

Workshop



Basic Programming with Python

For Beginner

Module

16 November 2019

Module Level :



Basic



Intermediate



Advanced



Python for DS

Tutor :



Moch. Chamdani Mustaqim

Laboratory Assistant and
Member Coordinator of Data Science Club

This module has prepared by :

Internal division collaborates with the infotech research division



Fajarisma A.P.



Dicky P.O.

Presented by :



Supported by :



1. Pengenalan Python :

Python adalah bahasa pemrograman yang populer yang dibuat oleh Guido van Rossum, dan dirilis pada tahun 1991.



Python biasa digunakan untuk :

- pengembangan web (sisi server),
- pengembangan perangkat lunak,
- matematika,
- system scripting.

Apa yang bisa Python lakukan?

- Python dapat digunakan di server untuk membuat aplikasi web.
- Python dapat digunakan bersama software lain untuk membuat alur kerja.
- Python dapat terhubung ke sistem database. Python juga dapat membaca dan memodifikasi file.
- Python dapat digunakan untuk menangani data besar dan melakukan matematika yang rumit.
- Python dapat digunakan untuk pembuatan prototipe cepat, atau untuk pengembangan perangkat lunak yang siap produksi.

Mengapa python?

- Python dapat bekerja pada platform yang berbeda (Windows, Mac, Linux, Raspberry Pi, dll).
- Python memiliki sintaks sederhana yang mirip dengan bahasa Inggris.

- Python memiliki sintaks yang memungkinkan pengembang untuk menulis program dengan lebih singkat dari beberapa bahasa pemrograman lainnya.
- Python berjalan pada sistem interpreter, artinya kode dapat dieksekusi segera setelah ditulis. Ini berarti prototyping bisa sangat cepat.
- Python dapat diperlakukan dengan cara prosedural, cara berorientasi objek atau cara fungsional.

Sekilas Info :

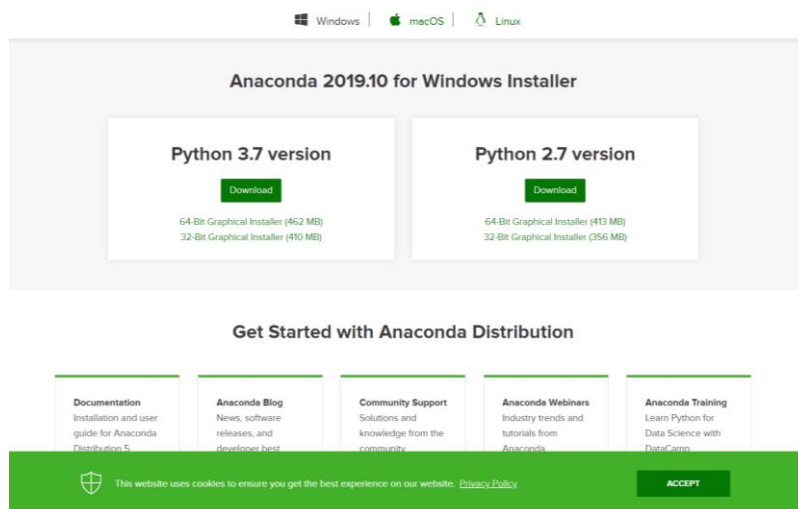
- Versi utama terbaru dari Python adalah Python 3, yang akan kita gunakan dalam tutorial ini. Namun, Python 2, meskipun tidak diperbarui dengan apa pun selain pembaruan keamanan, masih cukup populer.
- Dalam tutorial ini, Python akan ditulis dalam editor teks. Dimungkinkan untuk menulis Python di Integrated Development Environment, seperti Pycharm, Jupyter Notebook atau Google Colab yang sangat berguna ketika mengelola koleksi file Python yang lebih besar.

Sintaksis Python dibandingkan dengan bahasa pemrograman lain

- Python dirancang untuk readability dan memiliki beberapa kesamaan dengan bahasa Inggris dengan pengaruh dari matematika.
- Python menggunakan new lines untuk menyelesaikan perintah, berbeda dengan bahasa pemrograman lain yang sering menggunakan tanda titik koma ; atau tanda kurung ().
- Python bergantung pada indentasi, menggunakan whitespace, untuk mendefinisikan scope; seperti ruang lingkup loop, fungsi, dan kelas. Bahasa pemrograman lain sering menggunakan kurung kurawal {} untuk tujuan ini.

2. Instalasi Anaconda 3 :

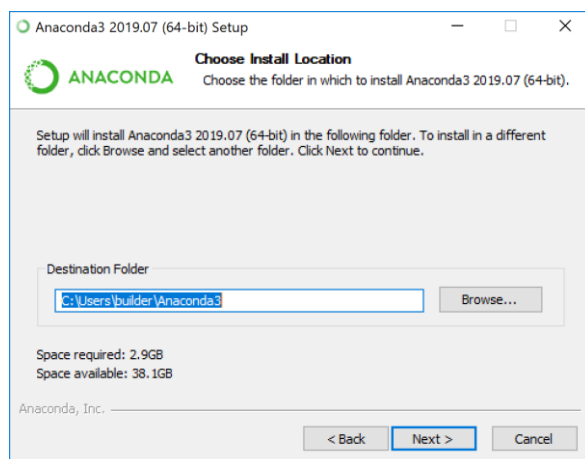
1. [Download Anaconda installer.](#)
2. Pilih **python 3.x version** download sesuai base system computer anda 32-Bit atau 64-Bit.



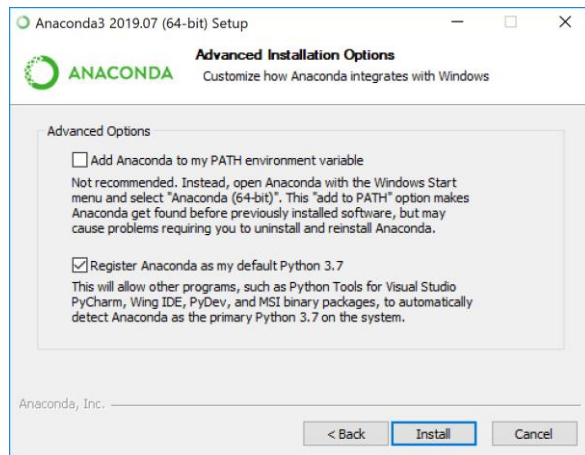
3. Setelah download selesai jalankan file yang telah anda download.
4. Click Next kemudian baca licensing terms dan klik **"I Agree"**.
5. Pilih install for "Just Me" atau anda dapat menginstall untuk semua user (yang memerlukan Windows Administrator privileges) lalu klik Next.
6. Pilih folder tujuan untuk tempat menyimpan file instalasi program Anaconda 3 kemudian klik next.

