MODUL INFORMATIKA

DASAR-DASAR KODING

DAFTAR ISI

DAFTAR ISI ii
DAFTAR GAMBAR v
DAFTAR TABEL vi
MODUL 1 PENGENALAN ALGORTIMA 7
1.1. Tujuan Pembelajaran 7
1.2. Dasar Teori 7
1.3. Percobaan 9
MODUL 2 FLOWCHART DAN PSEUDOCODE 10
2.1. Tujuan Pembelajaran 10
2.2. Dasar Teori 10
2.3. Flowchart 10
2.4. Pseudocode 11
2.5. Pemprograman Visual dan Console 14
2.6. Percobaan Tugas Mahasiswa 15
MODUL 3 PENGENALAN BAHASA PEMPROGRAMAN PYTHON 16
3.1. Tujuan Pembelajaran 16
3.2. Teori Dasar Python 16
3.3. Pengenalan Python 17
3.4. Mengapa Python 17
3.5. Kelebihan dan Kekurangan Python 18
3.6. Proses Instalasi Python 19
3.7. Cara Menjalankan Python 22
3.8. Perbandingan Penulisan Code 23
3.9. Cara Kerja Python Interpreter dan Compiler 23
MODUL 4 STRUKTUR DATA PEMPROGRAMAN PYTHON 25
4.1. Pokok Pembahasan 25
4.2. Tujuan Pembelajaran 25
4.3. Dasar Teori 25
4.4. Tipe Data 25
4.5. Variabel 29
4.6. Operator 30

4.7. Model Penulisan Program 31
4.8. Percobaan 35
MODUL 5 PERCABANGAN 36
5.1. Struktur Percabangan (IF) 36
5.2. Struktur Percabangan *If/Else* 38
5.3. Struktur Percabangan IF / ELIF / ELSE 40
MODUL 6 PERULANGAN 42
4.1. Perulangan For 42
4.2. Perulangan while 43
MODUL 7 LIST DAN TUPLE 45
7.1. Definisi List 45
7.2. Cara Membuat List di Python 46
7.3. Cara Pemanggilan List 46
7.4. Program Menggunakan List 47
7.5. List Multidimensi 50
7.6. Latihan Membuat Program List 52
7.7. Definisi Tuple 52
7.8. Cara Membuat Tuple di Python 53
7.9. Membuat Tuple Kosong 53
7.10. Mengakses Nilai Tuple 54
7.11. Memotong Tuple 54
7.12. Mengambil Panjang Tuple 54
7.13. Tuple Nested 55
7.14. Sequence Unpacking 55
MODUL 8 DICTIONARY 57
8.1. Definisi Dictionary 57
8.2. Deklarasi Dictionary 57
8.3. Membuat Dictionary 58
8.4. Mengakses Item Pada Dictionary 59
8.5. Menambahkan Item Pada Dictionary 61
8.6. Mengubah Item Pada Dictionary 62
8.7. Menghapus Item Pada Dictionary 62

8.8. Mengambil Panjang Dictionary 63
MODUL 9 FUNGSI DAN PROSEDUR 64
9.1. Cara Membuat Fungsi pada Python 64
9.2. Fungsi dengan Parameter 65
9.3. Fungsi Mengembalikan Nilai 65
9.4. Variabel Global dan Lokal 66
9.5. Program Menggunakan Fungsi 67
DAFTAR PUSTAKA 72

DAFTAR GAMBAR

Gambar 1.1. Proses Dalam Pembuatan Program 9

DAFTAR TABEL

Tabel 4.1. Operator Aritmatika 30
Tabel 4.2. Operator Perbandingan 31
Tabel 4.3. Operator Penugasan 31

MODUL 1
PENGENALAN ALGORTIMA

1.1. Tujuan Pembelajaran

a. Mampu memahami suatu masalah dan mampu mencari solusi pemecahannya dan mampu menuangkan langkah-langkah pemecahan masalah tersebut dalam bentuk algoritma

b. Mampu manganalisa masalah dan menerjemahkannya dalam bentuk algoritma deskriptif

1.2. Dasar Teori

Algoritma adalah urutan langkah langkah logis yang menyatakan suatu tugas dalam menyelesaikan suatu masalah yang disusun secara sistematis atau bisa juga diartikan dengan urutan aksi-aksi yang jelas dan tidak rancu untuk menyelesaikan suatu masalah.

a. Ciri Algoritma

Ciri-ciri Algoritma :

1. Algoritma harus memiliki paling tidak satu keluaran
2. Masukan algoritma dapat nol (tidak ada) atau banyak masukan
3. Setelah selesai mengerjakan langkah-langkah penyelesaian masalah, algoritma harus berhenti
4. Setiap langkah yang dibuat harus sederhana dan efektif
5. Setiap langkah dalam algoritma harus didefinisikan dengan tepat dan jelas.

b. Notasi Algoritma

Algoritma dalam penulisannya memiliki aturan penulisan sendiri yang disebut dengan Notasi Algoritma.

Jenis-jenis Notasi Algoritma

a. Algoritma Deskriptif, yaitu langkah-langkah algoritma dengan rangkaian kalimat deskriptif.

Contoh :

- ✓ Mencari bilangan terbesar dari tiga buah bilangan yang di inputkan

  Jawab!

Kalimat deskriptif :

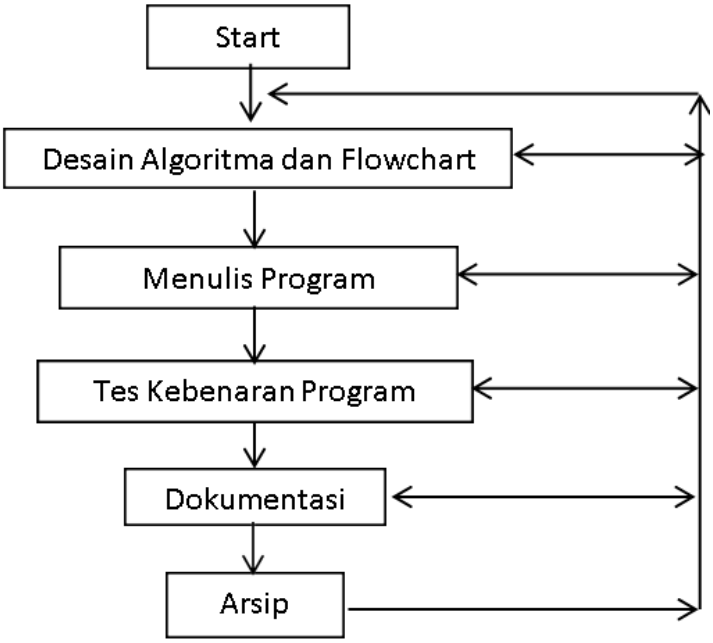
1. Masukkan sembarang bilangan sebanyak tiga buah
2. Ambil bilangan pertama dan set maksimum-nya sama dengan bilangan pertama.
3. Ambil bilangan kedua dan bandingkan dengan maksimum
4. Jika bilangan kedua lebih besar dari maksimum maka ubah maksimum-nya menjadi sama dengan bilangan kedua
5. Ambil bilangan ketiga dan bandingkan denganmaksimum
6. Jika bilangan ketiga lebih besar dari maksimum maka ubah maksimum-nya menjadi sama dengan bilangan ketiga
7. Variable maksimum akan berisi bilangan yang terbesar dan tampilkan hasilnya.

Algoritma deskriptif yang ditulis lebih sistematis adalah sebagai berikut :

1. Masukkan a,b,c
2. a → Maks
3. Jika b > maks, kerjakan langkah ke-4. Jika tidak kerjakan langkah ke-5
4. b → Maks
5. Jika c > maks, kerjakan langkah ke-6. Jika tidak kerjakan langkah ke-7
6. c → Maks
7. Tulis Maks

b. Bagan alir (flowchart) yaitu algoritma menggunakan bagan alir dengan memanfaatkan bentuk bentuk geometri.

Contoh Flowchart secara umum :



Gambar 1.1. Proses Dalam Pembuatan Program

1.3. Percobaan

Berikut ini terdapat beberapa persoalan, Pergunakanlah notasi algoritma deskriptif untuk menyelesaikannya dalam bentuk algoritma untuk pembuatan program:

1. Menjumlahkan 2 buah bilangan dan mencetak hasilnya
2. Memberikan pilihan untuk menghitung luas segitiga dan luas lingkaran
3. Menghitung rata-rata dari 3 buah data yang dimasukkan
4. Menentukan apakah umur yang di masukkan termasuk telah tua tau masih muda, dengan aturan jika lebih kecil dari 45 masih muda dan jika umur lebih besar dari 45 sudah tua
5. Mencari Faktor Persekutuan Terbesar (FPB) dari dua buah bilangan yang dimasukkan.

MODUL 2
FLOWCHART DAN PSEUDOCODE

2.1. Tujuan Pembelajaran

a. Mengenal dan memahami pemakaian simbol-simbol pada flowchart dan pseudocode
b. Mampu membuat flowchart dan pseudocode untuk memecahkan masalah
c. Mampu menganalisa masalah dan menerjemahkannya kedalam bentuk flowchart dan pseudocode
d. Mampu manganalisa masalah dan menerjemahkannya dalam bentuk flowchart dan psuedocode

2.2. Dasar Teori

Langkah yang umumnya dilakukan dalam proses pembuatan suatu program atau software adalah Mendefinisikan masalah dan menganalisanya. Dalam mendefinisikan masalah dan menganalisanya ini antara lain apa masukan dari permasalahan, keluaran dari permasalahan, proses dari masukan agar menjadi keluaran sebagai solusi permasalahan. Ketika pemrogram berfikir tentang proses, maka pemrogram akan berfikir parameter-parameter apa yang digunakan, kemudian menentukan metode atau algoritma apa yang akan diterapkan untuk menyelesaikan masalah tersebut dan terakhir menentukan bahasa program yang digunakan untuk membuat program.

2.3. Flowchart

Flow chart (diagram alir) adalah penggambaran secara grafik dari langkah- langkah pemecahan masalah yang harus diikuti oleh pemroses. Flow chart terdiri atas sekumpulan simbol, dimana masing-masing simbol menggambarkan suatu kegiatan tertentu. Flow chart diawali dengan penerimaan masukan (input), pemrosesan masukan, dan diakhiri dengan menampilkan hasilnya (output).

Adapun symbol-simbol flowchart :

	Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.		**Simbol Manual Input** Simbol untuk pemasukan data secara manual on-line keyboard
	Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan		**Simbol Preparation** Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.		**Simbol Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.		**Simbol Display** Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer		**Simbol disk and On-line Storage** Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer		**Simbol magnetik tape Unit** Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.		**Simbol Punch Card** Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya		**Simbol Dokumen** Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

Gambar 2.1. Simbol-simbol Flowchart

2.4. Pseudocode

Pseudo berarti imitasi atau mirip atau menyerupai. *Code* menunjukkan kode dari program. *seudocode* adalah kode yang mirip dengan instruksi kode program yang sebenarnya. Pseudocode berbasis pada bahasa pemrograman yang sesungguhnya seperti bahasa C, C++, Python, Pascal , sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada Programmer.

a. Gaya Penulisan Pseudocode

- Kata kunci (keywords) dan kata cadangan (reserved words) ditulis dengan huruf tebal atau kapital atau digaris bawahi dan kata-kata yang lainnya ditulis dengan huruf kecil.
- Kata kunci (if,then,else,repeat,until,for,do,while)yang membentuk struktur ditulis dengan menggunakkan huruf capital dan kata-kata yang tercantum di dalam kamus data ditulis dengan digaris bawah.

b. Pengenalan Struktur Dasar Pseudocode

- Struktur urut (*sequence structure*)

  Struktur ini terdiri dari sebuah instruksi atau blok dari instruksi yang tidak mempunyai perulangan atau keputusan di dalamnya. Struktur ini disebut juga struktur urut sederhana (simple sequence structure). Struktur ini semata-mata hanya berisi langkah-langkah yang berurut saja. Pseudocode juga menunjukkan proses membuka atau menutup file, meninisialisasi nilai awal dan lain sebagainya.

  Contoh:

  Baca data jam_kerja
  Hitung gaji = jam_kerja*tarif
  Tampilkan gaji di monitor

  Atau

  Read jam_kerja
  Let gaji = jam_kerja*tarif
  Print gaji

- Struktur keputusan (*decision structure*)

  Decision structure terdiri atas :

  1. IF
     IF Kondisi
     Tindakan

  2. IF - ELSE
     If kondisi
     else
     tindakan

  3. IF – ELIF - ELSE
     If kondisi Then
     tindakan-1
     elif
     tindakan 2
     else
     tindakan-3

4. Case

Select kasus

Case (nilai-1) Perform tindakan-1
Case (nilai-2) Perform tindakan-2
Case (nilai-3) Perform tindakan-3
……
Default Case Perform tindakan-n
Endselect

b. Struktur iterasi (*iteration structure*)

Iteration structure (struktur iterasi), ata loop structure (struktur perulangan), atau Repetition Structure (struktur repetisi) diterapkan pada situasi suatu instruksi atau group dari instruksi yang diproses berulangkali sampai kondisi yangdiinginkan sudah dipenuhi.

