

Short Course

FUNDAMENTAL PROGRAMMING WITH PYTHON

(Versi buku: 1.0)

Copyright © 2019,

BINUSCENTER Head Office

Jl. KH. Syahdan No. 20 Kemanggisan - Palmerah

Jakarta 11480

Telp: (+6221) 534 5830, 532 2157, 532 8091

Fax: (+6221) 536 3867, Email: info@binuscenter.com

***DIPERGUNAKAN HANYA UNTUK KEPENTINGAN
PEMBELAJARAN DI BINUS CENTER***

Hak Cipta dilindungi oleh undang-undang.

**Dilarang mengutip atau memperbanyak
sebagian atau seluruh isi buku ini tanpa izin tertulis dari
BINUSCENTER Pusat.**

DAFTAR ISI

DAFTAR ISI.....	II
BAB 1. PENGENALAN PEMROGRAMAN PYTHON.....	1
1.1. Instalasi Python.....	2
1.2. Aturan Penulisan Sintaks Pada Python.....	3
1.3. Editor Pemrograman Python.....	7
1.4. Membuat File Pemrograman.....	7
1.5. Membuat Teks “Hello World”	7
BAB 2. DASAR PEMROGRAMAN PYTHON.....	9
2.2. Tipe Data.....	10
2.3. Konversi Tipe Data	13
2.4. Operator	14
2.4.1 Operator Aritmatika	15
2.4.2 Operator Penugasan.....	17
2.4.3 Operator Pembandingan.....	19
2.4.4 Operator Logika.....	21
2.4.5 Operator Bitwise	22
2.4.6 Operator Ternary	25
BAB 3. SELECTION & LOOPING PADA PYTHON.....	27
3.1. Struktur Dasar If.....	27
3.2. Struktur Percabangan If/Elif/Else	28
3.3. Perulangan	29
3.4. Perulangan For	29
3.5. Perulangan While	30
BAB 4. FUNGSI PADA PYTHON.....	33

4.1. Fungsi dengan parameter	34
4.2. Fungsi yang mengembalikan nilai	34
4.3. Sifat Variabel Dalam Fungsi	36
4.4. Fungsi Built-In Python	37
BAB 5. LIST PADA PYTHON	41
5.1. List	41
5.2. Cara Mengambil Nilai dari List	42
5.3. Mengganti Nilai List	44
5.4. Menambahkan Item List	44
5.5. Menghapus Item List	47
5.6. Memotong List	48
5.7. Operasi List	48
5.8. List Multi Dimensi	51
BAB 6. TUPLES & SET PADA PYTHON.....	55
6.1. Membuat dan Mengakses Tuple	55
6.2. Memotong Tuple	56
6.4. Tuple Nested	57
6.5. Iterasi Pada Tuple	58
6.6. Jenis-Jenis Method (Fungsi) Pada Tuple	59
6.7. Set	59
6.8. Mengubah dan Menghapus Anggota Set	62
6.9. Operasi Set	65
BAB 7. DICTIONARY PADA PYTHON	74
7.1. Membuat Dictionary	74
7.2. Menggunakan Konstruktor	76

7.3. Mengakses Nilai Item Dari Dictionary.....	77
7.4. Menggunakan Perulangan.....	78
7.5. Mengubah Nilai dan Menghapus Item Dictionary	79
7.6. Menambahkan Item Ke Dictionary	81
7.7. Mengambil Panjang Dictionary	82
7.8. Jenis-Jenis Method (Fungsi) Dictionary.....	83

BAB 1. PENGENALAN PEMROGRAMAN PYTHON

Python adalah Bahasa Pemrograman interpretative yang dianggap mudah dipelajari serta berfokus pada keterbacaan kode. Dengan kata lain, Python diklaim sebagai bahasa pemrograman yang memiliki kode-kode pemrograman yang sangat jelas, lengkap dan mudah untuk dipelajari.

Python secara umum berbentuk pemrograman berorientasi objek, pemrograman imperative, dan pemrograman fungsional dan dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

Saat ini kode python dapat dijalankan di berbagai platform sistem operasi, beberapa diantaranya adalah:

- Windows
- Mac OS X
- Java Virtual Machine
- OS/2
- Amiga
- Palm
- Symbian (untuk produk-produk Nokia)

Kelebihan Pemrograman Python:

1. Multi-paradigm Programming Language.
2. Interpreted Language.
3. Support Dynamic Data Type.
4. Independent from platforms.
5. Focus on Development Time.
6. Simple and Easy Grammar.

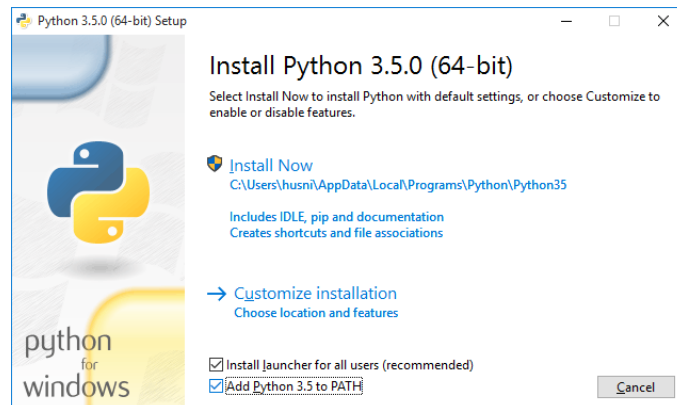
7. High-level internal objects data type.
8. Automatic Memory Management.
9. It's Free (Open Source).

Python dapat digunakan untuk mengembangkan:

1. Web Development (Tools Jango and flask).
2. Data Science.
3. Scripting.
4. GUI, Embedded applications, gaming.
5. Education.

1.1. Instalasi Python

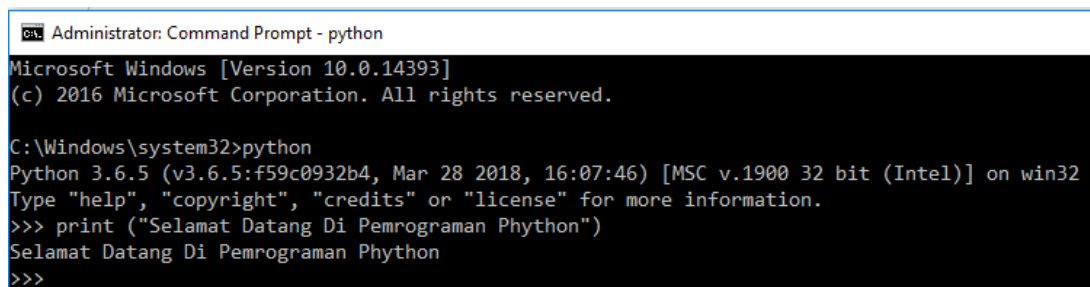
Python dapat diinstal dengan mudah. Langkah pertama yang harus dilakukan adalah mendownload versi python 3 keatas dari situsnya di python.org. Setelah download selesai, cukup *double click* untuk menjalankan program instalasinya.



Gambar 1. Jendela instalasi Python.

Gambar di atas memperlihatkan jendela pertama dari program instalasi Python. Tidak ada yang sangat perlu dikonfigurasi kecuali satu hal, yaitu

memastikan “Add Python 3.5 to PATH” dalam kondisi terpilih (dicentang). Ini maksudnya adalah menambahkan program Python ke dalam PATH dari sistem Windows sehingga dimana pun berada program Python dapat dipanggil untuk mengeksekusi modul-modul yang ditulis mengikuti kaidah bahasa pemrograman Python. Salah satu cara memulai program Python adalah dengan command line (shell) Windows (begitu pula di Linux, Macintosh dan sistem operasi yang lain). Berikut contoh command line (shell) di Windows.



```
Administrator: Command Prompt - python
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print ("Selamat Datang Di Pemrograman Python")
Selamat Datang Di Pemrograman Python
>>>
```

Gambar 1. Command line (shell) di windows.1.2. Aturan Penulisan Sintaks Python

1.2. Aturan Penulisan Sintaks Pada Python

Terdapat 5 aturan penulisan sintaks pada Python:

1. Penulisan Statement Python.

Statement adalah sebuah intruksi atau kalimat perintah yang akan dieksekusi oleh komputer.

Contoh:

```
print("Hello World!")
print("Belajar Python dari Nol")
```

Penulisan satu statement tidak diakhiri dengan tanda titik-koma. Sedangkan, bila kita ingin menulis lebih dari satu statement dalam satu baris, maka kita harus memisahkannya dengan titik-koma.

Contoh:

```
print("Hello"); print("World"); print("Tutorial Python untuk Pemula")
```

Tapi tidak dianjurkan menulis lebih dari satu statement dalam satu baris. Karena akan sulit dibaca

2. Penulisan String Pada Python.

String adalah teks atau kumpulan dari karakter. String dalam pemrograman biasanya ditulis dengan dibungkus menggunakan tanda petik. Bisa menggunakan tanda petik tunggal maupun ganda.

Contoh : judul="program python"

Penulis='Agus'

Atau kita juga bisa menggunakan triple tanda petik.

Contoh: judul="""program python"""

Penulis="""Agus"""

3. Penulisan Case Pada Python.

Sintak Python bersifat *case sensitive*, artinya teksini dengan Teksini dibedakan. Contoh:

judul = "Belajar Dasa-dasar Python"

Judul = "Belajar Membuat Program Python"

Antara variabel judul (j huruf kecil) dengan Judul (J huruf kapital) itu dibedakan

4. Penulisan Blok Program Pada Python.

Blok program adalah kumpulan dari beberapa statement yang digabungkan dalam satu blok. Penulisan blok program harus ditambahkan indentasi (tab atau spasi 2x/4x).