** Install Anaconda pada direktori path yang tidak mengandung spasi atau karakter unicode.*

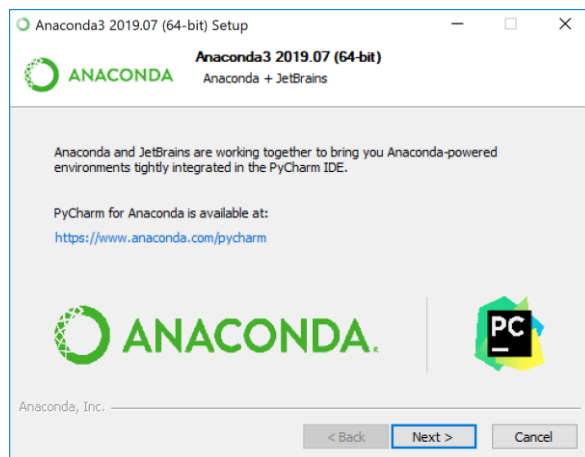
** jangan install Anaconda sebagai Administrator selama admin privileges tidak diperlukan.*



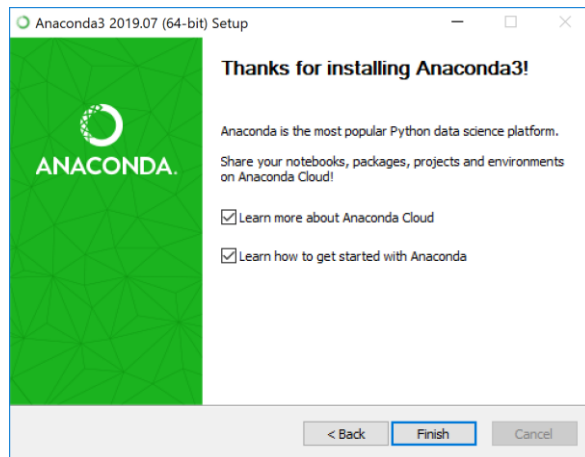
7. Pilih apakah anda akan menambahkan Anaconda to your PATH environment variable. Kami sarankan untuk tidak menambahkan Anaconda to your PATH environment variable, karena ini dapat mengganggu perangkat lunak lain. Anda dapat menggunakan Anaconda dengan membuka Anaconda Navigator atau Anaconda Prompt dari Start Menu.



8. Pilih apakah anda ingin Anaconda sebagai Python default Anda. Kecuali Anda berencana menginstal dan menjalankan beberapa versi Anaconda atau beberapa versi Python, terima defaultnya dan biarkan kotak ini dicentang.
9. Klik tombol install. Klik show details untuk melihat paket – paket anaconda diinstall.
10. Klik tombol next.
11. Disarankan untuk menginstall PyCharm atau PyCharm for Anaconda <https://www.anaconda.com/pycharm>. Klik next bila tidak ingin menginstall PyCharm.



12. Instalasi Anaconda selesai uncheck semua check box lalu klik finish.



13. Pada start menu jalankan anaconda navigator dan anaconda prompt.

3. Python Hello World :

3.1 Hello, World! Anaconda prompt

1. Setelah buka anaconda prompt ketikkan command berikut untuk melihat versi python yang terinstall di anaconda.

```
(base) C:\Users\UnNamed>python --version
Python 3.7.3
```

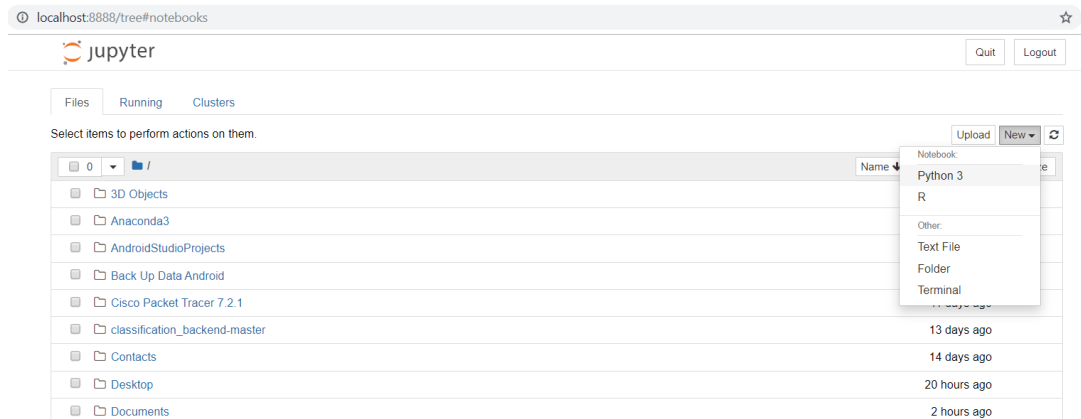
2. Anaconda prompt hello, world!

```
(base) C:\Users\UnNamed>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello, world!")
hello, world!
```

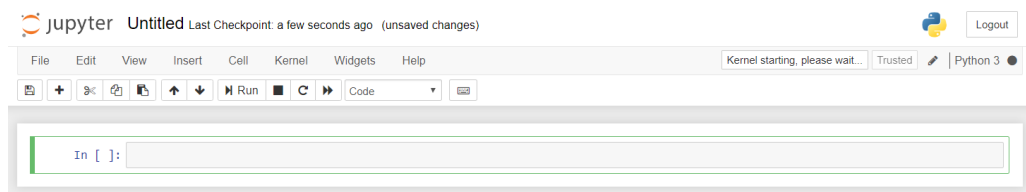
3. Ketikkan `>>> exit()` untuk keluar.

3.2 Hello, World! Jupyter Notebook

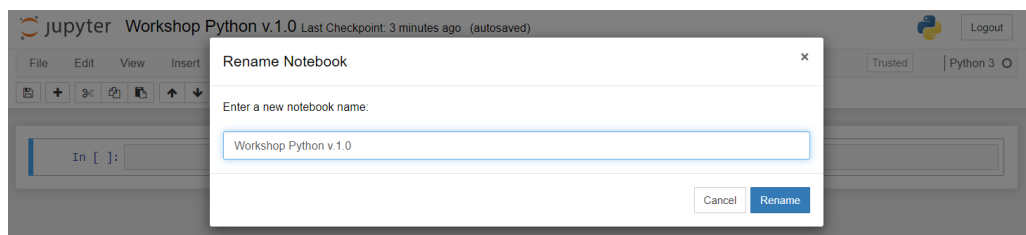
1. Buka jupyter notebook dengan mengetikkan command `jupyter notebook` pada anaconda prompt.
2. Setelah terbuka jupyter notebook pada browser anda buat file Python 3 baru.



3. Pastikan terbuka jendela baru seperti ini :



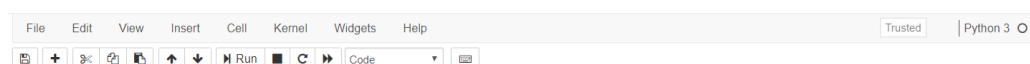
4. Rename dengan cara klik pada bagian **Untitled** ubah menjadi **Workshop Python v.1.0** lalu klik **Rename**



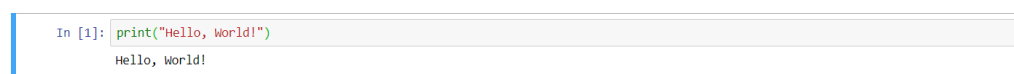
5. Pada kolom yang kosong tuliskan code berikut:



6. Lalu klik tombol **Run** :

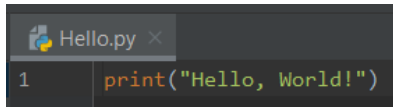


7. Maka akan keluar hasil seperti ini:



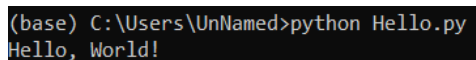
3.2 Hello, World! Hello.py

1. Buka PyCharm atau editor python lain yang anda miliki.
2. Buatlah file dengan nama Hello.py
3. Isi file tersebut dengan code berikut.



```
Hello.py x
1 print("Hello, World!")
```

4. Simpan file Hello.py
5. Buka Anaconda Prompt kemudian buka directory path dengan perintah **cd**.
6. Jika sudah sampai pada folder tempat file Hello.py disimpan ketikkan perintah berikut, dan perhatikan hasilnya.



```
(base) C:\Users\UnNamed>python Hello.py
Hello, World!
```

4. Python Syntax

4.1 Running Python Syntax

Python dapat menjalankan syntax nya langsung di command line atau dapat membuat file ber-ekstensi .py dan menjalankannya menggunakan command line seperti contoh di bab 3.