Struktur ini terdiri atas:

1. Do

   Do indeks = awal To akhir

   Perform tindakan

   End Do

2. Repeat

   Repeat

   Perform tindakan

   Until kondisi

3. Do - While

   While kondisi Do

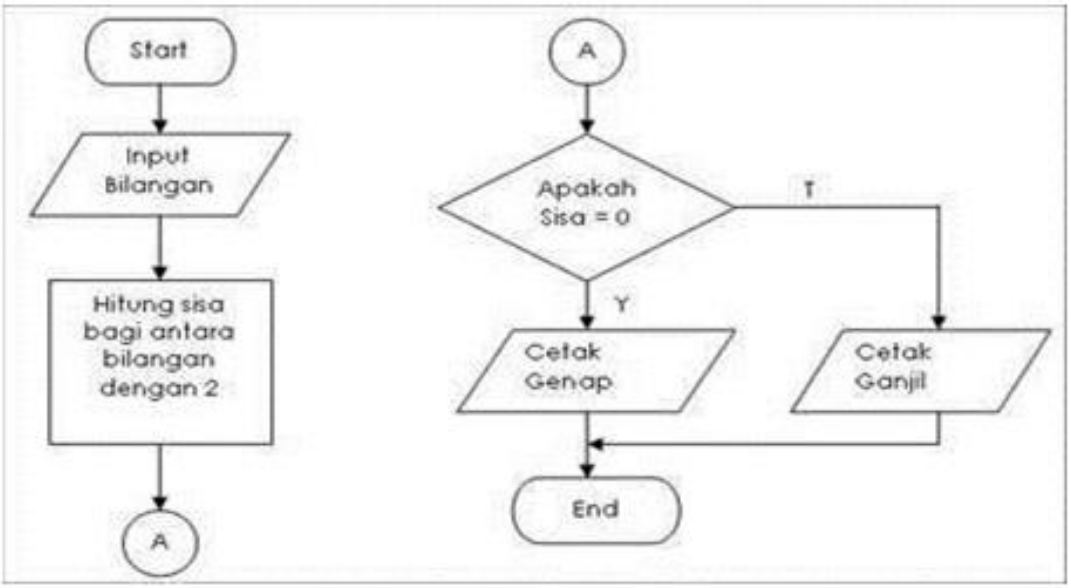
   Perform tindakan

   End While

Contoh Soal

Buatlah flowchart dan pseudocode untuk menentukan apakah suatu bilangan merupakan bilangan genap atau ganjil

Flowchart



Gambar 2.2. Flowchart untuk Menentukan Bilangan Genap atau Ganjil

Pseudocode

Read bilangan

If bil mod 2= 0

Then,

Write “genap”

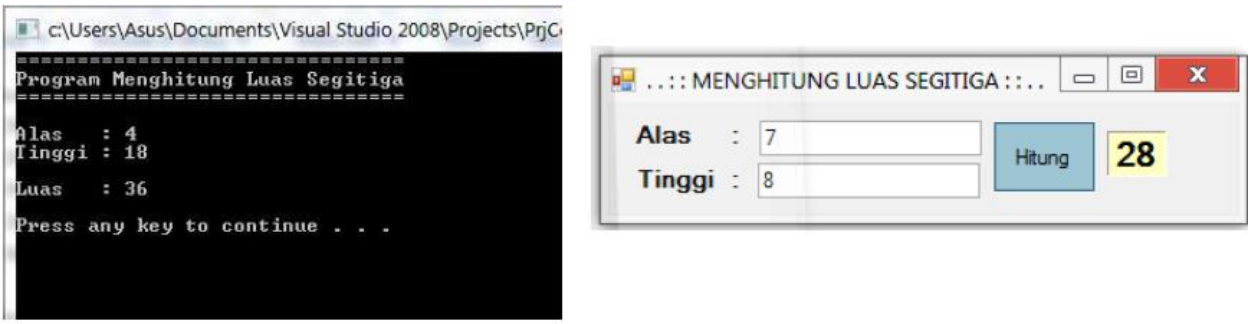
Else

Write “ganjil”

Endif

Penulisan Pseudocode sebaiknya juga mengikuti struktur penulisan bahasa pemrograman yang akan dipergunakan dalam membuat program agar programmer tidak mengalami kesulitan untuk memahami dan menerapkannya dalam bahasa pemrograman yang ditentukan.

2.5. Pemprograman Visual dan Console



Gambar 2.3. Tampilan Console (Kiri) dan Visual (kanan)

Gambar 2.3 memperlihatkan tampilan IDLE (*Integrated Development and Learning Environment*) yaitu Python sebagai lingkungan belajar berisi tampilan GUI yang menarik, bekerja pada OS (Windows, Linux dan Mac OS X), interaktif interpreter (penterjemah) berupa kode *input/output* dan *error messages, multi windows, multiple file (grep)* berupa *search within any windows, fiture debugger* (pencari kesalahan), konfigurasi/*browsers* dan dialog.

2.6. Percobaan Tugas Mahasiswa

Buatlah flowchart dan pseudocode dibawah ini :

a. Menjumlahkan 4 buah bilangan dan mencetak hasilnya
b. Memberikan pilihan untuk menghitung luas segitiga dan luas lingkaran
c. Menentukan apakah umur yang dimasukkan termasuk telah tua atau masih muda, dengan aturan jika umur lebih kecil dari 45 tahun masih muda dan jika umur lebih besar dari 45 tahun sudah tua.

MODUL 3
PENGENALAN BAHASA PEMPROGRAMAN PYTHON

3.1. Tujuan Pembelajaran

1. Mahasiswa dapat memahami sejarah perkembangan bahasa pemrograman Python.
2. Mahasiswa dapat mengetahui fitur-fitur penting yang terdapat pada Python.
3. Mahasiswa dapat mengetahui kelebihan dan kekurangan bahasa Python.
4. Mahasiswa dapat melakukan proses instalasi bahasa Python.
5. Mahasiswa mengerti konsep dan struktur bahasa pemrograman Python.
6. Mahasiswa mengerti konsep variabel, tipe data, dan operator pada Python.
7. Mahasiswa dapat membuat program sederhana menggunakan bahasa pemrograman Python.

3.2. Teori Dasar Python

Gambar 3.1. Logo 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 2.6.1 dan versi 3.0.

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.

3.3. Pengenalan Python

Python merupakan bahasa pemrograman dinamis yang mendukung pemrograman berbasis objek. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi. Saat ini skrip python dapat dijalankan pada sistem berbasis : Windows, Linux / Unix, Mac OS X, OS/2, Amiga. Python didistribusikan dengan beberapa lisensi yang berbeda dari beberapa versi. Lisensi Python tidak bertentangan baik menurut definisi Open Source maupun *General Public License (GPL)*. Interpreter Python dapat diperoleh diwebsite resminya di http://www.python.org

Python merupakan bahasa pemrograman tingkat tinggi yang diracik oleh Guido van Rossum. Python banyak digunakan untuk membuat berbagai macam program, seperti: program CLI, Program GUI (desktop), Aplikasi Mobile, Web, IoT, Game, Program untuk Hacking, dsb. Python juga dikenal dengan bahasa pemrograman yang mudah dipelajari, karena struktur sintaknya rapi dan mudah dipahami.

3.4. Mengapa Python

Sisi utama yang membedakan Python dengan bahasa lain adalah dalam hal aturan penulisan kode program. Bagi para programmer di luar python siap-siap dibingungkan dengan aturan indentasi, tipe data, tuple, dan dictionary. Python memiliki kelebihan tersendiri dibandingkan dengan bahasa lain terutama dalam hal penanganan modul, ini yang membuat beberapa programmer menyukai python. Selain itu python merupakan salah satu produk yang opensource, free, dan multiplatform.

Beberapa **fitur** yang dimiliki Python adalah :

- memiliki kepustakaan yang luas; dalam distribusi Python telah disediakan modulmodul
- siap pakai untuk berbagai keperluan.
- memiliki tata bahasa yang jernih dan mudah dipelajari.
- memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan
- kembali dan penulisan ulang kode sumber. berorientasi obyek.

- memiliki sistem pengelolaan memori otomatis (garbage collection, seperti java)
- modular, mudah dikembangkan dengan menciptakan modul-modul baru; modulmodul
- tersebut dapat dibangun dengan bahasa Python maupun C/C++.
- memiliki fasilitas pengumpulan sampah otomatis, seperti halnya pada bahasa
- pemrograman Java, python memiliki fasilitas pengaturan penggunaan ingatan
- komputer sehingga para pemrogram tidak perlu melakukan pengaturan ingatan
- komputer secara langsung.

3.5. Kelebihan dan Kekurangan Python

a. kelebihan bahasa Python antara lain :

- Tidak ada tahapan kompilasi dan penyambungan (link) sehingga kecepatan perubahan pada masa pembuatan system aplikasi meningkat.
- Tidak ada deklarasi tipe sehingga program menjadi lebih sederhana, singkat, dan fleksible.
- Manajemen memori otomatis yaitu kumpulan sampah memori sehingga dapat menghindari pencatatan kode.
- Tipe data dan operasi tingkat tinggi yaitu kecepatan pembuatan system aplikasi menggunakan tipe objek yang telah ada
- Pemrograman berorientasi objek
- Pelekatan dan perluasan dalam C
- Terdapat kelas, modul, eksepsi sehingga terdapat dukungan pemrograman skala besar secara modular
- Pemuatan dinamis modul C sehingga ekstensi menjadi sederhana dan berkas biner yang kecil
- Pemuatan kembali secara dinamis modul phyton seperti memodifikasi aplikasi tanpa menghentikannya
- Model objek universal kelas Satu
- Konstruksi pada saat aplikasi berjalan
- Interaktif, dinamis dan alamiah
- Akses hingga informasi interpreter
- Portabilitas secara luas seperti pemrograman antar platform tanpa ports

- Kompilasi untuk portable kode byte sehingga kecepatan eksekusi bertambah dan melindungi kode sumber
- Antarmuka terpasang untuk pelayanan keluar seperti perangkat Bantu system, GUI, persistence, database, dll

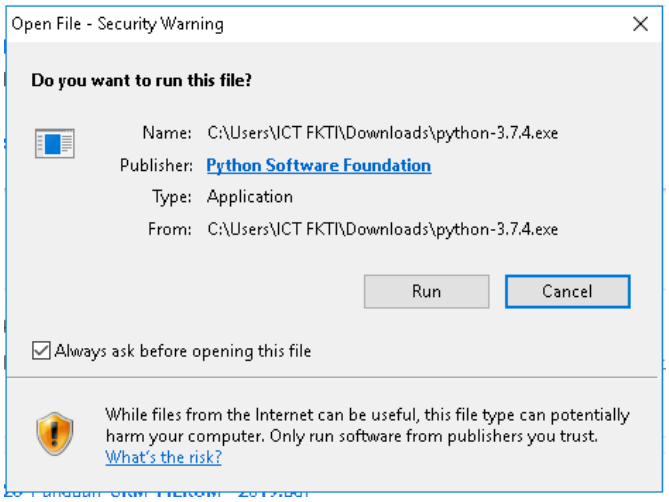
b. Beberapa kekurangan bahasa Python antara lain :

- Beberapa penugasan terdapat diluar dari jangkauan python, seperti bahasa pemrograman dinamis lainnya, python tidak secepat atau efisien sebagai statis, tidak seperti bahasa pemrograman kompilasi seperti bahasa C.
- Disebabkan python merupakan interpreter, python bukan merupakan perangkat bantu terbaik untuk pengantar komponen performa kritis.
- Python tidak dapat digunakan sebagai dasar bahasa pemrograman implementasi untuk beberapa komponen, tetapi dapat bekerja dengan baik sebagai bagian depan skrip antarmuka untuk mereka.
- Python memberikan efisiensi dan fleksibilitas tradeoff by dengan tidak memberikannya secara menyeluruh.
- Python menyediakan bahasa pemrograman optimasi untuk kegunaan, bersama dengan perangkat bantu yang dibutuhkan untuk diintegrasikan dengan bahasa pemrograman lainnya.

3.6. Proses Instalasi Python

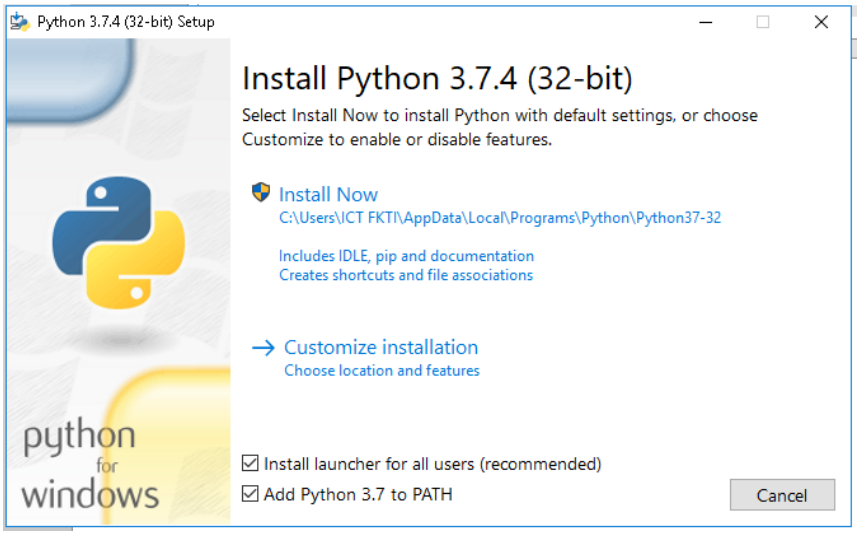
Proses instalasi python pada Windows dapat dilakukan langkah-langkah sebagai berikut :

1. Pilih software Python yang diinginkan, contoh memakai di modul ini menggunakan python versi 3.7 yang dapat di download pada alamat situs berikut https://www.python.org/ftp/python/3.7.4/python-3.7.4.exe. (Boleh menggunakan versi yang lebih tinggi dan sesuaikan dengan komputer)



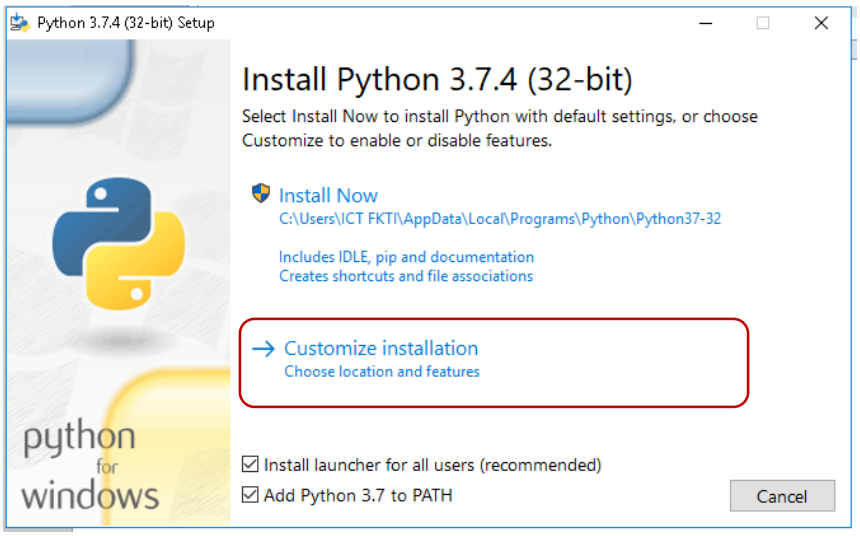
Gambar 3.2. Windows Installer Package

2. Klik software python running program kemudian lakukan peng-instalan pada computer dan ikuti langkah selanjutnya :



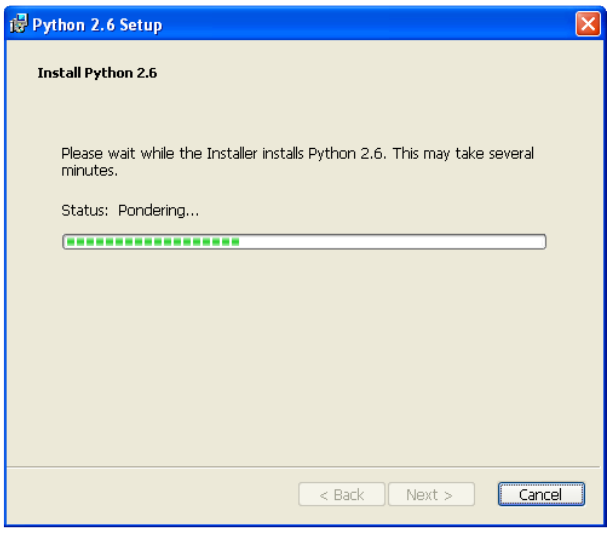
Gambar 3.3. Jendela Instalasi Python

3. Pilih direktori tujuan untuk tempat menyimpan program python, lalu klik tombol next. Jika kita klik tombol Disk Usage maka muncul form yang berisi informasi berapa besar kapasitas disk yang dibutuhkan/digunakan untuk menginstal python(sekitar 49 MB) :

