

Buku ajar Pemrograman Komputer dengan dasar-dasar Python, sebagai buku pegangan perkuliahan Pemrograman Komputer di Perguruan Tinggi bagi Mahasiswa Program Studi Pendidikan Matematika ataupun Matematika. Dalam buku ini akan dibahas teori dan prakteknya serta contoh agar Mahasiswa mudah untuk memahaminya dengan bahasa yang mudah.

Python merupakan salah satu bahasa pemrograman yang populer belakangan ini karena beberapa faktor fleksibilitas dapat digunakan di berbagai platform (Windows, Mac, Linux dan lain sebagainya). Bahasa yang dibangun sangat mudah dan sederhana, sehingga kesederhanaan bahasanya tersebut ada yang berpendapat bahwa Python merupakan salah satu Bahasa Pemrograman yang mendekati bahasa manusia. Selain kesederhanaan bahasanya, Python juga memiliki modul yang lengkap dan mudah digunakan di berbagai aplikasi. Selain itu, Python dapat di peroleh secara gratis (Opensource) dan berkembang sangat cepat yang akan membantu Mahasiswa dalam pengembangan lebih lanjut.

FIP UMJ

Pemrograman Komputer

ISMAH, M.Si

2017

Pemrograman Komputer

ISMAH, M.Si
dasar
dasar python



Fakultas Ilmu Pendidikan
Universitas Muhammadiyah Jakarta



FAKULTAS ILMU PENDIDIKAN
UNIVERSITAS MUHAMMADIYAH JAKARTA
E-mail : fip@umj.ac.id | Website : www.fip.umj.ac.id



Buku Ajar
PEMROGRAMAN KOMPUTER
Dasar-Dasar Python

Ismah, M.Si

FAKULTAS ILMU PENDIDIKAN UMJ

Perpustakaan Nasional RI : Katalog Dalam Terbitan (KDT)

Pemrograman Komputer Dasar-Dasar Python

Penulis

Ismah, M.Si

Editor

Rahmita Nurul Muthmainnah, M.Pd, M.Sc

Desain Sampul dan Tata Letak

Muhamad Farid Fachrurozi, M.Pd

ISBN : 978-602-74522-8-2

Penerbit :

Fakultas Ilmu Pendidikan UMJ
Jln. KH. Ahmad Dahlan Cireundeu-Ciputat
Jakarta Selatan Tel +6221744 2028
Fax +6221744 2330
E-Mail fip_umj@yahoo.co.id

Cetakan Pertama, September 2017

Hak Cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan cara apapun tanpa izin tertulis dari penerbit

UCAPAN TERIMA KASIH

Dalam penyusunan buku ajar ini tidak lepas dari bantuan dan bimbingan berbagai pihak. Penulis menyampaikan terima kasih kepada Direktorat Pembelajaran Direktorat Jenderal Pembelajaran dan Kemahasiswaan Kementerian Riset Teknologi dan Pendidikan Tinggi, yang telah memberikan dana kepada UMJ dalam Hibah Revitalisasi Lembaga Pendidikan Tenaga Kependidikan tahap 2 tahun 2017. Melalui hibah revitalisasi tahap 2 melalui pengembangan perangkat pembelajaran, telah memberikan kesempatan dan memfasilitasi penulis dalam penyusunan buku ajar ini. Selain itu penulis juga mengucapkan terima kasih kepada :

1. Prof. Dr. Syaiful Bakhri., SH, MH., selaku rektor UMJ yang telah memberikan motivasi dalam penyusunan buku ini.
2. Dr. Herwina., MA., selaku Dekan Fakultas Ilmu Pendidikan UMJ yang telah memberikan sarana dan prasarana yang menunjang proses penyusunan buku ini.
3. Bapak/Ibu dosen Program Studi Pendidikan Matematika Fakultas Ilmu Pendidikan UMJ, yang telah memberikan kontribusi pengetahuan sehingga bermanfaat dalam penyusunan buku ini.
4. Kedua orang tua, suami, serta anak-anak yang selalu mendukung sepenuh hati dan memberikan doa restu, arahan, dan motivasi untuk segera menyelesaikan buku ajar ini.

Semoga amal baik yang telah diberikan mendapatkan balasan dari Allah SWT. Amin Ya Rabbal'alamin.

KATA PENGANTAR

Puji dan syukur kehadiran Allah SWT yang telah memberikan rahmat dan hidayahNya, sehingga buku ajar yang berjudul “Buku Ajar Pemrograman Komputer Dasar-Dasar Python” ini dapat diselesaikan. Buku ajar ini disusun sebagai salah satu sumber belajar mahasiswa Program Studi Pendidikan Matematika pada mata kuliah Pemrograman Komputer di Fakultas Ilmu Pendidikan Universitas Muhammadiyah Jakarta.

Komputasi sangat penting dalam menunjang perkembangan ilmu pengetahuan saat ini. Berbagai permasalahan kompleks yang apabila diselesaikan dengan cara manual (tanpa menggunakan komputer) mungkin membutuhkan waktu yang sangat lama, dan proses perulangan dalam menyelesaikan permasalahan tersebut dapat menimbulkan kesalahan karena keterbatasan kemampuan manusia (*human errors*).

Permasalahan kompleks yang biasa ditemukan yaitu pada bidang matematika. Sehingga dalam penyelesaian masalah matematika seringkali membutuhkan bantuan program komputer untuk memudahkan proses penyelesaian dan waktu yang diperlukan cukup singkat. Konsep matematika berkaitan dengan logika, dalam penyusunan program komputer menyelesaikan masalah juga dibutuhkan logika untuk memperoleh hasil (*output*) yang sesuai dengan yang dituju.

Aplikasi komputer yang digunakan dalam pembuatan pemrograman pada buku ini adalah Python. Python memiliki kode program yang sederhana dibandingkan bahasa pemrograman yang lainnya, sehingga memudahkan mahasiswa dalam memahaminya. Buku ajar ini memfasilitasi mahasiswa Prodi Pendidikan Matematika dalam membuat program komputer untuk menyelesaikan permasalahan di berbagai bidang, khususnya matematika.

Buku ajar ini masih kurang dari kesempurnaan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat diharapkan demi kesempurnaan buku ajar ini. Penulis juga berharap buku ajar ini bermanfaat dan berguna bagi kita semua, khususnya mahasiswa yang mengambil matakuliah pemrograman komputer. Amin

Cirendeu, September 2017

P e n u l i s

DAFTAR ISI

Ucapan Terima Kasih	i
Kata Pengantar	iii
Daftar Isi	v
Pengantar Matakuliah	1
Pendahuluan	5
Deskripsi Matakuliah Pemrograman Komputer	11
BAB 1. Instalasi dan Cara menggunakan Python	13
BAB 2. Tipe Data dan Operator	35
BAB 3. Pemilihan Kondisi	55
BAB 4. Menyusun Looping	65
BAB 5. Mengenal Data Struktur Python Tingkat Lanjut	81
BAB 6. Function	101
BAB 7. Exception	121
BAB 8. File	141
BAB 9. Penggunaan Class	157
BAB 10. Penggunaan Module	167
BAB 11. Operasi Matriks	193
Daftar Pustaka	205
Rencana Pembelajaran Semester	207
Biodata Penulis	215

PERANAN MATAKULIAH

Apa sih program komputer?

Program komputer adalah sebuah rangkaian tahapan yang terstruktur untuk menghasilkan sebuah solusi dari permasalahan yang dijalankan menggunakan komputer.



Mengapa kita membuat program komputer?

Banyak permasalahan kompleks yang mungkin kita tidak mampu menyelesaikannya atau mampu menyelesaikan namun membutuhkan waktu yang cukup lama, dikarenakan solusi untuk menyelesaikan permasalahan tersebut diperlukan langkah dengan perulangan (*looping*) dengan jumlah yang banyak.

Sebagai contoh menjumlahkan 10 bilangan puluhan dari sebuah barisan seperti berikut 23; 42; 65; 23; 80; 56; 38; 39; 83; 99. Untuk menyelesaikan permasalahan ini yaitu dengan menjumlahkan dua bilangan puluhan yang berada dalam barisan terlebih dahulu, hasil penjumlahan dari kedua bilangan puluhan tersebut akan dijumlahkan dengan satu bilangan puluhan berikut, dan seterusnya hingga bilangan puluhan akhir yang berada di barisan.

Proses menjumlahkan bilangan-bilangan puluhan tersebut merupakan proses *looping*. Proses yang dilakukan seperti yang digambarkan dalam ilustrasi ini membutuhkan waktu yang lama, misalkan penjumlahan dua buah bilangan puluhan membutuhkan waktu 10 detik, maka untuk mendapatkan jumlah dari 10 bilangan puluhan tersebut membutuhkan waktu 90 detik atau 1 menit 30 detik. Waktu 1 menit 30 detik memang

tampak singkat, namun waktu tersebut hanya untuk menyelesaikan permasalahan penjumlahan 10 bilangan puluhan dalam barisan, **bagaimana dengan penjumlahan 1000 bilangan ratusan dalam barisan? Atau 100 bilangan ribuan dalam barisan?**

Cukup jelaskah ilustrasi yang diberikan untuk menggambarkan peran program komputer dalam menyelesaikan permasalahan yang kompleks??

Dapat disimpulkan bahwa peran program komputer dapat mempermudah dalam menyelesaikan permasalahan yang kompleks. Setiap permasalahan pasti akan ada penyelesaiannya, demikian yang disampaikan di dalam Al-Qur'an surat Al-Insyiroh ayat 5 yang berbunyi.

فَإِنَّ مَعَ الْعُسْرِ يُسْرًا إِنَّ مَعَ الْعُسْرِ يُسْرًا

Karena sesungguhnya sesudah kesulitan itu ada kemudahan, sesungguhnya sesudah kesulitan itu ada kemudahan.

Pertanyaan yang sering timbul:

1. Tujuan dari pembuatan program komputer?

Jawabannya:

- a. Untuk menyelesaikan masalah yang kompleks dengan waktu yang efisien
 - b. Untuk mempersingkat langkah proses penyelesaian masalah yang kompleks
2. Apakah program yang sudah tersedia tidak dapat digunakan sehingga kita dianjurkan membuat program komputer?

Jawabannya:

- a. Program yang sudah ada tidak mampu menyelesaikan permasalahan kita

- b. Program yang tersedia memiliki spesifikasi yang tidak sesuai dengan spesifikasi yang dimiliki oleh komputer kita
 - c. Membuat program menyenangkan
3. Bagaimana memanfaatkan program yang ada?

Jawabannya:

- a. Dapat menggunakan program yang sudah ada jika program tersebut mampu menyelesaikan permasalahan kita
- b. Jika program yang ada tidak mampu menyelesaikan masalah kita, maka kita dapat mengembangkan program yang sudah ada disesuaikan dengan kebutuhan dari masalah yang akan diselesaikan
- c. Membuat program baru

PENDAHULUAN

Dalam pembuatan program komputer dibutuhkan sebuah bahasa yang dapat di transformasi ke dalam bahasa mesin.

Mengapa bahasa mesin? Karena program yang akan kita jalankan menggunakan komputer, dan komputer tidak memahami bahasa yang digunakan oleh manusia. Komputer adalah sebuah mesin yang dirancang oleh manusia, sehingga bahasa yang digunakan oleh komputer adalah bahasa mesin yang disebut dengan *assembly* dengan bilangan biner 0 dan 1. Oleh karena itu, agar perintah yang diberikan oleh manusia kepada komputer dapat dijalankan, maka bahasa yang digunakan oleh manusia sebagai perintah harus ditransformasi ke dalam bahasa mesin yang disebut bahasa pemrograman.

Tidak banyak orang yang mampu membuat kode program sebagai perintah manusia yang dapat dijalankan oleh komputer. Hanya orang-orang yang mempelajari pemrograman komputer yang mampu membuat kode program. Hal tersebut seperti yang tercantum didalam Al-Qur'an surat Al-Ankabut ayat 43 seperti berikut:

وَتِلْكَ الْأَمْثَلُ نَضْرِبُهَا لِلنَّاسِ ۖ وَمَا يَعْقِلُهَا إِلَّا الْعُلَمَاءُ

Dan kami jadikan contoh-contoh tersebut untuk manusia dan tidak ada yang bisa memahaminya kecuali orang-orang yang berilmu.

Apa yang dimaksud dengan bahasa pemrograman?

Bahasa pemrograman adalah bahasa yang digunakan untuk menyusun struktur (langkah-langkah) sebagai instruksi dalam menyelesaikan masalah menggunakan komputer. Melalui bahasa pemrograman seorang programmer dapat menentukan data yang akan diolah dan informasi yang akan dicetak.

Bahasa pemrograman disusun menjadi satu set (himpunan) instruksi yang saling terkait, atau dapat dikatakan memberi masukan (input), yang kemudian akan menghasilkan luaran (*output*) setelah program dijalankan (*running*). Alur bahasa pemrograman dapat digambarkan secara umum seperti berikut ini.



Proses dalam alur pemrograman berupa compile, yang merupakan sebuah proses untuk memeriksa kebenaran kode yang telah dibuat untuk kemudian diubah ke dalam bahasa mesin.

Bahasa pemrograman yang telah dikenal sejak dari sebelum tahun 1940an hingga saat ini, terus mengalami perkembangan. Bahasa pemrograman memiliki jenis sesuai dengan kebutuhan *user* (pengguna). Jenis bahasa pemrograman sesuai dengan kebutuhan *user* dibagi dalam dua bagian.

Pertama, bahasa pemrograman untuk kebutuhan khusus, seperti **sql** (*structured query language*) yang dikhususkan untuk pembuatan database dan **OpenGL** yang dikhususkan untuk pembuatan grafik.

Kedua, bahasa pemrograman untuk kebutuhan yang beraneka macam, seperti python yang sejenis dengan C, C++, pascal, basic, dan java, dapat digunakan untuk pembuatan tidak hanya berkaitan dengan perhitungan matematika, tetapi mampu juga untuk membuat database, grafik, bahkan game.

Mengapa menggunakan python?



Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Versi terakhir yang dikeluarkan CWI adalah 1.2.

Tahun 1995, Guido pindah ke CNRI sambil terus melanjutkan pengembangan Python. Versi terakhir yang dikeluarkan adalah 1.6. Tahun 2000, Guido dan para pengembang inti Python pindah ke BeOpen.com yang merupakan sebuah perusahaan komersial dan membentuk BeOpen PythonLabs. Python 2.0 dikeluarkan oleh BeOpen. Setelah mengeluarkan Python 2.0, Guido dan beberapa anggota tim PythonLabs pindah ke DigitalCreations.

Saat ini pengembangan Python terus dilakukan oleh sekumpulan pemrogram yang dikoordinir Guido dan Python Software Foundation. Python Software Foundation adalah sebuah organisasi non-profit yang dibentuk sebagai pemegang hak cipta intelektual Python sejak versi 2.1 dan dengan demikian mencegah Python dimiliki oleh perusahaan komersial. Saat ini distribusi Python sudah mencapai versi 3.5. Nama Python dipilih oleh Guido sebagai nama bahasa ciptaannya karena kecintaan Guido pada acara televisi Monty Python's Flying Circus. Oleh karena itu seringkali ungkapan-ungkapan khas dari acara tersebut seringkali muncul dalam korespondensi antar pengguna Python.

Perkembangan python terlihat dari beberapa versi yang yang dirilis yaitu versi 1 sampai dengan 3, yang dimulai pada bulan Januari tahun 1994 yaitu python versi 1.5. Sedangkan versi yang terakhir saat ini adalah python versi 3.4 yang dirilis tanggal 16 Maret 2014.

Python hadir sebagai solusi dalam mengatasi tantangan perkembangan perangkat lunak (*software*) yang sangat cepat dan beragam, untuk memenuhi kebutuhan pengguna (*user*). Python sebagai alat (*tool*) yang dapat digunakan untuk mengembangkan program-program aplikasi secara sederhana dan cepat. Selain itu, python juga dapat dijalankan diberbagai sistem operasi (*multiplatform*) seperti windows, linux, max OS, android, dan lainnya).

Open source python diberikan secara gratis, dan dapat didownload di web resmi python <https://www.python.org/>. Python mulai banyak dikenal oleh para programmer Indonesia, python masuk dalam kategori *high level language* (bahasa pemrograman tingkat tinggi) dan *object oriented dynamics language* (bahasa yang berorientasi objek dinamis), namun struktur dan kode program yang digunakan sangat sederhana sehingga mudah untuk diterapkan dalam membuat program.

Kini Python menjadi salah satu bahasa pemrograman yang populer digunakan oleh pengembangan web, aplikasi web, aplikasi perkantoran, simulasi, dan masih banyak lagi. Hal ini disebabkan karena Python bahasa pemrograman yang dinamis dan mudah dipahami (dikutip dari <https://teknojurnal.com> pada tanggal 20 Agustus 2017).

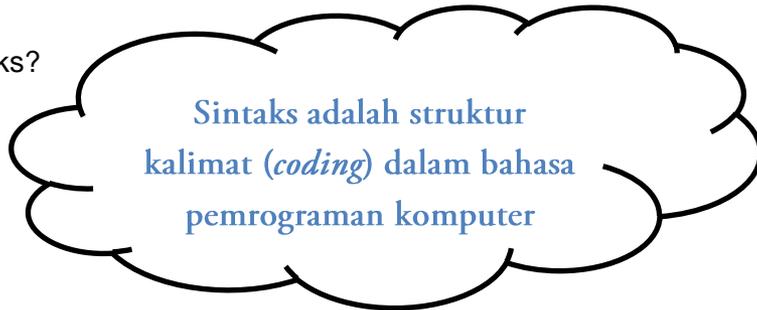
Bahasa pemrograman python menggunakan teknik interpretasi (*interpretation*) yang artinya kode program yang dituliskan akan dicek di setiap baris dan langsung dieksekusi apabila tidak ditemukan kesalahan (*error*). Namun apabila ditemukan kesalahan dibaris tertentu maka program akan di hentikan pada baris tersebut.

Secara umum keunggulan yang ditawarkan oleh python yaitu:

1. Berorientasi pada objek
2. Struktur pemrograman yang handal

3. Arsitektur yang dapat dikembangkan (*extendible*) dan ditanamkan (*embeddable*) dalam bahasa lain
4. Sintaks yang mudah dibaca

Apa itu sintaks?



Bagaimana cara belajar hingga menguasai python?

Selain menggunakan buku ajar ini sebagai pegangan dalam memahami python, di internet ada beberapa situs yang menyediakan penjelasan serta tutorial untuk dapat memahami python. Situs-situs python yang tersedia di internet, diantaranya ada yang berbayar dengan menawarkan kepada pembaca mengikuti kursus belajar python secara *online* atau daring (dalam jaringan), namun ada juga situs yang tidak berbayar (*free*). Berikut ini situs python yang dapat digunakan sebagai sumber pembelajaran, beberapa materi yang ada di dalam buku ini juga mengacu dari situs-situs berikut ini:

1. <http://www.learnpython.org/>
2. <https://www.codecademy.com/>
3. <https://teamtreehouse.com/>
4. <https://trinket.io/>
5. <http://pythontutor.com/>

Diantara situs-situs tersebut memiliki keunggulan masing-masing, selamat membaca dan mempelajari python!.

DESKRIPSI MATAKULIAH PEMROGRAMAN KOMPUTER

Capaian Pembelajaran Lulusan :

1. Menguasai konsep pedagogi-didaktik matematika untuk melaksanakan pembelajaran matematika inovatif di pendidikan menengah pertama dan atas yang berorientasi pada kecakapan hidup.
2. Menguasai konsep matematika yang meliputi logika matematika dan himpunan, aljabar, geometri, teori peluang dan statistika, matematika diskrit, pemodelan matematika, program linear, kalkulus, persamaan diferensial, metode numerik, dan analisis yang mendukung pembelajaran matematika di pendidikan dasar dan menengah serta untuk studi lanjut.
3. Menguasai pengetahuan faktual tentang fungsi dan manfaat teknologi khususnya teknologi informasi dan komunikasi yang relevan untuk pembelajaran matematika.

Capaian Pembelajaran Mata Kuliah :

1. Instalasi Python
2. Memahami cara menggunakan python
3. Memahami tipe data dan operator python
4. Memahami konsep pemilihan kondisi
5. Memahami konsep looping
6. Memahami data struktur Python tingkat lanjut
7. Memahami cara membuat *Function*
8. Memahami cara membuat exception
9. Memahami cara membuat file
10. Memahami penggunaan class

11. Memahami penggunaan module
12. Memahami cara menyelesaikan masalah matematika menggunakan python

Dekripsi Singkat Mata Kuliah :

Mata kuliah ini membahas mengenai bahasa pemrograman Python sebagai bahan pembuatan program matematika, matakuliah ini mencakup materi tentang: pengenalan python, tipe data dan operator, pemilihan kondisi, looping, data struktur python tingkat lanjut, *Function*, exception, membuat file, pengenalan class dan module dan aplikasi pada masalah matematika.

Materi Pembelajaran/Pokok Bahasan :

1. Cara menggunakan Python
2. Mengetahui tipe data dan operator
3. Membuat pemilihan kondisi
4. Menyusun looping
5. Mengetahui data struktur Python tingkat lanjut
6. Membuat *Function*
7. Membuat exception
8. Membuat file
9. Pengenalan class
10. Pengenalan module
11. Operasi Matriks

Media Pembelajaran :

Power Point, video pembelajaran

BAB 1. Instalasi dan Cara Menggunakan Python

Pokok Bahasan :

1. Instalasi python
2. Cara menggunakan Python

Deskripsi Materi :

Sebelum masuk materi cara menggunakan python, mahasiswa akan menginstal program python di komputer masing-masing. Program python terlebih dahulu di unduh dari <https://www.python.org/> serta referensi yang telah di sediakan di berbagai situs resmi python. Setelah python terinstal mahasiswa akan diperkenalkan cara menggunakan python, membuat program dengan prompt dan tanpa prompt, serta fungsi *sintaks print* melalui ilustrasi penggunaan **print** dengan berbagai kondisi.

Sub CPMK :

- 1.1 Download python
- 1.2 Instal python
- 2.1 Memahami cara menginstalasi python
- 2.2 Memahami cara Menggunakan Python Interpreter Prompt dan Teks Editor
- 2.3 Memahami cara Mencetak Informasi dengan *Function* “print”
- 2.4 Memahami cara Menerima Masukan Data dengan *Function* “raw_input” dan “input”
- 2.5 Memahami Hal Lain yang Harus Diingat dalam Penggunaan Python

Waktu : 2x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Python terlahir sudah cukup lama, namun pengguna python di Indonesia mulai ramai di Indonesia pada tahun 2000an. Python mulai digemari oleh kalangan akademisi dalam mempelajari pemrograman, struktur data serta berbagai sarana yang ditawarkan oleh python.

Python merupakan bahasa pemrograman yang *freeware* atau perangkat bebas dalam arti sebenarnya, tidak ada batasan dalam penyalinannya atau mendistribusikannya. Lengkap dengan *source codenya*, *debugger* dan *profiler*, fungsi sistem, GUI (antarmuka pengguna grafis), dan basis datanya.

Penting diketahui :

open source : istilah yang digunakan pada sebuah software atau perangkat lunak yang bisa digunakan oleh siapa saja dengan membuka atau membebaskan source codenya (sumber kode program) sehingga dapat mengetahui dengan jelas cara kerja software tersebut, selain itu siapa saja dibebaskan untuk mengubah ataupun memperbaiki jika ditemukan kelemahan-kelemahan pada software tersebut.

debugger : berasal dari kata *bug* yang merupakan suatu kesalahan desain pada suatu perangkat keras komputer atau perangkat lunak komputer yang menyebabkan peralatan atau program itu tidak berfungsi semestinya. Sedangkan debugger adalah sebuah proses menganalisa alur kerja program, mencari dan mengurangi *bug*, atau kerusakan di dalam sebuah program komputer atau perangkat keras sehingga perangkat tersebut bekerja sesuai dengan harapan.

GUI (Graphical User Interface), adalah antarmuka pada sistem operasi atau komputer yang menggunakan menu grafis agar mempermudah para penggunanya untuk berinteraksi dengan komputer atau sistem operasi

Matematika memiliki permasalahan yang kompleks, sehingga membutuhkan waktu yang lama untuk menyelesaikannya secara manual. Diperlukan alat komputasi sebagai solusi yang mampu menyelesaikan permasalahan yang kompleks tersebut dengan waktu yang singkat.

Matematika memiliki keterkaitan dengan komputasi, logika matematika digunakan sebagai dasar pemrograman. Banyak fungsi matematika yang digunakan untuk mengembangkan perangkat komputer. Di dalam matematika dan komputer dibutuhkan sebuah algoritma sebagai rancangan yang berisi kumpulan perintah untuk menyelesaikan masalah. Kesederhanaan kode program yang ada dalam python, memberikan kemudahan bagi kalangan yang berlatar belakang selain IT (*Information and Technology*) seperti matematika, dalam memahami cara pembuatan kode program python untuk menyelesaikan permasalahan yang terkait dengan bidangnya.

1. Instalasi Python

Sebelum melakukan proses instalasi python, terlebih dahulu unduh (*download*) *software* python di web resmi python <https://www.python.org/>. Versi python yang di gunakan dalam buku ini adalah python 2.7.13 dan digunakan pada perangkat komputer dengan sistem operasi windows.

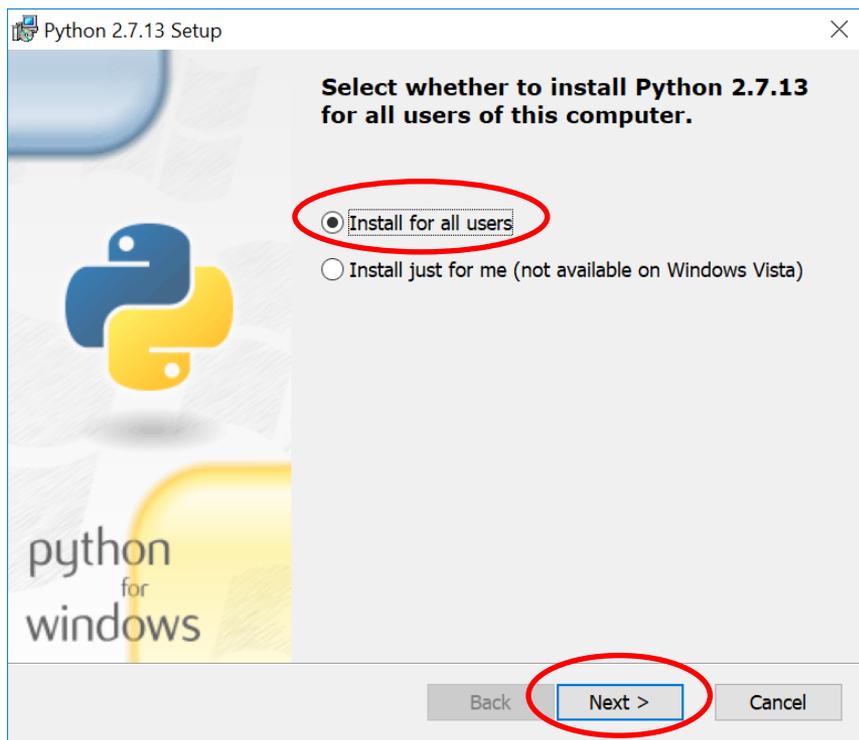
Python versi 2 dan 3 dirilis secara paralel yang artinya pada saat versi 2 masih digunakan dan mengalami perkembangan, telah dirilis pula dalam waktu yang hampir bersamaan python versi 3. Python versi 2.6 dirilis pada tanggal 1 Oktober 2008 dan versi 2.7 dirilis pada tanggal 3 Juli 2010, namun pada tanggal 3 Desember 2008 telah dirilis pula python dengan versi 3.

Tidak terlalu banyak mengalami perubahan yang signifikan dari python versi 2 ke versi 3, dalam hal bahasa dan sintaks. Hanya ada beberapa kode yang dihilangkan atau berubah fungsi seperti kode **raw_input** dan **input** yang tersedia dalam python versi 2.7, sedangkan dalam python versi 3.4 hanya tersedia kode **input** saja.

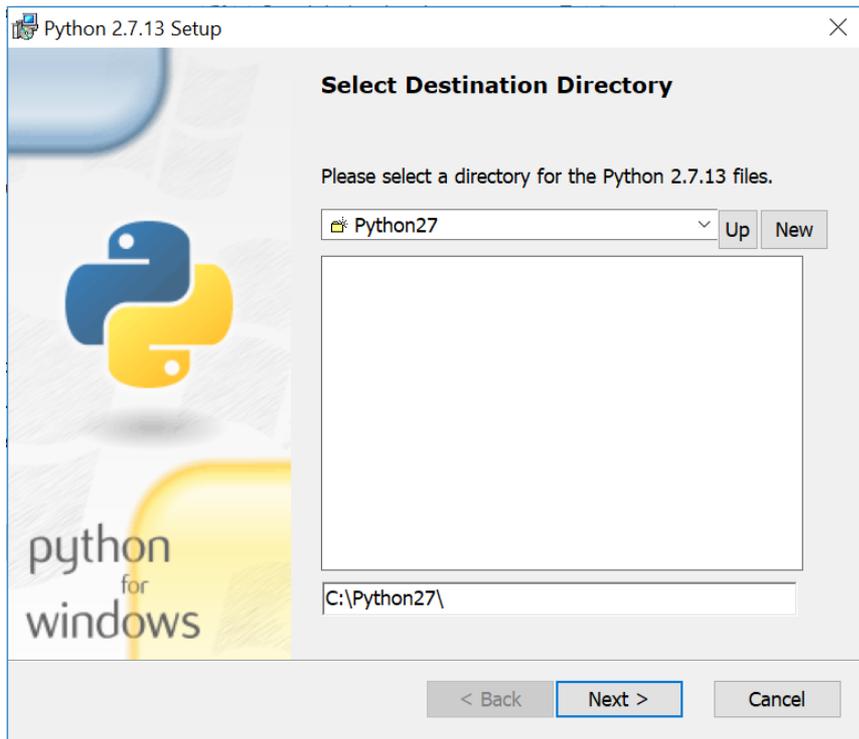
a. Buka software python 2.7.13

 python.rar	25/07/2017 10.46	WinRAR archive	21.454 KB
<input checked="" type="checkbox"/>  python-2.7.13.msi	25/07/2017 11.19	Windows Installer Pa...	18.712 KB
 python-3.6.2.exe	25/07/2017 11.19	Application	29.793 KB

b. Setelah muncul jendela seperti gambar berikut ini, lakukan proses penginstalan dengan memilih *install for all users* kemudian klik *next*.



- c. Selanjutnya akan muncul jendela berikut ini, untuk memilih direktori tempat penyimpanan/ruang program python. Setelah tempat penyimpanan ditentukan, klik *next* untuk lanjut ke langkah berikutnya.