Ada dua cara berbeda untuk mengubah suatu program dari bahasa pemrograman tingkat tinggi menjadi bahasa mesin :

COMPIRATION - program sumber diterjemahkan satu kali (namun, tindakan ini harus diulang setiap kali Anda mengubah kode sumber) untuk mendapatkan file (misalnya, file .exe jika kode ini dimaksudkan untuk dijalankan di bawah MS Windows) yang berisi mesin kode, sekarang Anda dapat mendistribusikan file ke seluruh dunia, program yang melakukan terjemahan ini disebut kompiler atau penerjemah.

INTERPRETATION - Anda (atau pengguna kode lain) dapat menerjemahkan source program setiap kali harus dijalankan. Program yang melakukan transformasi semacam ini disebut interpreter, karena ia menginterpretasikan

kode setiap kali dimaksudkan untuk dieksekusi. Hal ini juga berarti bahwa Anda tidak bisa hanya mendistribusikan kode sumber apa adanya, karena pengguna akhir juga membutuhkan penerjemah untuk menjalankannya.

Compilation vs. interpretation - advantages and disadvantages		
	COMPIRATION	INTERPRETATION
ADVANTAGES	<ul style="list-style-type: none"> the execution of the translated code is usually faster; only the user has to have the compiler - the end-user may use the code without it; the translated code is stored using machine language - as it is very hard to understand it, your own inventions and programming tricks are likely to remain your secret. 	<ul style="list-style-type: none"> you can run the code as soon as you complete it - there are no additional phases of translation; the code is stored using programming language, not the machine one - this means that it can be run on computers using different machine languages; you don't compile your code separately for each different architecture.
DISADVANTAGES	<ul style="list-style-type: none"> the compilation itself may be a very time-consuming process - you may not be able to run your code immediately after any amendment; you have to have as many compilers as hardware platforms you want your code to be run on. 	<ul style="list-style-type: none"> don't expect that interpretation will ramp your code to high speed - your code will share the computer's power with the interpreter, so it can't be really fast; both you and the end user have to have the interpreter to run your code.
<p>What does this all mean for you?</p> <ul style="list-style-type: none"> Python is an interpreted language. This means that it inherits all the described advantages and disadvantages. Of course, it adds some of its unique features to both sets. If you want to program in Python, you'll need the Python interpreter. You won't be able to run your code without it. Fortunately, Python is free. This is one of its most important advantages. <p>Due to historical reasons, languages designed to be utilized in the interpretation manner are often called scripting languages, while the source programs encoded using them are called scripts.</p>		

4.2 Python Indentation

Python tidak menggunakan tanda { } untuk menandai blok / grup kode. Blok kode di python menggunakan tanda indentasi (spasi). Jumlah spasi untuk setiap baris yang ada dalam satu blok kode harus sama. Contoh yang benar adalah sebagai berikut :

```
if nilai <= 5:
    print("Nilai merah")
    print("Tidak lulus")
else:
    print("Nilai biru")
    print("Lulus")

if 5 > 2:
    print("Five is greater than two!")
if 5 > 2:
    print("Five is greater than two!")
```

Contoh kesalahan dalam indentasi :

```
if 5 > 2:
    print("Five is greater than two!")
    print("Five is greater than two!")

if 5 > 2:
    print("Five is greater than two!")
    print("Five is greater than two!")
```

Catatan : disarankan untuk menggunakan tab / 4 spasi untuk setiap indentasi atau block code di python untuk memudahkan dalam pembacaan.

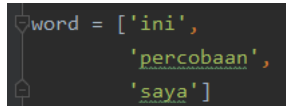
4.3 Python Statements

Semua perintah yang bisa dieksekusi oleh Python disebut statement. Misalnya, `a = 1` adalah sebuah statement penugasan. Selain statement penugasan ada statement lain seperti statement if, statement for, dan lain

sebagainya. Kita dapat membuat sebuah statement terdiri dari beberapa baris menggunakan tanda backslash (\). Misalnya :

```
a = panjang1 + panjang2 + \
    panjang3 + \
    panjang4
print(a)
```

Statement yang ada di dalam tanda kurung [], { }, dan () tidak memerlukan tanda \ Contohnya :



```
word = ['ini',
        'percobaan',
        'saya']
```

4.4 Python Comments

Python menggunakan tanda pagar # untuk mengaktifkan komentar. Tanda ini dapat digunakan untuk memberi komentar perbaris saja. Jika anda ingin memberikan penjelasan panjang yang memerlukan beberapa baris komentar maka disarankan untuk menggunakan triple quotes seperti ''' atau """. Hal ini dikarenakan python akan mengabaikan apa saja didalam triple quotes selama itu tidak diisikan kedalam suatu variabel.

Contoh :

```
#This is a comment.
```

```
print("Hello, World!")
```

```
'''Multi lines comment.
```

```
In python'''
```

Quis : Mengapa Guido van Rossum menamai bahasa pemrograman buatanya dengan python, dari manakah ia mendapatkan inspirasi nama tersebut?



5. Python Variables

5.1 Assign a Value to the Variable

Di python, variabel tidak perlu dideklarasikan secara eksplisit. Deklarasi atau pembuatan variabel terjadi secara otomatis pada saat kita memberi suatu nilai ke variabel. Tanda sama dengan (=) digunakan untuk memberikan nilai ke variabel. Operand di sebelah kiri tanda = adalah nama variabel dan di sebelah kanan tanda = adalah nilai yang disimpan di dalam variabel. Sebagai contoh:

```
x = 5
```

```
y = "Python3"
```

```
print(x)
```

```
print(y)
```

Bahkan anda dapat mengubah type data dari suatu variabel dengan mudah hanya dengan cara memberikan nilai bertipe data baru untuk variabel tersebut.

Kriteria penamaan variable di python :

- Variabel dalam python hanya dapat diawali oleh huruf atau underscore.
- Variabel tidak dapat diawali dengan angka.
- Nama variabel hanya boleh mengandung angka, huruf dan underscore (A-Z, a-z, 0-9 dan _).
- Variabel di python adalah case-sensitive (varA berbeda dengan VarA).