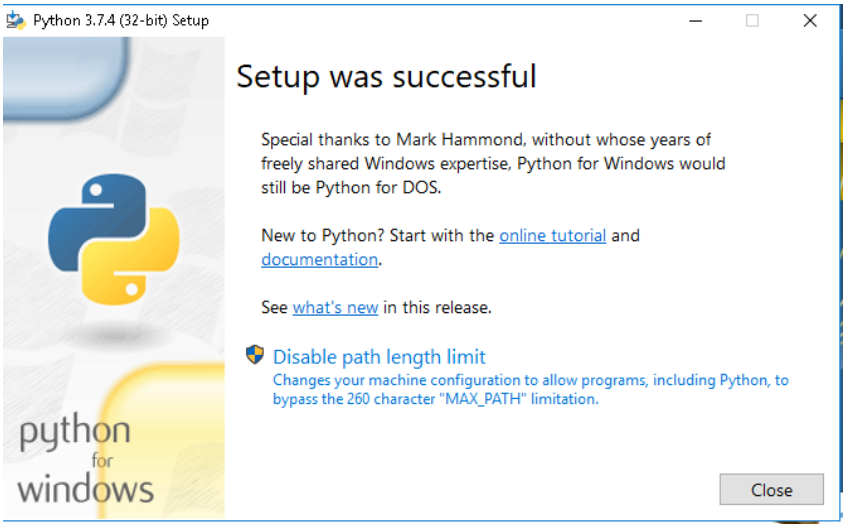


Gambar 3.4. Jendela Customize Python

4. Jika kita klik tombol advanced, pilih compile .py ke byte code setelah instalasi, jika tidak memilih juga tidak apa-apa. Tunggu beberapa menit selama proses instalasi berlangsung dan tekan finish, ikuti petunjuk selanjutnya :



Gambar 3.5. Jendela Proses Instalasi

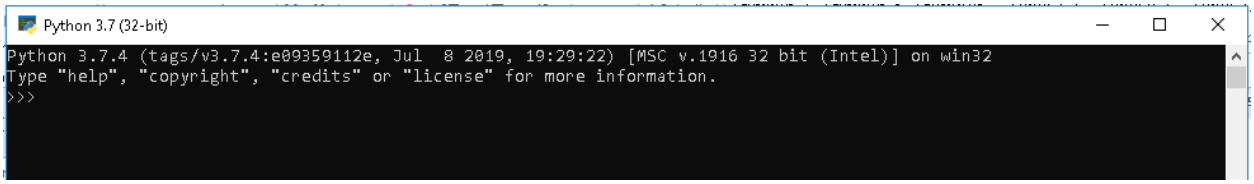


Gambar 3.6. Jendela Instalasi Python selesai dikerjakan

5. Selesai instalasi, komputer Anda telah ada Python, khususnya versi 3.7 pada folder \AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.7. Anda juga bisa mendownload langsung package **ANACONDA** lalu pergunakan IDE SPYDER untuk Python bagi beberapa komputer Package ANACONDA termasuk berat sehingga direkomendasikan IDLE untuk spek komputer dengan ram dan processor yang rendah.

3.7. Cara Menjalankan Python

Klik tombol start Program Python26 IDLE (Python GUI), IDLE(GUI-Integrated Development Environment) dengan tampilan sebagai berikut :



Gambar 3.7. Tampilan Awal Python

Pada window diatas,,didalam prompt (>>>), tuliskan : print instalasi python selesai. Kemudian interpreter merespon dengan menampilkan pada layar :

print("Instalasi Python telah Selesai.. Siap Ngoding")

Outputnya adalah sebagai berikut :

```
>>> print("Instalasi Python telah Selesai.. Siap Ngoding")
Instalasi Python telah Selesai.. Siap Ngoding
>>>
```

Gambar 3.8. Contoh menjalankan perintah sederhana pada python

3.8. Perbandingan Penulisan Code

C++ "Hello World"

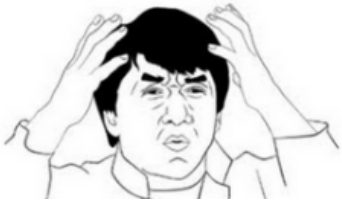
```
#include <iostream.h>
main()
{
cout << "Hello World! ";
}
return 0
```

Java "Hello World"

```
class HelloWorldApp
  {
    public static void main(String[] args)
        {
         System.out.println("Hello World!");
         }
    }
```

Python

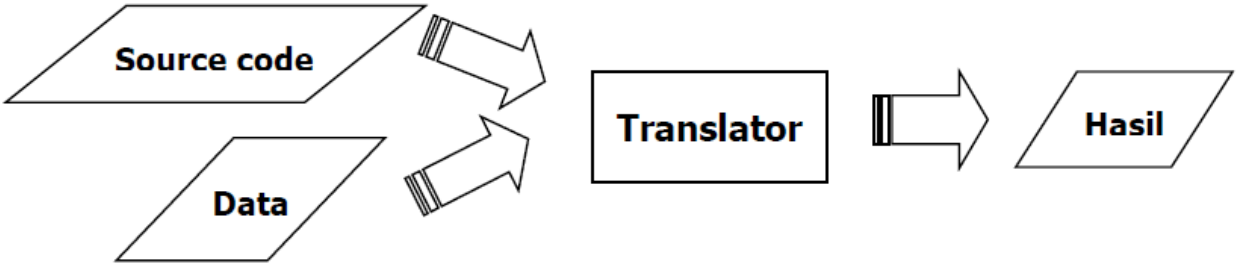
```
print "Hello world"
```



Gambar 3.9. Deklarasi Output Python yang sederhana

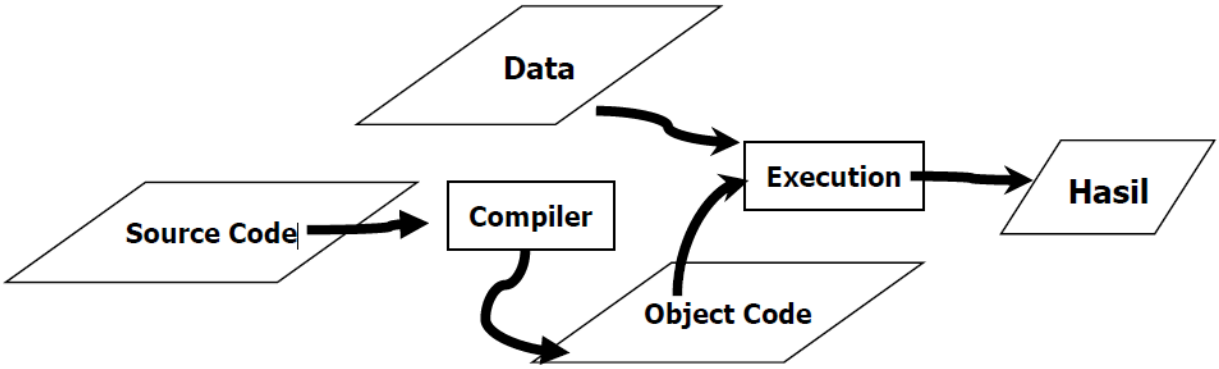
3.9. Cara Kerja Python Interpreter dan Compiler

Penterjemah bahasa python menggunakan interpreter (satu per-satu pernyataan), berbeda dengan penterjemah compiler yang menterjemahkan kode program sekaligus (blok pernyataan). Interpreter: Interpreter tidak menghasilkan bentuk *object code*, tetapi hasil translasinya hanya dalam bentuk internal, dimana program induk harus selalu ada-berbeda dengan *compiler*.



Gambar 3.10. Skema Proses Interpreter

Compiler : Source code adalah bahasa tingkat tinggi, object code adalah bahasa mesin atau bahasa assembly. Source code dan data diproses secara berbeda.



Gambar 3.11. Skema Proses Compiler

MODUL 4
STRUKTUR DATA PEMPROGRAMAN PYTHON

4.1. Pokok Pembahasan

1. Operator Aritmatika
2. Operator Perbandingan
3. Operator Logika
4. Operator Penugasan
5. Operator Bitwise

4.2. Tujuan Pembelajaran

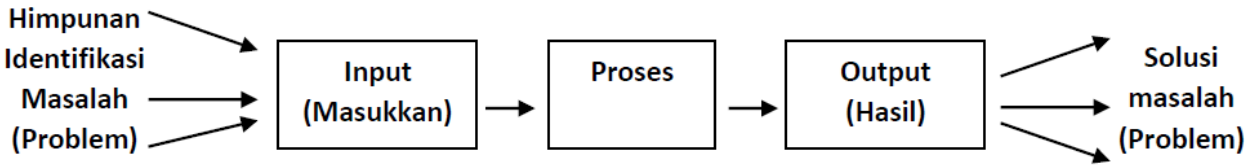
1. Mahasiswa mengetahui macam-macam operator serta fungsi dan penggunaannya
2. Mahasiswa dapat menerapkan fungsi operator dalam pemrograman

4.3. Dasar Teori

Operator adalah symbol-simbol khusus yang digunakan untuk mengoperasikan suatu nilai data (Operand). Operand adalah data-data yang akan dioperasikan oleh operator , ada 2 macam operator, yaitu Operator Unary dan Operator Binary. **Operator Unary** adalah operator yang operasinya hanya memerlukan satu buah operand, Misalnya : a++. Sedangkan **Operator Binary** adalah operator yang operasinya lebih dari satu operand Misalnya : a+b.

4.4. Tipe Data

Nilai (*value*) adalah hal yang paling mendasar seperti sebuah huruf, karakter khusus, atau sebuah angka yang akan dimanipulasi oleh program.



Gambar 4.1. Sistematis Pemecahan Masalah

Beberapa tipe data pada python, diantaranya :

1. **Number**

   Tipe data Number merepresentasikan nilai-nilai berupa angka. Python menggolongkan beberapa tipe data umum seperti, Integer (bilangan bulat) dan Floating-point (bilangan desimal) ke dalam tipe data Number.

   Contoh :

   print (100 + 200)

   Output : 300

   print (5.21 + 6.234)

   Output : 11.443999999999999

   Operator penugasan (=) digunakan untuk memasukkan nilai kedalam variabel. Tidak ada hasil yang akan muncul sampai statemen selanjutnya.

   tambah = 100 + 50

   print (tambah)

   Output : 150

   kali = 100 * 50

   print (kali)

   Output : 5000

   bagi = 50 / 5

   print (bagi)

   Output : 5000

   Nilai dapat di masukkan kedalam beberapa variabel secara simultan.

   a = b = c = 20

   print (a)

   print (b)

   print (c)

Tipe data angka dibagi menjadi beberapa jenis lagi:

1. int (Integer): bilangan bulat, contoh 32, 22, 12, 10, dsb.

2. float: bilangan pecahan, contoh 1.3, 4.2, 22.3, dsb.

   Contoh:

   harga = 15000 #tipe int

   berat = 28.12 #float

   jarak = 3e3 #float 3000.0, huruf e artinya eksponen 10

2. String

Selain angka, python juga mampu melakukan manipulasi string, yang dapat di ekspresikan dengan beberapa cara. Penulisan nilai string pada python menggunakan tanda petik satu (') atau tanda petik dua (“). Contohnya :

```
print ("Selamat Datang di Python")
print ('Selamat Datang di Python')
```

String literal juga dapat menggabungkan beberapa baris dalam berbagai cara. Dengan menggunakan operator (\n) di akhir kalimat untuk menyambung kalimat selanjutnya yang berada di baris selanjutnya.

```
kalimat1 = “Saat ini saya belajar Python \n”
kalimat2 = “belajar Python Menyenangkan \n”
print (kalimat1)
print (kalimat2)
```

Penulisan string untuk multiple line juga dapat dilakukan dengan menggunakan tanda petik dua atau satu sebanyak 3 kali, (“ “ “ atau ' ' ').

```
print ('''Selamat
      Datang
         di Python''')
```

Untuk menggabungkan dua buah string atau lebih dapat dilakukan dengan dua cara. Pertama, dengan menulis langsung dua buah string yang diapit dengan tanda kutip atau dengan penggunaan operator tambah (+).

```
print ('Selamat Datang' 'di Program Python Dasar')
```

Output : Selamat Datangdi Program Python Dasar

```
print ('Selamat Datang'+'di Program Python Lanjut')
```

Output : Selamat Datang di Program Python Lanjut

Sebuah string, setiap karakternya dapat di index, seperti pengindexan pada bahasa C. Karakter pertama pada sebuah string berindex 0, karakter ke-dua berindex 1 dan seterusnya.

```
kalimat = “Belajar Pemrograman Python"
print(kalimat[0])
```

Output : B

```
print(kalimat[5])
```

Output : a

```
print(kalimat[0:5])
```

Output : Belaj

```
print(kalimat[9:18])
```

Output : emrograma

```
print(kalimat[:9])
```

Output : Belajar P

```
print(kalimat[4:])
```

Output : jar Pemrograman Python

3. Boolean

Tipe data *boolean* adalah tipe data yang hanya memiliki dua nilai yaitu True dan False atau 0 dan 1. Penulisan True dan False, huruf pertamnya harus kapital dan tanpa tanda petik.

Contoh:

```
bergerak = True
nyala = 1 #sebenarnya tipenya int, tapi bisa juga menjadi bool
```

4. Fungsi untuk mengubah tipe data

1. int() untuk mengubah menjadi integer;
2. long() untuk mengubah menjadi integer panjang;

3. float() untuk mengubah menjadi float;
4. bool() untuk mengubah menjadi boolean;
5. chr() untuk mengubah menjadi karakter;
6. str() untuk mengubah menjadi string.
7. bin() untuk mengubah menjadi bilangan Biner.
8. hex() untuk mengubah menjadi bilangan Heksadesimal.
9. oct() untuk mengubah menjadi bilangan okta.

4.5. Variabel

Variabel merupakan simbol yang mewakili nilai tertentu. Pembuatan variabel dalam python sangat sederhana. Berikut adalah ketentuan mengenai variabel dalam pyton,

- Variabel tidak perlu dideklarasikan mempunyai tipe data tertentu
- Jenis data dalam variabel dapat berubah-ubah
- Penulisan variabel harus diawali dengan huruf, dan untuk karakter selanjutnya bisa berupa huruf atau angka
- Penulisan variabel tidak boleh dipisah oleh <spasi>
- Untuk variabel terdiri dari 2 suku kata, dapat dipisah dengan simbol underscore (_)

Aturan Penulisan Variabel

1 Nama variabel boleh diawali menggunakan huruf atau garis bawah (_), contoh: nama, _nama, namaKu, nama_variabel.

2 Karakter selanjutnya dapat berupa huruf, garis bawah (_) atau angka, contoh: __nama, n2, nilai1.

3 Karakter pada nama variabel bersifat sensitif (*case-sensitif*). Artinya huruf besar dan kecil dibedakan. Misalnya, variabel_Ku dan variabel_ku, keduanya adalah variabel yang berbeda.