Contoh yang benar:

```
# blok percabangan if
if username == 'agus':
    print("Selamat Datang Agus")
    print("Silahkan memilih menu")
```

```
# blok percabangan for
for i in range(10):
    print i
```

Contoh yang salah:

```
# blok percabangan if
if username == 'agus':
print("Selamat Datang Agus")
print("Silahkan memilih menu")
```

```
# blok percabangan for
for i in range(10):
print i
```

Ada beberapa macam blok program:

- Blok Percabangan
- Blok Perulangan
- Blok Fungsi

- Blok Class
- Blok Exception
- Blok With

5. Penulisan Blok Program Pada Python.

Komentar merupakan baris kode yang tidak akan dieksekusi. Komentar digunakan untuk memberikan informasi tambahan dan untuk menonaktifkan kode. Ada beberapa cara menulis komentar pada pemrograman Python:

- Menggunakan Tanda Pagar (#)

Cara ini paling sering digunakan.

Contohnya:

```
# ini adalah komentar
```

```
# Ini juga komentar
```

- Menggunakan Tanda Petik

Selain untuk mengapit teks (*string*), tanda petik juga dapat digunakan untuk membuat komentar.

Contoh:

```
"Ini adalah komentar dengan tanda petik ganda"
```

```
'Ini juga komentar, tapi dengan tanda petik tunggal'
```

Penulisan komentar dengan tanda petik jarang digunakan, kebanyakan orang lebih memilih untuk menggunakan tanda pagar. Jadi tidak direkomendasikan.

- Menggunakan Triple Tanda Petik

Sedangkan triple tanda petik, sering digunakan untuk menuliskan dokumentasi.

Contohnya: `'''Ini juga komentar, tapi dengan triple tanda petik'''`

1.3. Editor Pemrograman Python

Terdapat banyak sekali editor pemrograman Python yang dapat digunakan seperti NetBeans, PyCharm, Eclipse, Geany, Sublime Text dan lainnya. Dalam hal ini kita menggunakan PyCharm karena jauh lebih praktis digunakan karena tidak perlu melakukan berbagai konfigurasi seperti beberapa editor lainnya. Kelebihan PyCharm adalah lebih ringan dan praktis, karena PyCharm dibangun menggunakan Python, maka dukungan terhadap bahasa pemrograman tersebut cukup lengkap dan praktis.

Editor PyCharm dapat didownload dari situs:

<https://www.jetbrains.com/pycharm/download/>.

1.4. Membuat File Pemrograman

Langkah-langkah membuat file Python pada PyCharm:

1. Klik menu **File > New**.
2. Pilih **Python File**.
3. Buat nama untuk file script Python di dalam textbox **Name**. Misalnya, **HelloWorld**, kemudian tekan tombol **Ok**. Panel baru dengan nama file yang tercatat di atasnya dengan nama HelloWorld.py akan terlihat di dalam jendela PyCharm.

1.5. Membuat Teks "Hello World"

Dalam bahasa Python kita tuliskan seperti berikut : `print("Hello World")`, selanjutnya untuk menguji script yang kita tulis sekaligus menampilkan hasilnya , kita dapat meng-klik menu **Run > Run**

```
print("Hello World")
```

Hasil akan ditampilkan dalam panel run sebagai berikut :

```
Hello World

Process finished with exit code 0
```

Latihan :

1. Buatlah sebuah program untuk mencetak karakter seperti berikut:



2. Buatlah sebuah program yang dapat menampilkan bentuk sebuah *box*, *oval*, *arrow* dan *diamond* dengan menggunakan tanda *asterisks* (*)

Berikut contoh yang ditampilkan:



BAB 2. DASAR PEMROGRAMAN PYTHON

Bab ini secara khusus digunakan untuk memperkenalkan dasar-dasar pemrograman Python seperti pembuatan variable, tipe data dan operator.

2.1. Aturan Pembuatan dan penulisan Variabel

Terdapat beberapa aturan dan 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.

Variabel di python dapat dibuat dengan format seperti ini:

```
nama_variabel = <nilai>
```

Contoh:

```
variabel_ku = "ini isi variabel"
```

```
variabel2 = 20
```

Kemudian untuk melihat isi variabel, kita dapat menggunakan fungsi print.

```
print (variabel_ku)
```

```
print (variabel2)
```

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

Contoh:

```
>>> nama = "Agus"
>>> print nama
Agus
>>> del(nama)
>>> print nama
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nama' is not defined
>>>
```

Pada perintah terakhir, kita akan mendapatkan `NameError`. Artinya variabel tidak ada di dalam memori karena sebelumnya sudah dihapus.

Ada berbagai cara membuat variable dan memasukkan data ke dalam variable tersebut. Pada contoh dibawah ini , proses memasukkan data ke dalam sebuah variable dilakukan dengan memanfaatkan input, berikut contohnya:

```
user_input = input ("Masukkan Nama Anda : ")
```

```
print ("Nama Anda Adalah:",user_input)
```

2.2. Tipe Data

Cara mengisi nilai variabel ditentukan dengan jenis datanya, misalkan untuk tipe data teks (*string*) maka harus diapit dengan tanda petik ("...").

Sedangkan untuk angka (*integer*) dan *boolean* tidak perlu diapit dengan tanda petik.

Contoh:

```
nama_ku = "Agus"  
umur = 20  
tinggi = 183.22
```

Python akan secara otomatis mengenali jenis data atau tipe data yang tersimpan dalam sebuah variabel.

Untuk memeriksa tipe data pada suatu variabel, kita bisa menggunakan fungsi `type()`.

Contoh:

```
>>> usia = 20  
>>> type(usia)  
<type 'int'>  
>>> usia = "20"  
>>> type(usia)  
<type 'str'>  
>>> usia = '20'  
>>> type(usia)  
<type 'str'>  
>>> usia = 20.5  
>>> type(usia)  
<type 'float'>  
>>> usia = True
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>> usia = True
>>> type(usia)
<type 'bool'>
```

Secara umum, tipe data primitif dalam python dibagi menjadi tiga jenis:

1. Tipe data angka

Tipe data angka dibagi menjadi beberapa jenis lagi:

- int (Integer): bilangan bulat, contoh 32, 22, 12, 10, dsb.
- float: bilangan pecahan, contoh 1.3, 4.2, 22.3, dsb.

Contoh:

```
harga = 12000 #tipe int
berat = 23.12 #float
jarak = 3e3 #float 3000.0, huruf e artinya eksponen 10
```

2. Tipe data teks

Tipe data teks dibagi menjadi dua jenis lagi:

- Char: Karakter, contoh 'R'.
- String: Kumpulan karakter, contoh "aku lagi makan".

Penulisan tipe data teks harus diapit dengan tanda petik. Bisa menggunakan petik tunggal ('...'), ganda ("..."), dan tiga ('''...''' atau ''''...''').

Contoh:

```
nama = "Agus"  
jenis_kelamin = 'L'  
alamat = """  
Jl. XYZ, No 88. RT Kode,  
Kelurahan BBB, Jakarta  
"""
```

3. Tipe data 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 pertamanya harus kapital dan tanpa tanda petik.

Contoh: bergerak = True

2.3. Konversi Tipe Data

Meskipun Python telah otomatis mendeteksi tipe data yang tersimpan dalam variabel, tapi ada kalanya kita perlu melakukan konversi tipe data.

Misalkan, pada contoh berikut ini:

```
a = 10  
b = 3  
c = a / b  
print c #  
output: 3
```

Pembagian nilai a dan b menghasilkan 3 (integer). Mengapa demikian?

Karena nilai a dan b bertipe integer, maka hasilnya pun berupa integer.

Bagaimana agar hasilnya ada komanya?

Tentu kita harus merubah tipe variabel a dan b menjadi bilangan pecahan (float) dulu, baru setelah itu dibagi.

```
a = 10
b = 3
c = float(a) / float(b) #output: 3.3333333333333335
print c
```

Fungsi float() akan mengubah nilai a menjadi 10.0 dan b menjadi 3.0.

Fungsi-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.

2.4. Operator

Ada enam jenis operator dalam pemrograman yang wajib diketahui:

1. Operator Aritmatika
2. Operator Pembandingan/Relasi
3. Operator Penugasan
4. Operator Logika
5. Operator Bitwise
6. Operator Ternary

2.4.1 Operator Aritmatika

Operator Aritmatika terdiri dari:

Operator	Simbol
----------	--------

Penjumlahan	+
-------------	---

Pengurangan	-
-------------	---

Perkalian	*
-----------	---

Pembagian	/
-----------	---

Sisa Bagi	%
-----------	---

Pemangkatan	**
-------------	----

Mari kita coba dalam program:

```
# Ambil input untuk mengisi nilai
a = int(input("Inputkan nilai a: "))
b = int(input("Inputkan nilai b: "))

# Menggunakan operator penjumlahan
c = a + b
print ("Hasil %d + %d = %d" % (a,b,c))
# Operator Pengurangan
c = a - b
```

```
print ("Hasil %d - %d = %d" % (a,b,c))
```

```
# Operator Perkalian
```

```
c = a * b
```

```
print ("Hasil %d * %d = %d" % (a,b,c))
```

```
# Operator Pembagian
```

```
c = a / b
```

```
print ("Hasil %d / %d = %d" % (a,b,c))
```

```
# Operator Sisa Bagi
```

```
c = a % b
```

```
print ("Hasil %d %% %d = %d" % (a,b,c))
```

```
# Operator Pangkat
```

```
c = a ** b
```

```
print ("Hasil %d ** %d = %d" % (a,b,c))
```

Hasilnya:

```
Inputkan nilai a: 7
Inputkan nilai b: 3
Hasil 7 + 3 = 10
Hasil 7 - 3 = 4
Hasil 7 * 3 = 21
Hasil 7 / 3 = 2
Hasil 7 % 3 = 1
Hasil 7 ** 3 = 343
```

Pada kode program di atas, kita menggunakan *string formatting* untuk mencetak hasil dari masing-masing operasi.