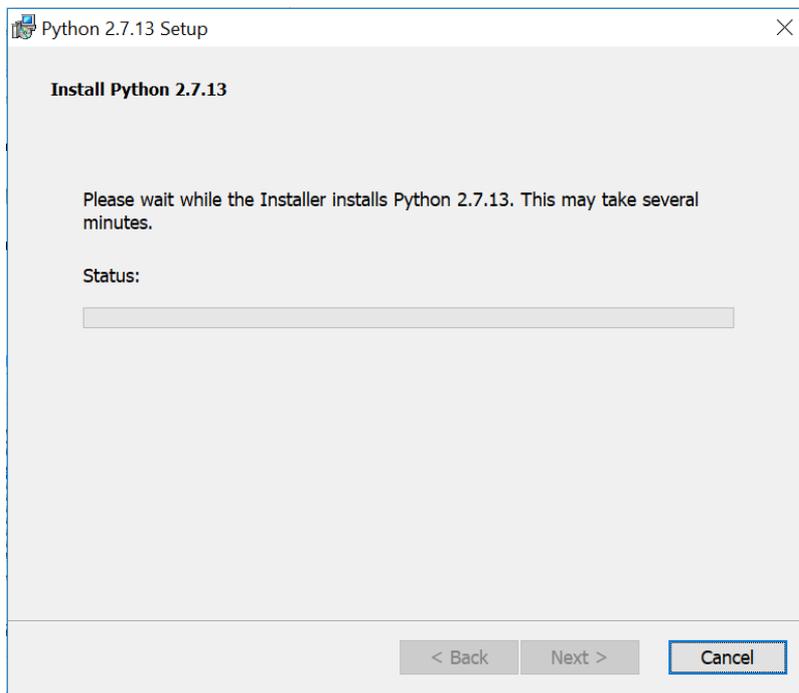


(catatan : upayakan memilih tempat penyimpanan program/software komputer di drive C tempat penyimpanan sistem operasi komputer, umumnya drive D, E, dst merupakan tempat penyimpanan data)

- d. Tahap selanjutnya akan muncul jendela seperti gambar berikut. Pada tahap ini langsung klik *next* dengan mengabaikan semua tool (*disk usage* dan *advanced*).



e. Proses instalasi sedang berlangsung



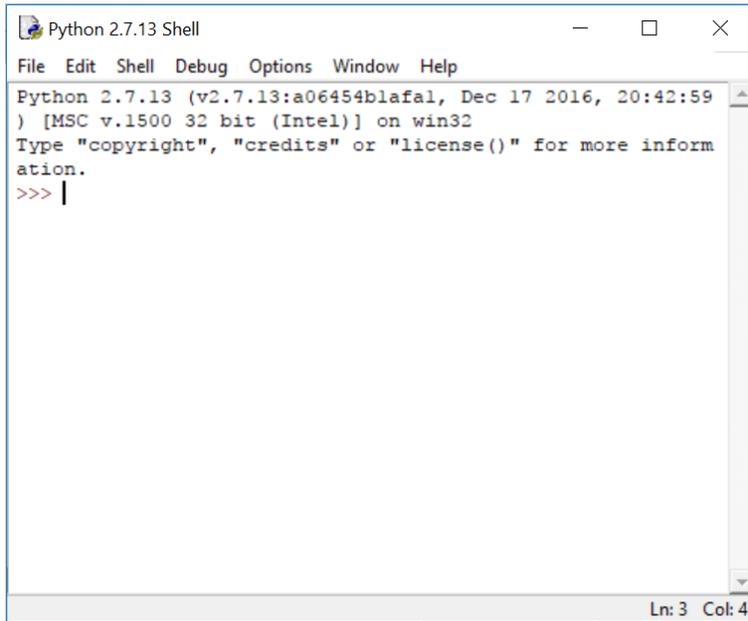
f. Proses instalasi selesai



Saat ini laptop atau komputer anda telah terinstal software python versi 2.7.13.

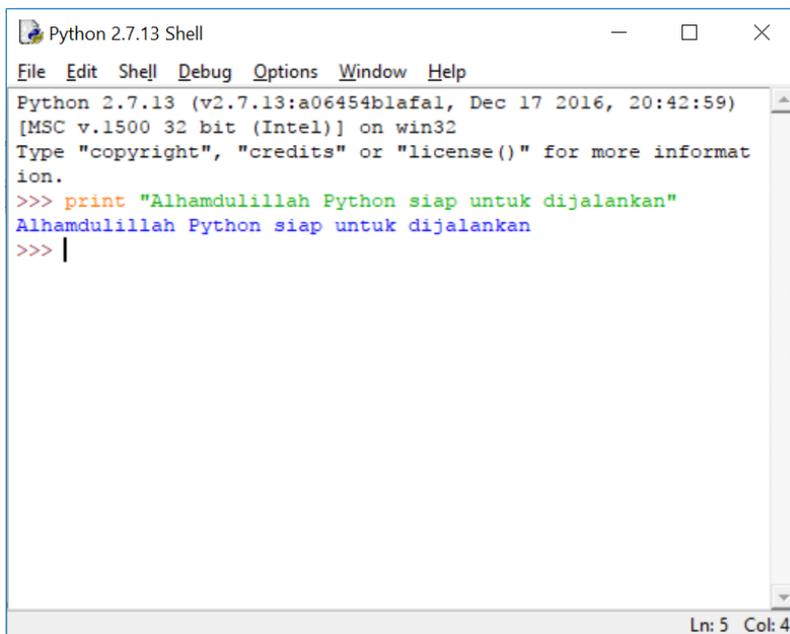
2. Cara Menggunakan Python

Dalam lingkungan windows untuk menjalankan python dapat menggunakan program IDLE (*Integrated Development Environment*). Langkah awal untuk menjalankan python adalah klik IDLE (Python GUI) dalam menu **start programs**. Kode program yang ditulis dalam python (*file.py*) dieksekusi menggunakan program Python Interpreter. Saat pertama kali python interpreter dijalankan akan muncul prompt dengan tanda `>>>` sebagai lokasi untuk menuliskan perintah-perintah. Tampilan awal IDLE python seperti berikut ini.



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454blafal, Dec 17 2016, 20:42:59
) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> |
```

Sebagai latihan tuliskan setelah prompt print "Alhamdulillah Python siap untuk dijalankan" kemudian enter, hasilnya akan seperti pada tampilan berikut ini.



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454blafal, Dec 17 2016, 20:42:59)
[MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informat
ion.
>>> print "Alhamdulillah Python siap untuk dijalankan"
Alhamdulillah Python siap untuk dijalankan
>>> |
```

Tulisan yang dicetak warna biru merupakan hasil dari perintah yang ditulis. Fungsi **print** dalam python sebagai perintah mencetak argumen (pernyataan) yang dituliskan ke layar. Python interpreter akan mengeksekusi semua argumen yang dituliskan. Dalam python versi 2.7 kode program **print** dapat dituliskan tanpa diiringi dengan tanda () atau dengan tanda (), sedangkan dalam python versi 3 kode program **print** harus diiringi tanda (), sebagai contoh `print ("Alhamdulillah Python siap untuk dijalankan")`.

Apabila argumen yang akan dieksekusi tidak cukup ditulis dalam satu baris, maka argumen dapat disambung menggunakan tanda *backslash* (\), seperti contoh berikut ini:

```
>>> panjang =10
>>> lebar = 5
>>> luas = panjang *\
        lebar
>>> luas
50
>>> print 'kita pasti \
bisa'
kita pasti bisa
```

Tanda *backslash* tidak perlu digunakan apabila argumen dituliskan menggunakan tanda (), [], maupun { }, seperti contoh berikut ini:

```
a = [1, 2,
     3, 4,
     5, 6]
>>> a
[1, 2, 3, 4, 5, 6]
```

Python dapat memunculkan nilai terakhir yang dioperasikan dengan menggunakan perintah *underscore* (`_`), seperti berikut:

```
>>> a = [1, 2,
         3, 4,
         5, 6]
>>> a
[1, 2, 3, 4, 5, 6]
>>> _
[1, 2, 3, 4, 5, 6]
>>> _ + [7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Tanda titik koma (`;`) dalam python bukan digunakan untuk mengakhiri perintah seperti pada program Pascal, C++, dan lainnya. Tetapi titik koma dalam python digunakan untuk memisahkan perintah-perintah yang dituliskan dalam satu baris.

```
>>> print 'perilakumu adalah refleksi dari \
kepribadianmu'; print 'sehingga jagalah \
perilakumu untuk pribadi yang baik'; 5 * 6

perilakumu adalah refleksi dari kepribadianmu
sehingga jagalah perilakumu untuk pribadi yang baik
30
>>>
```

Hasil yang dicetak setiap perintah di tuliskan pada baris yang baru. Untuk mencetak semua perintah dalam satu baris diakhir perintah dituliskan tanda koma (`,`).

```
>>> print '6 *',; print '5 =',;hasil=5*6; print hasil
6 * 5 = 30
```

Jika tidak ingin menggunakan prompt, maka kita dapat menggunakan *text editor* dengan cara klik menu *file* kemudian pilih *new file*. Program yang dituliskan menggunakan prompt berfungsi untuk menjalankan program yang sederhana atau sekedar menguji coba modul-modul yang ada dalam python (seperti contoh sebelumnya), sedangkan untuk menjalankan program yang kompleks, seperti *looping* dapat menggunakan *text editor*.

a. Menjalankan program menggunakan *text editor*

Ketika sebuah *ekspresi* diketikkan pada Python prompt, ia akan dinilai dan hasilnya akan dengan otomatis diperlihatkan pada baris dibawahnya. Seperti menggunakan kalkulator, jika kamu mengetik ekspresi ini kamu akan mendapatkan hasil.

Namun, lain dengan skrip *text editor*. Penilaian dari ekspresi tidak ditampilkan secara otomatis. Perintah-perintah seperti pernyataan, pertanyaan, hasil perhitungan, dan grafik yang telah dituliskan didalam pemrograman akan dicetak sebagai *output* jika terdapat *sintaks* yang dikenali oleh bahasa pemrograman sebagai perintah mencetak. Dalam python *sintaks* untuk mencetak dari perintah menggunakan operator **print**.

Bentuk kode

```
print <argumen>
```

Python sensitif terhadap penulisan huruf kapital dan huruf kecil, semua operator dalam python dituliskan menggunakan huruf kecil. Penulisan kode operator yang benar dalam python ditandai dengan warna tulisan merah (orange). Apabila warna tulisan tidak berubah (hitam) artinya penulisan kode belum benar.

Argumen pada bentuk kode `print <argumen>` dituliskan dengan tanda petik apabila berbentuk string (karakter) yang akan di bahas lebih

lengkap pada materi tipe data. Sedangkan apabila argumen berbentuk numerik (angka) tidak memerlukan tanda petik.

Berikut ini akan diberikan ilustrasi penggunaan *sintaks print* dengan berbagai kondisi. Sebelumnya siapkan *text editor* kemudian tuliskan berbagai perintah menggunakan *sintaks print* seperti berikut ini.

```
# mencetak sebuah kalimat menggunakan tanda " "  
print "Bismillahirrahmannirrahim"  
  
# mencetak angka  
print 123456  
  
# mencetak variabel  
Nama_saya = "Ismah"  
print Nama_saya  
  
# mencetak operasi bilangan  
panjang = 10  
lebar = 5  
print panjang*lebar  
  
# mencetak variabel diantara statement  
print "Nama Saya ",Nama_saya  
print panjang," x ",lebar,"=",panjang*lebar  
  
#mencetak variabel dengan format string  
luas = panjang*lebar  
print "Luas Persegi Panjang"  
print "%d x %d = %d" %(panjang,lebar,luas)
```

Pada ilustrasi program diatas terdapat beberapa keterangan:

1. Tanda # digunakan untuk memberikan komentar dan tidak dieksekusi oleh program.
2. Keterangan variabel diiringi dengan tanda petik tunggal (' ') atau petik dua ("") apabila berbentuk *text* (string).
3. Mencetak variabel yang berada diawal, ditengah atau diakhir *statement* dipisahkan dengan tanda koma.
4. Mencetak variabel dengan format string menggunakan kode %d (akan dibahas lebih rinci pada bab berikutnya).

Program diatas dapat kita jalankan dengan memilih menu *Run* kemudian pilih *Run Module* atau tekan F5 pada keyboard komputer Anda. Sebelum *output* muncul, Anda akan diminta untuk menyimpan file program anda dengan ekstensi file *.py. Setelah file tersimpan, maka akan keluar *output* dari program yang dijalankan di jendela baru *python 2.7.13 shell* seperti tampilan berikut ini.

```
Bismillahirrahmannirrahim
123456
Ismah
50
Nama Saya  Ismah
10 x 5 = 50
Luas Persegi Panjang
10 x 5 = 50
```

Untuk meng*copy* kode program dalam python sangat sederhana, dapat langsung memblok semua kode program yang ingin di*copy* kemudian tekan ctrl+c atau pilih edit pada menu bar lalu pilih *copy*. Setelah itu pada program aplikasi yang akan dituliskan kode program seperti microsoft word tekan ctrl+v atau pilih file pada menu bar kemudian pilih *paste*. Cara ini sama dengan yang dilakukan pada aplikasi microsoft, pada

menu bar python seperti halnya melakukan undo (ctrl+z) yaitu untuk mengembalikan tulisan yang telah terhapus, ataupun untuk memotong tulisan (ctrl+x) yaitu untuk menghilangkan dan mengkopi tulisan, dan lain sebagainya.

b. Fungsi `raw_input` dan `input`

`raw_input` merupakan perintah untuk memasukkan data melalui keyboard oleh user (pengguna) yang dikonversi ke dalam *string* (karakter), meskipun user memasukkan data numerik (angka), tanggal, waktu dan lain sebagainya, output yang dihasilkan tetap berupa *string*. Contoh penggunaan `raw_input` dapat dilihat pada ilustrasi program berikut untuk menghasilkan jumlah kembalian dalam bentuk rupiah, dengan menginput total belanja dan uang yang dimiliki.

```
# berlatih menggunakan raw_input

Nama=raw_input("Nama Anda=")
Total_belanja=raw_input("\nTotal Belanja = Rp ")
Uang_anda=raw_input("\nUang Anda = Rp ")
Kembalian = Uang_anda-Total_belanja
Print ("Kembalian = Rp "), kembalian
```

Output yang dihasilkan setelah *running* program diatas akan terjadi kesalahan (*error*). Hal tersebut dikarenakan apabila *user* menginput angka pada *statement* **Total Belanja**, komputer akan membaca angka tersebut sebagai string karena menggunakan perintah `raw_input ()`, sehingga komputer tidak dapat melakukan proses operasi matematika (penjumlahan, pengurangan, perkalian, pembagian, dan lainnya), dalam hal ini proses pengurangan untuk memperoleh variabel **kembalian**.

Berikut ini output yang dihasilkan setelah menjalankan program diatas, keterangan kesalahan (*TypeError*) dapat dilihat pada tulisan yang dibold.

```
= RESTART: D:/Latihan Python/latihan raw input 1.py =
Nama Anda=Ismah

Total Belanja = Rp 30000

Uang Anda = Rp 50000

Traceback (most recent call last):
  File "D:/Latihan Python/latihan raw input 1.py",
line 6, in <module>
    Kembalian = Uang_anda-Total_belanja
TypeError: unsupported operand type(s) for -: 'str'
and 'str'
```

Output yang dihasilkan dapat sesuai dengan jenis data yang diinginkan, apabila terlebih dahulu melakukan konversi ke tipe data yang diinginkan. Untuk bilangan bulat dapat dikonversi dengan menambahkan kode int (*integer*) seperti ilustrasi berikut ini.

```
# berlatih menggunakan raw_input

Nama=raw_input("Nama Anda=")
Total_belanja=raw_input("\nTotal Belanja = Rp ")
Uang_anda=raw_input("\nUang Anda = Rp ")
Kembalian = int(Uang_anda)-int(Total_belanja)
Print ("Kembalian = Rp "), kembalian
```

int, adalah singkatan dari integer yang berarti tipe data numerik atau bilangan bulat, penjelasan lengkapnya akan dibahas pada materi tipe data.

Kode program di atas untuk menghitung variabel kembalian yang berasal dari pengurangan total belanja dengan uang yang Anda miliki. Sehingga output yang dihasilkan setelah **running** program sebagai berikut ini.

```
= RESTART: D:/Latihan Python/latihan raw input 1.py =  
Nama Anda=Ismah  
  
Total Belanja = Rp 30000  
  
Uang Anda = Rp 50000  
  
Kembalian = Rp 20000
```

Selain menggunakan cara diatas, kita dapat menggunakan perintah **input ()** untuk variabel dengan tipe data numerik. **Input ()** memiliki fungsi yang sama dengan **raw_input ()**, tetapi pada **input ()** tipe data yang dimasukkan harus numerik, sehingga tidak perlu mendeklarasikan jenis data (int) ke dalam variabel seperti pada contoh sebelumnya.

```
# Berlatih menggunakan raw_input dan input  
  
Nama=raw_input('Nama Anda=')  
Total_belanja=input('Total Belanja= Rp ')  
Uang_anda=input('Uang Anda= Rp ')  
kembalian=Uang_anda-Total_belanja  
print('Kembalian= Rp '),kembalian
```

raw_input yang tertera pada program berfungsi untuk memerintahkan user menginput string (karakter) melalui *keyboard*. Sedangkan **input** berfungsi untuk memerintahkan user menginput numerik (angka) melalui *keyboard*. Sehingga output yang dihasilkan setelah program di **run** seperti tampilan berikut ini.

```
= RESTART: D:/Latihan Python/latihan raw input 2.py =  
Nama Anda= Ismah  
Total Belanja= Rp 1435000  
Uang Anda= Rp 1500000  
Kembalian= Rp 65000
```

c. Hal-hal yang perlu diingat dalam python

Didalam python versi 2.7 terdapat beberapa karakter khusus yang harus dituliskan sebelum menuliskan karakter/kalimat, seperti berikut ini:

\n	garis baru
\f	karakter sesudahnya pada garis baru
\t	Tab
\a	Bell

Contoh:

```
print '\tTRUKUN ISLAM'  
print '1. Syahadat'  
print '\a2. Shalat'  
print '\n3. Zakat'  
print '\n4. Puasa'  
print '5. Pergi haji'
```

Output dari kode program diatas dapat dilihat berikut ini:

RUKUN ISLAM

1. Syahadat
2. Shalat
3. Zakat
4. Puasa
5. Pergi haji

Selain itu, ada format khusus yang digunakan untuk memperoleh data sesuai dengan yang diinginkan, seperti pada tabel berikut.

Simbol	Keterangan
%c	Mencetak karakter
%s	Mencetak semua jenis data menjadi string
%d, %i, %u	Mencetak angka sebelum koma, atau bilangan bulat
%o	Mencetak angka okta
%x	Mencetak angka heksa
%f	Mencetak angka real, 6 digit di belakang koma
%e	Mencetak bilangan eksponensial
%g	Mencetak angka real, 3 digit di belakang koma

Ada empat basis bilangan yang sering digunakan yakni :

1. Bilangan berbasis dua atau yang sering disebut dengan bilangan biner (*binary*), digit yang digunakan adalah 0 dan 1
2. Bilangan berbasis delapan atau sering juga disebut oktal (*octal*), digit yang digunakan adalah 0, 1, 2, ..., 7
3. Bilangan berbasis sepuluh atau desimal yang sering kita digunakan dalam kehidupan sehari-hari, digit yang digunakan adalah 0, 1, 2, ..., 8, 9

4. Bilangan berbasis enambelas atau heksadesimal (hexadecimal), dengan digit yang digunakan adalah 0, 1, 2, 3, ..., 8, 9, A, B, ..., E, F. Dimana A sebagai pengganti nilai 10, B=11, C=12, dst.

Sebagai contoh konversi bilangan desimal ke oktal dan heksadesimal, dengan angka desimal 270

Cara Manual:

Konversi desimal ke oktal

270 dibagi 8 → 33 sisa 6

33 dibagi 8 → 4 sisa 1

4 dibagi 8 → 0 sisa 4

dituliskan dari bawah ke atas menjadi 416.

Sehingga hasil konversi 270 (desimal) menjadi 416 (oktal)

Konversi desimal ke heksadesimal

270 dibagi 16 → 16 sisa 14 (e)

16 dibagi 16 → 1 sisa 0 (0)

1 dibagi 16 → 0 sisa 1 (1)

dituliskan dari bawah ke atas menjadi 10e.

Sehingga hasil konversi 270 (desimal) menjadi 10e (heksadesimal)

Kode Program

```
>>> a=270
>>> print "%o"%(a)
416
>>> print "%x"%(a)
10e
>>>
```

-Ganbatte Kudasai-

B. Kesimpulan

1. Python merupakan aplikasi yang dapat digunakan oleh pemrogram secara gratis (*open source*), dan dapat di unduh di web resmi python <https://www.python.org/>.
2. Aplikasi python dapat dijalankan dengan interpreter. Penulisan program dapat dilakukan di dalam python interpreter, setiap kode program yang dituliskan setelah tanda prompt (>>>) akan langsung dieksekusi.
3. Penulisan program pada python interpreter digunakan untuk program yang sederhana atau untuk mencoba sintaks program. Sedangkan untuk program yang kompleks dapat dituliskan pada jendela text editor.
4. **Print** adalah sintaks untuk mencetak perintah di dalam python. Sedangkan untuk menginput data menggunakan sintaks **raw_input ()** atau **input ()**. **raw_input ()** berfungsi untuk menginput data bertipe string, dan **input ()** berfungsi untuk menginput data bertipe numerik.

Tugas mandiri 1**Penilaian :**

1. Proses pengerjaan 35%
2. Kreativitas 45%
3. Output 20%

Dibawah ini adalah potongan program yang belum sempurna, sempurnakan program dibawah ini dan tuliskan outputnya tanpa menggunakan bantuan komputer.

```
# Tugas Mandiri 1
"Selamat Datang di Fakultas Ilmu Pendidikan UMJ"
raw_input ('Nama : ')
Raw_input ('NIM : ')
input (Prodi : )
input (Usia : )
Print 'Selamat Mengikuti Peruliahan'
```


BAB 2.

Tipe Data dan Operator

Pokok Bahasan :

Mengenal tipe data dan operator

Deskripsi Materi :

Mengenal tipe data dan operator, tipe data python yang akan diperkenalkan diantaranya **boolean, complex, date, float, hexadecimal, integer, long, none, string, list, tuple, dictionary dan objek.** Sedangkan operator python diantaranya aritmatika, perbandingan, penugasan, biner, keanggotaan, identitas dan logika.

Sub CPMK :

- 3.1 Memahami jenis tipe data dan operator python
- 3.2 Memahami prioritas eksekusi operator di Python

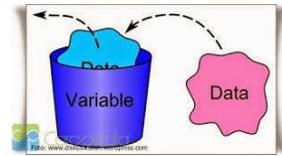
Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

1. Tipe Data

Tipe data adalah jenis nilai yang dapat ditampung oleh suatu variabel, dan pengertian dari variabel adalah tempat untuk menampung data yang berisi nilai konstanta ataupun yang dapat berubah-ubah selama proses program berjalan.



Penulisan variabel pada python memiliki aturan tertentu, yaitu :

- Karakter pertama harus berupa huruf atau garis bawah/underscore (`_`). Sebagai contoh **nama** atau **_nama_**
- Karakter selanjutnya dapat berupa huruf, garis bawah/underscore (`_`) atau angka. Sebagai contoh **jeniskelamin** atau **jenis_kelamin** atau **jenis2kelamin**
- Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya huruf kecil dan huruf besar dibedakan. Sebagai contoh, variabel **fipUmj** dan **fipumj** adalah variabel yang berbeda.

Akan terjadi kesalahan sintaks (*SyntaxError*) apabila penulisan variabel sebagai berikut :

- Karakter pertama dalam variabel berupa angka. Sebagai contoh **2nama**
- Di dalam karakter terdapat operator matematika (seperti tanda +, -, *, /, <, >). Sebagai contoh **nama-nama**, tanda - dianggap sebagai operator matematika untuk pengurangan
- Antar karakter dipisahkan oleh spasi. Sebagai contoh **pengalaman kerja**

Membuat variabel dalam Python caranya sangat mudah, kita cukup menuliskan variabel lalu mengisinya dengan suatu nilai dengan cara

menambahkan tanda sama dengan (=) diikuti dengan nilai yang ingin dimasukkan. Sebagai contoh apabila ingin membuat variabel **nama** maka dapat dituliskan,

```
nama = 'ismah'
```

Variabel yang memiliki tipe data tertentu pada python tidak perlu dideklarasikan seperti pada pemrograman PASCAL atau C. Pada python variabel dapat langsung dituliskan dengan tipe data tertentu, tanpa menuliskan tipe datanya python mampu membedakan tipe data yang di maksud.

Berikut contoh penulisan program untuk deklarasi variabel pada pemrograman PASCAL atau C.

```
var
p,l,luas:integer;//Deklarasi semua variabel bertipe integer
begin
  writeln('Masukkan nilai panjang');
  readln(p);           // Input nilai p
  writeln('Masukkan nilai lebar');
  readln(l);           // Input nilai l
  v := p * l ;         // Operasi matematika
  writeln(luas);       // Tampilkan nilai luas
  readln; //membaca nilai v sebelum program otomatis keluar
end.
```

Adapun tipe data python :

- a. **Boolean**, menyimpan data yang bernilai benar (*true*) atau salah (*false*)
- b. **Complex**, bilangan gabungan real dan imajiner, misalnya $3 + 2j$ (j = bilangan imajiner)

- c. **Date**, bilangan yang dapat dikonversi menjadi format tanggal, misalnya 17-08-2017
- d. **Float**, bilangan real atau bilang desimal, misalnya 3.14, 6.387
- e. **Hexadecimal**, bilangan dalam format heksa, misalnya 7b, 4d2
- f. **Integer**, bilangan bulat, misalnya 2, 9, 10, 20, 30, 15, 37, dan seterusnya
- g. **Long**, bilangan bulat yang panjang, misal 123456789123456789
- h. **None**, data yang tidak terdefinisi tipe data apapun
- i. **String**, data yang berisi karakter atau huruf. Bisa dibentuk dengan diapit tanda ' dan ', atau diapit " dan ".
- j. **List**, sebuah data array atau sekumpulan data yang menyimpan berbagai tipe data dan isinya bisa diubah. Ciri dari tipe data *list* adalah memiliki tanda kurung siku "[]". Akan lebih lengkap dibahas pada pertemuan berikutnya.
- k. **Tuple**, sebuah data array atau sekumpulan data yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah. Ciri dari tipe data *tuple* adalah memiliki tanda kurung buka dan kurung tutup "()". Akan lebih lengkap dibahas pada pertemuan berikutnya.
- l. **Dictionary**, sebuah data berupa untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai. Akan lebih lengkap dibahas pada pertemuan berikutnya.
- m. **Objek**, sebuah data yang berisi atribut dan method. Akan lebih lengkap dibahas pada pertemuan berikutnya.

Berikut ini contoh untuk mendapatkan tanggal dan waktu saat ini.

```
import time;
waktu = time.localtime(time.time())
print "Waktu lokal :", waktu
```

Contoh penulisan program python untuk berbagai tipe data seperti berikut ini.

```
# Latihan tipe data

# Tipe data string
string1 = "Semangat mencoba!!"
string2 = 'Kita pasti bisa!!'
print string1,string2

# Tipe data numerik (integer, complex, float, long)
numerik1 = 345
numerik2 = 1+89j
numerik3 = 67.090
numerik4 = 12390391000
print numerik1,numerik2,numerik3,numerik4

# Tipe data list dan tuple
tuple1 = ( 'Ismah', 2017, 'UMJ', '0330' )
tuple2 = (123, 'Pemrograman')

# Menampilkan semua data pada tuple
print tuple1

# Menampilkan elemen pertama pada tuple
print tuple1[0]

# Menampilkan data dari elemen kedua sampai ketiga
print tuple1[1:3]

# Menampilkan data dimulai dari elemen ketiga
print tuple1[2:]

# Menampilkan data tuple dua kali
print tuple2 * 2
```

```

# Menggabungkan tuple1 dengan tuple2
print tuple1 + tuple2
list1 = [ 'Ismah', 2017, 'UMJ', '0330' ]
list2 = [123, 'Pemrograman']

# Menampilkan semua data pada list
print list1
# Menampilkan elemen pertama pada list
print list1[0]
# Menampilkan data dari elemen kedua sampai ketiga
print list1[1:3]
# Menampilkan data dimulai dari elemen ketiga
print list1[2:]
# Menampilkan data list dua kali
print list2 * 2
# Menggabungkan list1 dengan list2
print list1 + list2

```

Bilangan yang didahului angka nol seperti 0330 dianggap sebagai string, dikarenakan tidak ada numerik (nilai) yang didahului angka nol. Oleh karena itu penulisan 0330 harus menggunakan tanda petik, dan apabila tidak menggunakan tanda petik maka akan keluar keterangan `SyntaxError: invalid token`.

Jenis data string dapat digunakan untuk variabel yang tidak memiliki nilai dan tidak dapat dioperasikan secara matematika seperti penjumlahan, pengurangan, perkalian, pembagian, perbandingan, dan lain-lain. Sebagai contoh variabel nomor telepon/handphone memiliki tipe data string, dikarenakan diawali dengan angka nol yang tidak memiliki nilai atau besaran serta tidak dapat dilakukan operasi matematika, sehingga penulisan nilai variabel harus menggunakan tanda petik.

Output yang dihasilkan setelah **run** program diatas dapat dilihat sebagai berikut:

```
Semangat mencoba!! Kita pasti bisa!!
345 (1+89j) 67.09 12390391000
('Ismah', 2017, 'UMJ', '0330')
Ismah
(2017, 'UMJ')
('UMJ', '0330')
(123, 'Pemrograman', 123, 'Pemrograman')
('Ismah', 2017, 'UMJ', '0330', 123, 'Pemrograman')
['Ismah', 2017, 'UMJ', '0330']
Ismah
[2017, 'UMJ']
['UMJ', '0330']
[123, 'Pemrograman', 123, 'Pemrograman']
['Ismah', 2017, 'UMJ', '0330', 123, 'Pemrograman']
```

Didalam penulisan kode program dapat dilakukan sesuai dengan keindahan atau seni menulis. Ada programmer yang menyukai penulisan melalui tahapan panjang, tetapi ada pula programmer yang menyukai penulisan kode program yang singkat. Sebagai contoh kode program untuk menghitung luas lingkaran.

Kode program 1:

```
respon = input("Berapa radiusnya ?")
r = float(respon)
luas = 3.14159 * r**2
print("Luas lingkarannya adalah ", luas)
```

kode program 2:

```
r = float( input("Berapa radiusnya ?") )
print("Luas lingkarannya adalah ", 3.14159 * r**2)
```

Kode program 3:

```
print("Luas lingkarannya adalah ", 3.14159*float(  
input("Berapa radiusnya ?")**2))
```

Terlihat perbedaan penulisan kode program 1 sampai dengan 3, namun output yang dihasilkan dari ketiga kode program tersebut adalah sama. Kode program 1 melalui tahapan yang panjang, sedangkan untuk kode program 3 penulisan terlihat lebih sederhana

2. Operator

Operator dalam python diklasifikasi dalam 7 kriteria berikut:

a. Aritmatika

Operator	Fungsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
**	Pangkat
/	Pembagian. Operator ini akan mengembalikan nilai integer jika kedua nilai (pembilang dan penyebut) bertipe integer (sisa bagi akan diabaikan), Dan akan mengembalikan nilai float jika salah satu atau keduanya bertipe float.
%	Modulus, sisa hasil bagi
//	Pembulatan hasil bagi. Menghasilkan nilai integer

Contoh:

```
# penjumlahan dua bilangan float  
>>> 14.55+15.35  
29.9
```

```

# pengurangan dua bilangan float
>>> 14.55-15.35
-0.79999999999999989
# angka awal adalah angka yang dipangkatkan 24
>>> 2**4
16
>>> 3.5**2
12.25
# pembagian dua bilangan integer
>>> 25/4
6          # menghasilkan bilangan integer
# pembagian bilangan float dengan integer
>>> 8.4/2
4.2       # menghasilkan bilangan float
# sisa hasil bagi
>>> 25%4
1
# pembagian (//) bilangan float dengan integer
>>> 8.4//2
4.0       # menghasilkan bilangan integer

```

Jika operator + atau – digunakan untuk nilai tunggal seperti +5, maka operator tersebut menunjukkan nilai positif atau negatif, +5 adalah positif 5 dan -5 adalah negatif 5.

b. Perbandingan (*comparison*)

Operator	Fungsi
==	Mengecek apakah kedua nilai sama atau berbeda
!=, <>	Mengecek apakah kedua nilai berbeda atau sama
>	Mengecek apakah nilai lebih besar dari
<	Mengecek apakah nilai kurang dari

>=	Mengecek apakah nilai lebih besar sama dengan dari
<=	Mengecek apakah nilai kurang dari sama dengan dari

Contoh:

```
>>> 35==35
True
>>> 35==25
False
>>> 35!=35
False
>>> 35!=25
True
>>> 35>25
True
>>> 35>=35
True
>>> 35<25
False
>>> 25<=35
True
```

c. Penugasan (*Assignment*)

Operator	Fungsi
=	Meletakkan nilai sebelah kiri ke nilai yang berada di sebelah kanan
+=	Menjumlahkan kedua nilai, dan hasilnya diletakkan di nilai yang berada di sebelah kanan
-=	Mengurangkan nilai di sebelah kanan dengan nilai yang disebelah kiri, dan hasilnya diletakkan di nilai yang berada di sebelah kanan
*=	Mengalikan kedua nilai, dan hasilnya diletakkan di nilai

yang berada di sebelah kanan

/= Membagi nilai di sebelah kanan dengan nilai yang disebelah kiri, dan hasilnya diletakkan di nilai yang berada di sebelah kanan

%= Sisa hasil nilai di sebelah kanan dengan nilai yang disebelah kiri, diletakkan di nilai yang berada di sebelah kanan

****=** Memangkatkan nilai di sebelah kanan dengan nilai yang disebelah kiri, hasilnya diletakkan di nilai yang berada di sebelah kanan

//= Pembulatan hasil bagi nilai di sebelah kanan dengan nilai yang disebelah kiri, hasilnya diletakkan di nilai yang berada di sebelah kanan

Contoh:

```
>>> a,b=6,5
>>> a=b
>>> a
5
>>> a+=b
>>> a
10
>>> a-=b
>>> a
5
>>> a*=b
>>> a
25
>>> a/=b
>>> a
5
```

```

>>> a%=b
>>> a
0
>>> b**=a
>>> b
1

```

d. Biner (*Bitwise*)

Operator	Fungsi
&	AND. Mengkonversi biner untuk kedua nilai, apabila terdapat angka 1 dalam bit yang sama maka akan dikonversi menjadi angka 1, selain itu dikonversi angka 0.
	OR. Mengkonversi biner untuk kedua nilai, apabila ada minimal ada angka 1 di bit, maka akan diberi angka 1 di bit hasil
^	XOR. Mengkonversi biner untuk kedua nilai, apabila ada angka yang sama (1 atau nol) di bit yang sama, maka akan diberi angka 0 di bit hasil, selain itu dikonversi angka 0.
~	NOT. Negasi biner, bit 0 menjadi 1 dan bit 1 menjadi 0
<< n	<i>Shift left</i> . Menggeser biner ke bit kanan sebanyak n
>> n	<i>Shift right</i> . Menggeser biner ke bit kiri sebanyak n

Contoh :

```

>>> a,b=8,10
>>> c=a&b
>>> c
8
>>> c=a|b
>>> c
10
>>> c=a^b

```

```

>>> c
2
>>> ~c
-3
>>> a<<2
32
>>> b>>2
2

```

Konsep operasi bitwise yaitu kedua bilangan integer yang akan dioperasikan dikonversi terlebih dahulu ke dalam bentuk biner basis 2 (0 dan 1). Setelah itu kedua bilangan tersebut baru diproses sesuai dengan fungsi operator yang dituliskan. Perhatikan cara mengkonversi bilangan integer 8 ke dalam bentuk biner basis 2 sebagai berikut:

8 dibagi 2 hasil 4 sisa 0
 4 dibagi 2 hasil 2 sisa 0
 2 dibagi 2 hasil 1 sisa 0
 1 dibagi 2 hasil 0 sisa 1
 Sehingga diperoleh biner 1000

Berikut ini ilustrasi konversi bilangan integer ke dalam bentuk biner serta pengoperasiannya:

```

1000          bentuk biner dari 8
1010          bentuk biner dari 10
----- &
1000          hasil = 8

1000          bentuk biner dari 8
1010          bentuk biner dari 10
----- |

```

```

1010      hasil = 10

1000      bentuk biner dari 8
1010      bentuk biner dari 10
----- ^
0010      hasil = 2

1000      bentuk biner dari 8
----- << 2
100000    hasil = 32

1000      bentuk biner dari 8
----- >> 2
0010      hasil = 2

```

e. Keanggotaan (*Membership*)

Operator	Fungsi
in	Mengecek nilai yang merupakan item di dalam list, jika benar maka akan menghasilkan <i>true</i> , jika sebaliknya akan menghasilkan <i>false</i>
not in	Mengecek nilai yang bukan merupakan item di dalam list, jika benar nilai bukan item di dalam list, maka akan menghasilkan <i>true</i> , dan jika sebaliknya akan menghasilkan <i>false</i>

Contoh:

```

>>> list = [1, 2, 3, 4, 5]
>>> 5 in list
True
>>> 7 in list
False
>>> list = [1, 2, 3, 4, 5]
>>> 2 not in list

```

```
False
>>> 7 not in list
True
```

f. Identitas (*Identity*)

Operator	Fungsi
Is	Mengecek apakah kedua variabel memiliki identitas yang sama, jika benar maka akan menghasilkan true
is not	Mengecek apakah kedua variabel memiliki identitas yang berbeda, jika benar maka akan menghasilkan true

Contoh:

```
>>> a,b=3,3
>>> a is b
True
>>> a is not b
False
>>> a,b='adi','ida'
>>> a is b
False
>>> a is not b
True
```

g. Logika (*Logic*)

Operator	Fungsi
And	Mengecek kebenaran dua kondisi
Or	Mengecek kebenaran salah satu dari dua kondisi
Not	Mengembalikan nilai (negasi)

Operator AND akan menghasilkan nilai TRUE apabila kedua kondisi bernilai TRUE, selain itu maka akan bernilai FALSE.

Sedangkan operator OR akan menghasilkan nilai FALSE apabila kedua kondisi bernilai FALSE, selain itu maka akan bernilai TRUE. Perhatikan tabel berikut ini.

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Contoh:

```
>>> print 5>2 and 3>2
True
>>> print 5>2 and 3<2
False
>>> print 5<2 or 3>2
True
>>> print 5<2 or 3<2
False
>>> print not 10<2
True
```

3. Prioritas Eksekusi Operator di Python

Dalam matematika dikenal prioritas operasi (proses perhitungan). Apabila dalam satu persamaan mengandung beberapa operasi, maka terdapat aturan prioritas dalam menjalankan operasi-operasi tersebut, yang disebut dengan istilah kabataku yang merupakan singkatan dari kali bagi tambah kurang. Perkalian dan pembagian akan di kerjakan lebih dulu sebelum penambahan dan pengurangan.

Dikarenakan ada aturan prioritas operasi pada matematika maka di dalam python juga memiliki aturan yang sama. Tetapi prioritas dalam python tidak hanya berlaku pada operator aritmatika saja, melainkan untuk operator perbandingan, bitwise, penugasan, identitas, keanggotaan serta logika.

Adapun prioritas operasi pada python ditunjukkan pada tabel berikut ini, dengan urutan prioritas mengacu pada nomor urut pada tabel, nomor urut pertama menjadi prioritas utama untuk dijalankan.

No	Operator	Keterangan
1	**	Aritmatika
2	~, +, -	Bitwise
3	*, /, %, //	Aritmatika
4	+, -	Aritmatika
5	>>, <<	Bitwise
6	&	Bitwise
7	^,	Bitwise
8	<=, <, >, >=	Perbandingan
9	<>, ==, !=	Perbandingan
10	=, %=, /=, //=, -=, +=, *=, **=	Penugasan
11	is, is not	Identitas
12	in, not in	Membership (Keanggotaan)
13	not, or, and	Logika

<https://www.belajarpython.com/2015/05/operator-python.html>

B. Kesimpulan

1. Tipe data adalah jenis nilai yang dapat ditampung oleh suatu variabel
2. Variabel adalah tempat untuk menampung data yang berisi nilai konstanta ataupun yang dapat berubah-ubah selama proses program berjalan
3. Jenis tipe data boolean, complex, date, float, hexadecimal, integer, long, none, string, list, tuple, dictionary, dan objek
4. Operator dalam python dibagi dalam 7 bagian, aritmatika (+, -, *, **, /, %, //), perbandingan (==, != atau <>, >, <, >=, <=), penugasan (=, +=, -=, *=, /=, %=, **=, //=), Biner (&, |, ^, ~, <<, >>), keanggotaan (in, not in), identitas (is, is not), logika (And, Or, Not).

Tugas mandiri 2**Penilaian :**

1. Proses pengerjaan 35%
2. Kreativitas 45%
3. Output 20%

1. Rumus untuk menghitung nilai akhir jika seseorang mendapatkan bunga majemuk, di wikipedia adalah sebagai berikut:

$$A = P \left(1 + \frac{r}{n} \right)^{nt}$$

dimana,

P = Jumlah pokok (*initial investment*)

r = Tingkat bunga nominal tahunan (as a decimal)

n = Jumlah bunga majemuk per tahun

t = Jumlah tahun

Tulis sebuah program Python yang memberikan jumlah pokok dari Rp. 100.000 untuk variabel P, berikan n nilai 12, dan berikan ke r tingkat bunga 8%. Lalu buat programnya menanyakan pengguna jumlah tahun dari seberapa lama uangnya akan diangsur. Hitung dan cetak jumlah final setelah t tahun.

2. Buatlah satu program python sederhana konversi waktu (detik) dari jam dan menit, dan konversi hari dari tahun dan bulan.

BAB 3. Pemilihan Kondisi

Pokok Bahasan :

Membuat pemilihan kondisi

Deskripsi Materi :

Sintaks untuk membuat pemilihan kondisi pada python diantaranya **if**, **if else** dan **if elif else**. Materi yang akan dibahas adalah penggunaan **if**, **if else** dan **if elif else**. Selain itu akan diberikan ilustrasi pembuatan program menggunakan if bersarang.

Sub CPMK :

- 4.1 Memahami Penggunaan Operator Kondisional dan Logika pada *Keyword* "if"
- 4.2 Memahami Penggunaan "else" pada "if"
- 4.3 Memahami Penggunaan "elif" pada "if"
- 4.4 Memahami cara menyusun program penyelesaian permasalahan matematika menggunakan kondisi

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Dalam kehidupan sehari-hari biasanya ditemukan klasifikasi atau pengelompokan data berdasarkan pada kategori yang ditentukan. Tujuan klasifikasi adalah mengelompokkan data sehingga terjadi keseragaman, selain itu dapat meringkas data dengan jumlah besar. Sebagai contoh nilai akhir mahasiswa yang dikategorikan ke dalam 5 tingkatan, A untuk nilai akhir 80-100, B untuk nilai akhir 68-79, C untuk nilai akhir 56-67, D untuk nilai akhir 45-55, E kurang dari 45.

Dalam pemrograman untuk mengklasifikasi data berdasarkan kategori, yaitu melalui pemilihan kondisi. Pemilihan kondisi sering ditemukan kehidupan sehari-hari, di dalam Al-quran pemilihan kondisi juga telah dijelaskan dalam surat An-Nahl ayat 93.

وَلَوْ شَاءَ اللَّهُ لَجَعَلَكُمْ أُمَّةً وَاحِدَةً وَلَكِنْ يُضِلُّ مَنْ يَشَاءُ وَيَهْدِي مَنْ يَشَاءُ ۗ وَلَتُسْأَلُنَّ عَمَّا
كُنتُمْ تَعْمَلُونَ

Artinya:

Dan kalau Allah menghendaki, niscaya Dia menjadikan kamu satu umat (saja), tetapi Allah menyesatkan siapa yang dikehendaki-Nya dan memberi petunjuk kepada siapa yang dikehendaki-Nya. Dan sesungguhnya kamu akan ditanya tentang apa yang telah kamu kerjakan. (16: 93)

Dalam python pemilihan kondisi dapat dilakukan menggunakan **if**, **if else**, dan **if elif else**. Kode **if** akan diakhiri tanda titik dua dan di baris yang baru penulisan program yang akan dieksekusi jika kondisi terpenuhi. Dalam membuat pemilihan kondisi Anda juga membutuhkan operator perbandingan (**==**, **<=**, **>=**, **>**, **<**, **<>**, **!=**) atau operator logika untuk menyusun kondisi yang Anda butuhkan.

Penting untuk diingat (bentuk umum penulisan) :

```
# gunakan operator perbandingan misal ==
# diakhiri tanda (:)
If var == kategori :
# penulisan blok program diberikan jarak (tab)
    print .....
```

1. Kondisi if

Berikut ini ilustrasi penggunaan **if** pada program.

```
#latihan if

Nama=raw_input('Tuliskan Nama Anda:')
print 'Selamat datang',Nama,\
      'Di Universitas Muhammadiyah Jakarta'
print '\n Apakah Anda mahasiswa \
      prodi pendidikan matematika?'
jawab1=raw_input ('\ny/t=')
if jawab1=='y':
    print Nama,'Silahkan mengikuti \
          kuliah kalkulus'
    print '\n Tuliskan jurusan \
          di sekolah asal Anda :'
    jawab2=raw_input ('\nIPA/IPS/Lainnya =')
    if jawab2=='IPA':
        print Nama, "Anda masuk kelompok A"
    if jawab2=='IPS':
        print Nama, "Anda masuk kelompok B"
    if jawab2!='IPA' and jawab2!='IPS':
        print Nama, "Anda masuk kelompok C"
if jawab1=='t':
```

```
print Nama, 'Tidak diperkenankan \
mengikuti kuliah kalkulus'
```

Program tersebut adalah untuk melakukan pengelompokan mahasiswa berdasarkan jurusan yang diambil di sekolah asal. Sebelumnya akan ditanyakan apakah user adalah mahasiswa dari prodi pendidikan matematika?, jika user menulis **t** maka perintah selanjutnya tidak dieksekusi dan akan tercetak **Anda tidak dapat mengikuti kuliah kalkulus**, tetapi jika user menulis **y** artinya user adalah **mahasiswa dari prodi pendidikan matematika** dan perintah berikutnya akan dieksekusi. Perintah berikutnya user atau mahasiswa akan ditanyakan jurusan di sekolah asal, jika menuliskan **IPA** maka mahasiswa tersebut akan masuk di **kelompok A**, jika mahasiswa mengambil jurusan **IPS** di sekolah asalnya maka mahasiswa tersebut akan masuk di **kelompok B**, dan jika jurusan **lainnya** maka mahasiswa akan masuk di **kelompok C**. Berikut ini berbagai kondisi output yang dihasilkan dari program tersebut.

```
Tuliskan Nama Anda:ismah
Selamat datang ismah Di Universitas Muhammadiyah Jakarta

Apakah Anda mahasiswa prodi pendidikan matematika?

y/t=y
ismah Silahkan mengikuti kuliah kalkulus

Tuliskan jurusan di sekolah asal Anda :

IPA/IPS/Lainnya =mesin
ismah Anda masuk kelompok C
```

```
Tuliskan Nama Anda:Ismah
Selamat datang Ismah Di Universitas Muhammadiyah Jakarta

Apakah Anda mahasiswa prodi pendidikan matematika?
```

```
y/t=y
```

```
Ismah Silahkan mengikuti kuliah kalkulus
```

```
Tuliskan jurusan di sekolah asal Anda :
```

```
IPA/IPS/Lainnya =IPA
```

```
Ismah Anda masuk kelompok A
```

```
Tuliskan Nama Anda:Ismah
```

```
Selamat datang Ismah Di Universitas Muhammadiyah Jakarta
```

```
Apakah Anda mahasiswa prodi pendidikan matematika?
```

```
y/t=t
```

```
Ismah Tidak diperkenankan mengikuti kuliah kalkulus
```

Contoh lain program penggunaan kode program **if** mencetak item yang berada dalam list seperti berikut ini:

```
for i in [20, 13, 16, 54, 81, 103]:  
    if i % 2 == 1: # Jika bilangannya adalah ganjil  
        continue # Jangan diproses  
    print(i)  
print("done")
```

Kode program `continue` berfungsi untuk melanjutkan perintah berikutnya apabila bilangan yang berada dalam **list** adalah ganjil (sisa pembagian dengan 2 sama dengan 1). Sehingga output yang dihasilkan dari kode program di atas adalah:

```
20  
16  
54  
done
```

2. Kondisi if else

else adalah sintaks yang digunakan dalam python apabila terdapat kondisi yang berlawanan dengan kondisi **if**, **else** digunakan beriringan dengan **if**. Statemen **else** harus ditulis lurus dengan statemen **if** dan setelah **else** harus diberi titik dua (:) serta perintah di baris yang baru diberi jarak (tab). Berikut ilustrasi program menggunakan kondisi **if else**.

```
#latihan kondisi if else

print 'Ayo main tebak-tebakan'
angka1=input('\nMasukkan angka pertama :')
angka2=input('\nMasukkan angka kedua :')
jumlah=angka1+angka2
jawaban=input('\nBerapa jumlah dari kedua angka tersebut?')
if jawaban==jumlah:
    print ('\n Good job!')
else:
    print ('\n Belajar lebih giat ya!')
```

Program diatas adalah permainan sederhana menjumlahkan dua buah bilangan. Dua buah bilangan diinput oleh user, kemudian user harus menuliskan hasil penjumlahan dari dua bilangan yang diinput. Apabila hasil penjumlahan salah maka akan tercetak komentar **Belajar lebih giat ya!**, tetapi apabila hasil penjumlahan yang dituliskan benar maka akan tercetak komentar **Good job**. Output setelah **run** program tersebut dapat dilihat pada tampilan berikut ini:

```
Ayo main tebak-tebakan
Masukkan angka pertama :28967
Masukkan angka kedua :18270
```

```
Berapa jumlah dari kedua angka tersebut?8209340
```

```
Belajar lebih giat ya!
```

```
Ayo main tebak-tebakan
```

```
Masukkan angka pertama :213
```

```
Masukkan angka kedua :3000
```

```
Berapa jumlah dari kedua angka tersebut?3213
```

```
Good job!
```

3. Penggunaan if Bersarang

Kondisi **if else** dapat digunakan untuk kondisi yang bersarang, seperti contoh mahasiswa dinyatakan lulus matakuliah apabila mendapatkan nilai UAS diatas 68 dan nilai UTS diatas 58, apabila mahasiswa mendapatkan nilai UAS diatas 68 dan nilai UTS dibawah 58 mahasiswa dinyatakan lulus tetapi harus lebih giat belajar, dan apabila mahasiswa mendapatkan nilai UAS dibawah 68 mahasiswa dinyatakan tidak lulus. Berikut ini ilustrasi program dari contoh penentuan kelulusan mahasiswa.

```
# Latihan if else bersarang

print 'Menentukan kelulusan'
nama=raw_input ('\n Nama mahasiswa:')
nilai_uts = input('\n Masukkan nilai UTS:')
nilai_uas = input('\n Masukkan nilai UAS:')
if nilai_uas>=68:
    if nilai_uts>=58:
        print ('Selamat',nama,'Anda lulus')
    else:
        print (nama,'lulus, lebih giat belajar ya')
```

```
else:  
    print (nama, 'Anda belum lulus')
```

Output setelah **run** program:

```
Menentukan kelulusan  
  
Nama mahasiswa: Ismah\  
  
Masukkan nilai UTS:70  
  
Masukkan nilai UAS:68  
( 'Selamat', ' Ismah\\', 'Anda lulus')
```

4. Kondisi if elif else

Dalam python pemilihan kondisi yang kompleks dapat menggunakan **elif**. **elif** sama dengan **else** yang digunakan beriringan dengan **if**. **elif** digunakan ketika dalam suatu program terdapat multi kondisi, proses dari program yang berjalan dimulai dengan pengecekan kesesuaian sebuah nilai dengan kondisi pertama, apabila sudah sesuai maka akan langsung dieksekusi perintah yang pertama. Namun apabila tidak sesuai sebuah nilai dengan kondisi yang pertama, maka akan dilanjutkan pengecekan kesesuaian nilai dengan kondisi yang kedua, dan proses ini akan terus berlanjut sampai dengan nilai dianggap telah sesuai dengan kondisi ke – n (n adalah jumlah kondisi dalam program), dan program akan mengeksekusi perintah ke – n.

Agar lebih jelas terkait kondisi **if elif else**, perhatikan ilustrasi penggunaan **if elif else** untuk program konversi nilai akhir matakuliah (0-100) ke bobot nilai (A-E). Mahasiswa yang mendapatkan nilai akhir diatas 80 akan memperoleh bobot nilai A, yang mendapatkan nilai akhir

diatas 68 dan kurang dari 80 akan memperoleh bobot nilai B, yang mendapatkan nilai akhir diatas 46 dan kurang dari 68 akan memperoleh bobot nilai C, dan seterusnya. Berikut contoh program ilustrasi diatas.

```
# latihan menggunakan kondisi if else

print 'konversi nilai akhir ke bobot nilai'
nama=raw_input('\n Nama mahasiswa :')
matakuliah=raw_input ('\n Masukkan nama matakuliah :')
nilai=input ('\n Masukkan nilai akhir \
yang di peroleh :')
if nilai>=80:
    print('Mahasiswa bernama', nama,'matakuliah',\
matakuliah,'mendapatkan nilai A')
elif nilai>=68:
    print('Mahasiswa bernama', nama,'matakuliah',\
matakuliah,'mendapatkan nilai B')
elif nilai>=46:
    print('Mahasiswa bernama', nama,'matakuliah',\
matakuliah,'mendapatkan nilai C')
elif nilai<=45:
    print('Mahasiswa bernama', nama,'matakuliah',\
matakuliah,'mendapatkan nilai D')
else:
    print('Mahasiswa bernama', nama,'matakuliah',\
matakuliah,'mendapatkan nilai E')
```

Output setelah **run** program:

```
konversi nilai akhir ke bobot nilai
```

```
Nama mahasiswa :Ismah
```

```
Masukkan nama matakuliah :Pemrograman Komputer
```

```
Masukkan nilai akhir yang di peroleh :85
```

```
('Mahasiswa bernama', 'Ismah', 'matakuliah',  
'Pemrograman Komputer', 'mendapatkan nilai A')
```

B. Kesimpulan

1. Pemilihan kondisi dalam python menggunakan kode program **if**, **if else**, **if elif else**.
2. **else** dan **elif** digunakan apabila terdapat lebih dari satu kondisi yang berlawanan
3. Aturan penulisan **if**, **else** dan **elif** diakhiri dengan tanda titik dua (:), dan penulisan statemen berikut harus diberi jarak (tab), serta **if**, **else** dan **elif** ditulis sejajar.

TUGAS KELOMPOK 1

Penilaian:

Waktu : 60 menit

1. Kekompakan : 20%
2. Perencanaan : 15%
3. Kualitas soal : 20%
4. Alur dan kreatifitas program : 35%
5. Output : 10%

Tugas yang telah dikerjakan dikumpulkan dalam bentuk *softcopy* dengan format nama *file* (Nama kelompok)_Tugas1.py

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan kondisi **if** atau **if else** atau **if elif else**.

BAB 4. Menyusun Looping

Pokok Bahasan :

Menyusun *looping*

Deskripsi Materi :

Instruksi untuk melakukan looping (pengulangan) dalam program python yaitu menggunakan kode **for** atau **while**. Selain itu akan dibahas kode **range** yang dapat digunakan untuk membuat sebuah list tanpa harus menuliskan semua item. Cara menggunakan **range** dalam **for** dan **while** bersarang juga akan dibahas, hingga menyusun program untuk menyelesaikan permasalahan matematika menggunakan looping.

Sub CPMK :

- 5.1 Menenal pengulangan “for” dan “while”
- 5.2 Menyusun pengulangan dengan “for”
- 5.3 Memahami *Function* “range”
- 5.4 Menggunakan *Function* “range” pada Pengulangan “for”
- 5.5 Menyusun pengulangan dengan “while”
- 5.6 Memahami cara menyusun program penyelesaian permasalahan matematika menggunakan looping

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Pengulangan (*looping*) adalah sebuah proses yang terkadang terjadi didalam matematika, diantaranya menentukan kelipatan persekutuan terkecil (KPK) dan faktor persekutuan terbesar (FPB).

Langkah-langkah menentukan kelipatan persekutuan terkecil (KPK) dari dua bilangan adalah:

1. Menentukan kelipatan dari masing-masing bilangan.
2. Menentukan kelipatan persekutuan dari dua bilangan tersebut.
3. Menentukan kelipatan persekutuan yang nilainya paling kecil.

Contoh:

Tentukan KPK dari 8 dan 12!

Jawab:

$K_8 = 8, 16, 24, 32, 40, 48, 56, 64, 72, \dots$

$K_{12} = 12, 24, 36, 48, 60, 72, 84, \dots$

$KP\ 8\ dan\ 12 = 24, 48, 72, \dots$

$KPK\ dari\ 8\ dan\ 12 = 24.$

Langkah-langkah menentukan faktor persekutuan terbesar (FPB) dari dua bilangan adalah:

1. Menentukan faktor dari masing-masing bilangan.
2. Menentukan faktor persekutuan dari dua bilangan tersebut.
3. Menentukan faktor persekutuan yang nilainya paling besar.

Contoh:

Tentukan FPB dari 15 dan 20!

Jawab:

$F_{15} = 1, 3, 5, 15$

$F_{20} = 1, 2, 4, 5, 10, 20$

$FP\ 15\ dan\ 20 = 1, 3, 5.$

$FPB\ 15\ dan\ 30 = 5$

Tentukan FPB dari 24 dan 45!

Jawab:

F 24 = 1, 2, 3, 4, 6, 8, 12, 24

F 45 = 1, 3, 5, 9, 15, 45

FP 24 dan 45 = 1, 3.

FPB 24 dan 45 = 3

Contoh:

Ibu mempunyai 16 apel dan 40 jeruk. Ibu akan memasukkan buah-buahan tersebut dalam beberapa kantong plastik. Isi setiap plastik sama. Berapa jumlah plastik terbanyak yang dibutuhkan Ibu?

Penyelesaian:

Masalah diatas dapat diselesaikan dengan menggunakan FPB.

F16 = 1, 2, 4, 8, 16

F40 = 1, 2, 4, 5, 8, 10, 20, 40

FPB 16 dan 40 = 8

Jadi, jumlah plastik terbanyak yang dibutuhkan Ibu adalah 8

Dengan kata lain untuk menentukan FPB dari dua buah bilangan a dan b, akan dilakukan proses dengan alur sebagai berikut:

1. Apabila $r = 0$ ($r = a$ modulus b) maka FPB dari a dan b adalah b
2. Apabila $r \neq 0$, maka $a = b$ dan $b = r$,
3. Kemudian kembali ke langkah 1, dan seterusnya sampai ditemukan $r = 0$, sehingga diperoleh FPB adalah b.

Sedangkan alur untuk menentukan KPK dapat diperoleh dari nilai FPB

yaitu $KPK(a, b) = \frac{a \times b}{FPB(a, b)}$.

Perintah untuk melakukan proses pengulangan (*looping*) dalam program python ada dua jenis **for** dan **while**.

1. Pengulangan dengan for

for digunakan untuk melakukan proses pengulangan sebanyak n kali, dan n adalah jumlah item di dalam untaian atau himpunan (Ingat jenis tipe data *list* dan *tuple*!). Berikut ilustrasi program menggunakan **for**.

Penulisan kode program:

```
for item in nama_list:  
    <perintah berikutnya di tab>
```

```
# Latihan perintah looping  
print 'Hasil pangkat bilangan PRIMA 1-13'  
n=(1, 3, 5, 7, 9, 11, 13)  
for i in n:  
    a=i**i  
    print '\nIterasi ke-',i,'hasil pangkat',a
```

Program diatas ingin mencetak hasil pangkat bilangan prima 1 sampai dengan 13. Bilangan prima terdapat di dalam untaian bilangan dengan nama variabel n.

Berikut hasil output setelah program dijalankan.

```
Hasil pangkat bilangan PRIMA 1-13  
  
Iterasi ke- 1 hasil pangkat 1  
  
Iterasi ke- 3 hasil pangkat 27  
  
Iterasi ke- 5 hasil pangkat 3125
```

```
Iterasi ke- 7 hasil pangkat 823543
```

```
Iterasi ke- 9 hasil pangkat 387420489
```

```
Iterasi ke- 11 hasil pangkat 285311670611
```

```
Iterasi ke- 13 hasil pangkat 302875106592253
```

Tipe data item di dalam list tidak hanya numerik (*integer*, *float*, *complex*, dll), tetapi bisa juga dalam bentuk *string* (karakter), yang berarti harus menggunakan tanda petik.

Apabila list dalam kode **for** hanya dituliskan dalam satu kata ("universitas") atau kumpulan angka ("123456") tanpa menggunakan kurung buka tutup '(') dan koma (,) pada masing-masing angka, seperti pada contoh program berikut ini.

```
a = 0
for i in "universitas":
    a=1+a
    print '\nhuruf ke-',a,'adalah',i
```

maka output yang dihasilkan setelah program di **run**, seperti berikut:

```
huruf ke- 1 adalah u
huruf ke- 2 adalah n
huruf ke- 3 adalah i
huruf ke- 4 adalah v
huruf ke- 5 adalah e
```

```
huruf ke- 6 adalah r
huruf ke- 7 adalah s
huruf ke- 8 adalah i
huruf ke- 9 adalah t
huruf ke- 10 adalah a
huruf ke- 11 adalah s
```

2. Pengulangan dengan while

Perintah untuk melakukan proses pengulangan berikutnya dalam python yaitu **while**. Kode **while** membutuhkan operator perbandingan atau logika, pengulangan **while** akan bekerja apabila kondisi bernilai *true* dan akan berhenti bekerja sampai dengan kondisi bernilai *false*. Hal tersebut membuat fungsi **while** mirip dengan **if**.

Kode penulisan

```
while suatu_kondisi :
```

Berikut contoh program menggunakan **while**.

```
print 'penjumlahan 10 buah bilangan'
i=0
a=0
while (i<=10):
    angka=input('Masukkan angka')
    i=i+1
    angka=a+angka
    a=angka
print '\njumlah bilangan yang anda masukkan adalah:',a
```

Program diatas memerintahkan user untuk menginput 10 bilangan, yang kemudian 10 bilangan tersebut akan dijumlahkan. Output dari program diatas seperti yang terlihat pada tampilan berikut ini.