4 Nama variabel tidak boleh menggunakan kata kunci yang sudah ada dalam python seperti if, while, for, dsb

Statemen yang tidak boleh dijadikan nama variabel :

and	continue	else	for	import	not	raise
assert	def	except	from	in	or	return
break	del	exec	global	is	pass	try
class	elif	finally	if	lambda	print	while

Menghapus Variabel

Ketika sebuah variabel tidak dibutuhkan lagi, maka kita bisa menghapusnya dengan fungsi del().

```
kata = "Belajar Coding Python Unmul"
print kata
#output Belajar Coding Python Unmul
del(kata)
print kata
#Error karena variabel kata telah dihapus
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'kata' is not defined
```

Pada perintah terakhir, kita akan mandapatkan NameError. Artinya variabel tidak ada di dalam memori alias sudah dihapus.

4.6. Operator

Operator dalam Python dibagi menjadi 3 bagian, yaitu :

- operator aritmatika : +, - , *, /, %
- operator perbandingan : >= , <= , != , > , < , ==
- operator penugasan : *= , /= , %= , +=, -=

1. Operator Aritmatika

Tabel 4.1. Operator Aritmatika

Operator	Deskripsi	Contoh	Hasil
*	Perkalian	7 * 3	21
/	Pembagian	7 / 3	2
%	Modulus	7 % 3	1
+	Penjumlahan	7 + 3	10
-	Pengurangan	7 - 3	4

2. Operator Perbandingan

Tabel 4.2. Operator Perbandingan

Operator	Deskripsi	Contoh	Hasil
>=	Lebih besar atau sama dengan	7 >= 9	FALSE
<=	Lebih kecil atau sama dengan	3 <= 8	TRUE
!=	Tidak sama dengan	1 != 10	TRUE
<	Lebih kecil	14 < 6	FALSE
>	Lebih besar	5 > 3	TRUE
==	Sama dengan	4 == 4	TRUE

3. Operator Penugasan

Tabel 4.3. Operator Penugasan

Operator	Contoh	Sama dengan
*=	x *= 100	x = x * 100
/=	x /= 100	x = x / 100
%=	x %= 100	x = x % 100
+=	x += 100	x = x + 100
-=	x -= 100	x = x – 100

4.7. Model Penulisan Program

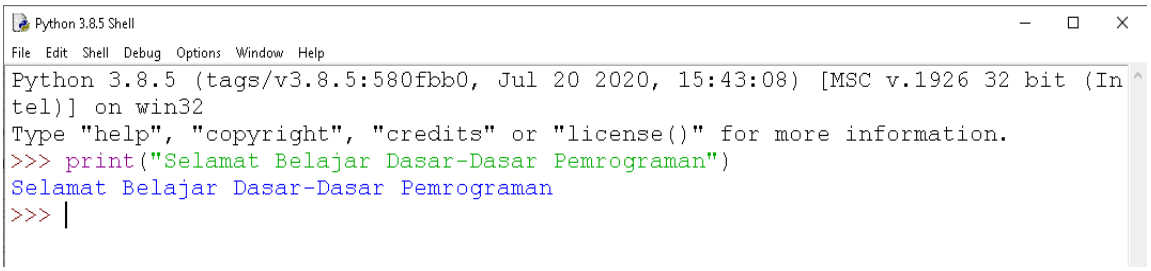
a. Contoh Program Sederhana Menggunakan Python

Modus penulisan kode python dapat dilakukan dengan dua cara, yaitu

1. Menggunakan mode interaktif
2. Menggunakan Skrip.

1. **Mode Interaktif**

Menggunakan mode interaktif berarti kita bekerja menggunakan Prompt interpreter dari python. Penulisan kode python dilakukan per-statemen, contohnya,



Gambar 4.2. Mode Interaktif

Statemen "Selamat Belajar Dasar-Dasar Pemrograman" adalah hasil eksekusi dari perintah "print "Selamat Belajar Dasar-Dasar Pemrograman"". Dengan menggunakan prompt interpreter python, kita dapat melakukan uji coba setiap statemen- statemen atau fungsi-fungsi yang ada pada Python.

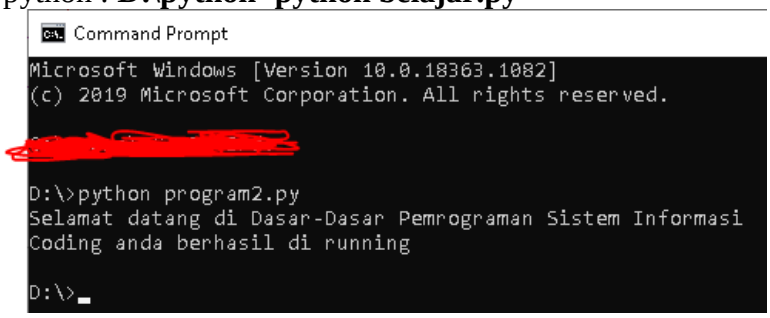
2. **Mode Skrip**

Menggunakan skrip berarti menyusun statemen-statemen menjadi sebuah satu kesatuan file python. Dengan membuat skrip berarti kita melakukan kompilasi file python dengan bantuan Interpreter dari Python lewat Command Prompt (Windows) atau Terminal (Linux/Unix). Contohnya :

Contoh Kasus 1.
program 1. belajar.py

```
# Program 1
print ("Selamat datang di Dasar-Dasar Pemrograman Sistem Informasi")
print ("Coding anda berhasil di running")
```

Tulis skrip diatas pada Text Editor, kamudian lakukan kompilasi,
Lokasi file python : **D:\python>python belajar.py**



Gambar 4.3. Tampilan Output program belajar.py

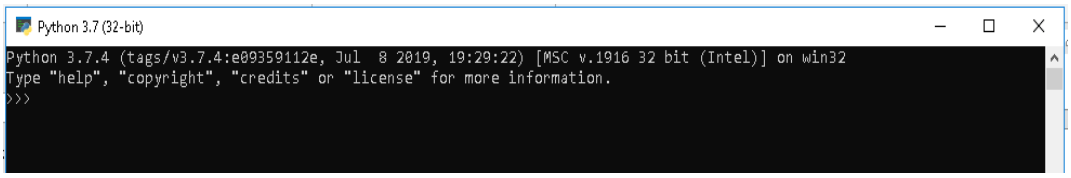
Contoh Kasus 2 :

Buatlah program menampilkan hasil penjumlahan dua buah bilangan menggunakan tipe data dan operator pada python. Output yang akan ditampilkan adalah sebagai berikut :

```
Ini adalah program yang ditulis dengan bahasa Python
Berikut ini contoh program penjumlahan
hasil = a + b
hasil = 10 + 20
hasil = 30
```

Langkah-langkah pengerjaan adalah sebagai berikut :

1. Klik tombol start Program Python 3.7 app dengan tampilan sebagai berikut :



Gambar 4.4. Tampilan Python 3.7

2. Klik Menu File -> New Window lalu ketikkan listing program sebagai berikut.

   # Program Contoh 2

   print ("Coding Anda berhasil di running")

   a = 10

   b = 30

   hasil = a + b

   print ("Hasil a + b")

   print ("Hasil = %d + %d" % (a,b))

   print ("Hasil = %d " % (hasil))

3. Setelah selesai mengetikkan code, langkah selanjutnya menyimpan file tersebut dengan cara klik menu File -> Save As. Masukkan nama file dengan nama program1.py

```
print ("Coding anda berhasil di running")
a = 10
b = 30
hasil = a + b
print ("Hasil a + b")
print ("Hasil = %d + %d" % (a,b))
print ("Hasil = %d " % (hasil))
```

Gambar 4.5. Tampilan Penulisan Program

4. Setelah itu menjalankan program dengan cara klik menu Run -> Run Module atau dengan menekan tombol F5.

```
Coding anda berhasil di running
Hasil a + b
Hasil = 10 + 30
Hasil = 40
```

Gambar 4.6. Tampilan Hasil Output

5. Selain itu kita juga dapat menjalankan program pada command prompt dengan cara mengetikkan D:\python> python program1.py (berdasarkan file python berada di drive)

```
Coding anda berhasil di running
Hasil a + b
Hasil = 10 + 30
Hasil = 40

D:\python>
```

Gambar 4.7. Tampilan menjalankan python pada command prompt

Contoh Kasus 3 :

Buatlah program menginput dan menampilkan kalimat menggunakan bahasa python.

Langkah-langkah pengerjaan adalah sebagai berikut :

1. Klik tombol start Program Python 3.7 app dengan tampilan sebagai berikut :

```
Python 3.7 (32-bit)
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 4.8. Tampilan Python 3.7

2. Klik Menu File -> New Window lalu ketikkan listing program sebagai berikut.

```
# Program Contoh 3
praktikum = "Dasar-Dasar Pemrograman"
nama = input("Masukkan Nama : ")
nim = input("Masukkan NIM : ")
prodi = input("Masukkan Prodi : ")
print()
print ("Praktikum : ", praktikum)
```

3. Setelah selesai mengetikkan code, langkah selanjutnya menyimpan file tersebut dengan cara klik menu File -> Save As. Masukkan nama file dengan nama program1.py

```
temp.py   untitled0.py   program2.py
1 # Program Contoh 3
2 praktikum = "Algoritma dan Pemprograman Dasar"
3 nama = input("Masukkan Nama : ")
4 nim = input("Masukkan NIM : ")
5 prodi = input("Masukkan Prodi : ")
6 print()
7 print ("Praktikum : ", praktikum)
8
9
```

Gambar 4.9. Tampilan Penulisan Program

4. Setelah itu menjalankan program dengan cara klik menu Run -> Run Module atau dengan menekan tombol F5.

```
Masukkan Nama : Muhammad Jamil

Masukkan NIM : 2019849393

Masukkan Prodi : Teknik Informatika

Praktikum :  Algoritma dan Pemprograman Dasar
```

Gambar 4.10. Tampilan Hasil Output

5. Selain itu kita juga dapat menjalankan program pada command prompt dengan cara mengetikkan D:\python> python program2.py

```
D:\python>python program2.py
Masukkan Nama : Hario Jati Setyadi
Masukkan NIM : 201984848
Masukkan Prodi : Sistem Informasi

Praktikum :  Algoritma dan Pemprograman Dasar

D:\python>
```

Gambar 4.11. Tampilan menjalankan python pada command prompt

4.8. Percobaan

Buatlah program dibawah ini dengan menggunakan operator :

1 Perhitungan luas dari bangun ruang kubus

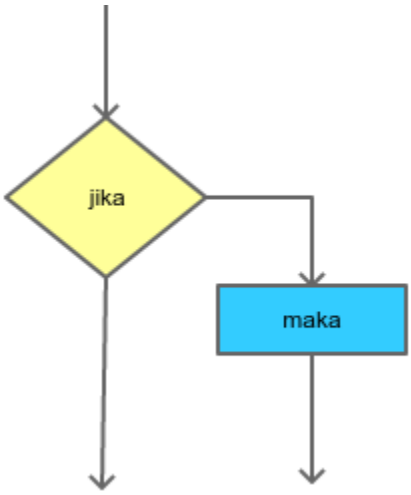
2 Perhitungan volume dari bangun ruang balok

3 Perhitungan rumus fisika dari energi potensial (ep)

4 Konversi dolar ke rupiah

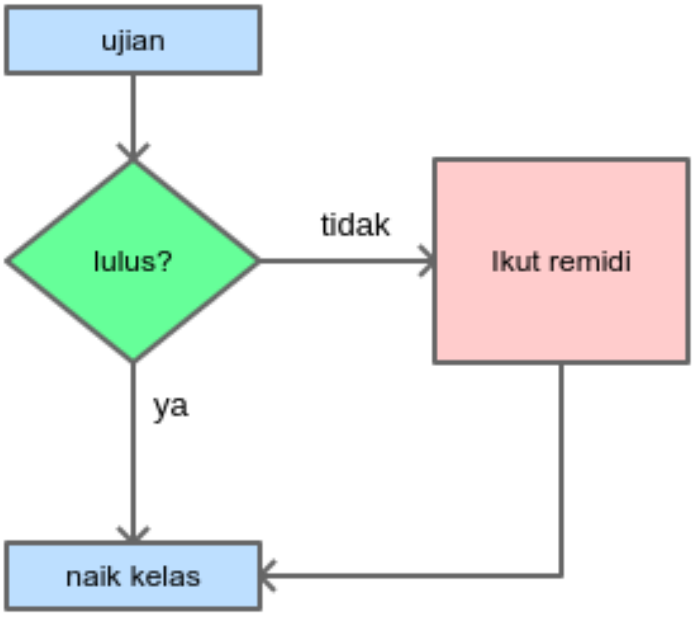
MODUL 5
PERCABANGAN

5.1. Struktur Percabangan (IF)



Gambar 5.1. Logika Percabangan

Selain percabangan, struktur ini juga disebut control flow, decision, struktur kondisi, Struktur if. Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang kita berikan. Percabangan If digunakan saat terdapat satu pilihan keputusan. Misalkan, kalau kita tidak lulus dalam ujian, maka kita ikut remidi. Sedangkan kalau lulus tidak perlu ikut remidi.



Gambar 5.2. Flowchart Program Percabangan

Maka kita bisa membuat kode-nya seperti ini:

```
if lulus == "tidak":
    print("kamu harus ikut remidi")
```

"Jika lulus == "tidak" maka cetak teks "kamu harus ikut remidi"" Kita menggunakan operator relasi sama dengan (==) untuk membandingkan isi variabel lulus. Sedangkan tanda titik-dua (:) adalah tanda untuk memulai blok kode *If*.

Contoh penulisan yang salah:

```
if lulus == "tidak":
print("Kamu harus ikut remidi")
```

Contoh penulisan yang benar:

```
if lulus == "tidak":
    print("kamu harus ikut remidi")
```

Perhatikan pada adanya jarak atau TAB pada codingan if yang artinya kondisi apabila terpenuhi akan menjalankan program sesuai perintah.

Buatlah Program Berikut
Contoh Program 1.