Operator % selain digunakan untuk *string formatting*, operator ini juga digunakan untuk menghitung operasi sisa bagi.

Misal: $5 \% 2$, maka hasilnya 1. Karena sisa dari hasil bagi antara 5 dengan 2 adalah 1.

2.4.2 Operator Penugasan

Seperti namanya, operator ini digunakan untuk memberikan tugas pada variabel.

Misalnya: umur = 17

Maka variabel umur telah kita berikan tugas untuk menyimpan angka 17. Selain menyimpan atau pengisian nilai, ada juga menjumlahkan, mengurangi, perkalian, pembagian, dsb.

Selengkapnya bisa dilihat di tabel berikut.

Operator	Simbol
Pengisian	=
Penjumlahan	+=
Pengurangan	-=
Perkalian	*=
Pembagian	/=
Sisa Bagi	%=
Pemangkatan	**=

Untuk lebih jelasnya, mari kita coba contohnya dalam program.

```
# Ambil input untuk mengisi nilai
a = int(input("Inputkan nilai a: "))
```

```
# contoh operator penugasan untuk mengisi nilai

print ("Nilai a = %d" % a)

# Coba kita jumlahkan nilai a dengan opertor penugasan
a += 5

# contoh operator penugasan untuk menjumlahkan

# Setelah nilai a ditambah 5, coba kita lihat isinya
print ("Nilai setelah ditambah 5:")
print ("a = %d" % a)
```

Hasilnya:

```
Inputkan nilai a: 10
Nilai a = 10
Nilai setelah ditambah 5:
a = 15
```

Pada awalnya kita mengisi nilai variabel a dengan 4. Kemudian dilakukan penjumlahan atau ditambah 5.

a += 5

Penjumlahan tersebut sama maksudnya seperti ini:

a = a + 5

Artinya, kita mengisi nilai variabel a dengan nilai a sebelumnya, lalu ditambah 5.

Contoh Program:

```
# Ambil input untuk mengisi nilai
a = int(input("Inputkan nilai a: "))

# contoh operator penugasan untuk mengisi nilai

print ("Nilai a = %d" % a)

# Coba kita jumlahkan nilai a dengan opertor penugasan
a += 5
# contoh operator penugasan untuk menjumlahkan

# Setelah nilai a ditambah 5, coba kita lihat isinya
print ("Nilai setelah ditambah 5:")
print ("a = %d" % a)
```

hasilnya:

```
Inputkan nilai a: 4
Nilai a = 4
Nilai setelah ditambah 5:
a = 9
```

2.4.3 Operator Pembandingan

Operator ini digunakan untuk membandingkan dua buah nilai. Operator ini juga dikenal dengan operator relasi dan sering digunakan untuk membuat sebuah logika atau kondisi.

Opertor ini terdiri dari:

Operator	Simbol
Lebih Besar	>

Operator	Simbol
Lebih Kecil	<
Sama Dengan	==
Tidak Sama dengan	!=
Lebih Besar Sama dengan	>=
Lebih Kecil Sama dengan	<=

Contoh:

```
a = 9
b = 5
c = a < b
```

Isi variable adalah False, karena nilai 9 lebih kecil dari 5 ($9 < 5$) adalah salah (False).

Contoh Program:

```
a = int(input("Inputkan nilai a: "))
b = int(input("Inputkan nilai b: "))

# apakah a sama dengan b?
c = a == b
print ("Apakah %d == %d: %r" % (a,b,c))

# apakah a < b?
c = a < b
print ("Apakah %d < %d: %r" % (a,b,c))

# apakah a > b?
c = a > b
print ("Apakah %d > %d: %r" % (a,b,c))

# apakah a <= b?
c = a <= b
print ("Apakah %d <= %d: %r" % (a,b,c))

# apakah a >= b?
c = a >= b
print ("Apakah %d >= %d: %r" % (a,b,c))

# apakah a != b?
```



```
c = a != b
print ("Apakah %d != %d: %r" % (a,b,c))
```

Hasilnya:

```
Inputkan nilai a: 10
Inputkan nilai b: 15
Apakah 10 == 15: False
Apakah 10 < 15: True
Apakah 10 > 15: False
Apakah 10 <= 15: True
Apakah 10 >= 15: False
Apakah 10 != 15: True
```

2.4.4 Operator Logika

Operator logika digunakan untuk membuat operasi logika, seperti logika AND, OR, dan NOT.

Operator logika terdiri dari:

Nama	Simbol di Python
Logika AND	And
Logika OR	Or
Negasi/kebalikan Not	

Contoh:

```
a = True
b = False

# Logika AND
c = a and b
print ("%r and %r = %r" % (a,b,c))
```

```
# Logika OR
c = a or b
print ("%r or %r = %r" % (a,b,c))

# Logika Not
c = not a
print ("not %r = %r" % (a,c))
```

Hasilnya:

```
True and False = False
True or False = True
not True = False
```

2.4.5 Operator Bitwise

Operator Bitwise adalah operator untuk melakukan operasi berdasarkan bit/biner.

Operator ini terdiri dari:

Nama	Simbol
AND	&
OR	
XOR	^
Negasi/kebalikan	~
Left Shift	<<
Right Shift	>>

Hasil operasi dari operator ini agak sulit dipahami, kalau kita belum paham operasi bilangan biner.

Contoh:

Misalnya, terdapat variabel $a = 60$ dan $b = 13$.

Bila dibuat dalam bentuk biner, akan menjadi seperti ini:

$a = 00111100$

$b = 00001101$

Kemudian, dilakukan operasi bitwise

Operasi AND

$a = 00111100$

$b = 00001101$

$a \& b = 00001100$

Operasi OR

$a = 00111100$

$b = 00001101$

$a \mid b = 00111101$

Operasi XOR

$a = 00111100$

$b = 00001101$

$a \wedge b = 00110001$

Operasi NOT (Negasi/kebalikan)

a = 00111100
~a = 11000011

Konsepnya memang hampir sama dengan operator Logika. Namun, Bitwise digunakan untuk biner.

Contoh:

```
a = int(input("Masukan nilai a: "))
b = int(input("Masukan nilai b: "))

# Operasi AND
c = a & b
print ("a & b = %s" % c)

# Operasi OR
c = a | b
print ("a | b = %s" % c)

# Operasi XOR
c = a ^ b
print ("a ^ b = %s" % c)

# Operasi Not
c = ~a
print ("~a = %s" % c)

# Operasi shift left (tukar posisi biner)
c = a << b
```

```
print ("a << b = %s" % c)

# Operasi shift right (tukar posisi biner)
c = a >> b
print ("a >> b = %s" % c)
```

Hasilnya:

```
Masukan nilai a: 10
Masukan nilai b: 15
a & b = 10
a | b = 15
a ^ b = 5
~a = -11
a << b = 327680
a >> b = 0
```

2.4.6 Operator Ternary

Operator ternary juga dikenal dengan operator kondisi, karena digunakan untuk membuat sebuah ekspresi kondisi seperti percabangam if/else.

Operator ternary sebenarnya tidak ada dalam Python, tapi python punya cara lain untuk menggantikan operator ini.

Pada bahasa pemrograman lain operator ternary menggunakan tanda tanya (?) dan titik dua (:).

kondisi ? <nilai true> : <nilai false>

Contoh:

```
aku = (umur < 10) ? "bocah" : "dewasa"
```

Dalam Python bentuknya berbeda, yaitu menggunakann IF/ELSE dalam satu baris.

<Nilai True> if Kondisi else <Nilai False>

Contoh:

```
umur = int(input("berapa umur kamu? "))  
v_aku = "anak kecil" if umur < 10 else "dewasa"  
print (v_aku)
```

Hasilnya:

```
berapa umur kamu? 11  
dewasa
```

Latihan:

Buatlah sebuah program untuk melakukan perhitungan rata-rata sebuah nilai dari lima buah data yang dimasukkan kedalam variable?

BAB 3. SELECTION & LOOPING PADA PYTHON

Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang kita berikan.

3.1. Struktur Dasar If

Dalam Python, struktur dasar if sangat sederhana, yaitu seperti contoh di bawah ini:

If kondisi:

- Kondisi adalah bagian yang harus dipenuhi agar bagian kode yang ada di bawahnya dijalankan oleh python.
- Setelah kondisi, akhiri dengan tanda titik dua.
- Tulis kode pemrograman yang akan dijalankan apabila kondisi bernilai TRUE.