5.2 Assign Value to Multiple Variable

Python memberikan kemudahan untuk menentukan nilai dari beberapa variabel. Contoh :

```
x, y, z = "Orange", "Banana", "Cherry"
```

```
print(x)
print(y)
print(z)
```

Python juga dapat menentukan nilai dari beberapa variabel dengan nilai yang sama hanya dengan single line of code. Contoh :

```
x = y = z = "Orange"
```

```
print(x)
print(y)
print(z)
```

5.3 Concat Multiple String Variables

Anda dapat melakukan concat beberapa variabel yang bertipe string dan menyimpannya kedalam variabel baru dengan menggunakan + seperti pada contoh berikut :

```
x = "Python is "
```

```
y = "awesome"
```

```
z = x + y
```

```
print(z)
```

Namun tanda + untuk variabel bertipe angka akan bekerja sebagai operator matematika biasa. Contoh :

```
x = 5
```

```
y = 7
```

```
z = x + y
```

```
print(z)
```

Catatan : Pada python anda akan mendapati error bila anda mencoba mengkombinasikan string dengan angka.

5.4 Global Variables

Semua variabel yang dibuat diluar fungsi disebut variabel global. Variabel global dapat digunakan oleh siapa saja, baik didalam maupun diluar fungsi.

```
x = "awesome"
```

```
def myfunc():  
    print("Python is " + x)
```

```
myfunc()
```

Jika anda membuat variabel didalam suatu fungsi dengan nama yang sama dengan variabel diluar fungsi maka variabel tersebut dinamakan variabel local yang mana hanya bisa diakses didalam fungsi tersebut.

```
x = "awesome"
```

```
def myfunc():  
    x = "fantastic"  
  
    print("Python is " + x)
```

```
myfunc()  
print("Python is " + x)
```

Untuk mengubah suatu variabel local menjadi global anda hanya perlu menambahkan code global didepan nama variabel yang ingin anda ubah menjadi global. Contoh :

```
x = "awesome"
```

```
def myfunc():  
    global x = "fantastic"  
  
    print("Python is " + x)
```

```
myfunc()  
print("Python is " + x)
```

6. Python Data Types

6.1 Build-in Data Types

Text :	str
Numeric :	int, float, complex
Sequence :	list, tuple, range
Mapping :	dict
Set :	set, frozenset
Boolean :	bool
Binary :	bytes, bytearray, memoryview

Contoh penerapan tipe data :

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Untuk mengetahui tipe data dari suatu variabel cukup gunakan `type()` seperti berikut :

```
x = 5
```

```
print(type(x))
```

6.2 Python Numbers

Untuk mendefinisikan atau menentukan nilai suatu variabel dengan angka didalam python dapat dibedakan menjadi 3 macam yaitu :

- **Int**

Untuk mendefinisikan suatu variabel bertipe int atau bilangan bulat cukup mudah dilakukan di python kita hanya perlu mendefinisikan nilai variabel seperti biasa, seperti yang dicontohkan diatas.

- **Float**

Untuk mendefinisikan suatu variabel bertipe float atau bilangan desimal kita cukup menambahkan titik dibagian belakang angka atau diantara angka-angka. Seperti contoh :

v = 1.

w = -1.

x = 1.10

y = 1.0

z = -35.59

- **Complex**

Untuk mendefinisikan suatu variabel bertipe complex number atau bilangan kompleks imajiner cukup menambahkan huruf "j" dibagian belakang angka. Seperti contoh :

x = 3+5j

y = 5j

z = -5j

Python tidak memiliki fungsi `random()` namun python memiliki built-in module yaitu `random` yang dapat digunakan untuk membuat random number. Module ini harus di import terlebih dahulu jika ingin menggunakannya sebagai contoh :

```
import random
print(random.randrange(1,10))
```

7. Casting in Python Data Types

Python menyediakan cara casting tipe data / pengubahan tipe data dengan cukup mudah yaitu dengan menggunakan fungsi konstruktor. Karena python adalah bahasa berorientasikan objek oleh sebab itu python menggunakan class untuk mendefinisikan tipe data termasuk tipe data primitif.

7.1 Casting to Int

`int()` – membangun bilangan dari suatu integer literal, float literal (dengan membulatkan bilangan ke bawah), atau string literal (berdasarkan nilai angka dari suatu string). Contoh :

```
x = int(1) # x menjadi 1
y = int(2.8) # y menjadi 2
z = int("3") # z menjadi 3
```

7.2 Casting to Float

`float()` – membangun bilangan desimal dari suatu integer literal, float literal, atau string literal (berdasarkan nilai angka dari suatu string). Contoh :

```
w = float(1) # w menjadi 1.0
x = float(2.8) # x menjadi 2.8
y = float("3.1") # y menjadi 3.1
z = float("3") # z menjadi 3.0
```

7.3 Casting to String

`str()` – membangun sebuah string dari berbagai macam tipe data suatu integer literal, float literal (dengan membulatkan bilangan ke bawah), atau string literal (berdasarkan nilai angka dari suatu string). Contoh :

```
x = str("s1") # x menjadi 's1'
```



```
y = str(2) # y menjadi '2'
```

```
z = str(3.0) # z menjadi '3.0'
```

8. Python Strings

8.1 Strings are Arrays

Seperti pada berbagai bahasa pemrograman lain, string di python merupakan array of bytes yang merepresentasikan karakter unicode. Sejalan ini python tidak memiliki tipe data karakter, sebuah karakter dapat dikatakan sebagai sebuah string dengan panjang 1.

Kurung kotak dapat digunakan untuk mengakses elemen dari string. Contohnya :

```
a = "Hello, World!"
```

```
print(a[1])
```

8.2 String Slicing

Anda dapat mengembalikan nilai string berupa rentang karakter dalam string dengan menggunakan sintaks slice. Untuk menggunakan slice perhatikan contoh berikut :

```
# program berikut akan menampilkan karakter yang memiliki index posisi 2  
hingga 4. "index 5 disini tidak termasuk"
```

```
b = "Hello, World!"
```

```
print(b[2:5])
```

8.3 String Length

Untuk mendapatkan panjang dari string python menyediakan fungsi `len()`. Fungsi tersebut akan mengembalikan nilai berupa panjang dari string. Contoh :

```
a = "Hello, World!"
```

```
print(len(a))
```

8.4 Check String

Untuk mengecek apakah frasa atau karakter terdapat disuatu string, kita dapat menggunakan keyword `in` atau `not in`. Keyword tersebut akan mengembalikan nilai `True` atau `False`

```
txt = "The rain in Spain stays mainly in the plain"
```

```
x = "ain" not in txt
```

```
print(x)
```

8.5 String Format

Kita telah mengetahui bahwa kita tidak dapat mengabungkan string dengan angka menggunakan metode concat. Namun didalam python kita dapat menggabungkannya menggunakan method `format()`. Method ini mengambil argument kemudian memformatnya dan menempatkan argument tersebut kedalam string dimana placeholder `{}` berada.

```
quantity = 3
```

```
itemno = 567
```

```
price = 49.95
```

```
myorder = "I want {} pieces of item {} for {} dollars."
```

```
print(myorder.format(quantity, itemno, price))
```

Atau kita juga bisa menggunakan angka index untuk menempatkan argument ketempat yang tepat seperti contoh :

```
quantity = 3
```

```
itemno = 567
```

```
price = 49.95
```

```
myorder = "I wanna pay {2} dollars for {0} pieces of item {1}."
```

```
print(myorder.format(quantity, itemno, price))
```

8.6 Escape Character

Escape character digunakan ketika kita ingin memasukkan karakter ilegal kedalam sebuah string. Gunakan backslash \ dan diikuti oleh karakter ilegal yang ingin anda masukkan. Sebagai contoh :

```
txt = "We are the so-called "Vikings" from the north."
```

Kode diatas akan mendapati error, untuk menanganinya kita bisa menggunakan escape character seperti berikut :

```
txt = "We are the so-called \"Vikings\" from the north."
```

Escape character lain didalam python :

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

8.7 String Methods

Berikut adalah beberapa build-in String methods di python :

capitalize()	expandtabs()	isalpha()
casefold()	find()	isdecimal()
center()	format()	isdigit()
count()	format_map()	isidentifier()
encode()	index()	islower()
endswith()	isalnum()	isnumeric()

isprintable()	partition()	splitlines()
isspace()	replace()	startswith()
istitle()	rfind()	strip()
isupper()	rindex()	swapcase()
join()	rjust()	title()
ljust()	rpartition()	translate()
lower()	rsplit()	upper()
lstrip()	rstrip()	zfill()
maketrans()	split()	

9. Python Operators

9.1 Arithmetic operators

Operator aritmatika adalah operator yang digunakan untuk melakukan operasi matematika. Pada tabel berikut menunjukkan jenis operator aritmatika.