```
# lulus.py
lulus = input("Apakah kamu lulus? [ya/tidak]: ")

if lulus == "tidak":
   print("Kamu harus ikut ujian")
```

Output :

```
Apakah kamu lulus? [ya/tidak]: tidak
Kamu harus ikut ujian

In [72]: runfile('D:/python/program2.py', wdir='D:/python')

Apakah kamu lulus? [ya/tidak]: ya

In [73]:
```

Pada Output diatas dapat dilihat skema percabangan berjalan dengan baik di mana saat pengguna menjawab **tidak** maka secara otomatis terlihat output : kamu harus ikut remidi. Sedangkan saat di isi dengan ya maka sistem tidak memberikan output sesuai yang ditentukan. Untuk dapat menjalankan perintah maka ditambahkan Else yang akan dibahas sesi ini.

Contoh Program 2

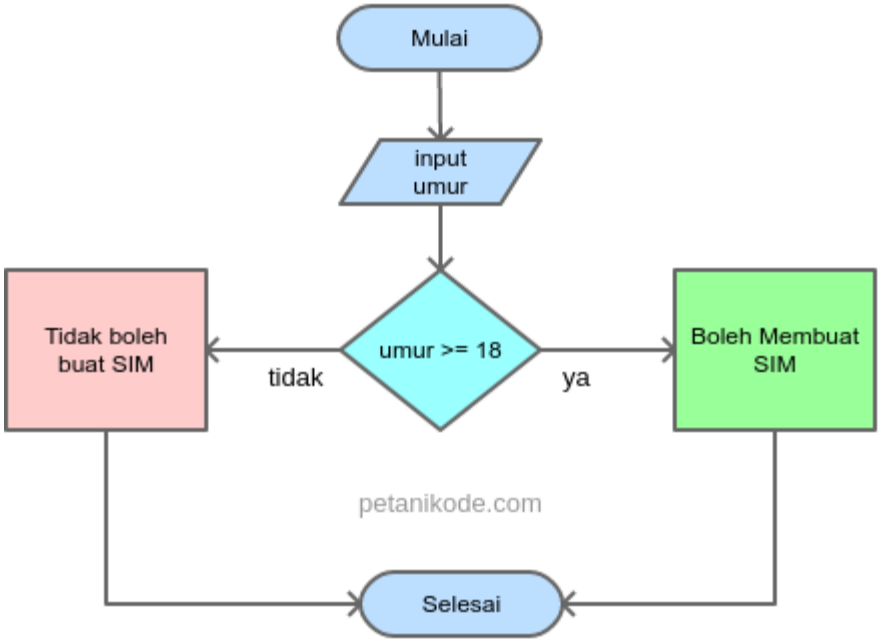
```
# program untuk mengecek bonus dan diskon
# file: bonus.py

total_belanja = input("Total belanja: Rp ")

# jumlah yang harus dibayar adalah berapa total belanjaannya
# tapi kalau dapat diskon akan berkurang
bayar = int(total_belanja)

# jika dia belanja di atas 100rb maka berikan bonus dan diskon
if int(total_belanja) > 100000:
    print("Selamat Karena Belanja > Rp. 100.000 anda mendapat Voucher Makan ")
    print("dan diskon 5%")

    # hitung diskonnya
    diskon = int(total_belanja) * 5/100 #5%
    bayar = int(total_belanja) - diskon

# cetak struk
print("Total yang harus dibayar: Rp %s" % bayar)
print("Terima kasih sudah berbelanja di Toko Kami")
print("Kami Tunggu kunjungan anda selanjutnya.")
```

Output :

```
Total belanja: Rp 500000
Selamat Karena Belanja > Rp. 100.000 anda mendapat Voucher Makan
dan diskon 5%
Total yang harus dibayar: Rp 475000.0
Terima kasih sudah berbelanja di Toko Kami
Kami Tunggu kunjungan anda selanjutnya.
```

5.2. Struktur Percabangan *If/Else*

Percabangan If/Else digunakan saat terdapat dua pilihan keputusan. Misalkan, jika umur diatas atau samadengan 18 tahun boleh membuat SIM. Sedangkan dibawah itu belum boleh.



Gambar 5.3. Flowchart Percabangan IF - Else

Maka Sourcecode Programnya adalah :

```
# cek_umur.py
umur = input("Berapa umur kamu: ")
if int(umur) >= 18:
    print("Kamu boleh membuat SIM")
else:
    print("Kamu belum boleh membuat SIM")
```

Selain blok *If*, terdapat juga blok *Else* yang akan dieksekusi apabila kondisi umur >= 18 salah (False). Hasil eksekusi dari kode di atas adalah sebagai berikut:

```
Berapa umur kamu: 19
Kamu boleh membuat SIM

In [85]: runfile('D:/python/program2.py', wdir='D:/python')

Berapa umur kamu: 15
Kamu belum boleh membuat SIM
```

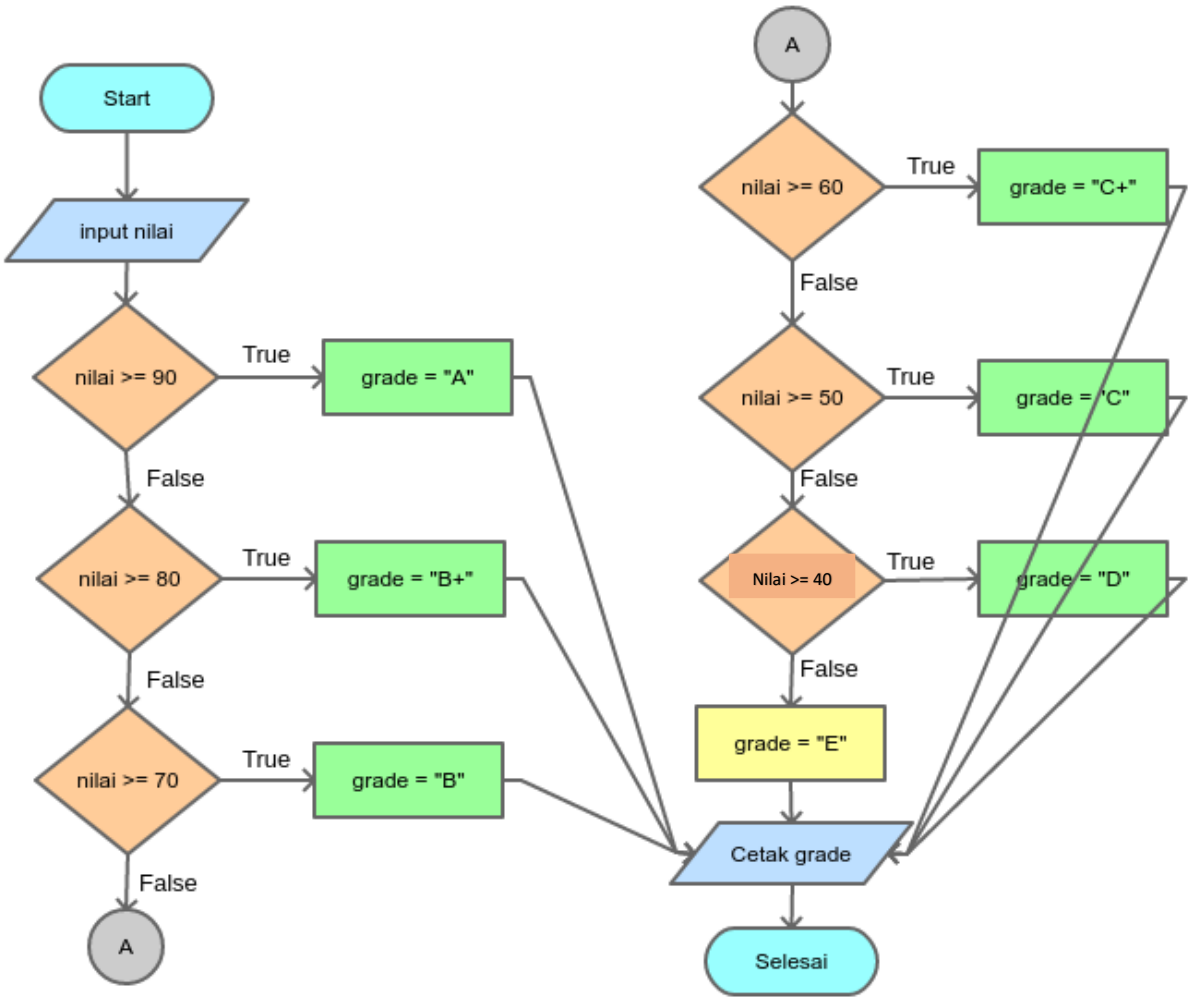
5.3. Struktur Percabangan IF / ELIF / ELSE

Percabangan *If/Elif/Else* digunakan apabila terdapat lebih dari dua pilihan keputusan.

```
if begini:
   maka ini
elif begitu:
   maka itu
else:
   pokoknya gitu dah!
```

Kata kunci elif artinya *Else if*, fungsinya untuk membuat kondisi/logika tambahan apabila kondisi pertama salah.

Contoh Program:
Misalkan kita akan membuat program untuk menentukan grade nilai dengan *flow chart* sebagai berikut:



Gambar 5.4. Flowchard Diagram Program Grade Nilai

Maka kode programnya bisa kita buat seperti ini:

```
nilai = input("Inputkan nilai Anda : ")
if nilai >= "90":
   grade = "Selamat Anda Mendapat Grade A"
elif nilai >= "80":
   grade = "Selamat Anda Mendapat Grade B+"
elif nilai >= "70":
   grade = "Selamat Anda Mendapat Grade B"
elif nilai >= "60":
   grade = "Selamat Anda Mendapat Grade C+"
elif nilai >= "50":
   grade = "Selamat Anda Mendapat Grade C"
elif nilai >= "40":
   grade = "Selamat Anda Mendapat Grade D"
else:
   grade = "Anda Tidak Lulus"
print("Hasil: %s" % grade)
```

Maka hasilnya :

```
In [91]: runfile('D:/python/program2.py', wdir='D:/python')

Inputkan nilai Anda : 88
Hasil: Selamat Anda Mendapat Grade B+

In [92]: runfile('D:/python/program2.py', wdir='D:/python')

Inputkan nilai Anda : 96
Hasil: Selamat Anda Mendapat Grade A

In [93]: runfile('D:/python/program2.py', wdir='D:/python')

Inputkan nilai Anda : 0
Hasil: Anda Tidak Lulus
```

MODUL 6
PERULANGAN

Perulangan dalam bahasa pemrograman berfungsi menyuruh komputer melakukan sesuatu secara berulang-ulang. Terdapat dua jenis perualangan dalam bahasa pemrograman python, yaitu perulangan dengan for dan while. Perulangan for disebut *counted loop* (perulangan yang terhitung), sementara perulangan while disebut *uncounted loop* (perulangan yang tak terhitung). Perbedaannya adalah perulangan for biasanya digunakan untuk mengulangi kode yang sudah diketahui banyak perulangannya. Sementara while untuk perulangan yang memiliki syarat dan tidak tentu berapa banyak perulangannya.

4.1. Perulangan For

Contoh Program :

```
# file: perulanganFor.py
ulang = 25
for i in range(ulang):
    print ("Perulangan ke-"+str(i))
```

Output :

```
In [96]: runfile('D:/python/program2.py', wdir='D:/python')
Perulangan ke-0
Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-4
Perulangan ke-5
Perulangan ke-6
Perulangan ke-7
Perulangan ke-8
Perulangan ke-9
Perulangan ke-10
Perulangan ke-11
Perulangan ke-12
Perulangan ke-13
Perulangan ke-14
Perulangan ke-15
Perulangan ke-16
Perulangan ke-17
Perulangan ke-18
Perulangan ke-19
Perulangan ke-20
Perulangan ke-21
Perulangan ke-22
Perulangan ke-23
Perulangan ke-24
```

Penjelasan :

Pertama kita menentukan banyak perulangannya sebanyak 10 kali

ulang = 10

Variabel i berfungsi untuk menampung indeks, dan fungsi range() berfungsi untuk membuat list dengan range dari 0-10. Fungsi str() berfungsi merubah tipe data ineger ke string.

```
for i in range(ulang):
   print ("Perulangan ke-"+str(i))
```

Contoh lain menggunakan List :

```
# berkas: perulanganFor.py
item = ['Nasi Kuning','Pecel Lele','Nasi Padang','Ikan Haruan']

for isi in item:
    print (isi)
```

Output :

```
In [98]: runfile('D:/python/program2.py', wdir='D:/python')
Nasi Kuning
Pecel Lele
Nasi Padang
Ikan Haruan
```

4.2. Perulangan while

Bentuk umum:

```
while(True):
   # jalankan kode ini
# kode ini berada di luar perulangan while
```

Contoh:

```
# berkas: perulanganWhile.py
jawab = 'ya'
hitung = 0

while(jawab == 'ya'):
    hitung += 1
    jawab = input("Apakah Anda Mau Mengulang ya/tidak ? ")
print ("Total perulagan: " + str(hitung))
```

Atau bisa juga dengan bentuk yang seperti ini, dengan menggunakan kata kuci break

```
# berkas: perulanganWhile.py
jawab = 'ya'
hitung = 0

while(True):
    hitung += 1
    jawab = input("Apakah Anda Ingin Mengulang ya/tidak ? ")
    if jawab == 'tidak':
        break
print ("Total perulangan: " + str(hitung))
```

Pertama menentukan variabel untuk menghitung, dan menentukan kapan perulangan berhenti. kalau pengguna menjawab tidak maka perulangan akan terhenti.

```
jawab = 'ya'
hitung = 0
```

Melakukan perulangan dengan *while*, kemudian menambah satu variabel hitung setiap kali mengulang. lalu menanyakan kepada pengguna, apakah mau berhenti mengulang atau tidak?