Contoh:

Usia = 17

If usia == 17:

 Print ("Kamu Sudah Dewasa")

Berikutnya adalah contoh statemen if dan else:

```
n = int(input("Angka? "))
```

```
if n < 0:
```

```
    print ("nilai absolut dari ", n, "adalah", -n)
```

else:

```
print ("nilai absolut dari ", n, "adalah", n)
```

Penulisan blok */f*, harus diberikan indentasi tab atau spasi 2x.

Contoh penulisan yang salah:

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

Contoh penulisan yang benar:

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

3.2. Struktur Percabangan If/Elif/Else

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

```
nilai = int(input("Inputkan nilaimu: "))  
if nilai >= 90:  
    grade = "A"  
elif nilai >= 80:  
    grade = "B+"  
elif nilai >= 70:  
    grade = "B"  
elif nilai >= 60:  
    grade = "C+"  
elif nilai >= 50:  
    grade = "C"
```



```
elif nilai >= 40:  
    grade = "D"  
else:  
    grade = "E"  
  
print("Grade: %s" % grade)
```

Hasilnya:

```
Inputkan nilaimu: 80  
Grade: B+
```

3.3. Perulangan

Terdapat dua jenis perulangan 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.

3.4. Perulangan For

contoh perulangan for:

```
ulang = 10  
for i in range(ulang):  
    print ("Perulangan ke-"+str(i))
```

Hasilnya:

```
-  
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
```

Pertama kita menentukan banyak perulangannya sebanyak 10x, Variabel `i` berfungsi untuk menampung indeks, dan fungsi `range()` berfungsi untuk membuat list dengan range dari 0-10. Fungsi `str()` berfungsi merubah tipe data integer ke string.

Contoh perulangan for menggunakan list:

```
item = ['kopi','nasi','teh','jeruk']  
for isi in item:  
    print (isi)
```

Hasilnya:

```
kopi  
nasi  
teh  
jeruk
```

3.5. Perulangan While

Contoh perulangan while:

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

```
while(jawab == 'ya'):
    hitung += 1
    jawab = str(input("Ulang lagi tidak? "))
print ("Total perulangan: " + str(hitung))
```

Hasilnya:

```
Ulang lagi tidak? ya
Ulang lagi tidak? a
Total perulangan: 2
```

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

```
jawab = 'ya'
hitung = 0
while(True):
    hitung += 1
    jawab = str(input("Ulang lagi tidak? "))
    if jawab == 'tidak':
        break
print ("Total perulangan: " + str(hitung))
```

Hasilnya:

```
Ulang lagi tidak? ya
Ulang lagi tidak? ya
Ulang lagi tidak? tidak
Total perulangan: 3
```

Latihan:

Buatlah sebuah program untuk melakukan tebak angka dari angka yang dimasukkan dengan ketentuan sebagai berikut:

- Buatlah sebuah variable angka yang kemudian diberikan default atau initial value.
- Lalu user diminta untuk memasukkan tebakan angka.
- Jika angka yang dimasukkan benar maka tampilkan selamat tebakan anda berhasil.
- Jika angka yang dimasukkan < dari variable angka initial value maka tampilkan angka yang dimasukkan lebih kecil dari angka yang ditebak.
- Jika angka yang dimasukkan > dari variable angka initial value maka tampilkan angka yang dimasukkan lebih besar dari angka yang ditebak.
- Program akan terus berulang sampai angka berhasil ditebak.

BAB 4. FUNGSI PADA PYTHON

Salah satu tujuan dibuat fungsi adalah kita dapat memecah program besar menjadi sub program yang lebih sederhana dan memudahkan untuk maintain program yang kita buat. Python memiliki struktur pembuatan fungsi yang simple. Struktur pembuatan fungsi adalah sebagai berikut:

```
def nama_fungsi():  
    print ("Hello ini Fungsi")
```

- Pembuatan fungsi menggunakan python dimulai dengan kata kunci (keyword) **def**. dilanjutkan dengan membuat nama fungsi.
- Sama seperti blok kode yang lain, kita juga harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi.
- Anda bisa membuat fungsi dengan atau tanpa argumen. Jika tanpa argument, maka cukup buat tanda kurung yang tidak diisi apapun. Tetapi jika mengandung argument, maka tulis nama argument di dalam tanda kurung.
- Jangan lupa untuk mengakhiri baris dengan tanda titik dua.
- Tulis kode-kode yang digunakan untuk menghasilkan nilai sesuai tujuan fungsi ini.

Contoh fungsi sederhana:

```
# Membuat Fungsi  
def f_pesan():  
    print ("Hello")  
  
## Pemanggilan Fungsi  
F_pesan()
```

4.1. Fungsi dengan parameter

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

```
# Membuat Fungsi
```

```
def f_pesan_parameter(vpesan):  
    print (vpesan)
```

```
## Pemanggilan Fungsi
```

```
F_pesan_parameter("Hello")
```

Contoh fungsi dengan jumlah parameter lebih dari satu:

```
def f_luas_segitiga(alas, tinggi):  
    luas = (alas * tinggi) / 2  
    print ("Luas segitiga: %f" % luas);
```

```
# Pemanggilan fungsi
```

```
F_luas_segitiga(3, 4)
```

Hasilnya:

Luas segitiga: 6.000000

4.2. Fungsi yang mengembalikan nilai

Cara mengembalikan nilai adalah menggunakan kata kunci return lalu diikuti dengan nilai atau variabel yang akan dikembalikan.

Contoh:

```
def f_luas_persegi(sisi):
```

```
luas = sisi * sisi  
return luas
```

```
# pemanggilan fungsi  
print ("Luas persegi: %d" % f_luas_persegi(6))
```

Hasilnya:

Luas persegi: 36

Apa bedanya dengan fungsi `f_luas_segitiga()` yang tadi?

Pada fungsi `f_luas_segitiga()` kita melakukan print dari hasil pemrosesan secara langsung di dalam fungsinya.

Sedangkan fungsi `f_luas_persegi()`, kita melakukan print pada saat pemanggilannya.

Jadi, fungsi `f_luas_persegi()` akan bernilai sesuai dengan hasil yang dikembalikan.

Sehingga kita dapat memanfaatkannya untuk pemrosesan berikutnya.

Misalnya seperti ini:

```
# rumus: sisi x sisi  
def f_luas_persegi(sisi):  
    luas = sisi * sisi  
    return luas
```

```
# rumus: sisi x sisi x sisi  
def f_volume_persegi(sisi):
```

```
volume = f_luas_persegi(sisi) * sisi
```

4.3. Sifat Variabel Dalam Fungsi

Contoh program:

```
# membuat variabel global
nama = "BUDI"

def help():
    # ini variabel lokal
    nama = "AGUS"
    # mengakses variabel lokal
    print ("Nama: %s" % nama)

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

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

Hasilnya:

```
Nama: BUDI
Nama: AGUS
```

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 kita memanggil fungsi `help()` maka nilai yang akan tampil adalah nilai yang ada di dalam fungsi `help()`.

Kenapa tidak tampil yang global?

Karena 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.

4.4. Fungsi Built-In Python

Fungsi built-in adalah fungsi yang sudah disediakan oleh Python, kita tinggal memakainya saja. Untuk detail serta contohnya Anda bisa mengunjungi situs <http://www.pythonindo.com/fungsi-built-in-python/>

Metode	Deskripsi
<code>abs()</code>	Mengembalikan nilai absolut dari suatu bilangan
<code>all()</code>	Mengembalikan True jika semua anggota data iterable bernilai True
<code>any()</code>	Menguji apakah satu atau lebih anggota iterable bernilai True
<code>ascii()</code>	Mengembalikan string yang berisi karakter yang bisa dicetak
<code>bin()</code>	Mengubah integer menjadi string biner
<code>bool()</code>	Mengubah suatu nilai menjadi nilai Boolean
<code>bytearray()</code>	Mengembalikan array dari byte
<code>bytes()</code>	Mengembalikan objek byte immutable
<code>callable()</code>	Menguji apakah sebuah objek bisa dipanggil
<code>chr()</code>	Mengembalikan karakter (string) dari suatu integer

<code>classmethod()</code>	Mengembalikan metode class untuk fungsi yang diberikan
<code>compile()</code>	Mengembalikan kode objek Python
<code>complex()</code>	Menciptakan Bilangan Kompleks
<code>delattr()</code>	Menghapus atribut dari objek
<code>dict()</code>	Menciptakan dictionary
<code>dir()</code>	Mengembalikan atribut objek
<code>divmod()</code>	Mengembalikan tuple dari hasil bagi dan sisa
<code>enumerate()</code>	Mengembalikan objek enumerasi
<code>eval()</code>	Menjalankan kode python dalam program
<code>exec()</code>	Mengeksekusi program secara dinamis
<code>filter()</code>	Menghasilkan iterator dari anggota – anggota yang bernilai True
<code>float()</code>	Mengembalikan bilangan float dari bilangan atau string
<code>format()</code>	Mengembalikan string yang terformat
<code>frozenset()</code>	Mengembalikan objek frozenset immutable
<code>getattr()</code>	Mengembalikan nilai dari atribut objek
<code>globals()</code>	Mengembalikan dictionary table simbol global
<code>hasattr()</code>	Menguji apakah objek memiliki atribut tertentu
<code>hash()</code>	Mengembalikan nilai hash dari suatu objek
<code>help()</code>	Memanggil sistem help built-in
<code>hex()</code>	Mengubah integer menjadi hexadecimal
<code>id()</code>	Mengembalikan identitas objek
<code>input()</code>	Membaca masukan dan mengembalikan string
<code>int()</code>	Mengembalikan integer dari suatu bilangan atau string
<code>isinstance()</code>	Menguji apakah suatu objek merupakan instance dari class

<code>issubclass()</code>	Menguji apakah suatu objek merupakan subclass dari class
<code>iter()</code>	Mengembalikan iterator objek
<code>len()</code>	Mengembalikan panjang objek
<code>list()</code>	Menciptakan list baru
<code>locals()</code>	Mengembalikan dictionary dari tabel simbol lokal
<code>map()</code>	Mengaplikasikan fungsi terhadap list dan mengembalikan list baru
<code>max()</code>	Mengembalikan nilai terbesar
<code>memoryview()</code>	Mengembalikan view memori dari argumen
<code>min()</code>	Mengembalikan nilai terkecil
<code>next()</code>	Mengambil nilai selanjutnya dari iterator
<code>object()</code>	Menciptakan objek
<code>oct()</code>	Mengubah integer menjadi octal
<code>open()</code>	Mengembalikan objek file
<code>ord()</code>	Mengembalikan kode unicode dari karakter
<code>pow()</code>	Mengembalikan pangkat bilangan
<code>print()</code>	Mencetak objek yang diberikan
<code>property()</code>	Mengembalikan atribut properti
<code>range()</code>	Mengembalikan integer yang berurut dari start sampai stop
<code>repr()</code>	Mengembalikan karakter yang bisa dicetak dari suatu objek
<code>reversed()</code>	Mengembalikan iterator sequence yang dibalik
<code>round()</code>	Membulatkan bilangan berkoma
<code>set()</code>	Mengembalikan set
<code>setattr()</code>	Mengeset nilai dari atribut objek
<code>slice()</code>	Menciptakan objek terpotong sesuai range

<code>sorted()</code>	Mengembalikan list yang disortir
<code>staticmethod()</code>	Menciptakan metode static dari suatu fungsi
<code>str()</code>	Mengembalikan string dari objek
<code>sum()</code>	Menambahkan semua anggota dari iterable
<code>super()</code>	Mengacu ke parent class dari objek
<code>tuple()</code>	Menciptakan tuple
<code>type()</code>	Mengembalikan tipe objek
<code>vars()</code>	Mengembalikan atribut <code>__dict__</code> dari kelas
<code>zip()</code>	Mengembalikan iterator dari tuple
<code>__import__()</code>	Fungsi lanjut yang dipanggil menggunakan import

Latihan:

1. Buatlah sebuah fungsi yang dapat memanggil dirinya sendiri, untuk menghitung nilai factorial dari angka yang dimasukkan oleh user?
2. Buatlah sebuah program dengan 4 buah fungsi yaitu penjumlahan, pengurangan, perkalian dan pembagian? Pertama user diminta untuk memilih menu yang akan digunakan lalu diminta untuk memasukkan angka pertama dan angka kedua lalu secara otomatis akan menampilkan hasilnya.

BAB 5. LIST PADA PYTHON

Salah satu kelebihan Python terletak pada kemampuannya mengolah struktur data. Struktur data adalah kumpulan elemen-elemen data seperti angka atau karakter yang disusun sedemikian rupa, misalnya dengan memberikan nomor yang spesifik pada elemen-elemen tersebut.

Dalam Python, struktur data yang paling sederhana disebut dengan sequence. Setiap elemen dalam sebuah sequence diberi nomor khusus yang menunjukkan lokasi dari elemen tersebut, atau biasa disebut dengan index. Index dimulai dari angka 0 seperti. Sequence ini memiliki keterkaitan dengan penggunaan list dan tuple pada pemrograman Python.

Python memiliki beberapa jenis sequence. Yang paling umum digunakan adalah list dan tuples. Yang membedakan list dan tuples adalah sifatnya. Anda dapat mengubah sebuah list, tetapi tidak dapat melakukannya pada tuples. Jadi anda bisa menambah elemen-elemen baru ke dalam list, sedangkan pada tuples tidak bisa dilakukan.

5.1. List

List adalah struktur data pada python yang mampu menyimpan lebih dari satu data, seperti array.

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

Contoh:

```
# Membuat List kosong  
L_warna = []
```

Membuat list dengan isi 1 item

```
L_hobi = ["membaca"]
```

Apabila list-nya memiliki lebih dari satu isi, maka kita bisa memisahkannya dengan tanda koma.

Contoh:

```
L_buah = ["jeruk", "apel", "mangga", "duren"]
```

list dapat diisi dengan tipe data apa saja, string, integer, float, double, boolean, object, dan sebagainya.

Kita juga bisa mencampur isinya.

Contoh:

```
L_laci = ["book", 17, True, 34.12]
```

Pada contoh list sebelumnya maka terdapat empat jenis tipe data pada list *L_laci*:

1. "book" adalah tipe data string;
2. 17 adalah tipe data integer;
3. True adalah tipe data boolean;
4. dan 34.12 adalah tipe data float.

5.2. Cara Mengambil Nilai dari List

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:

```
# list nama-nama buah
buah = ["apel", "anggur", "mangga", "jeruk"]
```

```
# Misanya kita ingin mengambil mangga
# Maka indeksinya adalah 2
print (buah[2])
```

Akan menghasilkan output:

"mangga"

Contoh Program dengan List:

```
L_my_friends = ["Anggun", "Dian", "Agung", "Agus", "Adam"]

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

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

Hasil outputnya:

```
Isi list L_my_friends indeks ke-3 adalah: Agus
Semua teman: ada 5 orang
Anggun
Dian
```

Agung

Agus

Adam

5.3. Mengganti Nilai List

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

Contoh:

```
# list awal
buah = ["pisang", "jambu", "mangga", "pepaya"]
# mengubah nilai index ke-2
buah[2] = "sawo"
```

Maka "mangga" akan diganti dengan "sawo".

```
["pisang", "jambu", "sawo", "pepaya"]
```

5.4. Menambahkan Item List

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

1. `prepend(item)` menambahkan item dari depan;
2. `append(item)` menambahkan item dari belakang.
3. `insert(index, item)` menambahkan item dari indeks tertentu

Contoh:

```
#list awal
buah = ["pisang", "jambu", "mangga", "pepaya"]
```



```
# Tambahkan sawo
buah.append("sawo")
```

Hasilnya "sawo" akan ditambahkan setelah item terakhir.

```
["pisang", "jambu", "mangga", "pepaya", "sawo"]
```

Metode yang kedua menggunakan `prepend()`, Metode `prepend()` akan menambahkan item dari depan atau awal list.

Contoh:

```
#list awal
buah = ["pisang", "jambu", "mangga", "pepaya"]
buah.prepend("sawo")
```

Maka "sawo" akan ditambahkan pada awal list.

```
["sawo", "pisang", "jambu", "mangga", "pepaya"]
```

Selain `prepend()` dan `append()` kita juga bisa menggunakan *method* `insert()` untuk menambahkan item pada indeks tertentu.

Contoh:

```
#list awal
buah = ["jeruk", "apel", "mangga", "duren"]
buah.insert(2, "pisang")
```

maka "pisang" akan ditambahkan di index ke 2

Contoh Program:

```
hobi = []
stop = False
i = 0
# Mengisi hobi
while (not stop):
    hobi_baru = str(input("Inputkan hobi yang ke-{}: ".format(i)))
    hobi.append(hobi_baru)
    # Increment i
    i += 1
    tanya = str(input("Mau isi lagi? (y/t): "))
    if (tanya == "t"):
        stop = True
# Cetak Semua Hobi
print("=" * 10)
print("Kamu memiliki {} hobi".format(len(hobi)))
for hb in hobi:
    print("- {}".format(hb))
```

Hasilnya:

```
Inputkan hobi yang ke-0: baca
Mau isi lagi? (y/t): y
Inputkan hobi yang ke-1: nonton
Mau isi lagi? (y/t): t
=====
Kamu memiliki 2 hobi
- baca
- nonton
```

5.5. Menghapus Item List

Untuk menghapus salah satu isi dari *List*, kita bisa menggunakan perintah `del`.

Perintah `del` akan menghapus sebuah variabel dari memori.

Contoh:

```
# Membuat List
todo_list = [
    "Balajar Python",
    "Belajar Django",
    "Belajar MongoDB",
    "Belajar Mobile",
    "Belajar Flask"
]

# Misalkan kita ingin menghapus "Belajar Mobile"
# yang berada di indeks ke-3
del todo_list[3]

print (todo_list)
```

Hasilnya, "Belajar Mobile" akan dihapus:

```
['Balajar Python', 'Belajar Django', 'Belajar MongoDB', 'Belajar Flask']
```

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

Contoh:

```
# List awal
L_a = ["a", "b", "c", "d"]
# kemudian kita hapus b
L_a.remove("b")

print (L_a)
```

Hasilnya:

```
["a", "c", "d"]
```

5.6. Memotong List

Seperti string, list juga dapat dipotong-potong.

Contoh:

```
# list warna
warna = ["merah", "hijau", "kuning", "biru", "pink", "ungu"]

# Kita potong dari indeks ke-2 sampai ke-5
print (warna[2:5])
```

Hasilnya:

```
['kuning', 'biru', 'pink']
```

5.7. Operasi List

Ada beberapa operasi yang bisa dilakukan terhadap List, diantaranya:

- Penggabungan (+)

- Perkalian (*)

Contoh:

```
# list lagu
list_lagu = [
    "No Women, No Cry",
    "Dear God"
]

# list lagu favorit
playlist_favorit = [
    "Break Out",
    "Now Loading!!!"
]

# Mari kita gabungkan keduanya
semua_lagu = list_lagu + playlist_favorit

print (semua_lagu)
```

Hasilnya:

```
['No Women, No Cry', 'Dear God', 'Break Out', 'Now Loading!!!']
```

Sedangkan untuk operasi perkalian hanya dapat dilakukan dengan bilangan.