```
penjumlahan 10 buah bilangan
Masukkan angka 10
Masukkan angka 23
Masukkan angka 34
Masukkan angka 15
Masukkan angka 12
Masukkan angka 10
Masukkan angka 26
Masukkan angka 30
Masukkan angka 34
Masukkan angka 11
Masukkan angka 10

jumlah bilangan yang anda masukkan adalah: 215
```

Selain itu contoh lain program untuk menentukan akar-akar persamaan kuadrat menggunakan operator looping **while**.

Untuk mencari akar-akar persamaan kuadrat dengan bentuk $y = ax^2 + bx + c$, $a \neq 0$, dapat diselesaikan dengan beberapa kriteria bergantung pada nilai $d = b^2 - 4ac$.

Jika $d = 0$, maka akar $x_{1,2}$ bernilai real dimana $x_1 = x_2$ dengan persamaan :

$$x_{1,2} = \frac{-b}{2a}$$

Jika $d > 0$, maka akar $x_{1,2}$ bernilai real dimana x_1 tidak sama dengan x_2 dengan persamaan :

$$x_{1,2} = \frac{-b \pm \sqrt{d}}{2a}$$

Jika $d < 0$, maka $x_{1,2}$ akar-akar persamaan bernilai imajiner

```
#program menentukan akar persamaan kuadrat
print 'Menentukan akar persamaan kuadrat'
i = 0
n = input('\n Berapa kali anda ingin\
        mencoba program ini ?')
while (n!=i):
    a = input('\n Masukkan nilai a=')
    b = input('\n Masukkan nilai b=')
    c = input('\n Masukkan nilai c=')
    if (a!=0):
        d=(b**2)-(4*a*c)
        i = i+1
        if d==0:
            x1 = (-b)/(2*a)
            x2 = x1
            print ('\n nilai akar-akar persamaan\
                    kuadrat',x1,'dan',x2)
        if d>=0:
            x1 = (-b+(d**(1/2)))/(2*a)
            x2 = (-b-(d**(1/2)))/(2*a)
            print ('\n nilai akar-akar persamaan\
                    kuadrat',x1,'dan',x2)
        if d<=0:
            print ('\n nilai akar-akar persamaan\
                    kuadrat adalah bilangan imajiner')
    else:
```

```
print ('\n Bukan persamaan kuadrat')
print ('\n Terima kasih')
```

Output dari program diatas, user akan diminta untuk memasukkan berapa kali user akan melakukan pengulangan untuk mencari akar-akar persamaan kuadrat.

Apabila user memasukkan angka nol (0) berarti perintah yang akan tercetak **Terima kasih** (dikarenakan $n=i$). Tetapi jika user memasukkan angka lebih besar dari nol yang berarti $n \neq i$, maka perintah akan dilanjutkan dengan memasukkan nilai a, b dan c sebagai konstanta pada persamaan kuadrat.

Apabila user memasukkan angka nol untuk a, maka akan tercetak **Bukan persamaan kuadrat** dan **Terima kasih**. Namun jika user memasukkan angka selain nol untuk a, maka perintah untuk mencari akar-akar persamaan dieksekusi tergantung dari nilai d yang diperoleh dari $b^2 - 4ac$, jika $d = 0$ maka akan diperoleh akar rela yang kembar, jika d positif akan diperoleh akar real yang berbeda, dan jika d bernilai negatif akan diperoleh akar berupa bilangan imajiner.

Ilustrasi output yang dihasilkan dari program diatas dapat dilihat pada gambar berikut.

```
Menentukan akar persamaan kuadrat
```

```
Berapa kali anda ingin mencoba program ini ? 2
```

```
Masukkan nilai a= 1
```

```
Masukkan nilai b= 2
```

```
Masukkan nilai c= 4
```

```
nilai akar-akar persamaan kuadrat adalah bilangan
imajiner
```

```
Masukkan nilai a= 1

Masukkan nilai b= 6

Masukkan nilai c= 8
('\n nilai akar-akar persamaan kuadrat', -3, 'dan', -
4)

Terima kasih
```

3. Menggunakan fungsi range dalam operator pengulangan for

Penggunaan *list* dalam **for** apabila semua item dituliskan maka sungguh menyulitkan. Item pada *list* dapat dituliskan melalui **range**, fungsi **range** digunakan untuk mempersingkat penulisan item dalam *list*. Item-item tidak perlu dituliskan satu persatu melainkan hanya menuliskan item pertama (*start*), item akhir (*stop*) dan selisih item (*step*). Kode penulisan **range (start,stop,step)**.

Berikut ini beberapa macam penulisan fungsi **range** dan hasilnya.

```
# range hanya dituliskan satu angka
print range(10)
Hasilnya : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# range hanya dituliskan dua angka
print range(1,10)
Hasilnya : [1, 2, 3, 4, 5, 6, 7, 8, 9]

# range dituliskan tiga angka
print range (0,10,2)
Hasilnya: [0, 2, 4, 6, 8]
```

```
# apabila step lebih besar dari stop
print range (1,5,6)
Hasilnya : [1]

# apabila start lebih besar dari stop
print range(10,1,1)
Hasilnya : [ ]

# apabila start lebih besar dari stop
# dan step bernilai negatif
print range (10,1,-1)
Hasilnya : [10, 9, 8, 7, 6, 5, 4, 3, 2]

# apabila step angka nol
print range(1,10,0)
Hasilnya : error
```

Contoh program pengulangan dengan kode **for** yang menggunakan `range`, dapat diilustrasikan untuk menentukan bilangan bulat genap dan ganjil.

Berikut ini contoh program untuk menentukan sebuah bilangan genap dan ganjil dari bilangan bulat positif 0 (*start*) sampai dengan 100 (*stop*) dan selisih antar bilangan (*step*) adalah 1. Apabila bilangan bulat yang diwakili oleh variabel `i` dibagi 2 sisanya 0 (`i%2==0`) atau habis dibagi 2 maka bilangan tersebut merupakan bilangan genap, dan sebaliknya apabila bilangan bulat yang diwakili oleh variabel `i` dibagi 2 sisanya tidak sama dengan 0 (`i%2!=0`) atau tidak habis dibagi 2, bilangan tersebut merupakan bilangan ganjil.

```
# Latihan perintah looping for dengan range
print 'menentukan bilangan genap dan ganjil'
```

```
for i in range (0,25,1):
    if i%2==0:
        print i, 'merupakan bilangan genap'
    else:
        print i, 'merupakan bilangan ganjil'
```

Perhatikan output yang dihasilkan dari program diatas yang ditampilkan sebagian seperti yang tampak pada gambar berikut ini.

```
menentukan bilangan genap dan ganjil
0 merupakan bilangan genap
1 merupakan bilangan ganjil
2 merupakan bilangan genap
3 merupakan bilangan ganjil
4 merupakan bilangan genap
5 merupakan bilangan ganjil
6 merupakan bilangan genap
7 merupakan bilangan ganjil
8 merupakan bilangan genap
9 merupakan bilangan ganjil
10 merupakan bilangan genap
11 merupakan bilangan ganjil
12 merupakan bilangan genap
13 merupakan bilangan ganjil
14 merupakan bilangan genap
15 merupakan bilangan ganjil
16 merupakan bilangan genap
17 merupakan bilangan ganjil
18 merupakan bilangan genap
19 merupakan bilangan ganjil
20 merupakan bilangan genap
21 merupakan bilangan ganjil
22 merupakan bilangan genap
```

```
23 merupakan bilangan ganjil
24 merupakan bilangan genap
25 merupakan bilangan ganjil
```

Penting untuk diketahui:

Apabila penulisan operator dalam python diakhiri oleh tanda titik dua (:) seperti **if**, **else**, **elif**, **for**, dan **while**, maka perintah selanjutnya di baris yang baru (*new line*) harus di beri jarak (tab).

4. While Bersarang

Jika sebelumnya kita telah membahas mengenai kondisi **if** bersarang, sekarang akan dibahas mengenai **while** bersarang, yang artinya di dalam blok **while** utama terdapat kode **while**, dengan kode penulisan sebagai berikut.

```
while kondisi_1:
    while kondisi_2:
```

Sebagai contoh perhatikan program berikut ini.

```
i = 1
while i <= 3:
    j = 1
    string1 = ' '
    while j <= 10:
        k = i*j
        string2 = "%d"%(k)
        string1 = string1+string2
        j = j+1
    print string1
    print '\n'
    i = i + 1
```

Sehingga output yang dihasilkan setelah **running** program

```
12345678910  
  
2468101214161820  
  
36912151821242730
```

B. Kesimpulan:

Kode untuk membuat looping dalam python yaitu **for** dan **while**.

kode penulisan untuk **for**

```
for item in nama_list:
```

kode penulisan untuk **while**

```
while kondisi_1:
```

list dalam for dapat menggunakan **range** yang berfungsi untuk mempersingkat penulisan item.

Kode penulisan

```
range (start, stop, step)
```

Tugas mandiri 3**Penilaian :**

1. Proses pengerjaan 35%
2. Kreatifitas 45%
3. Output 20%

1. Sebagai bahan latihan silahkan Anda mencoba membuat program menggunakan fungsi `for` dengan *list* yang berisi *string*.
2. Buatlah program menjumlahkan semua bilangan, dengan banyaknya bilangan tidak ditentukan, tergantung dari input yang dimasukkan oleh user.
3. Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan operator pengulangan `for` atau `while`.

BAB 5.

Mengenal Data Struktur Python Tingkat Lanjut

Pokok Bahasan :

Mengenal data struktur Python tingkat lanjut

Deskripsi Materi :

Materi yang akan dibahas terkait dengan pengenalan tipe data *list*, *tuple* dan *dictionary*. Selain itu penting untuk diketahui cara mengubah, menghapus dan menambahkan item pada uraian dengan ketiga tipe *list*, *tuple* dan *dictionary*.

Sub CPMK :

- 6.1 Memahami Mengenal List, Dictionary dan Tuple
- 6.2 Memahami Cara Akses List, Tuple, dan Dictionary
- 6.3 Memahami mengubah Isi List, Tuple, dan Dictionary
- 7.1 Memahami menambahkan Data pada List, Tuple, dan Dictionary
- 7.2 Memahami Menghapus Isi List, Tuple, dan Dictionary
- 7.3 Memahami cara menyusun program penyelesaian permasalahan matematika

Waktu : 2x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

1. List, tuple dan dictionary

List, *tuple* dan *dictionary* merupakan tipe data dalam bentuk uraian (himpunan) yang berisi item-item berupa string atau numerik atau campuran string dan numerik. Dalam bahasa program yang lain seperti C++ dan PASCAL penulisan himpunan (set) nilai biasanya menggunakan istilah *array* (larik). Di dalam *list* dan *tuple* item-item ditulis dalam bentuk indeks dengan nomor urut item yang dimulai dari nol.

Perbedaan ketiga tipe data *list*, *tuple* dan *dictionary* terletak pada penulisan sintaks. Pada *list* sintaks yang digunakan adalah “ [] ” dan menggunakan koma (,) untuk memisahkan antara item yang satu dengan yang lainnya. Sedangkan *tuple* menggunakan “ () ” dan juga menggunakan koma (,) untuk memisahkan antara item yang satu dengan yang lainnya, sedangkan pada *dictionary* menggunakan “ { } ” yang berisi **key** (kunci) berupa string atau numerik, dan juga dapat dicampur string dengan numerik. Penulisan **key** beserta nilainya dipisahkan dengan titik dua (:). Bentuk sintaks *dictionary* sebagai berikut:

```
d = { 'key1': 'n1', 'key2': 'n2' }
```

Item-item didalam *list* dan *dictionary* dapat dirubah, tetapi dalam *tuple* item-item tidak dapat dirubah (*fix*). Namun item-item pada ketiga tipe data tersebut dapat ditambahkan dan dihapus.

Contoh pengecekan ketiga tipe data tersebut sebagai berikut.

```
a=(1,2,3)
b=[1,2,3]
c={'angka pertama':"1",'angka kedua':"2"}
type (a)
```

```
type (b)
type (c)
```

Hasil program setelah di **run**:

```
<type 'tuple'>
<type 'list'>
<type 'dict'>
```

Dictionary biasanya digunakan untuk sebuah daftar data dengan urutan acak. Dalam hal ini *dictionary* lebih baik dibandingkan *list* ataupun *tuple*, karena pada *list* atau *tuple* untuk mencetak beberapa item didalamnya kita harus mengingat urutannya, sedangkan pada *dictionary* kita dapat mencetak salah satu itemnya dengan menuliskan **key**. Sebagai contoh berikut ini penggunaan tipe data *list* dan *dictionary*.

```
# variabel A1 list dan A2 dictionary
A1=['Ismah', '0330118401', 'dosen', 'travelling']
A2={'nama':'Ismah','NIDN':'0330118401',
    'pekerjaan':'dosen','hobi':'travelling'}

# akan dicetak item Nama dan pekerjaan
print A1[0],A1[2]
print A2['nama'],A2['pekerjaan']
```

Output dari program tersebut

```
Ismah dosen
Ismah dosen
```

Program tersebut akan mencetak nama dan pekerjaan yang berada di dalam *list* dan *dictionary*. Pada *list* A1 untuk mencetak item tertentu kita

harus hafal letak atau posisi item tersebut, seperti pada program diatas akan dicetak nama dan pekerjaan maka **print A1[0],A1[2]** maksudnya mencetak item yang berada pada list pertama dan ketiga. Sedangkan pada *dictionary* bisa langsung menuliskan item yang ingin dicetak nama dan pekerjaan maka dapat dituliskan **print A2['Nama'],A2['pekerjaan']**. *List* dapat digunakan pada operator **for** dan **if**, begitu juga *dictionary*. Untuk mengecek apakah item berada dalam *dictionary* atau tidak, menggunakan sintaks **in**, sebagai contoh dari program diatas akan dicek apakah terdapat *key* 'nama' didalam *dictionary* A2:

```
print 'Nama' in A2
print 'tempat tinggal' in A3
```

Output yang tercetak pada perintah pertama adalah *true* karena ada *key* nama pada *dictionary* A2, sedangkan pada perintah kedua *false* karena tidak ada *key* tempat tinggal pada *dictionary* A3.

Berikut ini contoh penggalan program penggunaan sintaks **if** dan *dictionary*.

```
if 'tempat tinggal' in A2:
    print A2['tempat tinggal']
else:
    print 'tempat tinggal tidak ada di A2'
```

Output yang dihasilkan dari penggalan program diatas adalah tempat tinggal tidak ada di A2.

2. Mengubah Item dalam *List* dan *dictionary*

Item didalam *list* dan *dictionary* dapat dirubah, sedangkan pada *tuple* item bersifat *fix* sehingga tidak dapat dirubah. Di *dictionary* untuk

memperbarui nilai suatu kunci yang sudah ada, dengan sintaks seperti berikut ini:

```
daftar ={'key1':'n1', 'key2':'n2', 'key3':'n3',.....}  
daftar['key1'] = 'nilai pengganti n1'
```

Sebagai contoh daftar menu makanan di sebuah restoran yang berisi makanan dan minuman, untuk daftar makanan terdiri dari nasi ayam bakar, nasi ayam goreng, nasi pecel, gado-gado, dan nasi goreng, sedangkan daftar minuman terdiri dari es teh manis, es tawar, es jeruk, es campur, es teler dan air mineral. Daftar makanan dan minuman berubah-ubah bergantung pada hari, sehingga daftar menu makanan dan minuman tersebut dapat kita buat program menggunakan *list* dan *dictionary* sebagai berikut.

```
# Membuat daftar menu makanan dan minuman  
  
hari={'hari1':'senin','hari2':'selasa','hari3':'rabu',  
      'hari4':'kamis','hari5':'jumat'}  
  
menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',  
              'nasi pecel', 'gado-gado', 'nasi goreng']  
  
menu_minuman=['es teh manis', 'es tawar', 'es jeruk',  
              'es campur', 'es teler', 'air mineral']  
  
print hari['hari1']  
menu_makanan[0]='nasi bebek bakar'  
menu_minuman [0]='ice lemon tea'  
print 'Menu Makanan:',menu_makanan  
print 'Menu Minuman:',menu_minuman
```

```
menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',  
'nasi pecel', 'gado-gado', 'nasi goreng']  
  
menu_minuman=['es teh manis', 'es tawar', 'es jeruk',  
'es campur', 'es teler', 'air mineral']  
  
print '\n'  
print hari['hari2']  
menu_makanan[1]='nasi bebek goreng'  
menu_minuman [1]='es buah'  
print 'Menu Makanan:',menu_makanan  
print 'Menu Minuman:',menu_minuman  
  
menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',  
'nasi pecel', 'gado-gado', 'nasi goreng']  
  
menu_minuman=['es teh manis', 'es tawar', 'es jeruk',  
'es campur', 'es teler', 'air mineral']  
  
print '\n'  
print hari['hari3']  
menu_makanan[2]='nasi keredok'  
menu_minuman [2]='es kopyor'  
print 'Menu Makanan:',menu_makanan  
print 'Menu Minuman:',menu_minuman  
  
menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',  
'nasi pecel', 'gado-gado', 'nasi goreng']
```

```
menu_minuman=['es teh manis', 'es tawar', 'es jeruk',
'es campur', 'es teler', 'air mineral']

print '\n'
print hari['hari4']
menu_makanan[3]='ketoprak'
menu_minuman [3]='es serut'
print 'Menu Makanan:',menu_makanan
print 'Menu Minuman:', menu_minuman

menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',
'nasi pecel', 'gado-gado', 'nasi goreng']

menu_minuman=['es teh manis', 'es tawar', 'es jeruk',
'es campur', 'es teler', 'air mineral']

print '\n'
print hari['hari5']
menu_makanan[4]='nasi bakar'
menu_minuman [4]='es buah'
print 'Menu Makanan:',menu_makanan
print 'Menu Minuman:',menu_minuman

menu_makanan=['nasi ayam bakar', 'nasi ayam goreng',
'nasi pecel', 'gado-gado', 'nasi goreng']

menu_minuman=['es teh manis', 'es tawar', 'es jeruk',
'es campur', 'es teler', 'air mineral']
```

TUGAS KELOMPOK 2

Penilaian:

1. Diskusi : 20%
2. Output : 50%
3. Presentasi : 30%

Tuliskan output dari program diatas tanpa bantuan komputer dan jelaskan proses bekerja program tersebut melalui presentasi di depan teman-teman kelas

3. Menambahkan isi dalam *list* dan *dictionary*

Selain merubah isi *list* dan *dictionary*, indeks *list* dan *dictionary* juga dapat ditambahkan. Sintaks untuk menambahkan item ke dalam variabel yang memiliki tipe data *list* menggunakan operator **append**, dengan bentuk penulisan dalam program `<nama_variabel>.append(<item yang ditambahkan>)`, sedangkan sintaks untuk penambahan item pada *dictionary* sama seperti memperbarui item, hanya saja pada key dibuat baru.

Selain itu, kita dapat mengetahui jumlah indeks dalam *list* ataupun *dictionary* menggunakan sintaks **len**, `len(<nama_variabel>)`. Berikut ini contoh sederhana penambahan data nama orang serta jumlah indeks setelah data ditambahkan.

```
# Menambahkan item dalam list
data = ['ria','ari','adi','ida','rian','rani']

# menambahkan item di akhir variabel
data.append('nida')
# menambahkan item di akhir variabel
data.append('andi')
# jumlah item dalam variabel
print len(data)
# cetak item indeks ke 7
print data[6]
# cetak item indeks ke 8
print data[7]

print data
```

Output yang dihasilkan setelah **run** program

```
8
nida
andi
['ria', 'ari', 'adi', 'ida', 'rian', 'rani', 'nida',
'andi']
```

Latihan:

Berikut penggalan program untuk mencetak nama-nama presiden RI yang pertama kali menjabat hingga saat ini. Nama-nama presiden yang ada di daftar masih belum lengkap, tolong teman-teman buat program yang lengkap dengan menambahkan nama-nama presiden sesuai urutan yang belum ada didalam *list* menggunakan sintaks **append.**, sehingga semua nama presiden RI dapat masuk ke dalam daftar.

```
#Cetak nama-nama presiden RI yang pertama hingga saat ini
Nama_presiden=['soekarno','soeharto','bj habibie','sby','joko widodo']
```

Selain penambahan indeks pada *list*, kita dapat juga membuat irisan pada *list*. Irisan adalah sebagian item yang berada pada *list*. Berikut contoh mencetak irisan dari *list*.

```
# membuat irisan list
angka=[1,2,3,4,5,6,7,8,9]
irisannya=angka[2:5]
print angka
print irisannya
```

Output yang dihasilkan dari program diatas,

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 4, 5]
```

Perlu diingat urutan indeks dalam *list* dimulai dari 0 yang sebenarnya berada dalam urutan pertama. Sehingga irisan =angka [2:5] membuat irisan pada *list* di variabel angka [1, 2, 3, 4, 5, 6, 7, 8, 9], mulai dari item yang berada pada urutan ketiga sampai item urutan kelima, maka irisan yang tercetak adalah [3, 4, 5].

Irisan juga dapat dilakukan pada string (kalimat), sebagai contoh perhatian program berikut ini

```
# mengiris kalimat
pernyataan='Saya pasti bisa belajar python'

#mengiris dari urutan pertama hingga kelima
a = pernyataan[:5]

# mengiris dari urutan keenam hingga keenambelas
b = pernyataan[5:16]

# mengiris dari urutan ke-17 hingga ke-32
c = pernyataan[16:32]

print '\n',a
print '\n',b
print '\n',c

d=a + b + c

print '\n',d
```

Hasil output dari program diatas

```
Saya
pasti bisa
belajar python
Saya pasti bisa belajar python
```

4. Menghapus item pada *list* dan *dictionary*

Item pada *list* dan *dictionary* dapat dihapuskan, operator yang digunakan adalah **del**, dengan sintaks untuk *list* `del <nama_variabel>[indeks]` dan untuk *dictionary* `del <nama_variabel>[key]`. Sebagai contoh

```
a = [1,2,3,4,5,6]
d = { 'key1': 'n1', 'key2': 'n2' }
del a[5]          #menghapus item urutan keenam dari list
del d['key1']     #menghapus key1 dari dictionary
print a
print d
```

Output yang dihasilkan

```
[1, 2, 3, 4, 5]
{'key2': 'n2'}
```

Selain menggunakan sintaks **del**, menghapus item di dalam list dapat menggunakan sintak **remove** yang hanya memiliki satu parameter yang akan dihapus. Kode program untuk menghapus item di dalam list menggunakan **remove** `nama_list.remove(item)`. Ilustrasi

penggunaan kode program **remove** dapat dilihat pada program berikut ini.

```
>>> prodi = ['PAUD', 'PGSD', 'PGMI', 'PENMAT', 'PBSI',  
            'PBI']  
>>> #menghapus PGMI dari list prodi  
>>> prodi.remove('PGMI')  
>>> prodi  
['PAUD', 'PGSD', 'PENMAT', 'PBSI', 'PBI']
```

Penggunaan *list* dalam program dapat digabungkan dengan *dictionary*, seperti berikut ini.

```
daftar = {'nama':'ismah','pendidikan':['SD',"SMP","SMA","S1",  
            "S2'],'pekerjaan':'dosen'}
```

Pembahasan terkait penggunaan *list*, *tuple*, dan *dictionary* tidak hanya untuk menambahkan, merubah, menghapus, dan mengiris item atau key didalam dictionary. Akan tetapi kita dapat mencari nilai minimum, nilai maksimum, membandingkan dua variabel dengan tipe data yang sama, merubah tipe data dari list ke tuple atau sebaliknya.

max() digunakan untuk mencari nilai maksimum dalam list dan tuple yang berbentuk numerik (angka), sedangkan nilai yang berbentuk string fungsi **max()** mencari urutan huruf abjad yang terbesar (a,b,c,d,e,... urutan e lebih besar dari urutan a), apabila terdapat dua nilai yang memiliki huruf awal yang sama, maka akan dilihat berdasarkan urutan terbesar dari huruf kedua, dan seterusnya.

min () digunakan untuk mencari nilai minimum dalam list dan tuple yang berbentuk numerik (angka), sedangkan nilai yang berbentuk string fungsi **min()** mencari urutan huruf abjad yang terkecil (a,b,c,d,e,... urutan a lebih kecil dari urutan e), apabila terdapat dua nilai yang memiliki huruf

awal yang sama, maka akan dilihat berdasarkan urutan terkecil dari huruf kedua, dan seterusnya.

cmp() digunakan untuk membandingkan dua variabel yang memiliki tipe data yang sama (misal list dengan list), apabila item list pertama lebih banyak dari item list kedua maka akan menghasilkan nilai 1, jika kedua list sama maka akan menghasilkan nilai 0, dan jika item list pertama kurang dari item list kedua maka akan menghasilkan nilai -1. Sebagai contoh :

```
#mendefinisikan list, tuple dan dictionary
prodi1 = ["penmat", "paud","pgsd"]
warna=("merah", "hijau","kuning","biru")
mahasiswa={"nama":"ismah","nim":2016830001,
"gender":"perempuan"}
print prodi1, "\n",warna, "\n",mahasiswa,"\n"

#menambahkan list dan membandingkan list
prodi2 = prodi1+["pbi","pbsi"]
print prodi2
print cmp(prodi1,prodi2)
print cmp(prodi1,prodi1)
print cmp(prodi2,prodi1)

#mencari nilai maksimum dan minimum
print max(prodi1)
print min (prodi1)
print max (mahasiswa)
print min (mahasiswa)
```

Output yang dihasilkan setelah **run** program diatas dapat diperhatikan berdasarkan kode program yang dibuat.

kode program :

```
print prodi1, "\n", warna, "\n", mahasiswa, "\n"
```

Output :

```
['penmat', 'paud', 'pgsd']  
( 'merah', 'hijau', 'kuning', 'biru')  
{ 'nama': 'ismah', 'gender': 'perempuan', 'nim':  
2016830001 }
```

Kode program :

```
print prodi2
```

Output :

```
['penmat', 'paud', 'pgsd', 'pbi', 'pbsi']
```

Kode program :

```
print cmp(prodi1,prodi2)
```

Output :

```
-1 # dikarenakan jumlah item list prodi1 kurang dari item list prodi2
```

Kode program :

```
print cmp(prodi1,prodi1)
```

output:

```
0 # dikarenakan jumlah item list prodi1 sama dengan item list prodi1
```

Kode program :

```
print cmp(prodi2,prodi1)
```

output:

```
1 # dikarenakan jumlah item list prodi2 lebih dari item list prodi1
```

Kode program :

```
print max(prodi1)
```

output:

```
pgsd # dikarenakan semua item pada list prodi1 memiliki huruf abjad  
awal p, maka dibandingkan urutan pada huruf kedua untuk semua item
```

dalam list `prodi1`. Pada item **pgsd** huruf kedua adalah **g** yang berarti memiliki urutan paling besar pada urutan huruf abjad (a, b, c, d,...,z) dibandingkan huruf **e** pada **penmat** dan **a** pada **paud**.

Kode program :

```
print min (prodi1)
```

output:

`paud` # Pada item **paud** huruf kedua adalah **a** yang berarti memiliki urutan paling besar kecil pada urutan huruf abjad (a, b, c, d,...,z) dibandingkan huruf **e** pada **penmat** dan **g** pada **pgsd**.

Kode program:

```
print max (mahasiswa)
```

output:

```
nim
```

kode program :

```
print min (mahasiswa)
```

Output:

```
gender
```

5. Mencari Item dalam List

Pencarian item di dalam list dapat menggunakan sintaks **index ()**. Sama seperti **remove**, **index** juga hanya memiliki satu parameter dari item yang ingin dicari. Kode program untuk mencari item dalam list `nama_list.index(item)`

Perhatikan contoh berikut ini.

```
>>> prodi = ['PAUD', 'PGSD', 'PENMAT', 'PBSI', 'PBI']
>>> #mencari item PENMAT dari list prodi
>>> prodi.index('PENMAT')
2
```

Hasil pencarian item PENMAT di list prodi adalah 2, yang artinya PENMAT berada pada urutan ke-3, karena urutan indeks dalam list diawali dari 0.

B. Kesimpulan

Dari pembahasan yang telah diuraikan maka disimpulkan eksekusi terhadap *list* dan *dictionary* dapat dilakukan sebagai berikut.

```
# mencetak list, tuple dan dictionary
prodi = ["penmat", "paud", "pgsd", "teknik"]
warna = ("merah", "hijau", "kuning", "biru")
mahasiswa = {"nama": "ismah", "nim": 2016830001,
              "gender": "perempuan"}

# mengakses indeks
print prodi[0]
print warna [2]
print mahasiswa["nama"]

# mengupdate suatu item, tidak berlaku untuk tuple
prodi[3] = "pbi"
mahasiswa["nama"] = "desi"
print "\n", prodi
print mahasiswa

# menambahkan item, tidak berlaku untuk tuple
prodi.append("pbsi")
mahasiswa["tempat tinggal"] = "jakarta"
print "\n", prodi
print mahasiswa
```

```
# mengiris item
prodi_bahasa = prodi[3:5]
colour=warna[2:3]
print "\n",prodi_bahasa
print colour
```

Output dari program diatas

```
penmat
kuning
ismah

['penmat', 'paud', 'pgsd', 'pbi']
{'nama': 'desi', 'gender': 'perempuan', 'nim':
2016830001}

['penmat', 'paud', 'pgsd', 'pbi', 'pbsi']
{'nama': 'desi', 'gender': 'perempuan', 'tempat
tinggal': 'jakarta', 'nim': 2016830001}

['pbi', 'pbsi']
('kuning',)
```

Tugas mandiri 4

Penilaian :

1. Proses pengerjaan 35%
2. Kreatifitas 45%
3. Output 20%

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan *list*, *tuple* dan *dictionary* dengan operator lainnya seperti **for**, **while** atau **if else**.

BAB 6

Function

Pokok Bahasan :

Membuat *Function*

Deskripsi Materi :

Function merupakan salah satu operator dalam python untuk mempermudah penulisan yang menggunakan kode yang berulang-ulang dengan nilai yang berbeda-beda. Didalam *Function* akan dibahas materi terkait pengenalan *Function* tanpa *return* dan dengan menggunakan *return*, *default argument*, *Variable-length Argument*, *Keyword Argument* pada *Function*, *Keyword-length Argument* pada *Function*, *Variable Scope* pada Python, dan cara menyusun program penyelesaian permasalahan matematika dengan *Function*.

Sub CPMK :

- 9.1 Memahami Pengenalan *Function* Tanpa “return”
- 9.2 Memahami *Function* yang Menggunakan “return”
- 9.3 Memahami Default Argument pada Python
- 9.4 Memahami Variable-length Argument pada Python
- 10.1 Memahami *Keyword Argument* pada *Function*
- 10.2 Memahami *Keyword-length Argument* pada *Function*
- 10.3 Memahami Variable Scope pada Python
- 10.4 Memahami cara menyusun program penyelesaian permasalahan matematika dengan *Function*

Waktu : 2x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

1. Konsep Fungsi

Di dalam matematika banyak fungsi atau formula yang biasa digunakan untuk menyelesaikan masalah matematika. Fungsi tersebut dapat digunakan secara berulang dengan nilai yang berubah-ubah. Sebagai contoh fungsi untuk mencari nilai rata-rata,

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Fungsi diatas dapat digunakan untuk mencari rata-rata dari sejumlah data (n), dengan nilai dari setiap data dapat berubah-ubah.

Fungsi matematika sebagai refleksi dari operator *Function* didalam bahasa pemrograman. Dalam penulisan program komputer terkadang dibutuhkan sebuah kumpulan kode yang akan digunakan berulang (lebih dari satu kali) dengan nilai yang berbeda-beda.

Function adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai sub-program (modul program) yang merupakan sebuah program kecil untuk memproses sebagian dari pekerjaan program utama.

Keuntungan menggunakan fungsi :

- a. Program besar dapat di pisah-pisah menjadi program-program kecil melalui *Function*.
- b. Kemudahan dalam mencari kesalahan-kesalahan karena alur logika jelas dan kesalahan dapat dilokalisasi dalam suatu modul tertentu.
- c. Memperbaiki atau memodifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa mengganggu keseluruhan program.
- d. Dapat digunakan kembali (*Reusability*) oleh program atau fungsi lain.
- e. Meminimalkan penulisan perintah yang sama.