```
while(jawab == 'ya'):
  hitung += 1
  jawab = input("Apakah Anda Ingin Mengulan ya/tidak ? ")
```

Setelah selesai mengulang, cetak berapa kali perulangan tersebut terjadi

```
print ("Total perulagan: " + str(hitung))
```

Hasil Luaran (**Output)** :

```
In [100]: runfile('D:/python/program2.py', wdir='D:/python')

Apakah Anda Mau Mengulang ya/tidak ? ya

Apakah Anda Mau Mengulang ya/tidak ? ya

Apakah Anda Mau Mengulang ya/tidak ? ya

Apakah Anda Mau Mengulang ya/tidak ? ya

Apakah Anda Mau Mengulang ya/tidak ? Tidak
Total perulagan: 5
```

MODUL 7
LIST DAN TUPLE

List adalah struktur data pada python yang mampu menyimpan lebih dari satu data, seperti array. Pada modul ini akan membahas cara menggunakan list di Python dari yang paling sederhana sampai yang sedikit kompleks.

Yang dipelajari pada modul List ini :

- Cara Membuat List dan Mengisinya
- Cara Mangambil nilai dari List
- Cara Menambahkan dan Menghapus isi List
- Operasi pada List
- List multi dimensi

7.1. Definisi List

List sering disebut juga array pada bahasa pemrograman lain. List adalah jenis data campuran yang bisa memiliki komponen penyusun yang berbeda-beda. Sebuah list dapat dibuat dengan menggunakan tanda kurung siku [ ]. Anggota list didaftar dalam kurung siku tersebut dan masing-masing dipisahkan oleh tanda koma. Sifat-sifat list bisa didaftar seperti ini:

- Komponen penyusunnya bisa diganti.
- Komponen penyusunya dapat dibaca dan dimanipulasi secara langsung.
- Komponen penyusunnya bisa ditambah.
- Komponen penyusunnya dapat diambil dengan menunjukkan indeksnya atau dengan notasi slice.
- Komponen penyusun sebuah list dapat juga berupa list yang lain.

Contoh :

```
prodi =["Ilmu Komputer","Teknik Informatika","Sistem Informasi"]
print(prodi[0])
print(prodi[1])
print(prodi[2])
```

Tampilan Luaran (Output) :

```
======================== RESTART: D:\program2.py =====
Ilmu Komputer
Teknik Informatika
Sistem Informasi
```

print(prodi[0:-1])

Output : ['Ilmu Komputer', 'Teknik Informatika']

print(prodi[:2]+["Fakultas Teknik","Unmul"])

Output : ['Ilmu Komputer', 'Teknik Informatika', 'Fakultas Teknik', 'Unmul']

print(len(prodi))

Output : 3

7.2. Cara Membuat List di Python

List dapat kita buat seperti membuat variabel biasa, namun nilai variabelnya diisi dengan tanda kurung siku ([]).

Contoh :

```
nama = ["Agus", "Bambang", "Nina", "Putra"]
```

Jenis data apa saja yang boleh diisi ke dalam List?

list dapat diisi dengan tipe data apa saja, string, integer, float, double, boolean, object, dan sebagainya. Kita juga bisa mencampur isinya.

Contoh:

laci = ["buku", 21, True, 34.12]

Ada empat jenis tipe data pada list laci:

- "buku" adalah tipe data **string**;
- 21 adalah tipe data **integer**;
- True adalah tipe data **boolean**;
- dan 34.12 adalah tipe data **float**.

7.3. Cara Pemanggilan List

Setelah mengetahui cara membuat dan menyimpan data di dalam List, mari kita coba mengambil datanya. List sama seperti array, list juga memiliki nomer indeks untuk mengakses data atau isinya. Nomer indeks list selalu dimulai dari nol (0). Nomer indeks ini yang kita butuhkan untuk

mengambil isi (item) dari list.

Contoh:

```
#nama list dari nama seseorang
nama = ["Agus", "Bambang", "Nina", "Putra"]
# Misanya kita ingin mengambil si Nina
# Maka indeknya adalah 2
print (nama[2])
```

Hasil Luaran :

```
In [104]: runfile('D:/python/program2.py', wdir='D:/python')
Nina
```

7.4. Program Menggunakan List

a. Mengambil Nilai List

Contoh Studi Kasus Perintah Program Aplikasi List Anggota suatu organisasi dengan perintah sebagai berikut :

1. Buat sebuah list untuk menyimpan Anggota
2. Isi list sebanyak 6
3. Tampilkan isi list indeks nomer 3
4. Tampilkan semua teman dengan perulangan
5. Tampilkan panjang list

Tuliskan kode berikut ini

```
# Buat list untuk menampung nama-nama Hero buat file dengan nama hero.py
sapujagad = ["Hario", "Medi", "Bambang", "Reza", "Gubtha", "Anton", "Putut"]

# Tampilkan isi list my_friends dengan nomer indeks 3
print ("Isi Nama Dosen indeks ke-3 adalah: {}".format(sapujagad[3]))

# Tampilkan semua daftar teman
print ("Jumlah Semua Dosen : ada {} orang".format(len(sapujagad)))
for dosen in sapujagad:
    print (dosen)
```

Pada kode di atas, kita menggunakan fungsi **len()** untuk mengambil panjang list.

Hasil outputnya :

```
Isi Nama Dosen indeks ke-3 adalah: Reza
Jumlah Semua Dosen : ada 7 orang
Hario
Medi
Bambang
Reza
Gubtha
Anton
Putut
```

b. Mengganti Nilai List

List bersifat *mutable*, artinya isinya bisa kita ubah-ubah.

Contoh:

```
# list mula-mula
prodi = ["Teknik Informatika", "Bahasa Coding",
         "Sistem Informasi", "Teknologi Informasi"]
# mengubah nilai index ke-2
print("Data pertama : ", prodi)
prodi[2] = "Ilmu Komputer"
print("Data perubahan : ", prodi)
```

Hasil Luaran Program :

```
Data pertama :  ['Teknik Informatika', 'Bahasa Coding', 'Sistem
Informasi', 'Teknologi Informasi']
Data perubahan :  ['Teknik Informatika', 'Bahasa Coding', 'Ilmu
Komputer', 'Teknologi Informasi']
```

c. Menambahkan Item List

Tedapat dua metode *(method)* atau fungsi yang bisa digunakan untuk menambahkan isi atau item ke List:

1. append(item) menambahkan item dari belakang.

   Contoh:

   ```
   #list awal
   prodi = ["Teknik Informatika", "Bahasa Coding",
            "Sistem Informasi", "Teknologi Informasi"]
   #penambahan prodi
   prodi.append("Rekayasa Sistem")
   print(prodi)
   ```

   Output :

   ```
   ===========
   ['Teknik Informatika', 'Bahasa Coding', 'Sistem Informasi',
   'Teknologi Informasi', 'Rekayasa Sistem']
   >>>
   ```

2. insert(index, item) menambahkan item dari indeks tertentu

   Contoh :

   ```
   #list awal
   prodi = ["Teknik Informatika", "Bahasa Coding",
            "Sistem Informasi", "Teknologi Informasi"]
   #penambahan prodi pada index ke 2
   prodi.insert(2, "Rekayasa Sistem")
   print(prodi)
   ```

   Output :

   ```
   ['Teknik Informatika', 'Bahasa Coding', 'Rekayasa Sistem',
   'Sistem Informasi', 'Teknologi Informasi']
   ```

d. Menghapus Nilai List

Perintah yang dipergunakan untuk menghapus list adalah del[index]

Contoh :

```
#list awal data prodi
prodi = ["Teknik Informatika", "Bahasa Coding",
         "Sistem Informasi", "Teknologi Informasi"]
#Penghapusan prodi Bahasa Coding pada index 1
del prodi[1]
print(prodi)
```

Selain menggunakan perintah del, kita juga bisa menggunakan method remove() dengan parameter item yang akan dihapus.

Contoh:

```
#list awal data prodi
prodi = ["Teknik Informatika", "Bahasa Coding",
         "Sistem Informasi", "Teknologi Informasi"]
#Penghapusan prodi Bahasa Coding pada index 1
del prodi[1]
print(prodi)
#Penghapusan prodi Teknologi Informasi
prodi.remove("Teknologi Informasi")
print(prodi)
```

Hasil Luaran :

```
================= RESTART: D:\program2.py =================
['Teknik Informatika', 'Sistem Informasi', 'Teknologi Informasi']
['Teknik Informatika', 'Sistem Informasi']
>>>
```

e. Memotong List

Contoh Program Memotong List

```
# Kita punya list Mobil
mobil = ["Avanza", "Xenia", "Pajero", "Fortuner", "Rush", "Sigra"]
# Kita potong dari indeks ke-1 sampai ke-5
print (mobil[1:5])
```

Hasil Luaran :

```
====================== RESTART: D:\program2.py ==
['Xenia', 'Pajero', 'Fortuner', 'Rush']
>>> |
```

f. Operasi List

- **Penggabungan (+)**

```
# Beberapa list lagu
list_lagu = [
    "Cinta Luar Biasa",
    "Hymne Unmul"
]
# playlist lagu favorit
playlist_favorit = [
    "Kasih Sayang Kepada Orang Tua",
    "Ruang Rindu"
]
# Mari kita gabungkan keduanya
semua_lagu = list_lagu + playlist_favorit
print (semua_lagu)
```

Hasil Luaran :

```
In [130]: runfile('D:/python/program2.py', wdir='D:/python')
['Cinta Luar Biasa', 'Hymne Unmul', 'Kasih Sayang Kepada Orang
Tua', 'Ruang Rindu']
```

- **Perkalian (*)**

```
# playlist lagu favorit
playlist_favorit = [
    "Is My Life",
    "Now Loading!!!"
]
# ulangi sebanyak 5 kali
ulangi = 5
now_playing = playlist_favorit * ulangi
print (now_playing)
```

Hasil Luaran :

```
In [131]: runfile('D:/python/program2.py', wdir='D:/python')
['Is My Life', 'Now Loading!!!', 'Is My Life', 'Now Loading!!!',
'Is My Life', 'Now Loading!!!', 'Is My Life', 'Now Loading!!!',
'Is My Life', 'Now Loading!!!']
```

7.5. List Multidimensi

Pada contoh-contoh di atas, hanya membuat list satu dimensi saja. List dapat juga memiliki lebih dari satu dimensi atau disebut dengan multi dimensi. List multi dimensi biasanya digunakan untuk menyimpan struktur data yang kompleks seperti tabel, matriks, graph, tree, dan lain-lain.

Contoh :

```
# List minuman dengan 2 dimensi
list_minuman = [
    ["Kopi Luwak", "Soda Sedih", "Teh"],
    ["Jus Apel", "Jus Buah Naga", "Jus Jeruk"],
    ["Es Kopyor Degan", "Es Campur", "Es Teler"]
]
# Cara mengakses list multidimensi
# misalkan kita ingin mengambil "es kopyor degan"
print (list_minuman[2][0])
```

Hasil Luaran :

```
In [133]: runfile('D:/python/program2.py', wdir='D:/python')
Es Kopyor Degan
```

Angka dua [2] pada kode di atas, menujukan indeks list yang akan di akses. Kemudian setelah dapat list-nya baru kita ambil isinya. Kemudian Bagaimana jika ingin menampilkan semua isi dalam list multi dimensi ?, jawabannya yaitu Tinggal gunakan perulangan bersarang.

Contoh Program Perulangan bersarang :

```
# List minuman dengan 2 dimensi
list_minuman = [
    ["Kopi Luwak", "Soda Sedih", "Teh"],
    ["Jus Apel", "Jus Buah Naga", "Jus Jeruk"],
    ["Es Kopyor Degan", "Es Campur", "Es Teler"]
]
for menu in list_minuman:
    for minuman in menu:
        print (minuman)
```

Hasil Luaran (Output) :

```
======================
Kopi Luwak
Soda Sedih
Teh
Jus Apel
Jus Buah Naga
Jus Jeruk
Es Kopyor Degan
Es Campur
Es Teler
```

7.6. Latihan Membuat Program List

Program Tambah Nama Pahlawan Super

```
# Membuat list kosong untuk menampung Super Hero
hero = []
stop = False
i = 0
# Mengisi super hero
while(not stop):
    hero_baru = input("Inputkan hero yang ke-{}: ".format(i))
    hero.append(hero_baru)
    # Increment i
    i += 1

    tanya = input("Mau mengisi nama Hero lagi? (y/t): ")
    if(tanya == "t"):
        stop = True

# Cetak Semua super hero
print ("=" * 10)
print ("Kamu memiliki {} hero Favoritmu Yaitu : ".format(len(hero)))
for hb in hero:
    print ("+ {}".format(hb))
```

Hasil Luaran Program :

```
Inputkan hero yang ke-0: Luffy Si Topi Jerami

Mau mengisi nama Hero lagi? (y/t): y

Inputkan hero yang ke-1: Naruro

Mau mengisi nama Hero lagi? (y/t): y

Inputkan hero yang ke-2: Shank

Mau mengisi nama Hero lagi? (y/t): y

Inputkan hero yang ke-3: Gundala

Mau mengisi nama Hero lagi? (y/t): y

Inputkan hero yang ke-4: Iron Man

Mau mengisi nama Hero lagi? (y/t): t
==========
Kamu memiliki 5 hero Favoritmu Yaitu :
+ Luffy Si Topi Jerami
+ Naruro
+ Shank
+ Gundala
+ Iron Man
```

7.7. Definisi Tuple

Tuple dalam Python adalah stuktur data yang digunakan untuk menyimpan sekumpulan data. Tuple bersifat immutable, artinya isi tuple tidak bisa kita ubah dan hapus. Namun, dapat kita isi dengan berbagai macam nilai dan objek. Tuple adalah salah satu struktur data di Python yang mampu menyimpan sekumpulan nilai dalam satu variabel.

Pada pokok pembahasan Tuple kali ini

- Cara Membuat Tuple
- Cara Mengakses Nilai Tuple
- Slicing Nilai Tuple
- Cara Mengambil panjang tuple
- Nested Tuple
- Unpacking Sequence

7.8. Cara Membuat Tuple di Python

Cara membuat Tuple dapat dibuat dengan tanda kurung seperti ini:

```
tuple = (1234, 4321, 'Hello')
```

atau bisa juga tanpa tanda kurung :

```
tuple2 = 1234, 432, 'World!'
```

Kedua-duanya valid.

7.9. Membuat Tuple Kosong

Apabila kita ingin membuat sebuah tuple tanpa isi, kita bisa menuliskannya seperti ini:

```
# Membuat tuple kosong
kosong = ()
```

Lalu untuk membuat Tuple yang hanya berisi satu (singleton), maka kita harus manambahkan tanda koma di belakangnnya.

Contoh:

```
# membuat tuple
data = ('Data 1')
data2 = "Data 2"
Jika tidak ditambahkan koma, akan bukan string.
data = ('Data 1') # <-- ini string
data2 = "Data 2" # <-- ini juga string
```

7.10. Mengakses Nilai Tuple

Sama seperti list, Tuple juga memiliki indeks untuk Mengakses item di dalamnya. Indeks Tuple dan list selalu dimulai dari nol 0.

Contoh:

```
# membuat tuple
nama = ('Amin', 'Rahmad', 'Reza')
# mengakses nilai tuple
print(nama[0])
print(nama[1])
print(nama[2])
```

Maka hasilnya:

```
====================
Amin
Rahmad
Reza
>>>
```

7.11. Memotong Tuple

Sama dengan proses di List pada tuple juga bisa melakukan proses pemotongan data :

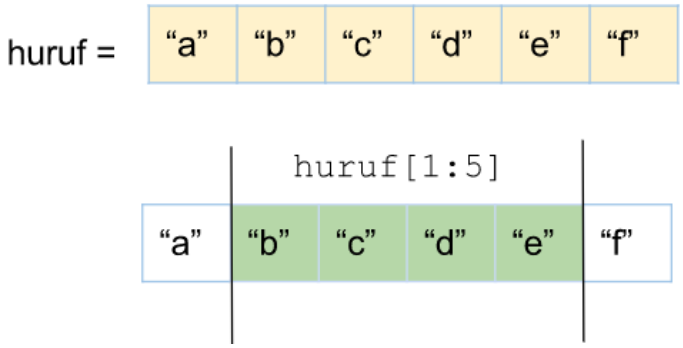
Contoh Program :

```
# mula-mula kita punya tuple seperti ini
fakultas = (123, 'Fakultas Teknik', 'https://ft.unmul.ac.id')
# lalu kita ingin potong agar ditampilkan
# dari indeks nomer 1 sampai 2
print(fakultas[1:3])
```

Maka hasilnya:

```
===================== RESTART: D:\program2.py =====
('Fakultas Teknik', 'https://ft.unmul.ac.id')
>>> |
```

Logikanya sama seperti di list sebagai berikut :



7.12. Mengambil Panjang Tuple

Untuk mengambil panjang atau jumlah item di dalam Tuple, kita bisa menggunakan fungsi len().

Contoh:

```
# Membuat Tuple dengan data hari
hari = ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jum\'at', 'Sabtu', 'Minggu')

# Mengambil panjang tuple hari
print(hari)
print("Jumlah hari: %d" % len(hari))
```

Maka hasilnya :

```
In [13]: runfile('D:/python/program5.py', wdir='D:/python')
('Senin', 'Selasa', 'Rabu', 'Kamis', "Jum'at", 'Sabtu',
'Minggu')
Jumlah hari: 7
```

7.13. Tuple Nested

Tuple juga bisa nested, artinya Tuple bisa diisi dengan Tuple.

#Contoh:

```
tuple1 = "Saya", "Bisa Lulus", "Cepat"
tuple2 = "Yakin", 3.5, "tahun", "Amin"
tuple3 = (tuple1, tuple2) # <- nested tuple
#tuple3 akan berisi nilai dari tuple1 dan tuple2.
print(tuple3)
print(tuple3[1])
print(tuple3[1][0])
```

Hasil Luaran :

```
In [17]: runfile('D:/python/program5.py', wdir='D:/python')
(('Saya', 'Bisa Lulus', 'Cepat'), ('Yakin', 3.5, 'tahun',
'Amin'))
('Yakin', 3.5, 'tahun', 'Amin')
Yakin
```

7.14. Sequence Unpacking

Proses pembuatan Tuble bisa kita sebut sebagai packing, sementara untuk mengambil (ekstrak) seluruh isinya disebut unpacking.

Contoh :

```
# mula-mula kita buat tuple seperti ini
fakultas = 123, "Fakultas Teknik", "https://ft.unmul.ac.id"
# lalu di-unpacking
id_fakultas, nama, web = fakultas
# maka sekarang tiga variabel tersebut akan memiliki nilai
# Lakukan pemanggilan luaran data
print(id_fakultas)
print(nama)
print(web)
```

Hasil Luaran Program :

```
====================== RESTART: D:\program2.py ==
123
Fakultas Teknik
https://ft.unmul.ac.id
>>> |
```

Dengan melakukan upacking, isi tuple akan di-copy ke variabel. Lalu dengan variabel kita bisa melakukan apapun, seperti mengubah isinya. Karena variabel bersifat mutable.

MODUL 8
DICTIONARY

Setelah kita mengenal stuktur data list yang mampu menyimpan berbagai macam hal. List biasanya digunakan untuk menyimpan koleksi data. Namun, list ternyata memiliki kekurangan. Kekurangannya : ia tidak bisa menggunakan kata kunci untuk mengakses itemnya. Hanya bisa menggunakan nomer indeks saja. Kekurangn ini sudah ditutipi oleh Dictionary. Pada kesempatan ini, kita akan belajar 7 hal dasar yang harus diketahui tentang Dictionary.

8.1. Definisi Dictionary

Dictionary adalah stuktur data yang bentuknya seperti kamus. Ada kata kunci kemudian ada nilainya. Kata kunci harus unik, sedangkan nilai boleh diisi dengan apa saja.

Contoh:

```
fakultas={"nama":"Fakultas Teknik Universitas Mulawarman",
          "URL":"https:\\ft.unmul.ac.id"}
print(fakultas["nama"])
```

Contoh di atas membuat sebuah Dictionary bernama fakultas dengan isi data nama dan URL. nama dan url adalah kunci (key) yang akan kita gunakan untuk mengakses nilai di dalamnya. Inilah perbedaanya dibandingkan list dan tuple. Dictionary memiliki kunci berupa teks bisa juga angka. sedangkan list dan tuple menggunakan indeks berupa angka saja untuk mengakses nilainya. Dalam bahasa pemrograman lain (seperti PHP), Dictionary juga dikenal dengan sebutan asosiatif array.

8.2. Deklarasi Dictionary

Berbeda dengan list yang memakai indeks angka untuk merujuk pada isi variabel, dictionary memakai *key* untuk merujuk pada isi variabelnya. Sifat kedua jenis data ini hanya berbeda dalam beberapa hal saja. Untuk mendeklarasikan sebuah dictionary, Python memakai tanda { }.

```
warna={"warna1":"Merah", "warna2":"Biru", "warna3":"Kuning"}
print(warna["warna1"])
print(warna["warna3"])
```

Hasil Luaran :

```
Merah
Kuning
```

Bentuk Lain :

```
prodi={}
prodi[1]="Ilmu Komputer"
prodi[2]="Teknik Informatika"
prodi[3]="Sistem Informasi"
print(prodi)
```

Output : {1: 'Ilmu Komputer', 2: 'Teknik Informatika', 3: 'Sistem Informasi'}

8.3. Membuat Dictionary

Hal yang wajib ada di dalam pembuatan Dictionary adalah:

- nama dictionary,
- key,
- value,
- buka dan tutupnya menggunakan kurung kurawal.

Antara key dan value dipisah dengan titik dua (:) dan apabila terdapat lebih dari satu item, maka dipisah dengan tanda koma (,).

Contoh satu item:

```
nama_dict = {
   "key": "value"
}
```

Contoh tiga item:

```
nama_dict = {
   "key1": "value",
   "key2": "value",
   "key3": "value"
}
```

Isi dari *Dictionary* dapat berupa:

- String
- Integer
- Objek
- List

- Tuple
- Dictionary
- dsb.

Contoh:

```
pak_dosen = {
    "nama": "Muhammad Bambang",
    "umur": 22,
    "hobi": ["Coding", "Membaca", "Traveling"],
    "menikah": True,
    "sosmed": {
        "facebook": "bengbeng",
        "twitter": "@mrbengbeng"
    }
}
```

Mari kita lihat isi dari Dictionary di atas:

- nama berisi string "Muhammad Bambang"
- umur berisi integer 22
- hobi berisi list dari string
- menikah berisi boolean True
- dan sosmed berisi Dictionary

Menggunakan Konstruktor

Selain menggunakan cara di atas, kita juga bisa membaut *Dictionary* dari constructor dict() dengan parameter *key* dan *value*.

Contoh:

almamater = dict(unmul="kuning", umkt="biru", untag="merah")

Maka akan menghasilkan *dictionary* seperti ini:

{'unmul': 'kuning', 'umkt': 'biru', 'untag': 'merah'}

8.4. Mengakses Item Pada Dictionary

Cara mengaksesnya sama seperti list. Namaun kunci yang digunakan bukan angka, melainkan keyword yang sudah kita tentukan di dalam Dictionary-nya.

Contoh Program :

```
pak_dosen = {
    "nama": "Anton Prafanto",
    "umur": 22,
    "hobi": ["Coding", "Mancing", "Membaca"],
    "menikah": True,
    "sosmed": {
        "facebook": "Anton Prafanto",
        "twitter": "@anton"
    }
}

# Mengakses isi dictionary
print("Nama saya adalah %s" % pak_dosen["nama"])
print("Facebook: %s" % pak_dosen["sosmed"]["facebook"])
print("Twitter: %s" % pak_dosen["sosmed"]["twitter"])
```

Hasil Luaran :

```
Nama saya adalah Anton Prafanto
Facebook: Anton Prafanto
Twitter: @anton
>>> |
```

Selain dengan cara di atas, kita juga bisa mengambil nilai Dictionary dengan method get().
Contoh:
print(pak_dosen.get("nama"))

Hasilnya:
Anton Prafanto

Menggunakan Perulangan :
Untuk mencetak semua isi Dictionary, kita bisa menggunakan perulangan seperti ini:

```
# Membuat dictionary
fakultas = {
    "name": "Teknik Universitas Mulawarman",
    "url": "https://ft.unmul.ac.id",
    "rank": "2"
}

# Mencetak isi dictionary dengan perulangan
for key in fakultas:
    print(fakultas[key])
```

Hasilnya Luarannya :

```
Teknik Universitas Mulawarman
https://ft.unmul.ac.id
2
```

Kita juga bisa melakukannya seperti ini:

```
# Membuat dictionary
fakultas = {
    "name": "Teknik Universitas Mulawarman",
    "url": "https://ft.unmul.ac.id",
    "rank": "2"
}

# Mencetak isi dictionary dengan perulangan
for key, val in fakultas.items():
    print("%s : %s" % (key, val))
```

Hasilnya Luaran :

```
name : Teknik Universitas Mulawarman
url : https://ft.unmul.ac.id
rank : 2
```

8.5. Menambahkan Item Pada Dictionary

Kita bisa menggunakan method update() untuk menambahkan isi ke Dictionary. Parameternya berupa Dictionary. Selain berfungsi untuk menambahkan, method ini juga berfungsi untuk mengubah nilai dictionary apabila kunci yang dimasukkan sudah ada di dalamnya.

```
# Membuat dictionary
fakultas = {
    "name": "Teknik Universitas Mulawarman",
    "url": "https://ft.unmul.ac.id",
    "rank": "2"
}

# Mencetak isi dictionary dengan perulangan
for key, val in fakultas.items():
    print("%s : %s" % (key, val))
```

Hasil Luaran Program :

```
===================== RESTART: D:\program2.py =====
name : Teknik Universitas Mulawarman
url : https://ft.unmul.ac.id
rank : 2
```

8.6. Mengubah Item Pada Dictionary

Dictionary bersifat *mutable*, artinya nilainya dapat kita ubah-ubah. Untuk mengubah nilai *Dictionary*, kita bisa lakukan seperti ini:

nama_dic["kunci"] = "Nilai Baru"

Contoh:

```
# membuat dictionary
bahasa = {
    "utama": "Python",
    "lainnya": ["PHP","Java", "HTML"]
}
# Mencetak isi skill utama
print("Bahasa Utama : ", bahasa["utama"])
# mengubah isi skill utama
bahasa["utama"] = "C++"
# Mencetak isi skill utama
print("Bahasa Utama setelah diubah : ", bahasa["utama"])
```

Hasil Luaran :

```
In [201]: runfile('D:/python/program2.py', wdir='D:/python')
Bahasa Utama :  Python
Bahasa Utama setelah diubah :  C++
```

8.7. Menghapus Item Pada Dictionary

Untuk menghapus nilai Dictionary, kita bisa menggunakan perintah del dan method pop(). Method pop() adalah method yang berfungsi untuk mengeluarkan item dari dictionary sedangkan fungsi del adalah fungsi untuk menghapus suatu variabel dari memori.

Contoh menghapus dengan del:

```
# membuat dictionary
bahasa = {
    "utama": "Python",
    "lainnya": ["PHP","Java", "HTML"]
}
# Mencetak isi skill utama
print("Bahasa Utama : ", bahasa["utama"])
# mengubah isi skill utama
bahasa["utama"] = "C++"
# Mencetak isi skill utama
print("Bahasa Utama setelah diubah : ", bahasa["utama"])
del bahasa["utama"]
print (bahasa)
```

Hasil Luaran Hapus del:

```
In [203]: runfile('D:/python/program2.py', wdir='D:/python')
Bahasa Utama :  Python
Bahasa Utama setelah diubah :  C++
{'lainnya': ['PHP', 'Java', 'HTML']}
```

Contoh menghapus dengan method pop():

```
# membuat dictionary
bahasa = {
    "utama": "Python",
    "lainnya": ["PHP","Java", "HTML"]
}
# Mencetak isi skill utama
print("Bahasa Utama : ", bahasa["utama"])
# mengubah isi skill utama
bahasa["utama"] = "C++"
# Mencetak isi skill utama
print("Bahasa Utama setelah diubah : ", bahasa["utama"])
bahasa.pop("utama")
print (bahasa)
```

Hasil Luaran Hapus Pop() :

```
In [203]: runfile('D:/python/program2.py', wdir='D:/python')
Bahasa Utama :  Python
Bahasa Utama setelah diubah :  C++
{'lainnya': ['PHP', 'Java', 'HTML']}
```

atau bila kita ingin menghapus semuanya sekaligus, kita bisa menggunakan method clear().

Contoh:

bahasa.clear()

8.8. Mengambil Panjang Dictionary

Untuk mengambil jumlah data atau panjang Dictionary, kita bisa menggunakan fungsi len().

Contoh :

```
# membuat dictonary
books = {
    "python": "Menguasai Python dalam 2028 jam",
    "java": "Tutorial Belajar untuk Pemula",
    "php": "Membuat aplikasi web dengan PHP"
}
# mencetak jumlah data yang ada di dalam dictionary
print("Jumlah buku dalam database : %d" % len(books))
```

Hasil Luaran :

```
In [197]: runfile('D:/python/program2.py', wdir='D:/python')
Jumlah buku dalam database: 3
```

MODUL 9
FUNGSI DAN PROSEDUR

Program yang kompleks dan memiliki banyak fitur, diharuskan menggunakan fungsi. Mengapa harus menggunakan fungsi? Karena jika tidak menggunakannya kita akan kerepotan menulis kode programnya, coding yang banyak yang harus ditulis dan kode akan menjadi sulit dibaca dan dirawat (*maintenance*). Dengan fungsi, kita dapat memecah program besar menjadi sub program yang lebih sederhana. Masing-masing fitur pada program dapat kita buat dalam satu fungsi. Pada saat kita membutuhkan fitur tersebut, kita tinggal panggil fungsinya saja. Hal ini akan kita coba pada contoh program yang sudah saya sediakan di bawah. Teori dasar dan hal apa saja yang harus kita ketahui tentang fungsi di Python.

9.1. Cara Membuat Fungsi pada Python

Fungsi pada Python, dibuat dengan kata kunci def kemudian diikuti dengan nama fungsinya.

Contoh:

```
def nama_fungsi():
   print ("ini adalah fungsi di Python")
```

Sama seperti blok kode yang lain, kita juga harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi. Setelah kita buat, kita bisa memanggilnya seperti ini:

```
nama_fungsi()
```

Sebagai contoh, coba tulis kode program berikut:

```
# Membuat Fungsi
def salam():
    print ("Selamat Pagi, Mulawarman Muda")
# Pemanggilan Fungsi
salam()
# Pemanggilan Fungsi 3 kali
salam()
salam()
salam()
```

Hasilnya:

```
====================== RESTART: D:\program2.py ==
Selamat Pagi, Mulawarman Muda
Selamat Pagi, Mulawarman Muda
Selamat Pagi, Mulawarman Muda
Selamat Pagi, Mulawarman Muda
>>>
```

9.2. Fungsi dengan Parameter

Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi.