Contoh:

```
# playlist lagu favorit
playlist_favorit = [
```

```
"Break Out",
"Now Loading!!!"
]
```

```
# ulangi sebanyak 5x
ulang = 5
```

```
now_playing = playlist_favorit * ulang
```

```
print (now_playing)
```

Hasilnya:

```
['Break Out', 'Now Loading!!!', 'Break Out', 'Now Loading!!!', 'Break Out', 'Now
Loading!!!', 'Break Out', 'Now Loading!!!', 'Break Out', 'Now Loading!!!']
```

Berikut adalah operasi dasar List:

Python Expression	Hasil	Penjelasan
<code>len([1, 2, 3, 4])</code>	<code>4</code>	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>['Halo!'] * 4</code>	<code>['Halo!', 'Halo!', 'Halo!', 'Halo!']</code>	Repetition
<code>2 in [1, 2, 3]</code>	<code>True</code>	Membership
<code>for x in [1,2,3]: print (x,end = ' ')</code>	<code>1 2 3</code>	Iteration

Method dan Fungsi Build-in Pada List Python

Python menyertakan fungsi built-in sebagai berikut:

Python Function	Penjelasan
<code>cmp(list1, list2) #</code>	Tidak lagi tersedia dengan Python 3
<code>len(list)</code>	Memberikan total panjang list.
<code>max(list)</code>	Mengembalikan item dari list dengan nilai maks.
<code>min(list)</code>	Mengembalikan item dari list dengan nilai min.
<code>list(seq)</code>	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut:

Python Methods	Penjelasan
<code>list.append(obj)</code>	Menambahkan objek obj ke list
<code>list.count(obj)</code>	Jumlah pengembalian berapa kali obj terjadi dalam list
<code>list.extend(seq)</code>	Tambahkan isi seq ke list
<code>list.index(obj)</code>	Mengembalikan indeks terendah dalam list yang muncul obj
<code>list.insert(index, obj)</code>	Sisipkan objek obj ke dalam list di indeks offset
<code>list.pop(obj = list[-1])</code>	Menghapus dan mengembalikan objek atau obj terakhir dari list
<code>list.remove(obj)</code>	Removes object obj from list
<code>list.reverse()</code>	Membalik list objek di tempat
<code>list.sort([func])</code>	Urutkan objek list, gunakan compare func jika diberikan

5.8. List Multi Dimensi

Pada contoh-contoh di atas, kita 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, dsb.

Contoh:

```
# List minuman dengan 2 dimensi
```

```
list_minuman = [  
    ["Kopi", "Susu", "Teh"],  
    ["Jus Apel", "Jus Melon", "Jus Jeruk"],  
    ["Es Kopi", "Es Campur", "Es Teler"]  
]
```

```
# Cara mengakses list multidimensi  
# misalkan kita ingin mengambil "es kopi"  
print (list_minuman[2][0])
```

Hasil outputnya:

"Es Kopi"

Contoh untuk menampilkan semua data pada list multidimensi

```
# List minuman dengan 2 dimensi  
list_minuman = [  
    ["Kopi", "Susu", "Teh"],  
    ["Jus Apel", "Jus Melon", "Jus Jeruk"],  
    ["Es Kopi", "Es Campur", "Es Teler"]  
]
```

```
for menu in list_minuman:  
    for minuman in menu:  
        print (minuman)
```

Hasilnya:

Kopi

Susu

Teh
Jus Apel
Jus Melon
Jus Jeruk
Es Kopi
Es Campur
Es Teler

Latihan:

Buatlah program dengan menggunakan konsep List untuk menampilkan data yang diinput oleh user. Berikut contoh tampilannya:

Masukkan jumlah data nilai mahasiswa yang ingin di masukkan = 2
<diinput oleh user>

Data ke -1 <ditampilkan oleh program berdasarkan indeks List>

Nama Siswa : Agus <diinput oleh user>

Nilai MID : 85 <diinput oleh user>

Nilai Final : 75 <diinput oleh user>

Data ke -2 <ditampilkan oleh program berdasarkan indeks List>

Nama Siswa : Ruby <diinput oleh user>

Nilai MID : 65 <diinput oleh user>

Nilai Final : 55 <diinput oleh user>

<Hasil tampilan dari program setelah selesai memasukkan data pada array>

No.	Nama Siswa	Nilai MID	Nilai Final	Hasil Ujian	Grade

1	Agus	85	75	79	B
2	Ruby	65	55	59	D

Ketentuan Perhitungan:

- Hasil ujian adalah $(40\% * \text{Nilai MID}) + (60\% * \text{Nilai Final})$

Grade ditentukan sebagai berikut :

Jika Hasil Ujian ≥ 80 maka Grade = A

Jika Hasil Ujian ≥ 70 dan < 80 maka Grade = B

Jika Hasil Ujian ≥ 60 dan < 70 maka Grade = C

Jika Hasil Ujian ≥ 50 dan < 60 maka Grade = D

Jika Hasil Ujian < 50 maka Grade = E

BAB 6. TUPLES & SET PADA PYTHON

Tuple seperti list, hanya saja elemen-elemen yang ada didalamnya tidak bisa diubah kembali (*immutable*). Langkah membuat tuple sangat mudah. Jika anda menulis beberapa nilai yang dipisah dengan koma, baik tanpa atau menggunakan tanda kurung biasa (bukan kurung siku yang merupakan ciri dari list), maka anda sudah membuat tuples:

Contoh:

angka =1,2,3 atau dengan tanda kurung angka =(1,2,3)

Jika menulis satu nilai di dalam tanda kurung, maka anda bisa menulis nilai itu dengan diakhiri koma, misalnya:

angka = (27,)

Print(angka)

Jika tidak ditambahkan koma maka akan dianggap sebagai string.

6.1. Membuat dan Mengakses Tuple

Membuat tuple kosong

kosong = ()

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 = ('agus', 'budi', 'ali')

```
# mengakses nilai tuple  
print(nama[1])
```

Apabila kita mencoba merubah nilainya maka akan terjadi error karena sifat dari tuple adalah (*immutable*)

6.2. Memotong Tuple

Sama seperti *list*, di Tuple juga kita bisa melakukan slicing.

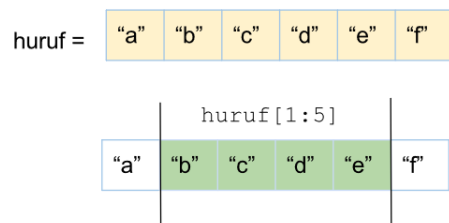
Contoh:

```
# data tuple  
web = (123, 'software isa', 'https://www.softwareisa.com')  
  
# lalu kita ingin potong agar ditampilkan  
# dari indeks nomer 1 sampai 2  
print(web[1:2])
```

Maka hasilnya:

('software isa',)

Logikanya sama seperti di list.



6.3. Mengambil Panjang Tuple

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

Contoh:

```
# data Tuple
```

```
hari = ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jum\'at', 'Sabtu', 'Minggu')
```

```
# Mengambil panjang tuple hari
```

```
print("Jumlah hari: %d" % len(hari))
```

Hasilnya:

```
>>> hari = ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jum\'at', 'Sabtu')
>>> print("Jumlah hari: %d" % len(hari))
Jumlah hari: 6
>>> hari = ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jum\'at', 'Sabtu', 'Minggu')
>>> print("Jumlah hari: %d" % len(hari))
Jumlah hari: 7
>>> █
```

6.4. Tuple Nested

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

Contoh:

```
tuple1 = "aku", "cinta", "kamu"
```

```
tuple2 = "selama", 3, "tahun"
```

```
tuple3 = (tuple1, tuple2) # <- nested tuple
```

tuple3 akan berisi nilai dari tuple1 dan tuple2.

```
>>> tuple1 = "aku", "cinta", "kamu"
>>> tuple2 = "selama", 3, "tahun"
>>> tuple3 = (tuple1, tuple2) # <- nested tuple
>>> tuple3
(('aku', 'cinta', 'kamu'), ('selama', 3, 'tahun'))
>>> print(tuple3[1])
('selama', 3, 'tahun')
>>> print(tuple3[1][0])
selama
>>> █
```

6.5. Iterasi Pada Tuple

Kita bisa menggunakan for untuk melakukan iterasi pada tiap anggota dalam tuple.

Contoh:

```
nama = ('Agus', 'Budi')
```

```
for name in nama:
```

```
    print('Hi', name)
```

Hasilnya:

Hi Agus

Hi Budi

6.6. Jenis-Jenis Method (Fungsi) Pada Tuple

Fungsi	Deskripsi
<code>all()</code>	Mengembalikan True jika semua anggota tuple adalah benar (tidak ada yang kosong)
<code>any()</code>	Mengembalikan True jika salah satu atau semua bernilai benar. Jika tuple kosong, maka akan mengembalikan False .
<code>enumerate()</code>	Mengembalikan objek enumerasi. Objek enumerasi adalah objek yang terdiri dari pasangan indeks dan nilai.
<code>len()</code>	Mengembalikan panjang (jumlah anggota) tuple
<code>max()</code>	Mengembalikan anggota terbesar di tuple
<code>min()</code>	Mengembalikan anggota terkecil di tuple
<code>sorted()</code>	Mengambil anggota tuple dan mengembalikan list baru yang sudah diurutkan
<code>sum()</code>	Mengembalikan jumlah dari semua anggota tuple
<code>tuple()</code>	Mengubah sequence (list, string, set, dictionary)

6.7. Set

Set adalah salah satu tipe data di Python yang tidak berurut (*unordered*). Set memiliki anggota yang unik (tidak ada duplikasi). Jadi misalnya kalau kita meletakkan dua anggota yang sama di dalam set, maka otomatis set akan menghilangkan yang salah satunya.