Kategori Fungsi

- a. **Standard Library Function** adalah fungsi-fungsi yang telah disediakan oleh Interpreter Python dalam file-file atau librarynya. Misalnya: `raw_input()`, `input()`, `print()`, `open()`, `len()`, `max()`, `min()`, `abs()` dll.
- b. **Programme-Defined Function** Adalah *Function* yang dibuat oleh programmer sendiri. *Function* ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri.

Dalam python terdapat dua perintah yang dapat digunakan untuk membuat sebuah fungsi, yaitu **def** dan **lambda**. **def** adalah perintah standar dalam python untuk mendefinisikan sebuah fungsi. Tidak seperti *Function* dalam bahasa pemrograman kompiler seperti C/C++, **def** dalam python merupakan perintah yang *executable*, artinya *Function* tidak akan aktif sampai python meruning perintah **def** tersebut. Sedangkan **lambda**, dalam python lebih dikenal dengan nama *Anonymous Function* (Fungsi yang tidak disebutkan namanya). Lambda bukanlah sebuah perintah (*statemen*) namun lebih kepada ekspresi (*expression*).

2. Mendeklarasikan dan Memakai Fungsi

Sintaks dalam *Function* dibagi kedalam 3 bagian:

- a. **Header**, bentuk penulisan

def <nama_Function> (<parameter_Function>) :

contoh: `def identitas (nama):`

`nama_Function 'identitas', dan parameter Function 'nama'`

- b. **Komentar**, sebagai tempat untuk menuliskan tujuan dari *Function*. Penulisan komentar ini *opsional*, jika diperlukan.

Contoh: "Menulis identitas"

c. **Block code**, area kode yang akan dijalankan oleh *Function*.

```
print "Nama anda : ", (nama)
```

Penulisan sintaks *Function* ada dua bentuk yaitu menggunakan parameter dan tanpa parameter. *Function* tanpa parameter bersifat *fix* (tetap) tidak ada perubahan nilai pada saat memanggil *Function* tersebut.

Contoh program tanpa parameter seperti berikut ini.

```
# Fungsi tanpa parameter dan nilai

def data_mahasiswa():
    print ('Nama saya adalah Rafa')
    print ('Prodi pendidikan matematika')
    print ('Fakultas Ilmu Pendidikan')

data_mahasiswa()
```

Program diatas menggunakan *Function* dengan nama "data_mahasiswa" tanpa parameter. Maka output setelah **run** program diatas.

```
Nama saya adalah Rafa
Prodi pendidikan matematika
Fakultas Ilmu Pendidikan
```

Operator *Function* juga dapat digunakan untuk memanggil *Function* lainnya, sebagai contoh dapat dilihat pada program untuk menentukan bilangan genap dan ganjil berikut ini.

```
def cekAngka ():
    x = input ('Masukkan angka yang Anda inginkan : ')
```

```

if x%2 == 0:
    print ('Angka yang Anda Masukkan adalah GENAP')
else:
    print ('Angka yang Anda Masukkan adalah GANJIL')

def utama():
    for i in range (1,4):
        cekAngka()

utama()

```

x = int (input ('Masukkan angka yg anda inginkan : '))
 bentuk lain penulisan dari x = input ('Masukkan angka yg anda inginkan : ').

Program diatas terdiri dari dua *function* “cekAngka” dan “utama”, kedua *function* tersebut tidak menggunakan parameter. *Function* utama memanggil *function* cekAngka, sehingga output yang dihasilkan seperti tampilan berikut ini.

```

Masukkan angka yang Anda inginkan : 6727
Angka yang Anda Masukkan adalah GANJIL
Masukkan angka yang Anda inginkan : 89219730
Angka yang Anda Masukkan adalah GENAP
Masukkan angka yang Anda inginkan : 64721
Angka yang Anda Masukkan adalah GANJIL

```

3. Function dengan return

Dalam *function* dapat diiringi statemen **return** yang digunakan untuk mengembalikan suatu nilai ke badan program yang memanggil fungsi. Sebagai contoh program untuk menghitung luas persegi panjang, dengan dua *function* luaspersegi panjang dan utama. Dalam blok *function* luaspersegi panjang terdapat statemen **return** yang akan mengembalikan

nilai L (luas persegi panjang) ke badan *function* utama yang telah memanggil *function* luaspersegi panjang. Berikut struktur program untuk menghitung luas persegi panjang.

```
def luaspersegi panjang ():
    l = input ('Masukkan lebar: ')
    p = input ('Masukkan panjang: ')
    L =p*l
    return (L)

def utama():
    luas = luaspersegi panjang()
    print '\n Luas= ',luas

print ("Menghitung Luas Persegi Panjang")
utama()
```

Perhatikan output dari **running** program berikut ini.

```
Menghitung Luas Persegi Panjang
```

```
Masukkan lebar: 50
```

```
Masukkan panjang: 25
```

```
Luas= 1250
```

Apa yang terjadi apabila program di atas tidak menggunakan return pada function ??? buatlah sebuah kesimpulan!

Selanjutnya diberikan contoh program sederhana menggunakan *function* dengan parameter dan *return*.

```
def sisa_bagi_0_dan_5(n):
    i = 0
    while n > 0:
        sisa_bagi = n % 10
```

```

        if sisa_bagi == 0 or sisa_bagi == 5:
            i = i + 1
            n = n // 10
        return i

a = input('Masukkan angka : ')
hitung=sisa_bagi_0_dan_5 (a)
print hitung

```

Output program diatas dapat dilihat berikut ini.

```

Masukkan angka : 10000
4
>>>
Masukkan angka : 38
0
>>>
Masukkan angka : 45
1

```

Contoh lain menghitung luas segitiga menggunakan *function* dengan parameter.

```

#Menghitung Luas Segitiga

def luasSegitiga (a,t):
    L = (a*t)/2
    print ('Luas = (%d x %d)/2' %(a,t))
    print ('          =%.2f' %L)

def hitung():
    alas = input ('Alas = ')

```

```
tinggi = input ('Tinggi = ')
luasSegitiga (alas,tinggi)
print ("Menghitung Luas Segitiga")
hitung()
```

setelah **running** program :

```
Menghitung Luas Segitiga
Alas = 35
Tinggi = 45
Luas = (35 x 45)/2
      =787.00
```

Program tersebut menggunakan operator *function* dengan parameter (a,t). Pada saat program memanggil *function* 'hitung', *function* tersebut memerintahkan user untuk memasukkan nilai alas dan tinggi, dan selanjutnya memanggil *function* luasSegitiga dengan mengkonversi nilai alas dan tinggi yang telah dimasukkan oleh user ke nilai parameter (a,t) pada *function* segiTiga. Setelah itu perintah yang berada dalam *function* segiTiga menghitung luas ($L=(a*t)/2$) dieksekusi.

Sekarang kita akan mencoba membuat program menggunakan *function* dengan parameter dan **return**. Sebagai contoh kita akan membuat program untuk menentukan volume balok, dengan nama *function* volumeBalok yang memiliki tiga parameter panjang, lebar dan tinggi. Di dalam badan program, *function* volumeBalok dipanggil dengan panjang=10, lebar=9 dan tinggi=9. **return** V berfungsi mengembalikan nilai V (volume balok) hasil perkalian panjang, lebar dan tinggi, ke badan program yang memanggil *function* volumeBalok.

```
# definisi fungsi
def volumeBalok(panjang, lebar, tinggi):
```

```
        V = panjang*lebar*tinggi
        return V

#Deklarasi data pada variable
panjang = 10
lebar = 9
tinggi = 9

#Pemanggilan fungsi
hasil = volumeBalok (panjang, lebar, tinggi)
print hasil
```

Sehingga output program yang dihasilkan adalah 810.

TUGAS KELOMPOK 3

Penilaian:

1. Diskusi : 20%
2. Output : 50%
3. Presentasi : 30%

Buatlah program untuk menyelesaikan permasalahan matematika menggunakan operator *function*:

- a. Dengan parameter serta dengan dan tanpa return
- b. Tanpa parameter serta dengan dan tanpa return

Masing-masing satu buah program. Sehingga produk yang dihasilkan adalah 4 buah program.

4. Scope Variabel

Scope variabel atau cakupan variabel merupakan suatu keadaan dimana pendeklarasian sebuah variabel di tentukan. Dalam scope variabel dikenal dua istilah yaitu *local* dan *global*. Variabel disebut *local* ketika variabel tersebut didefinisikan didalam sebuah fungsi (*def*). Artinya variabel tersebut hanya dapat di gunakan dalam cakupan fungsi tersebut saja. Dan jika sebuah variabel didefinisikan diluar fungsi maka variabel tersebut bersifat *global*. Artinya, variabel tersebut dapat digunakan oleh fungsi lain atau pun program utamanya.

```
def fungsiSatu ():
    x = 22
    print "Nilai x dalam fungsi :",x

def fungsiDua () :
    global x
    x = 33
    print "Nilai x dalam fungsi :",x

x = 44

print "Nilai x sebelum fungsi :",x
fungsiSatu ()
print "Nilai x sesudah fungsi :",x

print '\n\n'

print "Nilai x sebelum fungsi :",x
fungsiDua ()
print "Nilai x sesudah fungsi :",x
```

Program tersebut terdiri dari scope variabel lokal dan global. Pada lokal variabel ditunjukkan pada *function* fungsiSatu dengan x = 22, sehingga nilai x = 22 tidak dieksekusi dalam badan program (bagian diluar *function*) melainkan hanya didalam *function* fungsiSatu. Berbeda dengan scope variabel global pada *function* fungsiDua dengan x = 33, nilai x =

33 dieksekusi ke dalam badan program. Output program diatas adalah sebagai berikut.

```
Nilai x sebelum fungsi : 44  
Nilai x dalam fungsi : 22  
Nilai x sesudah fungsi : 44
```

```
Nilai x sebelum fungsi : 44  
Nilai x dalam fungsi : 33  
Nilai x sesudah fungsi : 33
```

Untuk lebih memahami fungsi dari scope variabel local dan global, silahkan Anda berlatih membuat program menggunakan scope variabel untuk meningkatkan pemahaman serta kreatifitas Anda dalam menyusun program.

```
# Dimana ada kemauan, pasti ada jalan...  
# Selamat Mencoba!!!
```

5. Statement Lambda

Selain statemen **def**, Python juga menyediakan suatu bentuk ekspresi yang menghasilkan objek fungsi sederhana yang mengembalikan nilai, yaitu **lambda**. Lambda dapat disebut sebagai *anonymous function* (fungsi yang tidak disebutkan namanya). **Lambda** bukanlah sebuah perintah atau statemen namun merupakan sebuah ekspresi.

Kode program:

```
lambda argument1, argument2,... argumentN :expression using arguments
```

Fungsi argumen sama dengan parameter pada **def**, argumen bisa lebih dari satu, dan diikuti oleh ekspresi setelah tanda titik dua (:).

Perbedaan **lambda** dengan **def** dapat kita lihat pada ilustrasi berikut ini:

Dalam bentuk **def**:

```
def Fungsi(): return "Assalamualaikum!"
```

Dalam bentuk **lambda**, maka penulisan akan lebih simpel.

```
lambda: "Assalamualaikum!"
```

Untuk memanggil fungsi **lambda**, tidak dapat memanggil secara langsung. Kita harus menggunakan variabel lain atau memanggil dalam bentuk fungsi. Berikut contoh penggunaan bentuk **lambda**,

```
x = lambda a,b,c: (a*b)/(b*c)
x(6,5,2) # hasilnya adalah 3
```

Atau untuk melihat perbedaan **def** dengan **lambda** dapat dilihat ilustrasi berikut ini.

```
#perbedaan fungsi def dengan fungsi lambda

#fungsi def
def f(x): return x*2
print f(10)

#fungsi lambda
g = lambda x: x*2
print g(20)
```

Output dari program diatas sebagai berikut.

```
20
40
```

6. Fungsi Rekursif

Rekursif adalah salah satu metode dalam dunia matematika, merupakan sebuah fungsi yang mengandung fungsi itu sendiri. Dalam dunia pemrograman, rekursif diimplementasikan dalam sebuah fungsi yang memanggil dirinya sendiri, sehingga terjadilah perulangan didalam fungsi tersebut. Pada proses rekursif, akan terjadi secara berulang-ulang. Oleh karena itu, perlu adanya *stopping role* atau penyetopan untuk penghentian proses perulangan tersebut.

Kelebihan

- a. Program lebih singkat.
- b. Pada beberapa kasus, lebih mudah menggunakan fungsi rekursif, contohnya: pangkat, factorial, dan fibonacci, dan beberapa proses deret lainnya.
- c. Lebih efisien dan cepat dibandingkan proses secara iteratif.

Kekurangan

- a. Memakan memori lebih besar, karena setiap bagian dari dirinya dipanggil, akan membutuhkan sejumlah ruang memori untuk penyimpanan.
- b. Rekursif sering kali tidak bisa berhenti sehingga memori akan terpakai habis dan program bisa hang.

Sebagai ilustrasi fungsi rekursif yaitu deret fibonacci. Barisan yang berawal dari 0 dan 1, kemudian angka berikutnya didapat dengan cara menambahkan kedua bilangan yang berurutan sebelumnya. Dengan aturan ini, maka barisan bilangan Fibonacci yang pertama adalah: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ... dan seterusnya.

```
# fungsi rekursif pada deret fibonacci

def fib (n):
    if n<=2:
        return 1
    else:
        return (fib(n-1)+fib(n-2))

print "Fungsi untuk menampilkan Deret Fibonacci\
sebanyak x"

n = input ('Masukkan nilai X: ')
for i in range (1,n):
    print (fib (i))
```

Output **running** program diatas.

```
Fungsi untuk menampilkan Deret Fibonacci sebanyak x
Masukkan nilai X: 12
1
1
2
3
5
8
13
21
34
55
89
```

7. Keyword Argumen pada Function

Perhatikan ilustrasi yang pernah diberikan sebelumnya.

```

#Menghitung Luas Segitiga

def luasSegitiga (a,t):
    L = (a*t)/2
    print ('Luas = (%d x %d)/2' %(a,t))
    print ('          =%.2f' %L)

def hitung():
    alas = input ('Alas = ')
    tinggi = input ('Tinggi = ')
    luasSegitiga (alas,tinggi)

hitung()

```

Pada program diatas, saat akan memanggil *function* luasSegitiga dengan parameter, nilai parameter yang dituliskan harus sesuai urutan parameter pada *function*. Seperti untuk memanggil *function* luasSegitiga **luasSegitiga (alas,tinggi)**, **alas** yang dituliskan di urutan pertama akan dikonversi ke parameter **a** yang berada di urutan pertama pada *function* luasSegitiga, dan **tinggi** yang dituliskan di urutan kedua dikonversi ke parameter **t** yang berada pada urutan kedua pula. Hal ini mungkin cukup mudah apabila jumlah parameter sedikit, Namun, bagaimana jika parameter yang dibutuhkan banyak dan berulang, hal tersebut mungkin akan sangat menyulitkan, dikarenakan kita harus mengingat posisi masing-masing parameter.

Oleh karena itu, solusi untuk mengatasi masalah tersebut, di dalam python pada saat memanggil *function* gunakan statemen kata kunci (*keyword*) untuk menuliskan parameter di badan *function* yang dipanggil, sehingga penulisan parameter tidak perlu menyesuaikan urutan. Sebagai contoh program dengan statemen yang menggunakan kata kunci pada saat memanggil *function*.

```
def hitung():
    alas = input ('Alas = ')
    tinggi = input ('Tinggi = ')
    luasSegitiga (t=tinggi,a=alas)
hitung()
```

Meski urutannya berbeda, dikarenakan dituliskan kata kunci dari parameter maka komputer akan memahaminya

Sebagai contoh program lain untuk melihat perbedaan statemen yang menggunakan *keyword* dengan statemen yang tanpa *keyword*.

```
def data_siswa (nama, kelas, jurusan) :
    print "nama      : ", nama
    print "kelas     : ", kelas
    print "jurusan   : ", jurusan

print "Menggunakan keyword parameter : "
data_siswa (kelas="XII", jurusan="IPA", nama="Zahwa")

print "\nTanpa menggunakan keyword parameter, \
urutan acak : "
data_siswa ("XII", "IPA", "Zahwa")

print "\nTanpa menggunakan keyword parameter, \
urutan sesuai : "
data_siswa ("Zahwa", "XII", "IPA")
```

Output yang dihasilkan setelah **running** program :

```
Menggunakan keyword parameter :
nama      :   Zahwa
kelas     :   XII
jurusan   :   IPA

Tanpa menggunakan keyword parameter, urutan acak :
```

```
nama      : XII
kelas     : IPA
jurusan   : Zahwa
```

```
Tanpa menggunakan keyword parameter, urutan sesuai :
nama      : Zahwa
kelas     : XII
jurusan   : IPA
```

8. *Keyword-length Argument* pada *Function*

Di dalam *function*, *keyword* pada parameter (argumen) dapat memiliki jumlah yang banyak dan nilai yang berubah-ubah. Agar nilai-nilai pada *keyword* berbentuk tuple, maka penulisan parameter harus diawali dengan lambang satu bintang (*). Sedangkan untuk nilai-nilai yang berbentuk *dictionary*, maka penulisan parameter harus diawali dengan lambang dua bintang (**). Sebagai contoh perhatikan berikut ini:

```
# lambang * dikenali sebagai tuple
# lambang ** dikenali sebagai dictionary
def biodata (nama, usia, *hobi, **lain2):
    print "Nama : ",nama
    print "Usia : ",usia
    print "Hobi : ",hobi
    print "Lain-lain : ",lain2

biodata ("Rafa", "7", "Membaca", "Futsal",
        "Berenang", Jenis_kelamin="Laki-Laki",
        Status="Belum Menikah")
```

Output yang dihasilkan setelah **running** program sebagai berikut :

```
Nama : Rafa
Usia : 7
Hobi : ('Membaca', 'Futsal', 'Berenang')
Lain-lain : {'Jenis_kelamin': 'Laki-Laki', 'Status':
'Belum Menikah'}
```

B. Kesimpulan:

1. *Function* dalam python memiliki dua jenis kode **def** dan **lambda**.
Kode penulisan untuk **def** dan **lambda**.
`def <nama_fungsi> (parameter)`
`lambda argument1, argument2,... argumentN: expression using arguments`
2. Di dalam *function* dikenal kode **return** yang berfungsi untuk mengembalikan nilai ke dalam badan program yang memanggil *function*.
3. Parameter pada function dapat didefinisikan ataupun tidak didefinisikan. Penulisan parameter untuk memanggil function jika sesuai urutan tidak menggunakan keyword, sebaliknya apabila dituliskan tidak sesuai dengan urutan maka harus menggunakan keyword.
4. Salah satu penerapan function adalah untuk menyelesaikan fungsi rekursi.
5. **Scope variabel** atau cakupan variabel merupakan suatu keadaan dimana pendeklarasian sebuah variabel di tentukan. Dalam scope variabel dikenal dua istilah yaitu *local* dan *global*.
6. Lambang yang digunakan untuk mendefinisikan argumen pada parameter berbentuk tuple dan dictionary adalah satu bintang (*) dan dua bintang (**)

Tugas mandiri 5

Penilaian :

1. Proses pengerjaan 25%
2. Kreativitas 55%
3. Output 20%

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan *function*.

BAB 7. Exception

Pokok Bahasan :

Membuat exception

Deskripsi Materi :

Pada materi ini akan dibahas mengenai pengertian exception, jenis-jenis exception, cara menyusun multiple except, penggunaan multiple exception, try-except bersarang, membuat exception sendiri, menggunakan finally pada try-except, cara menyelesaikan integral garis.

Sub CPMK :

- 11.1 Memahami Jenis – Jenis Exception
- 11.2 Memahami Menyusun Multiple Exception
- 11.3 Memahami penggunaan Multiple Exception
- 11.4 Memahami Try-Except Bersarang
- 11.5 Memahami membuat Exception Sendiri
- 11.6 Menggunakan “finally” pada Try-Except
- 11.7 Memahami cara menyusun program penyelesaian permasalahan matematika dengan exception

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Exception merupakan sebuah kesalahan (*error*) yang terkadang terjadi dalam penulisan program. Dalam programming sering kali ditemukan kesalahan (*Error*). Kesalahan tersebut dibagi menjadi 3 jenis:

1. Kesalahan sintaks

Kesalahan ini disebabkan oleh cara penulisan program yang tidak mengikuti kaidah yang benar sesuai bahasa pemrograman yang digunakan. Sebagai contoh:

Kode salah:

```
while 1 print "python"
```

Seharusnya ditulis seperti berikut diakhir **while** menggunakan titik dua (:)

Kode benar:

```
while 1:  
    print "python"
```

Contoh lainnya.

kode salah:

```
Print 12345
```

Seharusnya ditulis print dengan 'p' menggunakan huruf kecil, dikarenakan python sensitif terhadap penulisan huruf kapital dan huruf kecil. Sintaks python dituliskan menggunakan huruf kecil semua.

Kode benar:

```
print 12345
```

2. Kesalahan pada saat eksekusi

Kesalahan ini menyebabkan eksekusi segera dihentikan. Contoh kesalahan ini diantaranya pemasukan data yang salah (bukan bilangan) pada fungsi seperti `input()`, atau contoh lainnya adalah pemberian indeks diluar jangkauan indeks list, atau membuka berkas yang sebenarnya tidak ada dalam disk.

Contoh pada saat `IndexError` terjadi :

```
try :  
  
    z = (1,2,3,4,5,6,7,8,9)  
  
    s = input ("Masukkan nilai indeks yang  
diinginkan: ")  
  
    x = z[s]  
  
except :  
  
    print "Indeks tidak ada"  
  
else :  
  
    print "Nilai indeks : ",x
```

Kemungkinan output yang dihasilkan bergantung pada nilai `s` (indeks) yang diinput oleh user. Apabila user menginput nilai `s` mulai dari 0 sampai dengan 8 maka akan tercetak nilai item yang berada di dalam list `z` sesuai dengan indeks yang diinput oleh user. Tetapi apabila user menginput indeks di luar 0-8 maka akan tercetak **Indeks tidak ada**. Seperti tampilan output berikut ini hasil **running** program diatas.

```
Masukkan nilai indeks yang diinginkan: 5
```

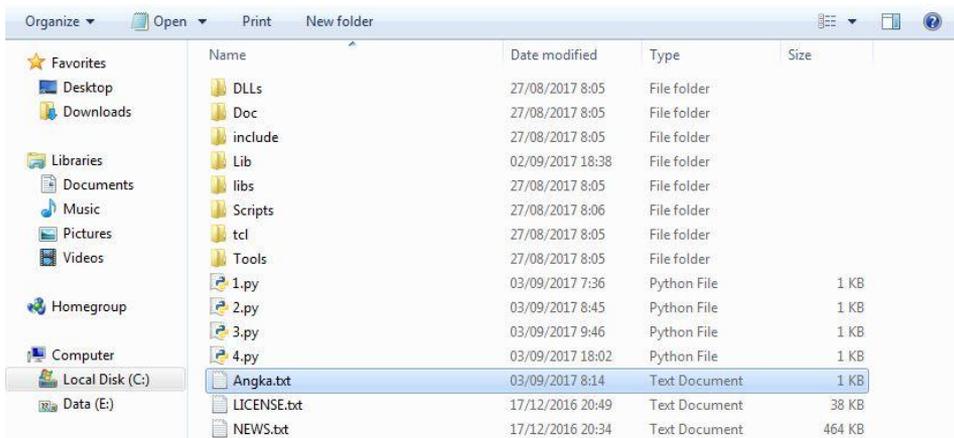
```
Nilai indeks : 6
```

```
Masukkan nilai indeks yang diinginkan: 10
```

```
Indeks tidak ada
```

Contoh pada saat FileNotFoundError terjadi :

Nama file Awal:



Contoh program:

```
try :  
    f = open ('Angka.txt')  
    print (f.read())  
    f.close()  
except:  
    print ('Nama File tidak tersedia, cek kembali.')
```

Program diatas ingin membaca file **Angka.txt**. Apabila file ditemukan maka Output yang dihasilkan seperti berikut ini:

```
1  
2  
3
```

```
4
```

```
5
```

Tetapi apabila nama file tidak sesuai dengan yang dituliskan dalam program, misal nama file sebenarnya **Angkat.txt**, maka output program setelah **run** seperti berikut ini.

```
Nama File tidak tersedia, cek kembali.
```

3. Kesalahan logika

Kesalahan ini terdeteksi sewaktu script dieksekusi tetapi tidak menghentikan eksekusi script. Kesalahan seperti ini umumnya karena logika dalam script yang salah. meskipun menghasilkan program dapat dijalankan dan menghasilkan output, tetapi sesungguhnya output yang dihasilkan tidak benar atau tidak sesuai dengan yang sebenarnya. Misal programmer yang seharusnya menuliskan bilangan phi 3.14 tetapi pada kenyataannya menuliskan 31.4 (*human error*), sehingga script yang seharusnya dapat menghitung luas lingkaran dengan benar menjadi salah.

Untuk melihat aneka eksepsi di python kita bisa menuliskan ini pada modus interaktif:

Kode:

```
dir(__builtins__)
```

Output yang dihasilkan :

```
['ArithmeticError', 'AssertionError',  
'AttributeError', 'BaseException', 'BufferError',  
'BytesWarning', 'DeprecationWarning', 'EOFError',  
'Ellipsis', 'EnvironmentError', 'Exception', 'False',
```

```
'FloatingPointError',          'FutureWarning',
'GeneratorExit',              'IOError',          'ImportError',
'ImportWarning',             'IndentationError', 'IndexError',
'KeyError',                  'KeyboardInterrupt', 'LookupError',
'MemoryError',               'NameError',        'None',          'NotImplemented',
'NotImplementedError',       'OSError',          'OverflowError',
'PendingDeprecationWarning', 'ReferenceError',
'RuntimeError',              'RuntimeWarning',   'StandardError',
'StopIteration',            'SyntaxError',      'SyntaxWarning',
'SystemError',               'SystemExit',       'TabError',     'True',
'TypeError',                  'UnboundLocalError',
'UnicodeDecodeError',        'UnicodeEncodeError',
'UnicodeError',              'UnicodeTranslateError',
'UnicodeWarning',           'UserWarning',     'ValueError',
'Warning',                   'ZeroDivisionError', '__debug__',
'__doc__',                   '__import__',       '__name__',     '__package__',
'abs',                        'all',              'any',           'apply',         'basestring',     'bin',
'bool',                       'buffer',           'bytearray',    'bytes',         'callable',
'chr',                         'classmethod',      'cmp',           'coerce',        'compile',
'complex',                    'copyright',        'credits',       'delattr',       'dict',
'dir',                         'divmod',           'enumerate',     'eval',          'execfile',
'exit',                       'file',             'filter',        'float',         'format',
'frozenset',                  'getattr',          'globals',       'hasattr',       'hash',
'help',                       'hex',              'id',            'input',         'int',            'intern',
'isinstance',                 'issubclass',      'iter',          'len',           'license',
'list',                       'locals',           'long',          'map',           'max',           'min',           'next',
'object',                     'oct',              'open',          'ord',           'pow',           'print',
'property',                   'quit',             'range',         'raw_input',     'reduce',
'reload',                     'repr',             'reversed',      'round',         'set',
'setattr',                    'slice',            'sorted',        'staticmethod',  'str',
'sum',                        'super',            'tuple',         'type',          'unichr',        'unicode',
'vars',                       'xrange',          'zip']
```

Macam-macam nama eksepsi ditandai dengan kata Error

Contoh pada saat IOError terjadi

Kode:

```
>>> f = open("tes exception.txt")

Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    f = open("tes exception.txt")
IOError: [Errno 2] No such file or directory: 'tes
exception.txt'
>>>
```

Pada contoh tersebut script tidak bisa membaca file dengan nama NULL.txt karena di dalam disk tidak ada file dengan nama seperti itu yang sudah disiapkan.

Contoh pada saat KeyboardInterrupt terjadi

Kode:

```
>>> while True:
    pass

Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    while True:
KeyboardInterrupt
>>>
```

Pada contoh tersebut memperlihatkan eksepsi KeyboardInterrupt (saya menekan tombol ctrl+C pada keyboard ketika program sedang proses loop)

Contoh pada saat KeyError terjadi

Kode:

```
>>> kamus = {1:"satu", 2:"dua", 3:"tiga"}
>>> kamus[4]

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    kamus[4]
KeyError: 4
>>>
```

Penyebab kesalahan diatas adalah key 4 yang tidak ada di dalam variabel dictionary.

Contoh pada saat NameError terjadi

Kode:

```
>>> print abc

Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print abc
NameError: name 'abc' is not defined
>>>
```

Error diatas memberitahukan bahwa variabel abc belum didefinisikan sebelumnya.

Contoh pada saat TypeError terjadi

Kode:

```
>>> t = (1,2,3)
>>> t[1] = 5
```

```
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    t[1] = 5
TypeError: 'tuple' object does not support item
assignment
>>>
```

Kesalahan diatas disebabkan adanya perubahan elemen pada *tuple*, kita tahu bahwa *tuple* merupakan struktur data yang *immutable*. Contoh pada saat ValueError terjadi.

Kode:

```
>>> range(1,10,0)

Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    range(1,10,0)
ValueError: range() step argument must not be zero
>>>
```

Kesalahan disebabkan argumen ketiga pada range(), dimana tidak boleh bernilai nol, contoh lain:

Kode:

```
>>> int("a")

Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    int("a")
ValueError: invalid literal for int() with base 10:
'a'
```

Tipe data integer seharusnya berupa bilangan, bukan huruf ("a").

Contoh program yang akan muncul ZeroDivisionError.

```
x = input ('masukkan sebuah angka: ')

try:

    x/0

except ZeroDivisionError, e:

    print 'proses perhitungan gagal karena :',e

print x/0
```

Apabila program dijalankan, maka output yang akan dihasilkan seperti berikut ini.

```
masukkan sebuah angka: 5
proses perhitungan gagal karena : integer division or
modulo by zero

Traceback (most recent call last):
  File "D:/Latihan Python/latihan raw input 1.py",
line 6, in <module>
    print x/0
ZeroDivisionError: integer division or modulo by zero
```

Terjadi kesalahan pada pembagian bilangan dengan 0.