```
def fungsi(parameter):
    print(parameter)
```

Cara pemanggilan fungsi yang memiliki parameter adalah seperti ini:

salam**("Selamat siang")**

"Selamat siang" adalah nilai parameter yang kita berikan. Lalu bagaimana kalau parameternya lebih dari satu. Kita bisa menggunakan tanda koma (,) untuk memisahnya.

Contoh Program:

```
# Membuat fungsi dengan parameter (panjang, lebar)
def luas_persegi_panjang(panjang, lebar):
    luas = panjang * lebar
    print ("Luas persegi panjang : %f" % luas)

# Pemanggilan fungsi luas_persegi_panjang
luas_persegi_panjang(4, 6)
```

Hasil Luaran:

Luas persegi : 24.000000

9.3. Fungsi Mengembalikan Nilai

Fungsi yang tidak mengembalikan nilai biasanya disebut dengan prosedur. Namun, kadang kita butuh hasil proses dari fungsi untuk digunakan pada proses berikutnya. Maka fungsi harus mengembalikan nilai dari hasil pemrosesannya. Cara mengembalikan nilai adalah menggunkan kata kunci `return` lalu diikuti dengan nilai atau variabel yang akan dikembalikan.

Contoh Program :

```
#Contoh Program kembalikan hasil fungsi
def luas_persegi(sisi):
    luas = sisi * sisi
    return luas

# pemanggilan fungsi luas persegi
print ("Luas persegi : %d" % luas_persegi(8))
```

Hasil Luaran :

Luas persegi : 64

Apa bedanya dengan fungsi luas_segitiga() yang tadi?. Pada fungsi luas_segitiga() kita melakukan print dari hasil pemrosesan secara langsung di dalam fungsinya. Sedangkan fungsi luas_persegi(), kita melakukan print pada saat pemanggilannya. Jadi, fungsi luas_persegi() akan bernilai sesuai dengan hasil yang dikembalikan. Sehingga kita dapat memanfaatkannya untuk pemerosesan berikutnya.

Contoh penggunaan pengembalian nilai (Return)

```
# rumus: sisi x sisi
def luas_persegi(sisi):
    luas = sisi * sisi
    return luas
# rumus: sisi x sisi x sisi
def volume_persegi(sisi):
    volume = luas_persegi(sisi) * sisi
    print ("Volume Persegi = ", volume)
#pemanggilan Fungsi
luas_persegi(4)
volume_persegi(6)
```

Hasil Luaran :

Volume Persegi = 216

Pada program diatas dapat dilihat pada hasil luaran def luas_persegi(sisi) dapat dipergunakan oleh def volume_persegi(sisi) menggunakan fungsi return.

9.4. Variabel Global dan Lokal

Variabel Global adalah variabel yang bisa diakses dari semua fungsi, sedangkan variabel lokal hanya bisa diakses di dalam fungsi tempat dia berada saja. Pada Python, urutan pengaksesan variabel (scope) dikenal dengan sebutan LGB (Local, Global, dan Build-in). Jadi program python mulai mencari vairabel lokal terlebih dahulu, kalau ada maka itu yang digunakan. Tapi kalau tidak ada, pencarian terus ke Global, dan Build-in. Variabel Build-in adalah variabel yang sudah ada di dalam Python.

Contoh program:
membuat variabel global

```
nama = "Universitas Mulawarman"
versi = "1.0.0"

def help():
    # ini variabel lokal
    nama = "Fakultas Teknik"
    versi = "1.0.2"
    # mengakses variabel lokal
    print ("Nama: %s" % nama)
    print ("Versi: %s" % versi)

# mengakses variabel global
print ("Nama: %s" % nama)
print ("Versi: %s" % versi)

# memanggil fungsi help()
help()
```

Hasil Luaran :

```
Nama: Universitas Mulawarman
Versi: 1.0.0
Nama: Fakultas Teknik
Versi: 1.0.2
>>> |
```

Perhatikanlah variabel nama yang berada di dalam fungsi help() dan diluar fungsi `help(). Variabel nama yang berada di dalam fungsi help() adalah variabel lokal. Jadi, saat program memanggil fungsi help() maka nilai yang akan tampil adalah nilai yang ada di dalam fungsi help(). Cara kerjanya Python mulai mencari dari lokal, ke global, dan build-in. Kalau di tiga tempat itu tidak ditemukan, maka biasanya akan terjadi NameError atau variabel tidak ditemukan.

9.5. Program Menggunakan Fungsi

Pertama kita buat sebuah variabel global berupa list untuk menampung judul-judul buku.

Variabel global untuk menyimpan data Buku

buku = []

Nanti program ini akan mampu melakukan operasi CRUD (*Create*, *Read*, *Update*, dan *Delete*). Maka kita membutuhkan fungsi-fungsi berikut:

- show_data() untuk menampilkan data dari list buku;
- insert_data() untuk menambahkan data ke list buku;
- edit_data() untuk mengedit data di list buku;
- delete_data() untuk untuk menghapus data dari list buku.

Dimulai dari fungsi show_data():

fungsi untuk menampilkan semua data

```
buku =[]
def show_data():
    if len(buku) <= 0:
        print ("Belum Ada data")
    else:
        for indeks in range(len(buku)):
            print ("[%d] %s" % (indeks, buku[indeks]))
```

Fungsi di atas akan mengecek isi dari list buku. Jika isinya kosong (len(buku) <= 0) maka tampilkan "BELUM ADA DATA". Namun, apabila ada isinya, maka tampilkan semua isinya dengan perulangan.

```
# fungsi untuk menambah data
def insert_data():
    buku_baru = input("Judul Buku : ")
    buku.append(buku_baru)
```

Fungsi di atas akan mengambil input dari user kemudian diisi ke dalam list buku dengan fungsi append(). Fungsi append() adalah fungsi untuk menambahkan item di akhir list. Selain append() ada juga prepend(). Namun, untuk kasus ini, kita pakai append() saja. Penjelasan lengkap tentang append() dan prepend() bisa dibaca pada:

Selanjutnya membuat fungsi edit_data():

```
# fungsi untuk edit data
def edit_data():
    show_data()
    indeks = input("Inputkan ID buku: ")
    if(indeks > len(buku)):
        print ("ID salah")
    else:
        judul_baru = input("Judul baru: ")
        buku[indeks] = judul_baru
```

Fungsi di atas akan menampilkan isi dari list buku dengan memanggil fungsi show_data() di dalamnya. Setelah itu, kita meminta user untuk menginputkan ID atau nomer indeks buku yang akan diedit. Lalu kita lakukan pengecekan, jika ID yang diinputkan melebihi dari isi list buku (indeks > len(buku)), maka tampilkan pesan "ID salah". Namun, apabila tidak melebihi dari isi buku, maka ambil input untuk judul baru dan simpan sesuai ID-nya.

Selanjutnya membuat fungsi delete_data():

```
# fungsi untuk menhapus data
def delete_data():
    show_data()
    indeks = input("Inputkan ID buku: ")
    if(indeks > len(buku)):
        print ("ID salah")
    else:
        buku.remove(buku[indeks])
```

Hampir sama dengan fungsi edit_data(). Fungsi delete_data() juga harus menampilkan isi list buku dan mengambil ID yang akan dihapus.
Kita dapat menghapus item pada list dengan fungsi remove().
Apakah sudah selesai?
Belum, masih ada dua fungsi lagi yang kita butuhkan:

- Fungsi untuk menampilkan menu
- Fungsi untuk keluar (sudah ada di python: exit())

Baik mari kita coba membuat menunya

```
# fungsi untuk menampilkan menu
def show_menu():
    print ("\n")
    print ("----------- MENU ----------")
    print ("[1] Show Data")
    print ("[2] Insert Data")
    print ("[3] Edit Data")
    print ("[4] Delete Data")
    print ("[5] Exit")

    menu = input("PILIH MENU> ")
    print ("\n")

    if menu == "1":
        show_data()
    elif menu == "2":
        insert_data()
    elif menu == "3":
        edit_data()
    elif menu == "4":
        delete_data()
    elif menu == "5":
        exit()
    else:
        print ("Salah pilih!")
```

Fungsi di atas akan menampilkan menu dari 1–5, lalu memanggil fungsi-fungsi yang sudah dibuat berdasarkan menu yang dipilih. Terakhir, kita harus membuat *main loop* programnya.

```
if __name__ == "__main__":
  while(True):
    show_menu()
```

Program akan mengulang terus-menerus sampai fungsi `exit()` dieksekusi. `if __name__ == "__main__":` adalah blok main di Python. Sebenarnya tanpa ini, programnya sudah bisa dijalankan. Sehingga kode lengkapnya akan seperti ini:

```
# fungsi untuk menampilkan semua data
buku =[]
def show_data():
    if len(buku) <= 0:
        print ("Belum Ada data")
    else:
        for indeks in range(len(buku)):
            print ("[%d] %s" % (indeks, buku[indeks]))

# fungsi untuk menambah data
def insert_data():
    buku_baru = input("Judul Buku : ")
    buku.append(buku_baru)

# fungsi untuk edit data
def edit_data():
    show_data()
    indeks = int(input("Inputkan ID buku: "))
    if(indeks > len(buku)):
        print ("ID salah")
    else:
        judul_baru = input("Judul baru: ")
        buku[indeks] = judul_baru

# fungsi untuk menghapus data
def delete_data():
    show_data()
    indeks = int(input("Inputkan ID buku: "))
    if(indeks > len(buku)):
        print ("ID salah")
    else:
        buku.remove(buku[indeks])

# fungsi untuk menampilkan menu
def show_menu():
    print ("\n")
    print ("----------- MENU ----------")
    print ("[1] Show Data")
    print ("[2] Insert Data")
    print ("[3] Edit Data")
    print ("[4] Delete Data")
```

```
    print ("[5] Exit")

    menu = input("PILIH MENU> ")
    print ("\n")

    if menu == "1":
        show_data()
    elif menu == "2":
        insert_data()
    elif menu == "3":
        edit_data()
    elif menu == "4":
        delete_data()
    elif menu == "5":
        exit()
    else:
        print ("Salah pilih!")

if __name__ == "__main__":
    while(True):
        show_menu()
```

Hasil Luaran :

```
----------- MENU ----------
[1] Show Data
[2] Insert Data
[3] Edit Data
[4] Delete Data
[5] Exit

PILIH MENU> 2



Judul Buku : Mengapa


----------- MENU ----------
[1] Show Data
[2] Insert Data
[3] Edit Data
[4] Delete Data
[5] Exit

PILIH MENU> 1


[0] Mengapa
```

Selamat Anda Telah Selesai menjadi programmer python Pemula.

DAFTAR PUSTAKA

Abidin, R. (2016). Pengertian Algoritma Pemrograman. Retrieved September 5, 2020, from https://teknojurnal.com/pengertian-algoritma-pemrograman/

Andre. (2018a). Tutorial Belajar Python Part 1: Pengertian Bahasa Pemrograman Python. Retrieved September 6, 2020, from https://www.duniailkom.com/tutorial-belajar-python-pengertian-bahasa-pemrograman-python/

Andre. (2018b). Tutorial Belajar Python Part 13: Tipe Data Tuple dalam Bahasa Python. Retrieved September 8, 2020, from https://www.duniailkom.com/tutorial-belajar-python-tipe-data-tuple-dalam-bahasa-python/

Andre. (2019). Tutorial Belajar Python Part 15: Tipe Data Dictionary dalam Bahasa Python. Retrieved September 8, 2020, from https://www.duniailkom.com/tutorial-belajar-python-tipe-data-dictionary-dalam-bahasa-python/

Belajarpython. (2020a). Dictionary Python. Retrieved September 8, 2020, from https://belajarpython.com/tutorial/dictionary-python

Belajarpython. (2020b). Fungsi Python. Retrieved September 8, 2020, from https://belajarpython.com/tutorial/fungsi-python

Belajarpython. (2020c). List Python. Retrieved September 8, 2020, from https://belajarpython.com/tutorial/list-python

Belajarpython. (2020d). Tuple Python. Retrieved September 8, 2020, from https://belajarpython.com/tutorial/tuple-python

Codeva. (2020). Pengertian Algoritma, Struktur dan Ciri-Ciri Pemrograman. Retrieved September 5, 2020, from https://codeva.co.id/pengertian-algoritma/

Duniailkom.com. (2020). Tutorial Belajar Bahasa Pemrograman Python Untuk Pemula. Retrieved August 27, 2020, from https://www.duniailkom.com/tutorial-belajar-bahasa-pemrograman-python-untuk-pemula/

Eril. (2019). Pengertian Apa itu Algoritma Pemrograman dan Contohnya. Retrieved September 5, 2020, from https://qwords.com/blog/apa-itu-algoritma-pemrograman/

GeeksforGeeks. (2020). Python Programming Language. Retrieved August 17, 2020, from https://www.geeksforgeeks.org/python-programming-language/

Jubilee Enterprise. (2019). Python untuk Programmer Pemula. Jakarta: Elex Media Komputindo.

Jogjaweb. (2020). Sejarah Python. Retrieved September 6, 2020, from https://jogjaweb.co.id/blog/sejarah-python

Kadir, A. (2019). Logika Pemprograman Python. Jakarta: Elex Media Komputindo.

Kadir, A. (2018). Dasar Pemrograman Python 3 : Panduan untuk Mempelajari Python dengan Cepat dan Mudah Bagi Pemula. Yogyakarta: Penerbit Andi.

Maxmanroe. (2020). Pengertian Algoritma: Ciri-Ciri, Fungsi, Jenis-Jenis, dan Contoh Algoritma. Retrieved September 5, 2020, from https://www.maxmanroe.com/vid/teknologi/pengertian-algoritma.html

Muhardian, A. (2017a). Belajar Python: Memahami Fungsi dan Prosedur pada Python. Retrieved September 8, 2020, from https://www.petanikode.com/python-fungsi/

Muhardian, A. (2017b). Belajar Python: Mengenal Struktur Data List. Retrieved September 8, 2020, from https://www.petanikode.com/python-list/

Muhardian, A. (2018a). 7 Hal Dasar yang Harus diketahui Tentang Dictionary pada Python. Retrieved September 8, 2020, from https://www.petanikode.com/python-dictionary/

Muhardian, A. (2018b). Belajar Python: Apa itu Tuple dalam Python? Retrieved September 8,

2020, from https://www.petanikode.com/python-tuple/
Muhardian, A. (2018c). Tutorial Dasar Python untuk Pemula. Retrieved August 17, 2020, from https://www.petanikode.com/tutorial/python/
Petani Kode. (2020). Tutorial Dasar Python untuk Pemula. Retrieved August 17, 2020, from https://www.petanikode.com/tutorial/python/
Pythonindo. (2020a). Dictionary. Retrieved September 8, 2020, from https://www.pythonindo.com/dictionary/
Pythonindo. (2020b). Fungsi. Retrieved September 8, 2020, from https://www.pythonindo.com/fungsi/
Supardi, Y. (2017). Semua Bisa Menjadi Programmer Python Basic. Jakarta: Elex Media Komputindo.
Thomas, H., Cormen, E.Leiserson, C., Ronald, L., & Rivest. (2003). Introduction to Algorithms. McGraw-Hill.
W3schools. (2020). Python Tutorial. Retrieved August 17, 2020, from https://www.w3schools.com/python/