Set bisa digunakan untuk melakukan operasi himpunan matematika seperti gabungan, irisan, selisih, dan lain - lain.

Membuat Set

Set dibuat dengan meletakkan anggota - anggotanya di dalam tanda kurung kurawal { }, dipisahkan menggunakan tanda koma. Kita juga bisa membuat set dari list dengan memasukkan list ke dalam fungsi set()

Set bisa berisi data campuran, baik integer, float, string, dan lain sebagainya. Akan tetapi set tidak bisa berisi list, set, dan dictionary.

Contoh:

set integer

```
my_set = {1,2,3}
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3}
```

set dengan menggunakan fungsi set()

```
my_set = set([1,2,3])
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3}
```

set data campuran

```
my_set = {1, 2.0, "Python", (3,4,5)}
```



```
print(my_set)
```

Hasilnya:

```
{'Python', 1, 2.0, (3, 4, 5)}
```

bila kita mengisi duplikasi, set akan menghilangkan salah satu

```
my_set = {1,2,2,3,3,3}
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3}
```

set tidak bisa berisi anggota list

contoh berikut akan muncul error TypeError

```
my_set = {1,2,[3,4,5]}
```

```
>>> my_set = {1,2,[3,4,5]}
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    my_set = {1,2,[3,4,5]}
TypeError: unhashable type: 'list'
>>> |
```

Untuk membuat set kosong, kita tidak bisa menggunakan { }, karena itu akan dianggap sebagai dictionary. Kita harus menggunakan fungsi set() tanpa argumen untuk membuat set kosong.

```
>>> # membuat variabel a dengan {}
>>> a = {}
>>> print(type(a))
<class 'dict'>

>>> # harus menggunakan fungsi set()
>>> a = set()
>>> print(type(a))
<class 'set'>
```

6.8. Mengubah dan Menghapus Anggota Set

Set bersifat *mutable*. Tapi, karena set adalah tipe data tidak berurut (unordered), maka kita tidak bisa menggunakan indeks. Set tidak mendukung indeks ataupun slicing.

Untuk menambah satu anggota ke dalam set, kita bisa menggunakan fungsi `add()`, dan untuk menambahkan beberapa anggota sekaligus kita bisa menggunakan fungsi `update()`. List, tuple, maupun string bisa digunakan sebagai masukan dari fungsi `update()`.

membuat set baru

```
my_set = {1,2,3}
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3}
```

menambah satu anggota

```
my_set.add(4)
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3, 4}
```

menambah beberapa anggota , set akan menghilangkan duplikasi

```
my_set.update([3,4,5,6])
```

```
print(my_set)
```

Hasilnya:

```
{1,2,3,4,5,6}
```

Menghapus Anggota Set

Kita bisa menghapus anggota set dengan menggunakan fungsi `discard()` dan `remove()`. Perbedaananya, fungsi `discard()` tidak akan memunculkan error bila anggota yang ingin dihapus ternyata tidak ada di dalam set, sedangkan `remove()` sebaliknya.

membuat set baru

```
my_set = {1, 2, 3, 4, 5}
```

```
print(my_set)
```

Hasilnya:

```
{1, 2, 3, 4, 5}
```

menghapus 4 dengan discard

```
my_set.discard(4)
```

```
print(my_set)
```

Hasilnya: {1, 2, 3, 5}

menghapus 5 dengan remove

```
my_set.remove(5)
```

```
print(my_set)
```

Hasilnya : {1, 2, 3}

anggota yang mau dihapus tidak ada dalam set, discard tidak akan memunculkan error

```
my_set.discard(6)
```

hasilnya:

{1, 2, 3}

Selain discard() dan remove(), kita bisa menghapus anggota set dengan menggunakan fungsi pop(). Dengan menggunakan fungsi pop(), kita menghapus salah satu anggota secara acak (random).

Untuk mengosongkan atau menghapus seluruh anggota set, kita bisa menggunakan fungsi clear()

Contoh:

membuat set baru , output : set berisi anggota yang unik

```
my_set = set("HelloPython")
```

```
print(my_set)
```

Hasilnya:

```
{'t', 'l', 'P', 'y', 'H', 'n', 'e', 'h', 'o'}
```

```
# pop anggota , output: anggota acak
```

```
print(my_set.pop())
```

Hasilnya:

```
t
```

```
# pop anggota lainnya , output: anggota acak
```

```
print(my_set.pop())
```

Hasilnya:

```
l
```

```
# mengosongkan set , output: set()
```

```
my_set.clear()
```

```
print(my_set)
```

Hasilnya:

```
set()
```

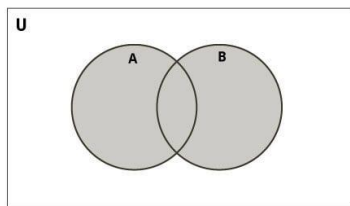
6.9. Operasi Set

Set dapat digunakan untuk melakukan operasi himpunan matematika seperti gabungan, irisan, selisih, dan komplemen.

Mari kita ambil dua contoh set berikut:

```
>>> A = {1, 2, 3, 4, 5}
>>> B = {4, 5, 6, 7, 8}
```

Operasi Gabungan (Union)



Gabungan (union) dari A dan B adalah himpunan atau set anggota yang ada di A dan B.

Gabungan dapat dibuat dengan menggunakan operator palang (`|`). Selain itu juga bisa dilakukan dengan menggunakan fungsi `union()`.

Contoh:

Membuat set A and B

```
A = {1, 2, 3, 4, 5}
```

```
B = {4, 5, 6, 7, 8}
```

Gabungan menggunakan operator `|`

```
print(A | B)
```

Hasilnya:

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Menggunakan fungsi union()

A.union(B)

Hasilnya:

{1, 2, 3, 4, 5, 6, 7, 8}

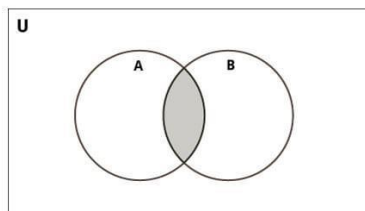
atau

B.union(A)

Hasilnya:

{1, 2, 3, 4, 5, 6, 7, 8}

Operasi Irisan (Intersection)



Irisan (intersection) dari A dan B adalah himpunan atau set anggota yang sama di A dan B.

Irisan dilakukan dengan menggunakan operator jangkar (&). Irisan juga bisa dilakukan dengan menggunakan fungsi intersection().

Contoh:

Membuat set A and B

```
A = {1, 2, 3, 4, 5}
```

```
B = {4, 5, 6, 7, 8}
```

Irisan menggunakan operator &

```
print(A & B)
```

Hasilnya:

```
{4,5}
```

Menggunakan fungsi intersection()

```
A.intersection(B)
```

Hasilnya:

```
{4,5}
```

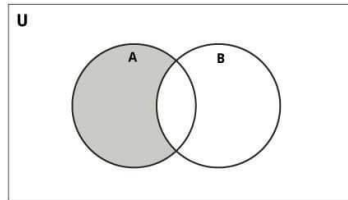
atau

```
B.intersection(A)
```

Hasilnya:

```
{4,5}
```


Operasi Selisih (Difference)



Selisih (intersection) dari A dan B adalah himpunan atau set anggota yang hanya ada di A dan tidak ada di B. Begitu juga sebaliknya, ada di B tapi tidak ada di A.

Selisih dilakukan dengan menggunakan operator kurang (-). Bisa juga dengan menggunakan fungsi difference().

Contoh:

membuat A and B

A = {1, 2, 3, 4, 5}

B = {4, 5, 6, 7, 8}

Menggunakan operator - pada A

print(A - B)

Hasilnya:

{1, 2, 3}

Atau

A.difference(B)

Hasilnya:

{1, 2, 3}

Menggunakan operator - pada B

print(B - A)

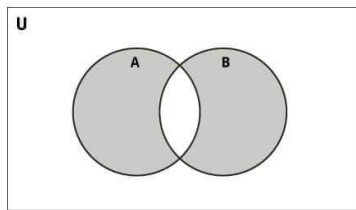
Atau

`B.difference(A)`

Hasilnya:

`{8, 6, 7}`

Operasi Komplemen (Symmetric Difference)



Operasi komplemen (symmetric difference) dari A dan B adalah himpunan atau set anggota yang ada di A dan di B, tapi tidak di keduanya.

Komplemen dilakukan dengan menggunakan operator `^`. Bisa juga dengan menggunakan fungsi `symmetric_difference()`.