Contoh lainnya ZeroDivisionError

```
def pembagianNol (x):

    try:

        y = 10
```

```

        z = y/x

        print (z)

    except ZeroDivisionError, e:

        print "Error,  bilangan  tidak  bisa
dibagi Nol"
pembagianNol (0)

```

Output setelah **running** program

```
Error, bilangan tidak bisa dibagi Nol
```

7. Menyusun Multiple Except

Program python dapat menyusun *multiple except*, yaitu menangkap *except* dengan baris *except* yang berbeda. Hal ini dilakukan apabila kita ingin memberikan perlakuan yang berbeda ketika terjadi kesalahan yang berbeda. Untuk lebih jelasnya perhatikan contoh program berikut ini.

```

# contoh multiple exception

try :

    x = input ("Masukkan angka :")

    print ("Akar dari ",x,"=",x**(0.5))

except ValueError,e:

    print "Akar dari bilangan tersebut\
gagal karena :",e

except NameError,e:

```

```
print "Akar dari bilangan tersebut\  
gagal karena :",e
```

Output yang mungkin dihasilkan apabila user salah memasukkan bilangan negatif atau string seperti tampilan berikut ini.

```
Masukkan angka :-4
```

```
Akar dari bilangan tersebut gagal karena : negative  
number cannot be raised to a fractional power
```

```
Masukkan angka :s
```

```
Akar dari bilangan tersebut gagal karena : name 's' is  
not defined
```

8. Menggunakan Multiple Exception

Sebelumnya penulisan pada *multiple exception* dapat menggunakan *except* di baris yang berbeda. Saat ini *multiple exception* yang akan dituliskan menggunakan satu baris. Sebagai contoh

```
# contoh multiple exception  
  
try :  
  
    x = input ("Masukkan angka :")  
    print ("Akar dari ",x,"=",x**(0.5))  
  
except (ValueError,NameError,TypeError), e:  
    print "Akar dari bilangan tersebut gagal  
karena :",e
```

Output yang mungkin dihasilkan ketika user salah memasukkan angka yang diperintahkan, dengan tampilan output sebagai berikut.

```
Masukkan angka :-9
```

```
Akar dari bilangan tersebut gagal karena : negative  
number cannot be raised to a fractional power
```

```
Masukkan angka :r
```

```
Akar dari bilangan tersebut gagal karena : name 'r' is  
not defined
```

```
Masukkan angka :1,2
```

```
Akar dari bilangan tersebut gagal karena : unsupported  
operand type(s) for ** or pow(): 'tuple' and 'float'
```

Kode program diatas dibuat untuk mencari nilai akar dari sebuah bilangan yang diinput oleh user melalui keyboard. Dalam contoh output diatas user menginput bilangan -9 hasilnya gagal dengan keterangan `negative number cannot be raised to a fractional power`, yang artinya bilangan negatif tidak dapat diakarkan. Selanjutnya user menginput `r`, dan hasilnya juga gagal dengan keterangan `name 'r' is not defined`, dikarenakan `'r'` adalah string sehingga tidak terdefinisi apabila dilakukan operasi matematika (akar) dari string. Selain itu ketika user menginput `1,2` hasilnya gagal dengan keterangan `unsupported operand type...`, dikarenakan nilai yang diinput oleh user adalah tipe data tuple atau float (desimal), maka python tidak dapat memprosesnya (operasi tidak didukung).

Selain kode program yang telah dituliskan sebelumnya. Kode **try-except** dapat diterapkan dengan cara bersarang, yaitu terdapat **try** didalam badan **try**. Seperti pada contoh program berikut.

```
try :  
    x = input ("Masukkan angka :")  
  
    try:  
        print ("Akar dari ",x,"=",x**(0.5))  
    except ValueError,e:  
        print "Akar dari bilangan tersebut\  
gagal karena :",e  
except NameError,e:  
    print "Akar dari bilangan tersebut\  
gagal karena :",e
```

Output yang dihasilkan sama dengan *multiple except* tetapi proses untuk mengeksekusi untuk tipe kesalahan **NameError** berada di **try** utama, sedangkan **ValueError** berada pada badan **try** bersarang.

Output setelah **running** program.

```
Masukkan angka :-9  
Akar dari bilangan tersebut gagal karena :\ negative  
number cannot be raised to a fractional power
```

```
Masukkan angka :a  
Akar dari bilangan tersebut gagal karena : name 'a' is  
not defined
```

9. Penanganan Eksepsi

Program yang baik seharusnya dapat menangani eksepsi. sebagai contoh seperti ini:

Pada program anda diminta untuk memasukkan bilangan dari keyboard, anda tidak sengaja memasukkan 27o yang seharusnya 270, dan ketika kesalahan tersebut dieksekusi program langsung dihentikan. "Masa sih salah seperti itu saja program dihentikan!" mungkin terbayang jika anda harus menginput ulang sebuah data yang banyak dalam mode text base, dan penanganan eksepsi tidak diberikan, program anda mungkin akan segera dihapus dan diganti dengan yang lain.

Program yang baik biasanya memberikan dialog eksepsi, contoh:

```
-----dialog program-----  
masukkan bilangan: 27o  
input yang anda berikan bukan bilangan, silahkan input ulang.  
masukkan bilangan: _  
-----
```

artinya user diberi kesempatan memberikan input yang valid ketika kesalahan pemberian input terjadi.

Format penanganan eksepsi dilakukan menggunakan pernyataan try...except. Format sederhana nya adalah sebagai berikut:

Kode:

```
try:  
    <pernyataan_pernyataan>  
except [<nama_eksepsi>]:
```

```
<pernyataan_pernyataan_yang_dieksekusi_jika_ada_eksepsi>
i>
[else:
    <pernyataan_pernyataan_yang_dieksekusi_jika_tidak_ada_eksepsi>]
```

tanda di dalam [] bisa ada, bisa tidak. contoh eksepsi dalam modus interaktif:

Kode program:

```
>>> def bagi(x,y):
    try:
        print x/y
    except:
        print "terjadi pembagian dengan bilangan nol"
    else:
        print "pembagian berhasil dilakukan"

>>> bagi (18,3)
6
pembagian berhasil dilakukan
>>> bagi (23,0)
terjadi pembagian dengan bilangan nol
>>>
```

B. Kesimpulan:

1. *Exception* merupakan sebuah kesalahan (*error*) yang terkadang terjadi dalam penulisan program. Dalam programming sering kali ditemukan kesalahan (*Error*).
2. Kesalahan dibagi menjadi 3 jenis: Kesalahan sintaks, kesalahan pada saat eksekusi dan kesalahan logika
3. Kode penulisan menggunakan **try-except**.
4. Eksekusi *except* dapat di tuliskan *multiple exception* melalui satu baris, atau beda baris
5. Kode penulisan juga dapat menggunakan **try-except** bersarang, yang artinya terdapat **try-except** didalam **try-except**.

Tugas mandiri 6**Penilaian :**

1. Proses pengerjaan 25%
2. Kreativitas 55%
3. Output 20%

Berikut ini program yang belum sempurna, ada beberapa kesalahan dalam program sehingga tidak dapat running program. Tugas Anda:

1. Tunjukkan letak kesalahan kode pada program dan tuliskan kode yang benar
2. Tuliskan output yang dihasilkan program setelah program di perbaiki, tanpa menggunakan komputer.

```
#program membuat menu restoran

menu = [paket 1, paket 2, paket 3, paket 4]
paket 1 = ['nasi', 'ayam bakar', 'kangkung', 'lalap',
           'tempe tahu', 'es jeruk']
paket 2 = ['nasi', 'ayam goreng', 'sayur asem', 'lalap',
           'tempe tahu', 'es teh manis']
paket 3 = ['nasi', 'bebek bakar', 'keredok', 'tahu isi',
           'es jeruk']
paket 4 = ['nasi', 'bebek goreng', 'tumis toge', 'tempe
medoan', 'es teh manis']
print "SELAMAT DATANG DI RESTORAN HARGA KAKI LIMA RASA
BINTANG LIMA"

pilihan=input('\n Silahkan pilih paket makanan sesuai
dengan selera Anda(1/2/3/4): ')

if pilihan == 1:
    print ('Anda telah memilih paket 1 :',
paket_1)
elif pilihan == 2:
    print ('Anda telah memilih paket 1 :',
paket_2)
elif pilihan == 3:
    print ('Anda telah memilih paket 1 :',
paket_3)
elif pilihan == 4:
```

```
        print ('Anda telah memilih paket 1 :',
paket_4)
    else
        print 'Anda tidak memilih paket, Silahkan
melakukan order makanan yang lain'
        x = input('tuliskan makanan :')
        y = input ('tuliskan minuman :')
        z = input ('tuliskan tambahan :')
        print 'Anda order makanan',y
        print 'Anda order minuman',z
        print 'Anda order tambahan',x

print '\n Mohon tunggu, pesanan akan segera diantar ke
meja Anda'
```


BAB 8. File

Pokok Bahasan :

Membuat file

Deskripsi Materi :

File pada python berfungsi menyimpan data-data yang dapat digunakan atau diubah isinya di lain waktu. **Python** memberikan kemudahan dalam bekerja yaitu dengan menggunakan file. Materi yang akan dibahas yaitu pengenalan file dan membuat file baru, mengisi file dan membaca isi file, memahami isi file dengan cara baris per baris dan posisi pointer file, serta mengganti nama file dan menghapus file.

Sub CPMK :

- 12.1 Memahami pengenalan file dan membuat file baru
- 12.2 Memahami mengisi file dan membaca isi file
- 12.3 Memahami isi file dengan cara baris per baris
- 12.4 Memahami posisi pointer file
- 12.5 Memahami cara mengganti nama file dan menghapus file

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Program python mampu membaca dan menuliskan dalam file teks dengan ekstensi .txt, yaitu format text file yang dapat dibuka menggunakan software notepad. Selain .txt python juga dapat membaca dan menulis dengan ekstensi .csv, .md, .json, dan sebagainya.

Kode untuk membaca dan menulis file dalam python adalah sebagai berikut:

```
<nama_variabel> = open ("<nama_file>",mode)
```

Beberapa mode yang dapat digunakan python untuk membaca dan menulis dalam file, yaitu :

MODE AKSES	DESKRIPSI
r	Membuka file untuk mode baca
w	Membuka file untuk mode tulis
a	Membuka file untuk mode tambah
rb	Membuka file untuk mode baca dalam format binari
wb	Membuka file untuk mode tulis dalam format binari
ab	Membuka file untuk mode tambah dalam format binari
r+	Membuka file untuk mode baca dan tulis
w+	Membuka file untuk mode tulis dan baca
a+	Membuka file untuk mode tambah dan baca
rb+	Membuka file untuk mode baca dan tulis dalam format binari
wb+	Membuka file untuk mode tulis dan baca dalam format binari
ab+	Membuka file untuk mode tambah dan baca dalam format binari

File teks akan dapat langsung dibaca apabila file berada dalam satu direktori bersama script python yang akan membaca file teks tersebut. Tetapi apabila file teks tidak berada dalam satu direktori dengan script python yang akan membaca file teks, maka harusnya dituliskan path file teks nya.

1. Cara Membaca File dalam Python

Sebagai contoh kita akan membuat sebuah file dengan format text file (.txt) di notepad dan di save dengan nama file latihan1.txt, seperti berikut ini:

```
1 Andi 2016830001
2 Andri 2016830002
3 Anwar 2016830003
4 Bella 2016830004
5 Betty 2016830005
6 Budi 2016830006
7 Chika 2016830007
```

Kemudian kita akan buat program dalam script sebagai berikut:

Script 1

```
data = open('latihan1.txt','r')

print data.readlines()

data.close()
```

`data = open('latihan1.txt','r')` membuka file dari file teks latihan1.txt dan disimpan dalam variabel data. Mode 'r' pada program diatas, apabila tidak dituliskan mode maka akan default 'r' yang artinya file hanya untuk dibaca.

`print data.readlines()` membaca isi file yang telah disimpan dalam variabel data dalam satu baris, dan mengembalikan nilai dalam list.

`data.close()` untuk menghapus data dalam memori, sehingga tidak tersimpan didalam memori yang akan mengakibatkan memori penuh.

Output yang dihasilkan setelah **running** program,

```
['1 Andi 2016830001\n', '2 Andri 2016830002\n',  
'3 Anwar 2016830003\n', '4 Bella 2016830004\n',  
'5 Betty 2016830005\n', '6 Budi 2016830006\n', '  
7 Chika 2016830007']
```

Script 2

```
data = open('latihan1.txt','r')  
  
print data.read()  
  
data.close()
```

`print data.read()` membaca seluruh teks, dan mengembalikan nilai dalam string. Hal inilah yang membedakan dengan metode `readlines()`, dan output yang dihasilkan sebagai berikut.

```
1 Andi 2016830001  
2 Andri 2016830002  
3 Anwar 2016830003  
4 Bella 2016830004  
5 Betty 2016830005  
6 Budi 2016830006  
7 Chika 2016830007
```

Dalam metode `readlines ()` juga dapat menampilkan output seperti metode `read ()` menggunakan looping (**for**), dengan script program berikut ini.

```
data = open('latihan1.txt','r')
data1 = data.readlines()
for x in data1:
    print x
data.close()
```

output yang dihasilkan sebagai berikut:

```
1 Andi 2016830001
2 Andri 2016830002
3 Anwar 2016830003
4 Bella 2016830004
5 Betty 2016830005
6 Budi 2016830006
7 Chika 2016830007
```

Metode `read()` dan `readlines()` hanya dapat digunakan untuk satu kali eksekusi. Eksekusi selanjutnya akan dikembalikan nilai kosong. Sebagai contoh :

```
data = open('latihan1.txt','r')
print data.read()
print data.read()
print data.readlines()
data.close()
```

Output yang dihasilkan setelah **running** program,

```
1 Andi 2016830001
2 Andri 2016830002
3 Anwar 2016830003
4 Bella 2016830004
5 Betty 2016830005
6 Budi 2016830006
7 Chika 2016830007

[]
```

2. Cara Menulis Data File di Python

Mode yang digunakan untuk menulis file dalam program python yaitu: **"w"**, **"a"**, dan **"r+"**. Akan di ilustrasikan ketiga mode ini:

Script 1, dengan mode 'w'

```
# Input data dalam python
no = input("No Urut Absen: ")
nama = raw_input("Nama: ")
nim = raw_input("NIM: ")

# format teks pada file yang ditulis
data = "No: {}\nNama: {}\nNIM: {}".format(no, nama,
nim)

# buka file untuk ditulis
data1 = open("latihan2.txt", "w")

# tulis teks ke file
data1.write(data)
```

```
# tutup file
data1.close()
```

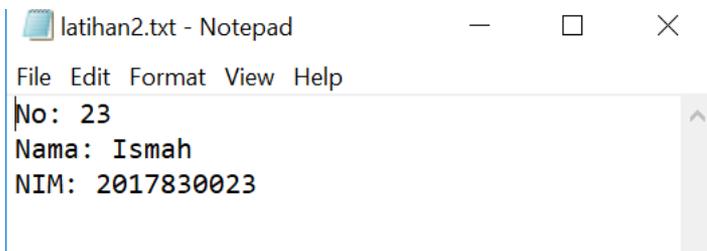
Perhatikan sintaks program tersebut, yaitu kode penulisan `data = "No: {}\nNama: {}\nNIM: {}".format(no, nama, nim)`. Tulisan yang di bold adalah kode penulisan untuk mencetak sebuah variabel string. `.format` akan memasukkan parameter `no`, `nama` dan `nim` kedalam `{}` yang berada pada `"No: {}\nNama: {}\nNIM: {}"`. Sehingga output yang dihasilkan setelah **running** program:

```
No Urut Absen : 23

Nama : Ismah

NIM : 2017830023
```

Sedangkan tampilan pada file latihan2.txt dari data yang telah diinput.



```
latihan2.txt - Notepad
File Edit Format View Help
No: 23
Nama: Ismah
NIM: 2017830023
```

untuk membaca file kita telah mengenal kode `read ()` dan `readlines()`, untuk membaca file dalam python juga dapat menggunakan kode `write()` dan `writelines()`. Perbedaan `write()` dan `writelines()`, hasil yang tercetak apabila menggunakan `write()` semua teks sesuai dengan yang ada pada file. Sedangkan pada `Writelines()` hasil yang tercetak teks dalam satu baris.

3. Menyisipkan Data ke File

Telah kita ketahui bahwa mode “w” memiliki fungsi untuk memasukkan data kedalam file, dan apabila dalam file sudah terisi data, maka isi file yang ada akan terhapus dan berganti dengan data yang baru.

Apabila kita ingin memasukkan tulisan kedalam sebuah file tanpa menghilangkan isi dari file sebelumnya, atau dengan kata lain ingin menyisipkan data ke dalam file, maka dalam program gunakan mode “a” (*append*). Sebagai ilustrasi kita akan menyisipkan data ke dalam file latihan2 .txt, yang telah dibuatkan pada program sebelumnya menggunakan mode “w”.

```
# Input data dalam python
alamat = raw_input("Alamat: ")
prodi = raw_input("Prodi: ")
hp = raw_input("No Hp: ")

# format teks pada file yang ditulis
data = "Alamat: {}\nProdi: {}\nNo Hp:
{}".format(alamat, prodi, hp)

# buka file untuk ditulis
data1 = open("latihan2.txt", "a")

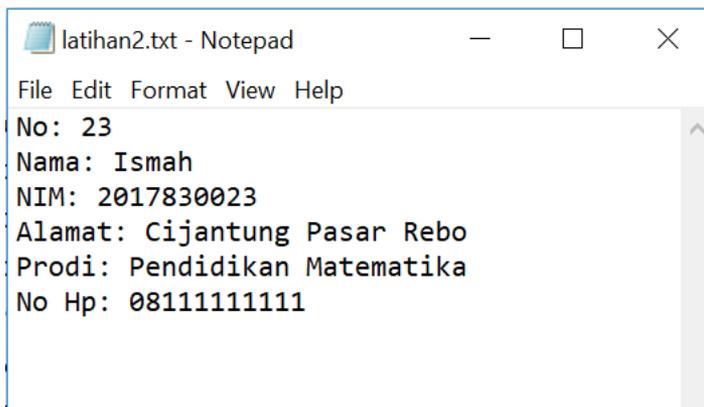
# tulis teks ke file
data1.write(data)

# tutup file
data1.close()
```

Misal kita input data sebagai berikut pada output yang dihasilkan setelah **running** program diatas.

```
Alamat: Cijantung Pasar Rebo  
Prodi: Pendidikan Matematika  
No Hp: 08111111111
```

Maka isi file pada latihan2.txt akan menjadi seperti pada tampilan gambar berikut ini.



4. Membaca dan Menulis File

Membaca dan menulis data dalam file dapat dilakukan dalam waktu bersamaan, yaitu menggunakan mode "r+". Fungsi pada mode "r+" sama dengan ketika kita menuliskan kode program mode "r" dan mode "w" sekaligus.

```
Var1 = open ("nama_file", "r")  
  
Var2 = open ("nama_file", "w")
```

Namun cara diatas akan membutuhkan memori yang banyak. Sehingga penggunaan mode "r+" dapat dijadikan sebagai solusi dari permasalahan tersebut.

Sebagai ilustrasi perhatian program berikut ini.

```
# Membaca file

data = open("latihan2.txt","r+")

data1 = data.read()

print (data1)

# Input data dalam python

data1 = "\n Selamat Anda telah Menjadi Mahasiswa Baru
Fakultas Ilmu Pendidikan UMJ"

# tulis teks

data.write(data1)

# tutup file

data.close()
```

sehingga output yang muncul sebagai berikut:

```
No: 23

Nama: Ismah

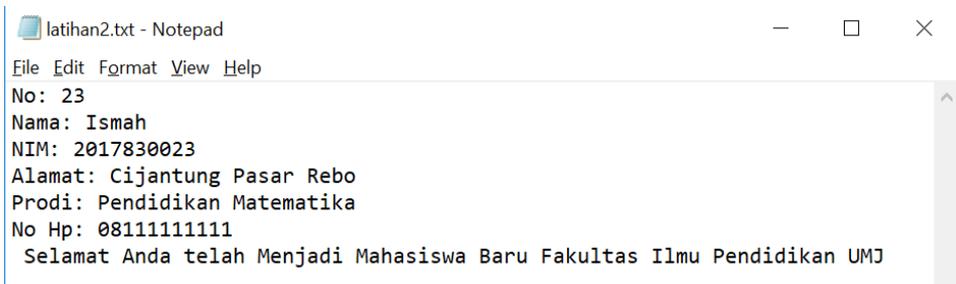
NIM: 2017830023

Alamat: Cijantung Pasar Rebo

Prodi: Pendidikan Matematika

No Hp: 081111111111
```

Sedangkan data di dalam file latihan1.txt bertambah seperti berikut ini



```
latihan2.txt - Notepad
File Edit Format View Help
No: 23
Nama: Ismah
NIM: 2017830023
Alamat: Cijantung Pasar Rebo
Prodi: Pendidikan Matematika
No Hp: 08111111111
Selamat Anda telah Menjadi Mahasiswa Baru Fakultas Ilmu Pendidikan UMI
```

Layaknya mode “a”, mode “r+” juga mampu menyisipkan data ke dalam file tanpa menindih atau menghapus data yang sudah ada.

Kode **with** dan **as**

Didalam python juga dikenal dengan kode **with** dan **as** yang dapat digunakan didalam program yang akan membaca atau menulis data ke dalam file. Apabila program-program yang telah dibahas dan diilustrasikan sebelumnya menggunakan kode close () diakhir program dengan tujuan agar tidak terjadi *memory leak* (kebocoran memori).

Kebocoran memori membuat pengurangan kapasitas memori (*free*) secara kontinu yang dijalankan oleh sistem komputer, dan dapat mengakibatkan seluruh sistem komputer menjadi *not responding*. Biasa kebocoran memori diakibatkan oleh sebuah program yang secara tidak sengaja akan menggunakan jumlah memori yang terus bertambah apabila program tersebut digunakan, sehingga jumlah RAM yang telah disediakan tersedot oleh program tersebut.

Kita dapat menggunakan kode program **with** dan **as** untuk menghindari *memory leak* selain dengan close (). Sebagai contoh :

```

# Membaca file

with open("latihan2.txt","r+") as data:

    data1 = data.read()

    print (data1)

    # Input data dalam python

    data1 = "\n Selamat Anda telah Menjadi

    Mahasiswa Baru Fakultas Ilmu Pendidikan UMJ"

    # tulis teks

    data.write(data1)

```

Setelah blok **with** dieksekusi, file akan ditutup dan otomatis dihapus dari memori.

5. Mengganti Nama File

Selain dapat membaca, menulis, menyisipkan data dalam file, python juga dapat mengganti nama file, menggunakan modul **os**.

Kode program

```

import os
os.rename('<nama_file_yang_akan_dirubah>',
'<nama_file_yang_baru>')

```

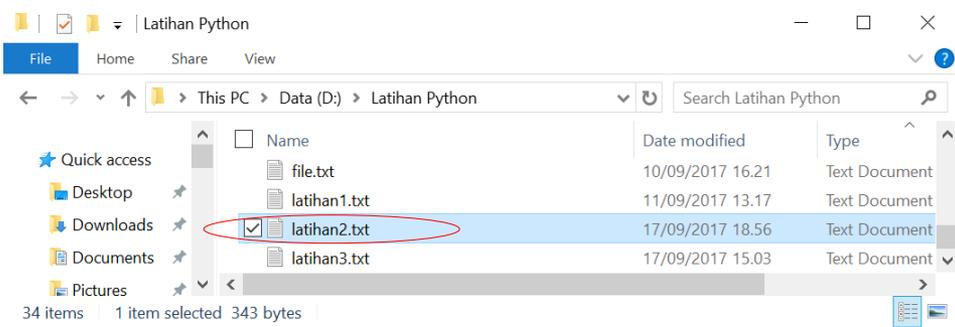
Sebagai contoh akan diubah nama file **latihan2.txt** menjadi **latihan dasar.txt**.

```

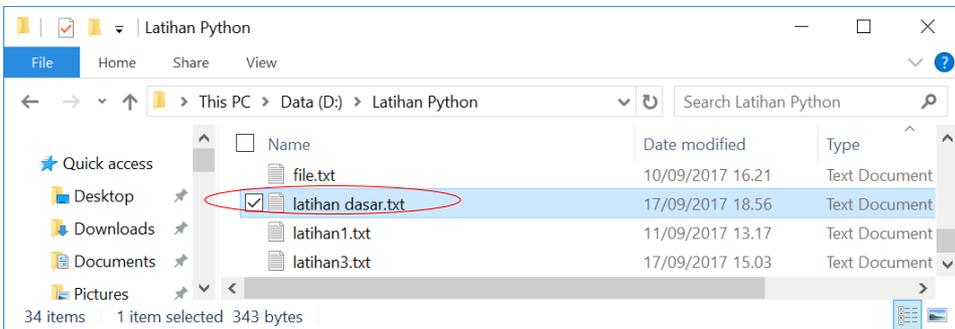
import os
os.rename('latihan2.txt', 'latihan dasar.txt')

```

Sebelum program dijalankan file bernama **latihan 2.txt**



Setelah running program, file berganti nama **latihan dasar.txt**



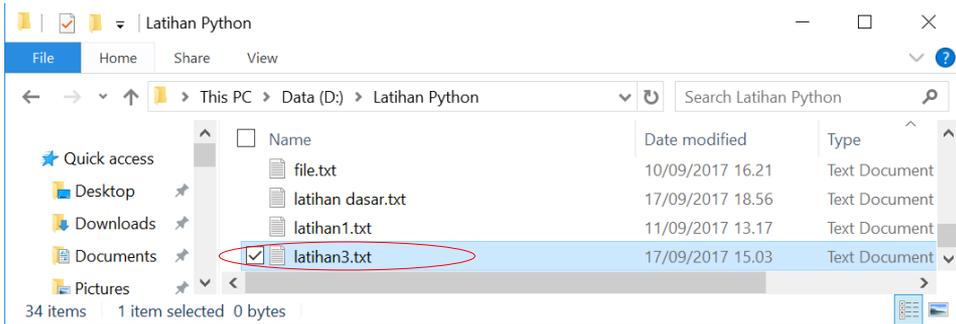
6. Menghapus File

Modul **os** selain dapat digunakan untuk mengganti nama file, juga dapat digunakan untuk menghapus file.

Kode program

```
import os
os.remove('<nama_file_yang_akan_dirubah>')
```

Sebagai contoh akan dihapus file **latihan3.txt**.



Setelah running program maka file **latihan3.txt** akan terhapus.

B. Kesimpulan

Python dapat terintegrasi dengan aplikasi lain seperti .txt (notepad). Kode program untuk membaca, menulis, mengubah dan lainnya tergantung mode. Berikut kode program:

```
<nama_variabel> = open ("<nama_file>",mode)
```

Sedangkan jenis mode dalam python sebagai berikut :

MODE AKSES	DESKRIPSI
r	Membuka file untuk mode baca
w	Membuka file untuk mode tulis
a	Membuka file untuk mode tambah
rb	Membuka file untuk mode baca dalam format binari
wb	Membuka file untuk mode tulis dalam format binari
ab	Membuka file untuk mode tambah dalam format binari
r+	Membuka file untuk mode baca dan tulis
w+	Membuka file untuk mode tulis dan baca
a+	Membuka file untuk mode tambah dan baca
rb+	Membuka file untuk mode baca dan tulis dalam format binari
wb+	Membuka file untuk mode tulis dan baca dalam format binari
ab+	Membuka file untuk mode tambah dan baca dalam format binari

Tugas mandiri 7

Penilaian :

1. Proses pengerjaan 25%
2. Kreatifitas 55%
3. Output 20%

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan *file*.

BAB 9. Penggunaan Class

Pokok Bahasan :

Membuat penggunaan class

Deskripsi Materi :

Cara membuat class dan modul serta menggunakannya. Class dan modul ini sangat lekat dengan sebuah aplikasi, biasanya pada aplikasi cukup besar akan menggunakan class dan modul menjadi sangat bermanfaat. Sebuah aplikasi seringkali melakukan suatu kegiatan yang berulang, jika program di buat secara berulang kali akan memakan memori yang cukup besar.

Sub CPMK :

- 13.1 Memahami cara membuat class dan object
- 13.2 Memahami built-i *Function* pada class dan object
- 13.3 Memahami penggunaan class dan object

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

1. Pengertian dan Fungsi Class

Sebelum kita masuk ke sesi pembuatan class, akan dijelaskan mengenai class pada bahasa pemrograman python. Sehingga pembelajaran pada pertemuan ini akan dibuat secara bertahap sehingga akan mempermudah mahasiswa untuk memahami materi ini.

Class merupakan sebuah objek yang di dalamnya biasanya terdapat beberapa metode program berupa penjabaran kongkrit dari suatu algoritma dalam bentuk sintaks python itu sendiri yang umum juga disebut OOP (*object oriented programming*). Pembuatan program yang telah dibahas sebelumnya dinamakan sebagai program yang terstruktur, sedangkan class merupakan program yang berbasis objek. Apabila program yang dibuat untuk menyelesaikan permasalahan sederhana, maka dapat menggunakan program yang terstruktur, tetapi apabila dirasa permasalahan kompleks maka program berbasis objek menjadi solusi yang tepat untuk menyelesaikan masalah tersebut.

Sebagai gambaran ada sebuah fungsi yang mana fungsi tersebut akan seringkali di gunakan pada program yang akan dibuat, penggunaan class akan lebih efisien secara memori dan kecepatan efektif, yang hanya memanggil fungsi dari class tersebut. Class mirip dengan sebuah atau kumpulan dari fungsi yang lebih besar dan kompleks yang akan menyederhanakan sebuah struktur program menjadi suatu dokumentasi-dokumentasi metode.

Class yang dapat dianalogikan dari suatu ruangan pada sekolah yakni kelas memiliki beberapa benda seperti meja, pulpen, buku, kursi, papan tulis, spidol dan lain-lainnya merupakan suatu benda-benda yang memiliki fungsi sesuai dengan prosedur peruntukannya. Sebagai ilustrasi “seorang guru dalam setiap aktifitas akan menggunakan spidol dan

papan tulis sebagai alat untuk menerangkan suatu materi yang diajarkannya, setelah selesai menerangkan bisa jadi kegiatan menerangkan dengan tertulis yang digambarkan dengan aktifitas di papan tulis dilanjutkan dengan sebuah kegiatan siswa yang menyalin penjelasan guru dengan menulis kembali apa yang ada dipapan tulis serta kegiatan pertanyaan siswa yang duduk diatas bangku lalu dilanjutkan dengan penjelasan guru menggunakan spidol dan menulisnya dipapan tulis”.

Dari ilustrasi diatas seorang guru melakukan kegiatan menulis dengan spidol di papan tulis secara berulang walau aktifitas penggunaan sedikit berbeda, yakni kegiatan pertama menuliskan materi dalam peruntukan menjelaskan suatu materi dan kegiatan kedua menuliskan penjelasan dalam menjawab pertanyaan siswa. Kegiatan berulang tersebut dapat dilakukan dengan menggunakan fungsi/prosedur atau yang sebelumnya dijelaskan diatas sebagai *function* dari program, kemudian kumpulan dari *function-function* menjadi sebuah kegiatan belajar mengajar siswa didalam 1 kelas yang setiap hari dilakukan secara berulang dengan materi yang berbeda-beda tersebut dapat dikatakan sebagai class, akan tetapi materi yang berbeda-beda setiap harinya dapat dikatakan sebagai data.

Selanjutnya agar lebih memahami penggunaan class akan dijelaskan dalam membuat class pada python.

2. Built-i *Function* pada class dan object

Dari gambaran penjelasan class diatas akan diawali dengan membuat sebuah struktur penulisan yaitu “class” dan di ikuti dengan “nama class”,

yang dilanjutkan dengan tanda kurung buka dan kurung tutup “()”, serta diakhiri dengan titik dua “:”.

Berikut kode dari struktur penulisan class.

```
class <nama_class> () :  
    def metode 1 (self) :  
        Isi metode 1  
    def metode 2 (self) :  
        Isi metode 2
```

Kalau kita lihat dan perhatikan cara penulisan class diatas maka kita akan mendapatkan semua metode yang ada di atas terdapat **self**. **Self** ini yang menjadi komponen yang dijalankan dari sebuah metode. Metode merupakan sebuah *function* yang lebih luas karena setiap aktifitas program yang akan dibuat merupakan penjabaran konkrit dari sebuah algoritma.