Operator	Nama dan Fungsi	Contoh
+	Penjumlahan, menjumlahkan 2 buah operand	$x + y$
-	Pengurangan, mengurangi 2 buah operand	$x - y$
*	Perkalian, mengalikan 2 buah operand	$x * y$
/	Pembagian, membagi 2 buah operand	x / y
**	Pemangkatan, memangkatkan bilangan	$x ** y$
//	Pembagian bulat, menghasilkan hasil bagi tanpa koma	$x // y$
%	Modulus, menghasilkan sisa pembagian 2 bilangan	$x \% y$

9.2 Assignment operators

Operator penugasan adalah operator yang digunakan untuk memberi nilai ke variabel. $a = 7$ adalah contoh operator penugasan yang memberi nilai 7 di kanan ke variabel a yang ada di kiri.

Operator	Penjelasan	Contoh
=	Menugaskan nilai yang ada di kanan ke operand yang ada di sebelah kiri	$c = a + b$ menugaskan $a + b$ ke c
+=	Menambahkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c += a$ sama dengan $c = c + a$
-=	Mengurangi operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c -= a$ sama dengan $c = c - a$
*=	Mengalikan operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c *= a$ sama dengan $c = c * a$
/=	Membagi operand yang di kanan dengan operand yang ada di kiri dan hasilnya di tugaskan ke operand yang di kiri	$c /= a$ sama dengan $c = c / a$
**=	Memangkatkan operand yang di kanan dengan operand yang ada di kiri dan hasilnya ditugaskan ke operand yang di kiri	$c **= a$ sama dengan $c = c ** a$
//=	Melakukan pembagian bulat operand di kanan terhadap operand di kiri dan hasilnya disimpan di operand yang di kiri	$c //= a$ sama dengan $c = c // a$
%=	Melakukan operasi sisa bagi operand di kanan dengan operand di kiri dan hasilnya di simpan di operand yang di kiri	$c \% = a$ sama dengan $c = c \% a$

9.3 Comparison operators

Operator perbandingan merupakan operator yang digunakan untuk membandingkan 2 buah nilai. Hasil perbandingannya adalah True atau False berdasar dari kondisi.

Operator	Nama dan Fungsi	Contoh
>	Lebih besar dari – Hasilnya True jika nilai sebelah kiri lebih besar dari nilai sebelah kanan	$x > y$
<	Lebih kecil dari – Hasilnya True jika nilai sebelah kiri lebih kecil dari nilai sebelah kanan	$x < y$
==	Sama dengan – Hasilnya True jika nilai sebelah kiri sama dengan nilai sebelah kanan	$x == y$
!=	Tidak sama dengan – Hasilnya True jika nilai sebelah kiri tidak sama dengan nilai sebelah kanan	$x != y$
>=	Lebih besar atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih besar atau sama dengan nilai sebelah kanan	$x >= y$
<=	Lebih kecil atau sama dengan – Hasilnya True jika nilai sebelah kiri lebih kecil atau sama dengan nilai sebelah kanan	$x <= y$

9.4 Logical operators

Operator logika adalah operator yang digunakan untuk melakukan operasi logika.

Operator	Penjelasan	Contoh
and	Hasilnya adalah True jika kedua operandnya bernilai benar	$x \text{ and } y$
or	Hasilnya adalah True jika salah satu atau kedua operandnya bernilai benar	$x \text{ or } y$
not	Hasilnya adalah True jika operandnya bernilai salah (kebalikan nilai)	$\text{not } x$

9.5 Identity operators

Operator identitas adalah operator yang memeriksa apakah dua buah nilai (atau variabel) berada pada lokasi memori yang sama.

Operator	Penjelasan	Contoh
is	True jika kedua operand identik (menunjuk ke objek yang sama)	x is True
is not	True jika kedua operand tidak identik (tidak merujuk ke objek yang sama)	x is not True

9.6 Membership operators

Operator keanggotaan adalah operator yang digunakan untuk memeriksa apakah suatu nilai atau variabel merupakan anggota atau ditemukan di dalam suatu data (string, list, tuple, set, dan dictionary).

Operator	Penjelasan	Contoh
in	True jika nilai/variabel ditemukan di dalam data	5 in x
not in	True jika nilai/variabel tidak ada di dalam data	5 not in x

9.7 Bitwise operators

Operator bitwise adalah operator yang melakukan operasi bit terhadap operand. Operator ini beroperasi bit per bit sesuai dengan namanya. Sebagai misal, angka 2 dalam bit ditulis 10 dalam notasi biner dan angka 7 ditulis 111. Pada tabel di bawah ini, misalkan x = 10 (0000 1010) dalam biner dan y = 4 (0000 0100) dalam biner.

Operator	Nama	Contoh
&	Bitwise AND	x & y = 0 (0000 0000)
	Bitwise OR	x y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

10. Python Decisions

10.1 Percabangan

Merupakan cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu, bisa satu atau lebih. Percabangan mengevaluasi hasil dengan ekspresi boolean (bernilai True atau False). Jika True maka blok kondisi akan dieksekusi, bila False, maka akan mengeksekusi pernyataan yang lain.

Terdapat 3 pernyataan untuk percabangan, yaitu :

No.	Pernyataan	Deskripsi
1	If	Pernyataan if terdiri dari ekspresi boolean diikuti oleh satu baris atau lebih pernyataan.
2	If...else	Bila pernyataan if benar, maka blok pernyataan if dieksekusi. Bila salah, maka blok pernyataan else yang dieksekusi.
3	If...elif...else	Disebut juga if bercabang. Bila ada kemungkinan beberapa kondisi bisa benar maka digunakan pernyataan if...elif atau if...elif...else

10.2 Pernyataan IF

Pernyataan ini menguji satu buah kondisi. Apabila benar maka pernyataan di dalam blok IF akan dieksekusi, apabila salah maka tidak dieksekusi

```
bil = 8
if bil%2==0 :
    print("Bilangan",bil,"adalah genap")
```

10.3 Pernyataan IF...ELSE

Pernyataan ini menguji 2 kondisi. Kondisi pertama kalau benar, dan kondisi kedua kalau salah


```

bil = 9
if bil%2==0 :
    print("Bilangan",bil,"adalah genap")
else :
    print("Bilangan",bil,"adalah ganjil")