Contoh:

membuat A and B

`A = {1, 2, 3, 4, 5}`

`B = {4, 5, 6, 7, 8}`

Menggunakan operator `^` pada A

`print(A ^ B)`

Hasilnya:

`{1, 2, 3, 6, 7, 8}`

Atau

```
A.symmetric_difference(B)
```

Hasilnya:

```
{1, 2, 3, 6, 7, 8}
```

Menggunakan operator ^ pada B

```
print(B ^ A)
```

Hasilnya:

```
{1, 2, 3, 6, 7, 8}
```

Atau

```
B.symmetric_difference(A)
```

Hasilnya:

```
{1, 2, 3, 6, 7, 8}
```

Berikut jenis-jenis Metode (Fungsi) Set

Metode	Deskripsi
<code>add()</code>	Menambahkan satu anggota ke set
<code>clear()</code>	Menghapus semua anggota set
<code>copy()</code>	Mengembalikan <i>shallow copy</i> dari set
<code>difference()</code>	Mengembalikan set baru berisi selisih dua atau lebih set
<code>difference_update()</code>	Menghapus semua anggota set lain yang ada di set ini
<code>discard()</code>	Menghapus satu anggota dari set
<code>intersection()</code>	Mengembalikan set baru berisi irisan antara dua atau lebih set
<code>intersection_update()</code>	Mengupdate set dengan irisan set bersangkutan dan set lainnya
<code>isdisjoint()</code>	Mengembalikan True jika dua set tidak memiliki irisan
<code>issubset()</code>	Mengembalikan True jika set lain berisi set ini
<code>issuperset()</code>	Mengembalikan True jika set ini berisi set lain
<code>pop()</code>	Menghapus dan mengembalikan anggota acak dari sebuah set
<code>remove()</code>	Menghapus satu anggota dari set
<code>symmetric_difference()</code>	Mengembalikan set baru berisi komplemen dari dua set
<code>symmetric_difference_update()</code>	Mengupdate set dengan komplemen dari set ini dan set lainnya
<code>union()</code>	Mengembalikan set baru berisi gabungan dua atau lebih set
<code>update()</code>	Mengupdate set dengan gabungan set ini dan set lainnya

Latihan:

1. Buatlah sebuah program dengan konsep tuple untuk dapat menampilkan bilangan terkecil dan bilangan terbesar dari kelompok data?
2. Buatlah sebuah program dengan konsep set dimana terdapat pilihan menu untuk user agar dapat menambahkan dan menghapus data pada set?

BAB 7. DICTIONARY PADA PYTHON

Sebelumnya kita sudah mengetahui tentang list yang memiliki kemampuan menyimpan berbagai macam hal. *List* biasanya digunakan untuk menyimpan koleksi data. Namun, *list* ternyata memiliki kekurangan. Kekurangannya adalah pada list tidak bisa menggunakan kata kunci untuk mengakses itemnya. Hanya bisa menggunakan nomer indeks saja.

Dictionary adalah tipe data yang anggotanya terdiri dari pasangan kunci:nilai (key:value). Dictionary bersifat tidak berurut (*unordered*) sehingga anggotanya tidak memiliki indeks.

Kata kunci harus unik, sedangkan nilai boleh diisi dengan apa saja.

Contoh:

```
aku = {  
    "nama": "agus",  
    "url": "https://www.softwareisa.com"  
}
```

Pada contoh di atas kita membuat sebuah *Dictionary* bernama aku dengan isi data nama dan URL. nama dan url adalah kunci (*key*) yang akan kita gunakan untuk mengakses nilai di dalamnya.

Inilah perbedaannya dibandingkan list dan *tuple*. *Dictionary* memiliki kunci berupa teks–bisa juga angka–sedangkan *list* dan *tuple* menggunakan indeks berupa angka saja untuk mengakses nilainya.

7.1. 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_agus = {  
    "nama": "Agus",  
    "umur": 40,  
    "hobi": ["coding", "Reading", "Listening"],  
    "menikah": True,  
    "sosmed": {  
        "facebook": "softwareisa",  
        "twitter": "@softwareisa"  
    }  
}
```

Mari kita lihat isi dari *Dictionary* di atas:

- nama berisi string "Agus"
- umur berisi integer 40
- hobi berisi list dari string
- menikah berisi boolean True
- dan sosmed berisi Dictionary

7.2. Menggunakan Constructor

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

Contoh:

```
warna_buah = dict(jeruk="orange", buah_naga="merah", pisang="kuning")
```

Maka akan menghasilkan *dictionary* seperti ini:


```
{'jeruk': 'orange', 'buah_naga': 'merah', 'pisang': 'kuning'}
```

7.3. Mengakses Nilai Item Dari Dictionary

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

Contoh:

```
pak_agus = {
    "nama": "Agus",
    "umur": 40,
    "hobi": ["coding", "Reading", "Listening"],
    "menikah": True,
    "sosmed": {
        "facebook": "softwareisa",
        "twitter": "@softwareisa"
    }
}

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

Maka akan menghasilkan:

```
Nama saya adalah Agus
Twitter: @software isa
```

Selain dengan cara di atas, kita juga bisa mengambil nilai *Dictionary* dengan *method* `get()`.

Contoh:

```
print(pak_agus.get("nama"))
```

Hasilnya:

Agus

7.4. Menggunakan Perulangan

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

```
# Membuat dictionary
web = {
    "name": "software isa",
    "url": "https://www.softwareisa.com",
    "rank": "1"
}

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

Hasilnya:

```
Software isa
https://www.softwareisa.com
5
```

Kita juga bisa melakukannya seperti ini:

```
web = {
    "name": "software isa",
```

```
"url": "https://www.softwareisa.com",  
"rank": "1"  
}
```

```
for key, val in web.items():  
    print("%s : %s" % (key, val))
```

Hasilnya:

```
name : software isa  
url : https://www.softwareisa.com  
rank : 1
```

7.5. Mengubah Nilai dan Menghapus Item 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  
skill = {  
    "utama": "Python",  
    "lainnya": ["PHP", "VB", "C#"]  
}
```

```
# Mencetak isi skill utama  
print(skill["utama"])
```

```
# mengubah isi skill utama  
skill["utama"] = "Java"
```

```
# Mencetak isi skill utama  
print(skill["utama"])
```

Maka akan menghasilkan:

```
Python  
Java
```

Menghapus Item dari 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`:

```
>>> del skill["utama"]  
>>> skill={  
"lainnya": ['PHP', 'VB', 'HTML']  
}
```

Contoh menghapus dengan method `pop()`:

```
>>> skill.pop("utama")
```

```
'Java'  
>>> skill{  
  "lainnya": ['PHP', 'VB', 'HTML']  
}
```

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

Contoh:

```
skill.clear()
```

7.6. Menambahkan Item Ke 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.

Contoh:

```
# membuat dictionary user  
user = {  
    "name": "softwareisa"  
}  
  
# menambahkan password  
user.update({"password": "software123"})
```

```
print(user)
```

```
# update name
```

```
user.update({"name": "softwareErp"})
```

```
print(user)
```

Hasilnya:

```
{'name': 'softwareisa', 'password': 'software123'}
```

```
{'name': 'softwareErp', 'password': 'software123'}
```

7.7. Mengambil Panjang Dictionary

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

Contoh:

```
# membuat dictionary
```

```
books = {
```

```
    "python": "Menguasai Python dalam 24 jam",
```

```
    "java": "Tutorial Belajar untuk Pemula",
```

```
    "php": "Membuat aplikasi web dengan PHP"
```

```
}
```

```
# mencetak jumlah data yang ada di dalam dictionary
```

```
print("total: %d" % len(books))
```

Hasilnya:

total: 3

7.8. Jenis-Jenis Method (Fungsi) Dictionary

Metode	Deskripsi
<code>clear()</code>	Menghapus semua anggota dictionary
<code>copy()</code>	Mengembalikan <i>shallow copy</i> dari dictionary
<code>fromkeys(seq[, v])</code>	Mengembalikan dictionary baru dengan kunci-kuncinya dari <code>seq</code> , dan nilainya sama dengan <code>v</code> (defaultnya <code>None</code>)
<code>get(key[, d])</code>	Mengembalikan nilai dari <code>key</code> . Bila <code>key</code> tidak tersedia, kembalikan <code>d</code> (defaultnya <code>None</code>)
<code>items()</code>	Mengembalikan view baru (berisi semua pasangan <code>key,value</code> dari dictionary)
<code>keys()</code>	Mengembalikan view baru (berisi semua kunci pada dictionary)
<code>pop(key[, d])</code>	Menghapus anggota yang memiliki kunci <code>key</code> , dan mengembalikan nilai <code>d</code> jika kunci tidak ada dalam dictionary. Bila <code>d</code> tidak dibuat, dan <code>key</code> tidak ditemukan, akan menghasilkan <code>KeyError</code>
<code>popitem()</code>	Menghapus anggota secara acak. Menghasilkan <code>KeyError</code> jika dictionary kosong
<code>setdefault(key[, d])</code>	Bila <code>key</code> ada dalam dictionary, kembalikan nilainya. Bila tidak, sisipkan <code>key</code> dengan nilai <code>d</code> dan kembalikan <code>d</code> (defaultnya <code>None</code>)
<code>update([other])</code>	Mengupdate dictionary dengan menambahkan anggota dari dictionary lain <code>other</code> , timpa (<i>overwrite</i>) bila ada kunci yang sama
<code>values()</code>	Mengembalikan view baru (berisi semua <i>value</i> pada dictionary)

Latihan:

Buatlah sebuah program dengan konsep dictionary dimana dapat menentukan nama bulan berdasarkan hasil inputan user?