3. Penggunaan class dan object

Dalam menggunakan class perlu di fahami bagaimana class tersebut dipanggil seperti halnya *function* dengan menyebutkan nama classnya dan di akhiri dengan tanda kurung buka dan tutup “()” seperti kode penulisan di bawah ini :

```
Nama_class()
```

Sedangkan untuk memanggil metodenya, kita cukup menggunakan memanggil class yang kemudian diikuti dengan pemanggilan nama metode yang tersedia di dalam class tersebut dengan dipisahkan oleh tanda titik. Seperti kode penulisan di bawah ini :

```
Nama_class().nama_metode()
```

Untuk memudahkan pemanggilan metode tersebut, umumnya class di tampung dalam sebuah variabel terlebih dahulu, yang kemudian kita panggil metodenya seperti di bawah ini.

```
penampung = nama_class()  
penampung.nama_metode()
```

Selain yang dijelaskan diatas dalam sebuah class juga terdapat sebuah metode yang namanya sudah disediakan oleh python yakni `__init__` yang merupakan sebuah inisialisasi seperti kode program dibawah ini :

```
class <nama_class> () :  
    def __init__(self, paramater) :  
        Isi yang ingin Anda masukkan  
    def metode 1 (self) :  
        Isi metode  
    def metode 2 (self) :
```

Self didalam kode program diatas harus dituliskan sebagai sintaks. **Self** digunakan pada metode yang akan dinyatakan sebagai metode dari kelas yang kita rancang. **Self** dituliskan di parameter pertama. Jika metode tersebut tidak disertakan **self** pada metode yang dimiliki kelas tersebut, akan muncul *error*: "TypeError: **nama_function()** takes exactly **n** arguments (1 given) ", yang artinya metode tersebut tidak bisa dipanggil oleh objek yang telah terinstansiasi.

Berikut ini akan diberikan iustrasi program menghitung luas persegi panjang menggunakan class dengan nama class bangunDatar.

```

class bangunDatar:

    def __init__(self, a, b):
        self.a = a
        self.b = b
        bangunDatar.luas = a*b

    #deklarasi method class
    def luasBangunDatar(self):
        print "Luas Bangun Datar : %d" % bangunDatar.luas

    def sisiBangunDatar(self):
        print "Alas : ", self.a, ", Panjang : ", self.b

#membuat objek pertama
a1 = input ('Masukkan alas:')
b1 = input ('Masukkan panjang:')
bidang1 = bangunDatar(a1, b1)

#perintah akses atribut objek
bidang1.sisiBangunDatar()
bidang1.luasBangunDatar ()

#membuat objek kedua
a2 = input ('Masukkan alas:')
b2 = input ('Masukkan panjang:')
bidang2 = bangunDatar(a2, b2)

#perintah akses atribut objek
bidang2.sisiBangunDatar()
bidang2.luasBangunDatar ()

```

Output yang dihasilkan setelah **running** program

```
Masukkan alas:23
Masukkan panjang:56
Alas : 23 , Panjang : 56
Luas Bangun Datar : 1288
Masukkan alas:178
Masukkan panjang:464
Alas : 178 , Panjang : 464
Luas Bangun Datar : 82592
```

Misal kita akan mencari jumlah dari bilangan genap, bilangan ganjil dan bilangan kelipatan 6 dalam suatu himpunan bilangan (*list*). Maka langkahnya kita harus membuat suatu class pencari bilangan dengan cara mendefinisikan terlebih dahulu bilangan genap (sisa hasil bagi 2 sama dengan 0), bilangan ganjil (sisa hasil bagi 2 tidak sama dengan 0), dan bilangan kelipatan 6 (sisa hasil bagi 6 sama dengan 0) dalam setiap bilangannya menjadi suatu *function-function*. Kemudian kita lakukan *looping* dari bilangan tersebut dari bilangan pertama sepanjang jumlah bilangan pada himpunan yang akan dibaca oleh program tersebut.

Untuk lebih jelasnya perhatikan contoh program berikut ini :

```
class pencariBilangan() :
    def __init__(self, bilangan):
        self.himpunan = bilangan
        self.genap = 0
        self.ganjil = 0
        self.kelipatan6 = 0

    def Genap (self) :
```

```

        for i in range(len(self.himpunan)):
            x = self.himpunan[i]%2
            if x == 0 :
                self.genap = self.genap + 1
        return self.genap
def Ganjil (self) :
    for i in range(len(self.himpunan)):
        x = self.himpunan[i]%2
        if x != 0 :
            self.ganjil = self.ganjil + 1
    return self.ganjil
def Kelipatan6 (self) :
    for i in range(len(self.himpunan)):
        x = self.himpunan[i]%6
        if x == 0 :
            self.kelipatan6 = self.kelipatan6 + 1
    return self.kelipatan6

bilangan = [1,2,3,4,5,6,7,8,9,10,11,12,
13,14,15,16,17,18,19,20]
temp = pencariBilangan(bilangan)

jmlgenap = temp.Genap()
jmlganjil = temp.Ganjil()
jmlkelipatan6 = temp.Kelipatan6()

print(jmlgenap)
print(jmlganjil)
print(jmlkelipatan6)

```

Ketika program di atas di jalankan, setiap *function* akan membaca bilangan yang berada dalam himpunan [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] dari bilangan pertama yakni “1” sampai bilangan terakhir “20”. Jumlah dari bilangan genap, ganjil dan kelipatan 6 tersebut menghasilkan output seperti dibawah ini :

```
10
10
3
```

Bilangan genap berjumlah 10

Bilangan ganjil berjumlah 10

Bilangan kelipatan 6 berjumlah 3

B. Kesimpulan

Python merupakan pemrograman berorientasi objek atau yang dikenal dengan *Object Oriented Programming* (OOP). Sehingga bagian-bagian dalam program disebut sebagai objek. Secara teknis objek merupakan kumpulan variabel dan fungsi yang dibungkus ke dalam suatu blok yang disebut dengan kelas.

Kelas berperan sebagai bentuk abstrak atau pemodelan dari suatu objek tertentu. Kode program class:

```
class <nama_class> () :
    def __init__(self, paramater) :
        Isi yang ingin Anda masukkan
    def metode 1 (self) :
        Isi metode
```

```
def metode 2 (self) :  
    Isi metode
```

Sedangkan kode program untuk memanggil class:

```
Nama_class().nama_metode()
```

Lembar Kerja Mahasiswa

waktu : 90 menit

Tugas mandiri 8

Penilaian :

1. Proses pengerjaan 25%
2. Kreatifitas 55%
3. Output 20%

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan class.

BAB 10. **Penggunaan Module**

Pokok Bahasan :

Penggunaan module

Deskripsi Materi :

Cara membuat class dan modul serta menggunakannya. Class dan modul ini sangat lekat dengan sebuah aplikasi, biasanya pada aplikasi cukup besar akan menggunakan class dan modul menjadi sangat bermanfaat. Sebuah aplikasi seringkali melakukan suatu kegiatan yang berulang, jika program di buat secara berulang kali akan memakan memori yang cukup besar.

Sub CPMK :

- a. Memahami module dan packages
- b. Memahami cara membuat module-module di dalam packages
- c. Memahami module di file utama
- d. Mampu menyelesaikan masalah matematika dengan phyton

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

1. Modul dan Packages

Kita telah mempelajari function yang berguna untuk membuat suatu kode program yang dapat digunakan berulang-ulang kali. Namun, apabila program yang dibuat sangat kompleks membutuhkan banyak kode program (sub program), kode-kode program dapat dibuatkan dalam file-file terpisah yang di sebut sebagai modul (*module*). Penggunaan modul hampir mirip dengan class, class merupakan kumpulan function dengan kode program hanya berada dalam satu file, sedangkan pada modul function digunakan dengan kode program dalam beberapa file. Sehingga dapat dikatakan bahwa modul di dalam python adalah file-file .py yang terdiri dari kode program python.

Python menyediakan beberapa modul di *python standard library* dan sudah terinstal bersamaan dengan instalasi python. Modul lainnya dapat diinstal melalui *python's package manager pip* seperti matplotlib. Selain itu, kita dapat membuat modul python sendiri yang terdiri dari file python .py, yang selanjutnya akan dibahas beserta ilustrasi penulisan kode program.

Selain modul dalam python juga dikenal dengan istilah packages. Packages merupakan folder yang berisi modul-modul dalam bentuk file-file .py dan file **`_init_.py`**. **`_init_.py`** sebagai file utama yang harus ada dalam folder sehingga folder tersebut dapat dikenali sebagai packages. File **`_init_.py`** bertugas untuk memuat semua modul-modul yang berada di dalam packages, agar isi di dalam modul dapat digunakan di file yang berisi *runnable code* (kode program untuk menjalankan class dan modul). Perhatikan contoh program pembuatan modul dalam python berikut ini:

```
import math

def pembukaan ():
    print "Bismillahirrahmannirrahim"

#Luas segitiga
def luasSegitiga (alas, tinggi):
    hasil = 0.5 * alas * tinggi
    return int (hasil) #konvert float ke integer

# Luas persegi
def luasPersegi (sisi):
    hasil = sisi* sisi
    return hasil

# Luas Lingkaran
def luasLingkaran (jari2):
    hasil = math.pi*(jari2**2)
    return hasil

# Faktorial
def faktorial (nilai):
    if 0 <= nilai <= 2:
        return nilai
    elif nilai < 0:
        return false
```

```
else:
    hasil = nilai*faktorial(nilai-1)
    return hasil
```

`import math` adalah kode program untuk mengimport modul `math` yang telah tersedia (*library*) di dalam python khusus untuk fungsi matematika. Pada kode program diatas untuk mencari luas lingkaran dibutuhkan nilai pi ($\pi = \frac{22}{7}$ atau 3.14), maka kita dapat memanggil modul `math` untuk nilai pi dengan menuliskan kode program `math.pi`.

Setelah kode program diatas ditulis simpan kode program tersebut dengan nama file `latihanModul.py`. File tersebut akan menjadi modul yang dapat digunakan di dalam kode program manapun. Sebagai latihan akan dibuatkan kode program yang menggunakan modul `latihanModul` seperti berikut ini.

```
import latihanModul

def main():
    print ("LUAS SEGITIGA")
    alassegitiga=input("Masukkan panjang \
    alas segitiga: ")
    tinggisegitiga=input("Masukkan tinggi \
    segitiga: ")
    luas_segitiga=latihanModul.luasSegitiga\
        (alassegitiga, tinggisegitiga)
    print "Luas Segitiga =",luas_segitiga
```

```

print ("LUAS PERSEGI")
sisipersegi=input("Masukkan sisi persegi: ")
luas_persegi=latihanModul.luasPersegi\
              (sisipersegi)
print "Luas Persegi =",luas_persegi

print ("LUAS LINGKARAN")
jariJari=input("Masukkan jari-jari \
lingkaran: ")
luas_lingkaran=latihanModul.luasLingkaran\
              (jariJari)
print "Luas Lingkaran =",luas_lingkaran

print ("FAKTORIAL")
bilangan=input("Masukkan bilangan: ")
faktorial=latihanModul.faktorial(bilangan)
print "Luas Lingkaran =",faktorial

main()

```

Diawal program tuliskan `import.latihanModul`, sebagai perintah untuk mengimport modul `latihanModul` agar dapat digunakan atau dipanggil pada kode program.

Penggunaan backslash di dalam kode program diatas dikarenakan kode program tidak dapat di tuliskan dalam satu baris. Untuk memanggil modul dengan fungsi tertentu seperti fungsi `luasLingkaran` menggunakan kode program berikut:

```
luas_lingkaran=latihanModul.luasLingkaran(jariJari)
```

`luas_lingkaran` adalah nama variabel, `latihanModul` adalah nama file (modul yang akan dipanggil) dan `luasLingkaran` adalah nama fungsi yang berada dalam modul yang akan dipanggil, sedangkan `jariJari` parameter yang akan dikonversi ke dalam parameter dalam fungsi `luasLingkaran`.

Berikut ini contoh menggunakan modul `random` untuk memperoleh angka secara acak yang telah tersedia dalam program python.

```
import random

acak = random.Random()
# Dapatkan bilangan acak antara [1 dan 100].
angka = acak.randrange(1, 100)

i = 0
temp = ""

while True:
    j = int(input(temp + "\nTebak bilangan \
saya antara 1 dan 100: "))
    i += 1
    if j > angka:
        temp += str(j) + " masih terlalu besar.\n"
    elif j < angka:
        temp += str(j) + " masih terlalu kecil.\n"
    else:
        break
```

```
input("\n\nBagus, berhasil ditebak \
dalam {0} tebakan!\n\n".format(i))
```

Output yang mungkin dihasilkan setelah running program sebagai berikut:

```
Tebak bilangan saya antara 1 dan 100: 46
46 masih terlalu besar.
```

```
Tebak bilangan saya antara 1 dan 100: 34
46 masih terlalu besar.
34 masih terlalu besar.
```

```
Tebak bilangan saya antara 1 dan 100: 26
46 masih terlalu besar.
34 masih terlalu besar.
26 masih terlalu besar.
```

```
Tebak bilangan saya antara 1 dan 100: 15
46 masih terlalu besar.
34 masih terlalu besar.
26 masih terlalu besar.
15 masih terlalu besar.
```

```
Tebak bilangan saya antara 1 dan 100: 8
```

46 masih terlalu besar.

34 masih terlalu besar.

26 masih terlalu besar.

15 masih terlalu besar.

8 masih terlalu kecil.

Tebak bilangan saya antara 1 dan 100: 12

46 masih terlalu besar.

34 masih terlalu besar.

26 masih terlalu besar.

15 masih terlalu besar.

8 masih terlalu kecil.

12 masih terlalu besar.

Tebak bilangan saya antara 1 dan 100: 10

46 masih terlalu besar.

34 masih terlalu besar.

26 masih terlalu besar.

15 masih terlalu besar.

8 masih terlalu kecil.

12 masih terlalu besar.

10 masih terlalu kecil.

Tebak bilangan saya antara 1 dan 100: 11

Bagus, berhasil ditebak dalam 8 tebakan!

Selain modul `math` dan `random` yang telah dibahas sebelum, dalam python juga disediakan modul *datetime* (waktu) untuk mengetahui waktu saat ini. Berikut contoh kode program modul waktu.

```
import datetime
import time

sekarang = datetime.datetime.now()

tanggal = sekarang.date()
waktu = sekarang.time()

print 'Hari : ', tanggal.day
print 'Bulan : ', tanggal.month
print 'Tahun : ', tanggal.year
print 'Jam : ', waktu.hour
print 'Menit : ', waktu.minute
print 'Detik : ', waktu.second

time.sleep(5)

sekarang2 = datetime.datetime.now()

delta = sekarang2 - sekarang

print 'selisih detik : ', delta.total_seconds()
```

Output yang dihasilkan sebagai berikut:

```
Hari : 23
Bulan : 10
Tahun : 2017
Jam : 1
Menit : 31
Detik : 3
selisih detik : 5.201
```

2. Modul Matplotlib

Dalam website librari dituliskan :

“Matplotlib adalah librari plotting 2D Python yang menghasilkan gambar publikasi bermutu di dalam berbagai format hardcopy dan lingkungan interaktif sepanjang platform. matplotlib dapat digunakan di dalam script Python, shell Python dan [ipython](#) (ala MATLAB® or Mathematica®), server aplikasi web, dan enam GUI toolkit. Matplotlib mencoba untuk membuat hal mudah menjadi lebih mudah dan hal sulit menjadi mungkin. Kamu dapat membuat plot, histogram, power spectra, grafik batang, grafik error, scatterplot, dll, hanya dengan beberapa baris code.”

Matplotlib adalah salah satu modul python yang berfungsi untuk membuat gambar dua dimensi (plot 2D). Plot yang dibuat menggunakan sistem cartesius, garis horizontal untuk sumbu x dan garis vertikal untuk sumbu y. Matplotlib dapat diinstal dengan mudah layaknya python, software matplotlib dapat diunggah di berbagai web yang menyediakan source code matplotlib seperti <https://sourceforge.net/projects/matplotlib/>.

Mekanisme kerja matplotlib sangat simpel dan sederhana, yaitu dengan menuliskan beberapa baris kode sebagai perintah untuk menampilkan data kita kedalam grafik. Untuk memanggil modul matplotlib menggunakan kode program sebagai berikut:

```
import matplotlib.pyplot as <nama_parameter>
```

Adapun jenis grafik yang dapat ditampilkan diantaranya.

- a. **Line Plot** : Membentuk garis dari setiap data yang dimasukkan sebagai x dan y. Untuk membentuk garis dibutuhkan minimal 2 titik, oleh karena itu, dalam plot 2D dibutuhkan dua buah titik pada sumbu x dan y.

Sebagai contoh akan dibuatkan list nama mahasiswa beserta list nilai-nilai mahasiswa tersebut.

```
nama = ['Ismah', 'Rafa', 'Zahwa', 'Farid']  
nilai = [98, 87, 87, 100]
```

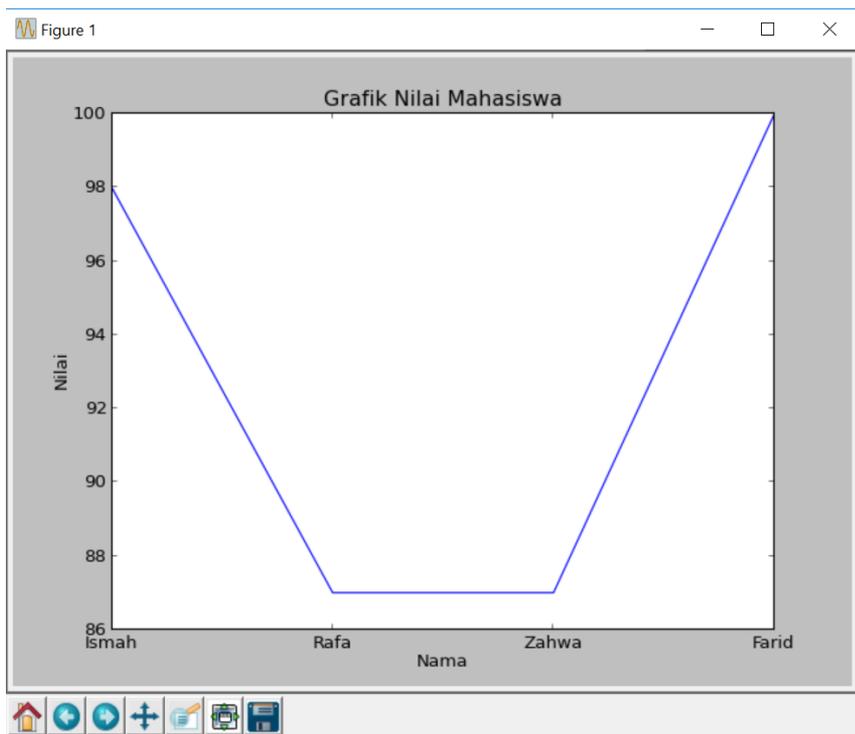
Kedua list tersebut harus memiliki jumlah indeks yang sama, posisi indeks harus sejajar, sehingga pada contoh kedua list tersebut menunjukkan Ismah mendapatkan nilai 98, Rafa mendapatkan nilai 87, dan seterusnya. Langkah selanjutnya membuat kode program, seperti berikut.

```
# memanggil modul matplotlib.pyplot  
# sebagai 'grafik'  
import matplotlib.pyplot as grafik  
  
# list nama dengan tipe data string  
nama= ['Ismah', 'Rafa', 'Zahwa', 'Farid']  
# list nilai dengan tipe data numerik  
nilai = [98, 87, 87, 100]  
  
# konversi string pada 'nama' menjadi  
# numerik dengan rentang 0-3  
x = range(len(nama))  
  
# membuat plot x dan nilai, kode 'grafik'
```

```
grafik.plot(x, nilai)
# menempelkan 'nama' dalam sumbu x
grafik.xticks(x, nama)
# membuat label 'Nilai' di sumbu y
grafik.ylabel('Nilai')
# membuat label 'Nama' di sumbu x
grafik.xlabel('Nama')
# membuat judul grafik
grafik.title('Grafik Nilai Mahasiswa')

# mencetak grafik
grafik.show()
```

Apabila program dijalankan maka akan muncul grafik seperti berikut ini.



- b. **Bar Plot** : grafik berbentuk batang, dengan penulisan kode program sebagai berikut:

```
# memanggil modul matplotlib.pyplot
# sebagai 'grafik'
import matplotlib.pyplot as grafik

# list nama dengan tipe data string
nama= ['Ismah','Rafa','Zahwa','Farid']
# list nilai dengan tipe data numerik
nilai = [98, 87, 87, 100]

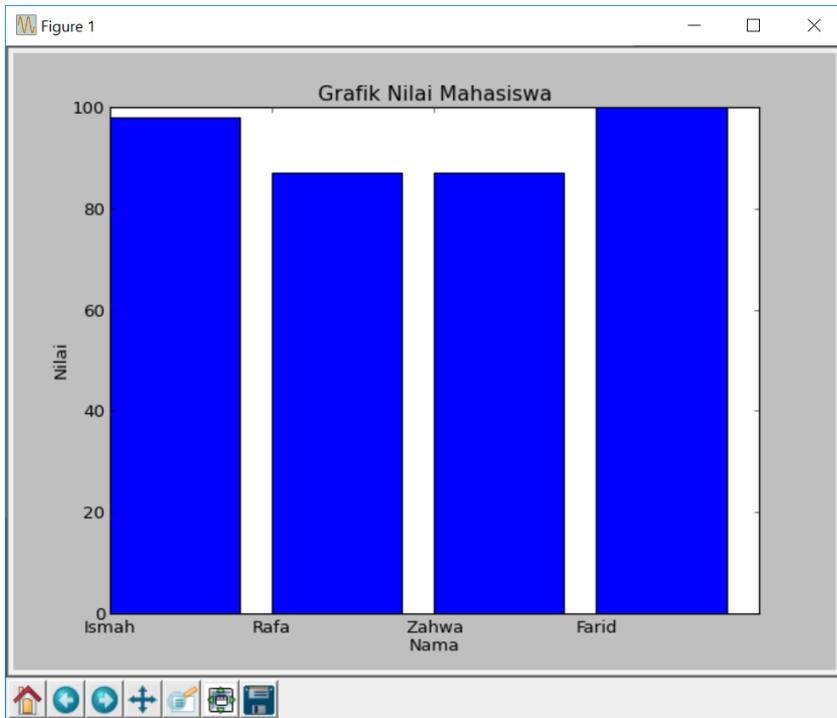
# konversi string pada 'nama' menjadi
# numerik dengan rentang 0-3
x = range(len(nama))

# membuat bar x dan nilai, kode 'grafik'
grafik.bar(x, nilai)
# menempelkan 'nama' dalam sumbu x
grafik.xticks(x, nama)
# membuat label 'Nilai' di sumbu y
grafik.ylabel('Nilai')
# membuat label 'Nama' di sumbu x
grafik.xlabel('Nama')
# membuat judul grafik
grafik.title('Grafik Nilai Mahasiswa')

# mencetak grafik
grafik.show()
```

Struktur program sama dengan pada saat membuat line grafik, yang membedakan hanya pada kode program `grafik.bar(x,`

nilai). Setelah program dijalankan muncul seperti gambar berikut.



- c. **Scatter Plot** : Grafik yang menyerupai sebuah pola hubungan dari dua variabel. Kode program dalam python sebagai berikut.

```
<nama_parameter>.scatter (<sumbu x>, <sumbu y>)
```

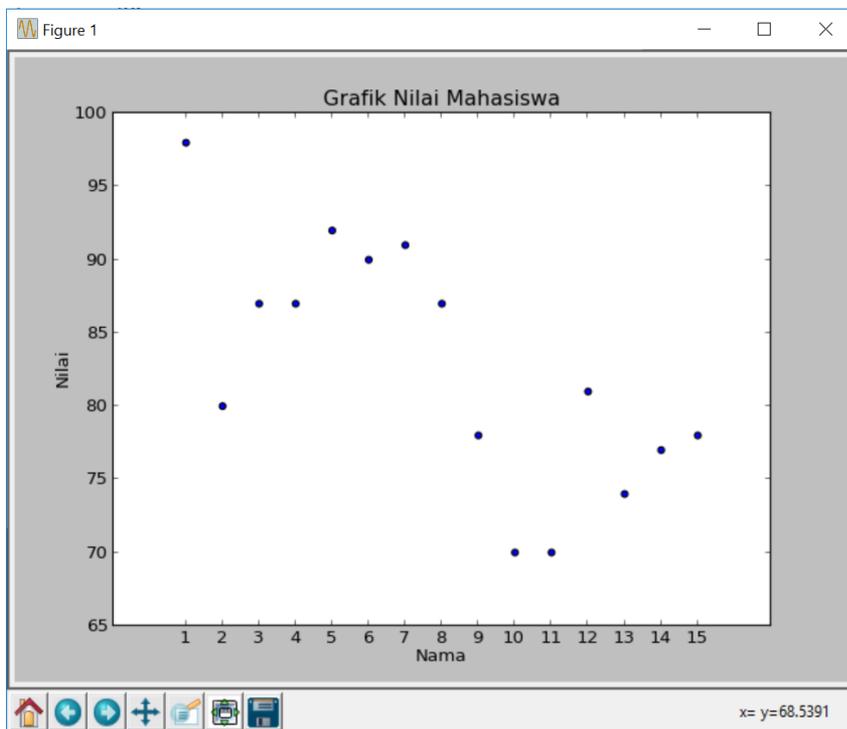
Sebagai ilustrasi perhatikan program berikut ini.

```
import matplotlib.pyplot as grafik
nama= [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
nilai = [98, 80, 87, 87,92,90,91,87,78,
70,70,81,74,77,78]

grafik.scatter(nama, nilai)
```

```
grafik.ylabel('Nilai')
grafik.xlabel('Nama')
grafik.title('Grafik Nilai Mahasiswa')
grafik.show()
```

Grafik yang dihasilkan setelah **running** program sebagai berikut.



`grafik.scatter(nama, nilai)` merupakan kode program untuk membuat grafik scatter dengan sumbu x adalah 'nama' dan sumbu y adalah 'nilai'. Program ini sedikit berbeda dengan program sebelumnya pada saat mengilustrasikan line plot dan bar plot. Pada program ini tidak dituliskan kode `grafik.xticks(x,nama)`, dikarenakan kedua variabel list bertipe numerik. Sedangkan pada program sebelumnya salah satu variabel list (nama) bertipe string.

3. Membuat modul-modul dalam packages

Cara membuat modul sama saja seperti membuat file Python lainnya. Modul bisa saja diisi dengan function, variable, pemilihan kondisi, looping atau kode lainnya yang dapat digunakan di program Python.

Agar memudahkan kita dalam memahami cara pembuatan modul, akan diberikan ilustrasi pembuatan modul dan packages python untuk menghitung luas bangun ruang kubus, balok, prisma, limas, tabung, dan bola. Sebagai langkah awal untuk membuat modul dalam python, buatlah file-file yang berisi kode program sebagai perintah untuk mencari luas bangun ruang kubus, balok, prisma, limas, tabung, dan bola, yang selanjutnya di sebut dengan modul. Sehingga file yang akan dibuat sebanyak 7 modul, modul 1 dengan kode program luas kubus, modul 2 dengan kode program luas balok, modul 3 dengan kode program luas prisma, modul 4 dengan kode program luas limas, modul 5 dengan kode program luas tabung, modul 6 dengan kode program luas bola, dan modul 7 sebagai program utama yang akan menjalankan program dengan memanggil modul 1 sampai dengan 6.

RUMUS BANGUN RUANG

a. Kubus

Rumus luas permukaan: $6 \times s^2 = 6s^2$

s = panjang sisi bidang

b. Balok

Rumus luas permukaan: $2\{(p \times l)+(p \times t)+(l \times t)\}$

p = panjang balok

l = lebar balok

t = tinggi balok

c. Prisma

Rumus luas permukaan : $(2 \times \text{Luas alas}) + (\text{Keliling alas} \times \text{tinggi})$

Kode modul 1 kubus.py :

```
# Define a function luas permukaan kubus

def luas_kubus(s):
    luas = 6 * (s*s)
    print 'luas kubus=',luas
```

Apabila **running** program kubus.py maka tidak akan terjadi apa-apa, karena belum ada perintah program untuk menggunakan function yang berada dalam kubus.py.

Sebelum kita memanggil function dalam file kubus.py, kita akan membuat modul berikutnya yaitu balok.py, prisma.py, limas.py, tabung.py dan bola.py.

Kode modul 2 balok.py :

```
# Define a function luas permukaan balok

def luas_balok(p,l,t):
    luas = 2*((p * l)+(p * t)+(l * t))
    print 'luas balok=',luas
```

Kode modul 3 prisma.py :

```
# Define a function luas permukaan prisma

def luas_prisma (a,t,p,l):
    print '\nMenghitung Volume Prisma\n'
    print '1. Prisma Segitiga'
    print '2. Prisma Segiempat'
```

```

print
j=input('Silakan pilih jenis Prisma \
yang akan dihitung: ')
if j==1:
    b='prisma segitiga'
    print '\nMenghitung Volume',b,
    print '\nRumus Menghitung Volume',b,'= \
luas alas x tinggi'
    print '\nMenghitung Luas Alas'
    print 'Rumus Menghitung Luas Alas = \
1/2(alas x tinggi)'
    L=(a*t)/2.0
elif j==2:
    b='prisma segiempat Volume',b,
    print '\nRumus Menghitung Volume',b,'= \
luas alas x tinggi'
    print '\nMenghitung Luas Alas'
    print 'Rumus Menghitung Luas Alas = \
panjang x lebar'
    L=p*l
else:
    print 'Pilihan Anda salah!!'

print "luas alas",b,"=",L
g=input ('Masukkan nilai tinggi prisma: ')
v=L*g
print "Volume",b,"=",v

```

Oleh karena itu, langkah selanjutnya kita akan membuat file yang kedua di direktori yang sama menggunakan nama main_program.py sehingga

kita bisa memasukkan modul yang sudah kita buat dan mengaktifkan fungsinya. File ini harus di dalam direktori yang sama, jika tidak maka Python tidak bisa menemukan filenya.

```
#import modul

import kubus

import balok

import prisma

# call function kubus

sisi = input("Masukkan panjang sisi bidang kubus = ")

kubus.luas_kubus (sisi)

#call function balok

panjang = input("Masukkan panjang balok= ")

lebar = input("Masukkan lebar balok= ")

tinggi = input("Masukkan tinggi balok= ")

balok.luas_balok(panjang,lebar,tinggi)

#call function prisma

alas=input ('\nMasukkan nilai alas : ')
```

```
tinggi=input ('Masukkan nilai tinggi : ')
panjang=input ('\nMasukkan nilai panjang : ')
lebar=input ('Masukkan nilai lebar : ')
prisma.luas_prisma(alas, tinggi,panjang,lebar)
```

Sehingga output yang dihasilkan seperti demikian:

```
Masukkan panjang sisi bidang kubus = 34
luas kubus= 6936
Masukkan panjang balok= 25
Masukkan lebar balok= 37
Masukkan tinggi balok= 23
luas balok= 4702

Masukkan nilai alas : 5
Masukkan nilai tinggi : 3

Masukkan nilai panjang : 8
Masukkan nilai lebar : 5

Menghitung Volume Prisma
```

```
1. Prisma Segitiga
```

```
2. Prisma Segiempat
```

```
Silakan pilih jenis Prisma yang akan dihitung: 1
```

```
Menghitung Volume prisma segitiga
```

```
Rumus Menghitung Volume prisma segitiga = luas alas x  
tinggi
```

```
Menghitung Luas Alas
```

```
Rumus Menghitung Luas Alas = 1/2 (alas x tinggi)
```

```
luas alas prisma segitiga = 7.5
```

```
Masukkan nilai tinggi prisma: 3
```

```
Volume prisma segitiga = 22.5
```

4. Membuat Modul dalam Direktori Python

Pengguna python dapat membuat sebuah fungsi atau modul yang akan di import ke dalam python untuk menambah library yang tersedia dalam python. Sebagai contoh sederhana membuat kode program menggunakan fungsi (**def**) dan **return** untuk mencari nilai sisa pembagian dari 1 sampai 10.

```
def mod1 (a):
```

```
mod1=a%1

return mod1

def mod2 (a):

    mod2=a%2

    return mod2

def mod3 (a):

    mod3=a%3

    return mod3

def mod4 (a):

    mod4=a%4

    return mod4

def mod5 (a):

    mod5=a%5

    return mod5

def mod6 (a):

    mod6=a%6

    return mod6

def mod7 (a):

    mod7=a%7

    return mod7
```

```
def mod8 (a):  
  
    mod8=a%8  
  
    return mod8  
  
def mod9 (a):  
  
    mod9=a%9  
  
    return mod9  
  
def mod10 (a):  
  
    mod10=a%10  
  
    return mod10
```

Kemudian save script pada library python, misal script diberi nama `sisabagi.py` dan di save di dalam direktori python `C:\Python27\Lib`. Selanjutnya script `sisabagi.py` dapat diimport ke dalam interpreter python atau IDLE dengan menuliskan `from sisabagi import*`. Setelah itu `sisabagi.py` menjadi modul baru di *library* python yang dapat kita gunakan. Sebagai contoh sisa hasil bagi 26719 dengan 8 dan sisa hasil bagi 13 dengan 3.

```
>>> from sisabagi import*  
>>> mod8(26719)  
7  
>>> mod3 (13)  
1  
>>>
```

B. Kesimpulan

Modul adalah sebuah file yang berisi sekumpulan fungsi – fungsi dan instruksi – instruksi program python. Modul tersebut disimpan dengan ekstension `.py`, pemanggilan modul dijalankan dengan perintah `import`.