```

10.4 Pernyataan IF...ELIF...ELSE

Pernyataan ini digunakan untuk menguji lebih dari 2 kondisi. Bila kondisi pada IF benar, maka pernyataan di dalamnya yang dieksekusi. Bila salah, maka akan mengeksekusi pada kondisi elif. Terakhir, bila kondisi if dan elif salah, maka dieksekusi pada blok else.

```

bil = -1
if bil==0 :
    print("Bilangan",bil,"adalah nol")
elif(bil>0) :
    print("Bilangan",bil,"adalah positif")
else :
    print("Bilangan",bil,"adalah negatif")

```

11. Python Loop

11.1 For Loop

Sintaks for adalah sebagai berikut :

```

for var in sequence:
    body of for

```

Var adalah variabel yang digunakan untuk penampung sementara nilai dari sequence pada saat terjadi perulangan. **Sequence** adalah tipe data berurut seperti string, list, dan tuple.

```

num = [1,2,3,4,5]
sum=0
for x in num :
    sum = sum+x
print("Jumlah semuanya :",sum)

```

11.2 range() Function

Fungsi ini dapat digunakan untuk menghasilkan deret bilangan. Fungsi `range()` juga dapat langsung ditentukan batas bawah, atas, interval dengan format **`range(batas bawah, batas atas, interval)`**. Fungsi `range` tidak menyimpan semua nilai dalam memori secara langsung.

Contoh :

`range(5)` akan menghasilkan angka 0 sampai 4 (5 bilangan).

Contoh Program :

<pre>print(range(10)) print(list(range(10))) print(list(range(0,5))) print(list(range(2,20,3)))</pre>	<pre>range(0, 10) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4] [2, 5, 8, 11, 14, 17]</pre>
---	--

<pre>buah = ['apel', 'nangka', 'jeruk'] for i in range(len(buah)): print("Saya suka buah", buah[i])</pre>	<pre>Saya suka buah apel Saya suka buah nangka Saya suka buah jeruk</pre>
---	---

11.3 While Loop

Pada contoh berikut kita memiliki **`count < 5`** sebagai batasan dalam perulangan `while` dan **`count += 1`** sebagai bagian increment nilai `count`.

<pre>count = 0 while (count < 5): print('Angka ke-:', count) count = count + 1</pre>	<pre>Angka ke-: 0 Angka ke-: 1 Angka ke-: 2 Angka ke-: 3 Angka ke-: 4</pre>
---	---

* Note : tanpa increment maka perulangan anda akan dilakukan terus menerus hingga anda menekan tombol `Ctrl +C`

11.4 Break and Continue

Syntax **`break`** digunakan untuk keluar dari looping sedangkan syntax **`continue`** digunakan untuk melompati step looping tertentu dan melanjutkan ke step selanjutnya.

Contoh program :

<pre>for letter in "Programming": if letter == "g": break print("Huruf sekarang:", letter) print("Good bye")</pre>	<pre>Huruf sekarang: P Huruf sekarang: r Huruf sekarang: o Good bye</pre>
--	---

```

for letter in "Programming":
    if letter == "g":
        continue
    print("Huruf sekarang:", letter)
print("Good bye")

```

```

Huruf sekarang: P
Huruf sekarang: r
Huruf sekarang: o
Huruf sekarang: r
Huruf sekarang: a
Huruf sekarang: m
Huruf sekarang: m
Huruf sekarang: i
Huruf sekarang: n
Good bye

```

11.5 While ... else

Python mendukung penggunaan **else** sebagai pasangan dari **while**. Blok pernyataan else hanya akan dieksekusi bila kondisi while bernilai salah.

```

count = 0
while (count < 5):
    print(count, "kurang dari 5")
    count = count + 1
else:
    print(count, "tidak kurang dari 5")

```

12. Python List

Ada 4 tipe collection data dalam python programming :

1. List merupakan collection yang **terurut** dan **dapat diubah** serta memperbolehkan anggota bernilai duplikat.
2. Tuple adalah collection yang **terurut** dan **tidak dapat diubah** namun memperbolehkan anggotanya bernilai duplikat.
3. Set merupakan collection yang **tak terurut** dan **tidak terindex** serta tidak memperbolehkan anggota bernilai duplikat.
4. Dictionary adalah collection yang **tidak terurut, dapat diubah, dan memiliki index**. Dictionary juga tidak memperbolehkan duplicate members.

12.1 Introduction to python list

List adalah tipe data yang berisi satu atau lebih di dalamnya yang disebut dengan item, elemen, atau anggota list. Item di dalam list berada di dalam tanda kurung [], dan dipisahkan dengan tanda koma. Namun di dalam satu list, dapat berisi

satu tipe data lebih. List juga dapat berisi list lain yang disebut dengan list bersarang.

```
lista = []
listint = [1,2,3]
listmix = [1,2,"Hello", 'There!']
listmixed = [[1,2], 'Hello']
print(lista)
print(listint)
print(listmix)
print(listmixed)
```

```
[]
[1, 2, 3]
[1, 2, 'Hello', 'There!']
[[1, 2], 'Hello']
```

Untuk mengakses dan mengubah nilai dari list kita dapat menggunakan index list terkait. Apabila anda mengakses suatu index diluar index list maka anda akan mendapati *IndexError*.

12.2 Negative indexing

Maksud dari index negatif adalah dimulai dari index -1 atau setara dengan index terakhir dari list. Index -2 , -3 dan seterusnya setara dengan index terakhir ke 2, ke 3 dan seterusnya.

```
listmix = [1,2,"Hello", 'There!']
print(listmix[-1])
```

12.3 Range of list index "slicing"

Kita dapat memotong anggota list dengan range tertentu dengan menggunakan slicing titik dua (:). Slicing ini berkaitan dengan pemahaman indeks dari suatu list.

0	1	2	3	4	5	6	7	8	9
P	Y	T	H	O	N	I	N	D	O
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Contoh program :

```
print(list[3:])
print(list[:5])
print(list[3:6])
```

12.4 Loop through a list

Untuk menampilkan keseluruhan isi dari list anda dapat menggunakan looping for. Seperti contoh :

```
thislist = ["apple", "banana", "cherry"]
```

```
for x in thislist:  
    print(x)
```

Untuk mengecek apakah suatu list mengandung element tertentu anda juga dapat menggunakan looping for. Seperti contoh :

```
thislist = ["apple", "banana", "cherry"]
```

```
if "apple" in thislist:  
    print("Yes, 'apple' is in the fruits list")
```

12.5 List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position

pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

13. Python Tuple

13.1 Introduction to tuple

Tuple dibuat dengan meletakkan semua item di dalam tanda kurung () dengan tiap item dipisah tanda koma. Tanda kurung di sini bersifat opsional, hanya untuk mempermudah pembacaan kode.

```
tupleone = ()
print(tupleone)

tupletwo = (1,2,3)
print(tupletwo)

tuplethree = (1,2,3,(4,5),[6,7])
print(tuplethree)

tuplefour = 1,2,3
print(tuplefour)

a,b,c = tuplefour
print(a,b,c)
```

()
(1, 2, 3)
(1, 2, 3, (4, 5), [6, 7])
(1, 2, 3)
1 2 3|

Kelebihan tuple adalah sebagai berikut :

1. Tuple bersifat immutable sehingga iterasi lebih cepat dibandingkan list.
2. Tuple bisa berisi anggota immutable yang dapat digunakan sebagai key pada dictionary. Sedangkan List tidak bisa dipakai untuk itu.