Python menyediakan beberapa modul di *python standard library* dan sudah terinstal bersamaan dengan instalasi python. Modul lainnya dapat diinstal melalui *python's package manager pip* seperti `matplotlib`. Selain itu, kita dapat membuat modul python sendiri yang terdiri dari file python `.py`, yang selanjutnya akan dibahas beserta ilustrasi penulisan kode program.

Packages adalah suatu direktori yang digunakan untuk menyimpan kumpulan modul (*file. py*). Di dalam direktori tersebut harus ada satu file khusus dengan nama `_init_.py`. File ini berisi informasi tentang nama-nama file di dalam direktori yang akan dibungkus ke dalam paket.

Tugas mandiri 8

Penilaian :

1. Proses pengerjaan 25%
2. Kreatifitas 55%
3. Output 20%

Buatlah sebuah soal matematika dan penyelesaiannya menggunakan program python yang memanfaatkan modul.

BAB 11. **Operasi Matriks**

Pokok Bahasan : Melakukan Operasi Matriks di Python

Deskripsi Materi :

Materi yang akan dibahas terkait operasi matriks di python diantaranya, mengakses elemen matriks, melakukan penjumlahan matriks, melakukan pengurangan matriks, melakukan perkalian matriks

Sub CPMK :

- 15.1 Memahami matriks cara operasi matriks
- 15.2 Mampu mengakses elemen matriks di python
- 15.3 Mampu melakukan penjumlahan matriks di python
- 15.4 Mampu melakukan pengurangan matriks di python
- 15.5 Mampu melakukan perkalian matriks di python

Waktu : 1x pertemuan

Metode Pembelajaran : Diskusi, praktek dan tanya jawab

A. Pembahasan

Matriks adalah kumpulan bilangan yang tersusun rapi membentuk persegi, yang memiliki baris dan kolom. Baris matriks ditunjukkan pada bilangan yang menjulur ke samping (horizontal), dan kolom matriks ditunjukkan pada bilangan yang menjulur tegak ke atas (vertikal).

$$\begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{array}{l} \longrightarrow \text{baris} \\ \longrightarrow \text{kolom} \end{array}$$

Dalam matriks terdapat istilah ordo yang menunjukkan jumlah baris dan kolom pada matriks. Contoh matriks A ordo 2 x 3

$$A_{2 \times 3} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Matriks digunakan untuk menyelesaikan sistem persamaan linier. Sebagai contoh untuk menyelesaikan dua persamaan dengan dua variabel x dan y .

$$\begin{array}{l} ax + by = c \\ dx + ey = f \end{array} \rightarrow \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ f \end{bmatrix}$$

$\begin{bmatrix} a & b \\ d & e \end{bmatrix}$ adalah matriks koefisien yang terdiri dari 2 baris dan 2 kolom

$\begin{bmatrix} x \\ y \end{bmatrix}$ adalah matriks variabel yang terdiri dari 2 baris dan 1 kolom

$\begin{bmatrix} c \\ f \end{bmatrix}$ adalah matriks konstanta yang terdiri dari 2 baris dan 1 kolom

1. Operasi Matriks

Sama seperti operasi aljabar, dalam matriks juga terdapat operasi penjumlahan, pengurangan, dan perkalian.

a. Penjumlahan dan pengurangan matriks

Kedua buah matriks dapat dijumlahkan dan dikurangkan apabila memiliki ordo yang sama. Penjumlahan dan pengurangan dua buah matriks menghasilkan matriks yang baru dengan ordo yang sama dengan matriks yang dijumlahkan atau dikurangkan. Aturan penjumlahan dan pengurangan dua buah matriks sebagai berikut:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \pm \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \pm e & b \pm f \\ c \pm g & d \pm h \end{bmatrix}$$

$$\text{Misalkan } P = \begin{bmatrix} -7 & 2 \\ 4 & 1 \end{bmatrix} \text{ dan } Q = \begin{bmatrix} 1 & -2 \\ 4 & -7 \end{bmatrix}$$

$$\begin{aligned} P+Q &= \begin{bmatrix} -7 & 2 \\ 4 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -2 \\ 4 & -7 \end{bmatrix} \\ &= \begin{bmatrix} -7+1 & 2-2 \\ 4+4 & 1-7 \end{bmatrix} \\ &= \begin{bmatrix} -6 & 0 \\ 8 & -6 \end{bmatrix} \end{aligned}$$

b. Perkalian matriks

Dua buah matriks dapat dikalikan apabila banyaknya kolom pada matriks pertama sama dengan banyaknya baris pada matriks kedua. Misal perkalian matriks $A_{2 \times 2}$ dengan $B_{2 \times 3}$, maka akan menghasilkan matriks misal $C_{2 \times 3}$.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix} = \begin{bmatrix} ae + bh & af + bi & ag + bj \\ ce + dh & cf + di & cg + dj \end{bmatrix}$$

A B C

Selain penjumlahan, pengurangan dan perkalian, dikenal operasi matriks lainnya seperti determinan dan matriks kebalikan (invers). Tetapi dalam materi pada pertemuan ini di batasi hanya untuk ketiga operasi matriks yang telah dijelaskan sebelumnya.

Matriks dengan ordo 2×2 dapat dengan mudah diselesaikan dengan cara perhitungan manual. Namun keterbatasan manusia dalam melakukan operasi perhitungan manual untuk menyelesaikan matriks rata-rata hanya terbatas pada ordo 3×3 , sedangkan matriks dengan ordo lebih dari 3×3 akan sangat sulit untuk diselesaikan dengan cara manual. Melalui komputasi permasalahan matriks dengan ordo lebih dari 3×3 akan dengan mudah teratasi, apabila dibuatkan sebuah program komputer yang terstruktur sesuai dengan alur tahapan dalam menyelesaikan matriks.

Contoh:

$$\text{Misalkan } A = \begin{bmatrix} -3 & -1 \\ 5 & 2 \end{bmatrix} \text{ dan } B = \begin{bmatrix} -2 & -1 \\ 5 & 3 \end{bmatrix}$$

$$\begin{aligned} AB &= \begin{bmatrix} -3 & -1 \\ 5 & 2 \end{bmatrix} \begin{bmatrix} -2 & -1 \\ 5 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 6-5 & 3-3 \\ -10+10 & -5+6 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

$$\text{Misalkan } P = \begin{bmatrix} -7 & 2 \\ 4 & 1 \end{bmatrix} \text{ dan } Q = \begin{bmatrix} 1 & -2 \\ 4 & -7 \end{bmatrix}$$

$$PQ = \begin{bmatrix} -7 & 2 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 4 & -7 \end{bmatrix}$$

$$= \begin{bmatrix} -7+8 & 14-14 \\ -4+4 & 8-7 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. Melakukan Operasi Matriks di Python

a. Cara mengakses elemen matriks di python

Matriks dalam python dibuat menggunakan kode list yang bersarang, artinya terdapat list di dalam list. Sebagai contoh

```
A = [[1,2],[3,4]]
```

A adalah variabel matriks dengan tipe data list yang memiliki dua indeks, masing-masing indeks merupakan list yang terdiri dari 2 indeks. Untuk mengakses elemen matriks di python dapat menggunakan kode pengulangan **for** dan **function**, seperti berikut ini:

```
def matriks(temp):
    for i in temp:
        print i

A = [[1,2],[3,4]]
B = [[1,2],[3,4]]

print 'Matriks A ='

matriks(A)

print 'Matriks B ='

matriks(B)
```

Program diatas untuk mengakses elemen dua buah matriks (A dan B) dengan ordo 2x2. Hasil output setelah running program sebagai berikut:

Matriks A =

[1, 2]

[3, 4]

Matriks B =

[1, 2]

[3, 4]

b. Penjumlahan dan Pengurangan Matriks

Setelah kita memahami cara mengakses elemen-elemen matriks, maka selanjutnya kita akan membahas operasi dua buah matriks atau lebih. Operasi penjumlahan dua buah matriks dapat dilakukan apabila ordo kedua matriks adalah sama. Contoh program untuk melakukan operasi penjumlahan dua buah matriks dapat di lihat seperti berikut ini.

```
def matriks(temp):  
    for i in temp:  
        print i  
  
def penjumlahan_matriks(a, b):  
    temp_i = []  
    temp_j = []  
  
    for i in range(0, len(a)):  
        for j in range(0, len(a[0])):  
            temp_i.append(a[i][j] + b[i][j])
```

```

        temp_j.append(temp_i)

        temp_i = []

    return temp_j

def perkalian_matriks(c,d):

    temp_k = []

    for x in range(0, len(c)):

        temp_l = []

        for y in range(0, len(c[0])):

            hasil = 0

            for z in range(0, len(c)):

                hasil = hasil + (c[x][z] * d[z][y])

            temp_l.append(hasil)

        temp_k.append(temp_l)

    return temp_k

A = [[1,2],[3,4],[3,2]]
B = [[1,2],[3,4],[3,2]]

print 'Matriks A ='
matriks(A)

print 'Matriks B ='
matriks(B)

```

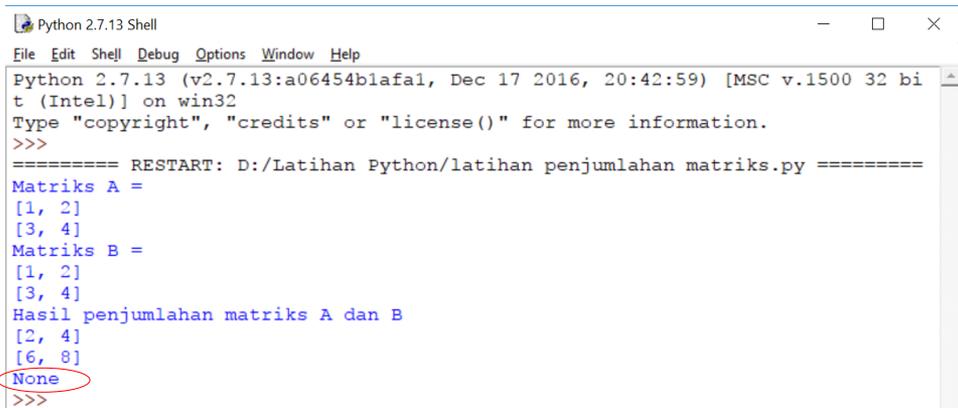
```
print 'Hasil penjumlahan matriks A dan B'  
jumlah=penjumlahan_matriks(A,B)  
print matriks (jumlah)  
print 'Hasil perkalian matriks A dan B'  
kali=perkalian_matriks(A,B)  
print matriks (kali)
```

Program diatas memiliki dua function **matriks** dan **penjumlahan_matriks**. Function **matriks** sebagai fungsi untuk mengakses elemen matriks, sedangkan function **penjumlahan_matriks** adalah fungsi untuk melakukan operasi penjumlahan matriks. Alur dari proses penjumlahan matriks dalam program diatas adalah sebagai berikut.

- 1). `temp_i = []` dan `temp_j = []` merupakan nama variabel list yang belum memiliki indeks
- 2). `for i in range (0, len(a))`, `i` menunjukkan baris pada matriks, `len (a)` artinya jumlah list dalam `a` (contoh diatas `len(a) = 2`).
- 3). `for j in range (0, len (a[0]))`, `j` menunjukkan kolom pada matriks, `len(a[0])` artinya jumlah list pada indeks list `a` yang pertama (ingat urutan indeks pada list dimulai dari angka nol)
- 4). `temp_i.append(a[i][j] + b[i][j])` merupakan penambahan indeks misal `i =0` dan `j =0`, maka `a[0][0]` yaitu baris pertama dan kolom pertama pada matriks `a` dengan indeks `b[0][0]` yaitu baris pertama dan kolom pertama pada matrik `b`.
- 5). `temp_j.append(temp_i)` merupakan penambahan indeks `temp_i` ke dalam list `temp_j`

- 6). `temp_i = []`, kita siapkan `temp_i` list kosong lagi untuk memproses kembali ke langkah 3 hingga `j =` jumlah kolom (contoh `j = 1`, dikarenakan indeks dimulai dari angka 0 dan jumlah kolom = 2).
- 7). Selanjutnya proses akan kembali ke langkah 2 dan seterusnya hingga `i =` jumlah baris (contoh `i = 1`, dikarenakan indeks dimulai dari angka 0 dan jumlah baris = 2).
- 8). Cetak `temp_j` yang merupakan matriks hasil penjumlahan dua buah matriks A dan B.

Sehingga output yang dihasilkan setelah **running** program sebagai berikut.



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Latihan Python/latihan penjumlahan matriks.py =====
Matriks A =
[1, 2]
[3, 4]
Matriks B =
[1, 2]
[3, 4]
Hasil penjumlahan matriks A dan B
[2, 4]
[6, 8]
None
>>>
```

Operasi pengurangan dua buah matriks dapat menggunakan program diatas dengan merubah penjumlahan indeks pada *function* penjumlahan_matriks.

Mengapa muncul none pada output??

c. Operasi Perkalian Matriks

Dua buah matriks dapat dikalikan apabila jumlah kolom pada matriks pertama sama dengan jumlah baris pada matriks kedua. Program python untuk menyelesaikan perkalian matriks disusun seperti dibawah ini.

```

def matriks(temp):

    for i in temp:

        print i

    return ''

def perkalian_matriks(a,b):

    if len(b)!=len(a[0]):

        return "Matriks tidak dapat dikalikan"

    else:

        temp_i = []

        for i in range(0,len(a)):

            temp_j = []

            for j in range(0,len(b[0])):

                temp_k = 0

                for k in range(0,len(b)):

                    temp_k=temp_k+a[i][k]*b[k][j]

                temp_j.append(temp_k)

            temp_i.append(temp_j)

        return temp_i

```

```

A = [[2, 1, 2], [3, 4, 2], [5, 1, 1]]

B = [[1, 2, 1], [5, 1, 0], [3, 1, 2]]

print 'Matriks A ='

matriks(A)

print 'Matriks B ='

matriks(B)

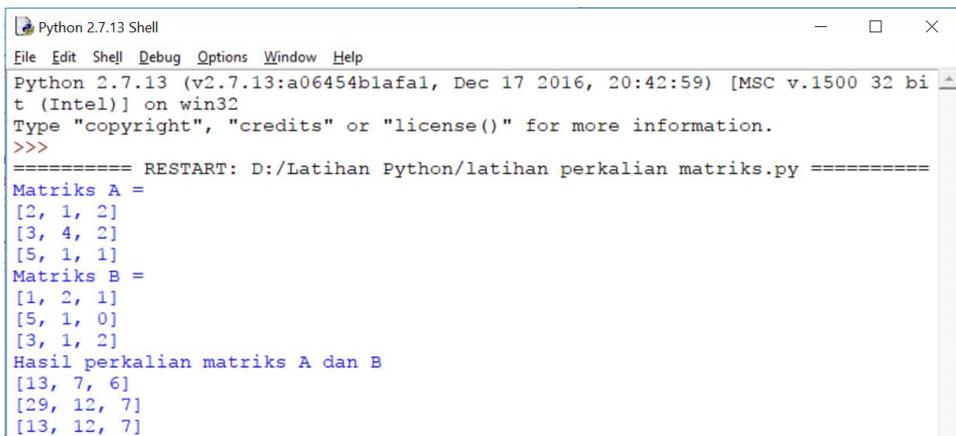
print 'Hasil perkalian matriks A dan B'

kali=perkalian_matriks(A,B)

print matriks (kali)

```

Output yang dihasilkan setelah **running** program diatas.



```

Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit
  t (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Latihan Python/latihan perkalian matriks.py =====
Matriks A =
[2, 1, 2]
[3, 4, 2]
[5, 1, 1]
Matriks B =
[1, 2, 1]
[5, 1, 0]
[3, 1, 2]
Hasil perkalian matriks A dan B
[13, 7, 6]
[29, 12, 7]
[13, 12, 7]

```

B. Kesimpulan

Penyelesaian matriks dengan ordo $n \times m$, apabila n atau m sama dengan 2 dapat dilakukan dengan mudah melalui perhitungan manual. Tetapi

apabila ukuran matriks untuk ukuran n atau m melebihi 3, penyelesaian cukup menyulitkan apabila menggunakan perhitungan manual. Melalui program komputer yang di susun menggunakan bahasa pemrograman python dapat memudahkan kita dalam menyelesaikan matriks dengan ordo yang besar. Selain itu logika berpikir dalam menyelesaikan soal matriks akan terasah pada saat pembuatan alur program.

TUGAS KELOMPOK 4

Penilaian:

Waktu : 90 menit

1. Kekompakan : 20%
2. Perencanaan : 15%
3. Kualitas soal : 20%
4. Alur dan kreatifitas program : 35%
5. Output : 10%

Buatlah program yang mampu mencari nilai determinan dan invers dari matriks. Kemudian jelaskan alur kerja program di depan kelas.

DAFTAR PUSTAKA

Fuadi, Kholid., 2013, *Tutorial Matplotlib*, Ebook, https://kholidfu.github.io/assets/matplotlib_docs.pdf, diakses pada tanggal 27 September 2017

Khalid, U. Y. M., *Intermediate Python*. First Edition. Ebook. pythontips.com. diakses pada tanggal 10 Agustus 2017

Lutz, Mark., 2011, *Programming Python*, O'Reilly

Raharjo, Budi., 2015, *Mudah Belajar Python untuk Aplikasi Desktop dan Web*, Informatika

Septian, Fajar R., 2013, *Belajar Pemrograman Python Dasar*, POSS-UPI

Zelle, M.J., 2002, *Python Programming: An Introduction to Computer Science*, Wartburg College Printing Services

<https://www.python.org/downloads/source/>, diakses pada tanggal 08 Juli 2017

http://sakti.github.io/python101/operator_dan_ekspresi.html, diakses pada tanggal 10 Agustus 2017

<https://matplotlib.org/users/beginner.html>, diakses pada tanggal 27 September 2017



**UNIVERSITAS MUHAMMADIYAH JAKARTA
FAKULTAS ILMU PENDIDIKAN
PROGRAM STUDI PENDIDIKAN MATEMATIKA**

RENCANA PEMBELAJARAN SEMESTER

MATA KULIAH	KODE	Bobot (sks)	SEMESTER	Tgl Penyusunan
Pemrogram Komputer	MAT 824	3	1	23 April 2017
OTORISASI	Dosen Pengembang RPS		Unit Kendali Mutu	Ka Prodi
CPL	<p>Pengusahaan Pengetahuan: PP1. Menguasai konsep pedagogi-didaktik matematika untuk melaksanakan pembelajaran matematika inovatif di pendidikan menengah pertama dan atas yang berorientasi pada kecakapan hidup. PP2. Menguasai konsep matematika yang meliputi logika matematika dan himpunan, aljabar, geometri, teori peluang dan statistika, matematika diskrit, pemodelan matematika, program linear, kalkulus, persamaan diferensial, metode numerik, dan analisis yang mendukung pembelajaran matematika di pendidikan dasar dan menengah serta untuk studi lanjut. PP4. Menguasai pengetahuan faktual tentang fungsi dan manfaat teknologi khususnya teknologi informasi dan komunikasi yang relevan untuk pembelajaran matematika.</p>			
CPMK	<ol style="list-style-type: none"> 1. Instalasi Python 2. Memahami cara menggunakan python 3. Memahami tipe data dan operator python 4. Memahami konsep pemilihan kondisi 5. Memahami konsep looping 6. Memahami data struktur Python tingkat lanjut 7. Memahami cara membuat <i>Function</i> 8. Memahami cara membuat exception 9. Memahami cara membuat file 10. Memahami penggunaan class 11. Memahami penggunaan module 12. Memahami cara menyelesaikan masalah matematika menggunakan python 			
Deskripsi Singkat MK	<p>Mata kuliah ini membahas mengenai bahasa pemrograman Python sebagai bahan pembuatan program matematika, matakuliah ini mencakup materi tentang: pengenalan python, tipe data dan operator,</p>			

	pemilihan kondisi, looping, data struktur python tingkat lanjut, <i>Function</i> , exception, membuat file, pengenalan class dan module dan aplikasi pada masalah matematika.	
Materi Pembelajaran/Pokok Bahasan	<ol style="list-style-type: none"> 1. Cara menggunakan Python 2. Mengetahui tipe data dan operator 3. Membuat pemilihan kondisi 4. Menyusun looping 5. Mengetahui data struktur Python tingkat lanjut 6. Membuat <i>Function</i> 7. Membuat exception 8. Membuat file 9. Pengenalan class 10. Pengenalan module 11. Operasi matriks 	
Pustaka	Utama :	
	a. Ridwan Fajar Septian., 2013. Belajar Pemrograman Python Dasar. Ebook. Bandung: Pustaka UPI.	
	Pendukung :	Kumpulan artikel. Playing with Python. Ebook Hendri. 2003. Cepat Mahir Python. Ebook. IlmuKomputer.com M. Octaviano Pratama. 2003. Mudah Belajar Python. Ebook: IlmuKomputer.com
Media Pembelajaran	Perangkat lunak :	Perangkat keras :
	Power Point	LCD&Proyektor
Team Teaching	Ismah, M.Si., Hastri Rosiyanti, MPmat., Rahmita NurulMuthmainah, M.Pd. M.Sc,	
Matakuliah Syarat	-	

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Pert Ke- Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)	
(1)	(3)	(4) (Menit)	(5)	(6)	(7)	
1-2	1.4 Download referensi python 1.5 Download python 1.6 Instal python 2.6 Memahami cara menginstalasi python 2.7. Memahami cara Menggunakan Python Interpreter Prompt dan Teks	Kriteria: Materi Pembelajaran Sikap, Pedagogis dan ketrampilan dan keaktifan di kelas. Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	6. Tugas (1) (3) (4) (5) (6) (7)	<ul style="list-style-type: none"> • Kuliah [TM: 2x(3x50’)] • Tugas [BT+BM: (2+2)x(3x 60’)] 	3. Instalasi Phyton 4. Cara menggunakan Python	14

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Bentuk Penilaian	Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)
(1)	(3)	(4)	(5)		(6)	(7)
	Editor 2.8. Memahami cara Mencetak Informasi dengan <i>Function</i> "print" 2.9. Memahami cara Menerima Masukan Data dengan <i>Function</i> "raw_input" dan "input" 2.10. Memahami Hal Lain yang Harus Diingat dalam Penggunaan Python					
3	3.3. Memahami jenis tipe data dan operator python 3.4. Memahami prioritas Eksekusi Operator di Python	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	<ul style="list-style-type: none"> • Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')] 	Mengenal tipe data dan operator	7
4	4.5 Memahami Penggunaan Operator Kondisional dan Logika pada <i>Keyword</i> "if" 4.6 Memahami Penggunaan "else" pada "if" 4.7 Memahami Penggunaan "elif" pada "if" 4.8 Memahami cara menyusun program penyelesaian permasalahan matematika menggunakan kondisi	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	7. Diskusi 8. Praktek 9. Tugas	<ul style="list-style-type: none"> • Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')] 	Membuat pemilihan kondisi	8
5	5.7 Mengenal Pengulangan "for" dan "while" 5.8 Menyusun Pengulangan dengan "for" 5.9 Memahami <i>Function</i> "range" 5.10 Menggunakan <i>Function</i> "range"	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	10. Diskusi 11. Praktek 12. Tugas	<ul style="list-style-type: none"> • Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')] 	Menyusun looping	8

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Bentuk Penilaian	Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)
(1)	(3)	(4)	(5)		(6)	(7)
	<p>pada Pengulangan "for"</p> <p>5.11 Menyusun Pengulangan dengan "while"</p> <p>5.12 Memahami cara menyusun program penyelesaian permasalahan matematika menggunakan looping</p>					
6-7	<p>7.4 Mengenal List, Dictionary dan Tuple</p> <p>7.5 Memahami cara Akses List, Tuple, dan Dictionary</p> <p>7.6 Memahami mengubah Isi List, Tuple, dan Dictionary</p> <p>7.7 Memahami menambahkan Data pada List, Tuple, dan Dictionary</p> <p>7.8 Memahami Menghapus Isi List, Tuple, dan Dictionary</p> <p>7.9 Memahami cara menyusun program penyelesaian permasalahan matematika</p>	<p>Kriteria: Sikap, Penguasaan dan ketrampilan</p> <p>Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.</p>	<p>4. Diskusi</p> <p>5. Praktek</p> <p>6. Tugas</p>	<p>• Kuliah [TM: 2x(3x50')]</p> <p>• Tugas [BT+BM:(2+2)x(3x60')]</p>	Mengenal data struktur Python tingkat lanjut	15
UJIAN TENGAH SEMESTER						
9-10	<p>9.5. Memahami Pengenalan <i>Function</i> Tanpa "return"</p> <p>9.6. Memahami <i>Function</i> yang Menggunakan "return"</p> <p>9.7. Memahami Default Argument pada Python</p> <p>9.8. Memahami Variable-length</p>	<p>Kriteria: Sikap, Penguasaan dan ketrampilan</p> <p>Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.</p>	<p>4. Diskusi</p> <p>5. Praktek</p> <p>6. Tugas</p>	<p>• Kuliah [TM: 2x(3x50')]</p> <p>• Tugas [BT+BM:(2+2)x(3x60')]</p>	Membuat <i>Function</i>	15

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Bentuk Penilaian	Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)
(1)	(3)	(4)	(5)		(6)	(7)
	Argument pada Python 10.10 Memahami <i>Keyword</i> Argument pada <i>Function</i> 10.11 Memahami <i>Keyword-length</i> Argument pada <i>Function</i> 10.12 Memahami Pass by Reference dan Pass by Value pada Python 10.13 Memahami Variable Scope pada Python 10.14 Memahami cara menyusun program penyelesaian permasalahan matematika dengan <i>Function</i>					
11	11.16 Memahami Jenis – Jenis Exception 11.17 Memahami Menyusun Multiple Except 11.18 Memahami penggunaan Multiple Exception 11.19 Memahami Try-Except Bersarang 11.20 Memahami membuat Exception Sendiri 11.21 Menggunakan “finally” pada Try-Except 11.22 Memahami cara menyelesaikan integral garis 11.23 Memahami cara menyusun program penyelesaian permasalahan matematika dengan exception	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	<ul style="list-style-type: none"> • Kuliah [TM: 1x(3x50’)] • Tugas [BT+BM:(1+1)x(3x60’)] 	Membuat exception	7

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Bentuk Penilaian	Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)
(1)	(3)	(4)	(5)		(6)	(7)
12	12.11Memahami pengenalan file dan membuat file baru 12.12Memahami mengisi file dan membaca isi file 12.13Memahami isi file dengan cara baris per baris 12.14Memahami posisi pointer file 12.15 Memahami cara mengganti nam file dan menghapus file	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	• Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')]	Membuat file	7
13	13.4 Memahami cara membuat class dan object 13.5 Memahami built-i <i>Function</i> pada class dan object 13.6 Memahami penggunaan class dan object	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	• Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')]	Pengenalan class	7
14	14.4 Memahami module dan packages 14.5 Memahami cara membuat module-module di dalam packages 14.6 Memahami module di file utama	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	• Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')]	Pengenalan module	8
15	15.6 Memahami matriks cara operasi matriks 15.7 Mampu mengakses elemen matriks di python 15.8 Mampu melakukan penjumlahan matriks di python 15.9 Mampu melakukan pengurangan matriks di python 15.10Mampu melakukan perkalian matriks di python	Kriteria: Sikap, Penguasaan dan ketrampilan Bentuk non-test: Penilaian berdasarkan keaktifan di kelas.	4. Diskusi 5. Praktek 6. Tugas	• Kuliah [TM: 1x(3x50')] • Tugas [BT+BM:(1+1)x(3x60')]	Matriks	10

Pert Ke-	Sub-CP-MK (sbg kemampuan akhir yg diharapkan)	Kriteria & Bentuk Penilaian	Metode Pembelajaran	Estimasi Waktu (Menit)	Materi Pembelajaran	Bobot Penilaian (%)
(1)	(3)	(4)	(5)		(6)	(7)
UJIAN AKHIR SEMESTER						

Catatan :

1. TM : Tatap Muka, BT : Belajar Terstruktur, BM : Belajar Mandiri
2. [TM: $2 \times (3 \times 50')$] dibaca : kuliah tatap muka 2 kali (minggu) x 3 sks x 50 menit = 300 menit (5 jam)
3. [BT+BM: $(2+2) \times (3 \times 60')$] dibaca : belajar terstruktur 2 kali (minggu) dan belajar mandiri 2 kali (minggu) x 3 sks x 60 menit = 720 menit (12 jam)

BIODATA PENULIS

Nama lengkap penulis **Ismah**, lahir di Jakarta pada tanggal 30-November-1984. Penulis telah menyelesaikan pendidikan S1 Matematika pada tahun 2006 peminatan matematika informatika, di Universitas Islam Negeri (UIN) Syarif Hidayatullah Jakarta dengan mendapatkan gelar Sarjana Sains (S.Si). Selanjutnya pada tahun 2010 penulis telah menyelesaikan magister bidang Statistika di Institut Pertanian Bogor dengan mendapatkan gelar Sarjana Magister Sains (M.Si).

Saat ini penulis berprofesi sebagai dosen Pendidikan Matematika di Fakultas Ilmu Pendidikan Universitas Muhammadiyah Jakarta (UMJ) sejak tahun 2012. Sebelumnya penulis pernah menjadi dosen Universitas Satyagama (2007-2010) dan Universitas Gunadarma (2011-2012).

Adapun matakuliah yang pernah diampu diantaranya adalah pemrograman komputer, statistika matematika 1 & 2, statistika pendidikan, metode penelitian 1 & 2, teori bilangan, matematika ekonomi, dan geometri transformasi.