3. Kalau kita memerlukan data yang memang tidak untuk diubah, maka menggunakan tuple bisa menjamin bahwa data tersebut akan write-protected.

13.2 Accessing tuple value

Untuk mengakses nilai dari tuple kita dapat menggunakan index tuple terkait dengan format `namatuple[index]`. Apabila anda mengakses suatu index diluar index tuple maka anda akan mendapati *IndexError*. Index dimulai dari 0 untuk anggota pertamanya, dan dimulai dari -1 dari anggota terakhir tuple.

```
tuple = (1,2,3,4,5,6,7,8,9,10)
print(tuple[3])           4
print(tuple[:3])          (1, 2, 3)
print(tuple[3:])          (4, 5, 6, 7, 8, 9, 10)
print(tuple[2:7])         (3, 4, 5, 6, 7)
print(tuple[-2])          9
```

13.3 Accessing tuple value

Pada tuple, kita dapat menguji apakah sebuah objek adalah anggota tuple atau tidak, yaitu dengan menggunakan operator `in` atau `not in` untuk kebalikannya.

```
tuple = (1,[2,3],4,5,6,7,8,9,10)
print(3 in tuple)         False
print(4 in tuple)         True
```

Angka 3 mengembalikan nilai False karena 3 tidak berada di keanggotaan Tuple asli, dia merupakan anggota list yang bersarang pada Tuple.

13.4 Tuple method

Tuple hanya memiliki dua buah metode yaitu `count()` dan `index()`

- Metode `count(x)` berfungsi mengembalikan jumlah item yang sesuai dengan x pada tuple

- Metode `index(x)` berfungsi mengembalikan indeks dari ite pertama yang sama dengan x

```
tuple = (1,[2,3],4,5,6,7,8,9,10,2,2,2)
print(tuple.count(2))
print(tuple.index(5))
```

Angka 2 di atas dihitung sebanyak 3, dikarenakan angka 2 pada list bersarang tidak termasuk dalam tuple. Begitu pula dengan angka 2 dan 3 pada list bersarang, akan dihitung menjadi 1 indeks, yaitu index ke 1 dalam tuple.

Selain fungsi yang telah disebutkan di atas, masih terdapat beberapa fungsi lain untuk melakukan operasi pada tuple. Berikut diantaranya :

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

14. Python Set

14.1 Introduction to Set

Set dibuat dengan meletakkan anggota-anggotanya di dalam tanda kurung kurawal {} dan itemnya dipisah dengan tanda koma. Kita dapat membuat set dari list dengan memasukkan list ke dalam fungsi `set()`.

Item set dapat berisi data dengan tipe data berbeda. Namun, set tidak bisa berisi list, set, dan dictionary.

```
myset = {1,2,3}
print(myset)
myset_one = set([1,2,3,4,5])
print(myset_one)
myset_two = {4,2,3,"trial",1.0}
print(myset_two)
myset_three = {1,1,2,3,3}
print(myset_three)
myset_three = {1,2,3,[4,5]}
print(myset_three)
```

{1, 2, 3}
{1, 2, 3, 4, 5}
{1.0, 'trial', 2, 3, 4}
{1, 2, 3}

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

```
b = {}
print(type(b))

a = set()
print(type(a))
```

<class 'dict'>
<class 'set'>

14.2 Manipulate set item

Set bersifat mutable. Tapi, karena set adalah tipe data tidak terurut, maka kita tidak bisa menggunakan indeks. Set tidak mendukung indeks maupun slicing.

Untuk menambah item pada set, kita bisa menggunakan fungsi `add()`, dan untuk menambahkan beberapa item sekaligus kita bisa menggunakan fungsi `update()`. list, tuple, maupun string bisa digunakan sebagai masukan dari fungsi `update()`.

```
myset = {1,2,3}
print(myset)

myset.add(4)
print(myset)

myset.update([5,6,7,8])
print(myset)
```

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

Kita bisa menghapus anggota set dengan fungsi `discard()` dan `remove()`. perbedaannya adalah `discard()` tidak memunculkan error apabila anggota yang ingin dihapus ternyata tidak di dalam set, dan `remove()` akan melakukan sebaliknya.

```
myset.remove(4)
print(myset)

myset.discard(5)
print(myset)

myset.discard(5)
print(myset)

myset.remove(4)
print(myset)
```

Selain `discard()` dan `remove()`, kita bisa menghapus anggota set dengan menggunakan fungsi `pop()`. dengan menggunakan fungsi `pop()`, kita menghapus salah satu anggota secara acak. Untuk mengosongkan set, kita dapat menggunakan fungsi `clear()`.

```
set_a = set("Hellow")
print(set_a)
#output : set berisi anggota yang unik
```

```
print(set_a.pop())
print(set_a.pop())
#output : anggota acak
```

```
set_a.clear()
print(set_a)
#output : mengosongkan set
```

```
{ 'w', 'H', 'o', 'e', 'l' }
w
H
set()
```

14.3 Union Sets

Menggabung 2 set dapat digunakan fungsi `union()` untuk mengembalikan nilai dari semua set yang digabungkan. Selain itu, dapat digunakan fungsi `update()` untuk menginputkan nilai dari satu set ke set lainnya

```
myset = {1,2,3}
myset_one = set(['a','b','c'])
myset.update(myset_one)
print(myset)
```

```
myset = {1,2,3}
myset_one = set(['a','b','c'])
set3 = myset.union(myset_one)
print(set3)
```

14.4 Intersection Sets

Irisan dari himpunan A dan B adalah anggota yang sama di kedua himpunan tersebut. Irisan dapat dilakukan dengan menggunakan operator (&) atau fungsi `intersection()`.

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

print(A & B)
print(A.intersection(B) )
```

14.5 Difference

Selisih dari himpunan A dan B adalah anggota yang hanya ada di A dan tidak ada di B, maupun sebaliknya. Intersection dilakukan dengan operator (-) atau fungsi `difference()`.

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

print(A - B)
print(A.difference(B) )
```

14.6 Symetric difference

Operasi komplemen dari himpunan A dan B adalah semua anggota di kedua himpunan kecuali bilangan irisan dari kedua himpunan. Jadi komplemen adalah gabungan dua himpunan dikurangi dengan irisan keduanya. Komplemen dapat menggunakan operator ^ atau fungsi `symmetric_difference()`.

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

print(A ^ B)
print(A.symmetric_difference(B) )
```

14.7 Set method

Metode	Deskripsi
<code>add()</code>	Menambahkan satu anggota ke set
<code>clear()</code>	Menghapus semua anggota set

<code>copy()</code>	Mengembalikan shallow copy 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 komplement dari dua set
<code>symmetric_difference_update()</code>	Mengupdate set dengan komplement dari set ini dan set lainnya

union()	Mengembalikan set baru berisi gabungan dua atau lebih set
update()	Mengupdate set dengan gabungan set ini dan set lainnya

15. Python dictionary

15.1 Introduction to dictionary

Dictionary adalah tipe data yang anggotanya terdiri dari pasangan **key:value**. Dictionary bersifat tidak berurut sehingga anggotanya tidak memiliki indeks. Dictionary dibuat dengan menempatkan anggotanya di dalam tanda kurung kurawal {}, dipisahkan oleh tanda koma. Key dalam dictionary harus bersifat unik, tidak boleh ada dua kunci yang sama dalam dictionary.

```
my_dicta = {}
my_dictb = {1:'sepatu', 2:'tas'}
my_dictc = {'warna' : 'merah', 1:[456]}
my_dictd = dict({1:'merah',2:'tas'})
print(my_dicta)
print(my_dictb)
print(my_dictc)
print(my_dictd)
```

```
{}
```

```
{1: 'sepatu', 2: 'tas'}
```

```
{'warna': 'merah', 1: [456]}
```

```
{1: 'merah', 2: 'tas'}
```

15.2 Manipulate dictionary items

Anggota dictionary diakses dengan menggunakan **Key**. Selain itu, bisa juga diakses dengan menggunakan fungsi get(). Bila menggunakan get() dan key tidak ada di dalam dictionary, maka akan mengembalikan **None**. Apabila tidak menggunakan fungsi get(), maka akan terjadi **KeyError** saat mengakses key yang tidak ada di dalam dictionary.

```
my_dicta = {}
my_dictb = {1:'sepatu', 2:'tas'}
my_dictc = {'warna' : 'merah', 1:[456]}
my_dictd = dict({1:'merah',2:'tas'})
print(my_dictb.get(1)) #akses menggunakan get
print(my_dictc['warna']) #akses menggunakan key
```

```
sepatu
```

```
merah
```

Anda dapat memperbarui dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti contoh di bawah ini

```
my_dict = {'warna' : 'merah', 1:[456]}
my_dict[1] = 10
my_dict['warna'] = 'hijau'
print(my_dict)          {'warna': 'hijau', 1: 10}
```

Untuk menghapus elemen dictionary secara individual digunakan statemen del dengan diikuti key pada dictionary. Untuk menghapus semua dictionary dapat digunakan statement del diikuti nama dictionary atau dengan fungsi clear()

```
my_dict = {'warna' : 'merah', 1:[456]}
del my_dict[1]
print(my_dict)

my_dict.clear()
print(my_dict)          {'warna': 'merah'}
del my_dict              {}
```

15.3 Dictionary method 1

Fungsi Python	Penjelasan
cmp(dict1, dict2)	Membandingkan unsur keduanya.
len(dict)	Memberikan panjang total Dictionary. Ini sama dengan jumlah item dalam Dictionary.
str(dict)	Menghasilkan representasi string yang dapat dicetak dari Dictionary
type(variable)	Mengembalikan tipe variabel yang lulus. Jika variabel yang dilewatkan adalah Dictionary, maka akan mengembalikan tipe Dictionary.

15.4 Dictionary method 2

Method Python	Penjelasan
dict.clear()	Menghapus semua elemen Dictionary
dict.copy()	Mengembalikan salinan Dictionary

<code>dict.fromkeys()</code>	Buat Dictionary baru dengan kunci dari sequence dan nilai yang disetel ke nilai.
<code>dict.get(key, default=None)</code>	For key, nilai pengembalian atau default jika tombol tidak ada dalam Dictionary
<code>dict.has_key(key)</code>	Mengembalikan true jika key dalam Dictionary, false sebaliknya
<code>dict.items()</code>	Mengembalikan daftari dari pasangan tuple dictionary (key, value)
<code>dict.setdefault(key, default=None)</code>	Mirip dengan get (), tapi akan mengatur dict [key] = default jika kunci belum ada di dict
<code>dict.update(dict2)</code>	Menambahkan pasangan kunci kata kunci dict2 ke dict
<code>dict.values()</code>	Mengembalikan daftar nilai dictionary
<code>dict.keys()</code>	Mengembalikan daftar key dictionary

16. Python function

16.1 Introduction to function

Fungsi adalah blok program untuk melakukan tugas tertentu yang berulang. Fungsi membuat kode program menjadi reusable, artinya hanya didefinisikan sekali saja dan bisa digunakan berulang kali dari tempat lain di dalam program.

Fungsi memecah keseluruhan program menjadi bagian-bagian yang lebih kecil sehingga lebih mudah diorganisir dan dimanage.

Contoh fungsi yang sering digunakan adalah `print()`, `type()` yang merupakan fungsi bawaan dari Python. Kita pun dapat membuat fungsi kita sendiri sesuai kebutuhan.

Mendefinisikan Fungsi

```
def function_name(parameters) :  
    statements(s)  
    return (expression)
```

Penjelasan :

1. Kata kunci def diikuti function_name (nama fungsi), tanda kurung dan tanda titik dua (:):a menandai header (kepala) fungsi.
2. Parameter / argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi
3. Setelah itu diletakkan baris-baris pernyataan (statements). jangan lupa indentasi untuk menandai blok fungsi
4. Return bersifat opsional, berguna untuk mengembalikan suatu nilai expression dari fungsi.

Memanggil Fungsi

Bila fungsi didefinisikan, maka ia sudah bisa dipanggil dari tempat lain di dalam program. Untuk memanggil fungsi adalah dengan mengetik nama fungsi beserta parameternya seperti contoh di atas

```
def fruit(buah) :  
    print("Saya suka buah",buah)  
fruit('Semangka')  
                                     Saya suka buah Semangka
```

Return

Pernyataan return digunakan untuk keluar dari fungsi dan kembali ke baris selanjutnya di mana fungsi dipanggil. Return bisa berisi satu atau beberapa ekspresi nilai yang dievaluasi dan nilai tersebut akan dikembalikan

16.2 Argument in function

Kita bisa memanggil fungsi dengan :

a. Argumen wajib

Argumen wajib adalah argumen yang dilewatkan ke dalam fungsi dengan urutan posisi yang benar yang mana jumlah argumen saat pemanggilan harus sama persis dengan jumlah argumen pada pendefinisian fungsi.

```
def fruit(buah) :  
    print("Saya suka buah", buah)  
fruit('Semangka')  
  
fruit() #akan muncul error
```

b. Argumen kata kunci

Argumen dengan kata kunci berkaitan dengan pemanggilan fungsi. Ketika menggunakan argumen ini, fungsi pemanggil menentukan argumen dari nama parameter-nya. Hal ini membuat kita bisa mengabaikan argumen atau menempatkannya dengan urutan sembarang.

```
def fruit(buah) :  
    print("Saya suka buah", buah)  
fruit(buah = 'Semangka')
```

Dan urutan parameter tidak menjadi masalah

```
def identitas(nama, nim) :  
    print("Nama saya ", nama)  
    print("NIM saya ", nim)  
identitas(nim=173, nama = 'Ayu')
```

c. Argumen default

Fungsi dengan argumen default menggunakan nilai default untuk argumen yang tidak diberikan nilainya saat pemanggilan fungsi. Pada contoh di bawah ini, fungsi akan menampilkan usia default bisa argumen usia tidak diberikan

```
def identitas(nama, nim=180) :  
    print("Nama saya ", nama)      Nama saya  Ayu  
    print("NIM saya ", nim)        NIM saya  173  
identitas(nim=173, nama = 'Ayu')  Nama saya  Ayu  
identitas(nama="Ayu")             NIM saya  180
```

d. Argumen dengan panjang sembarang

Argumen dengan panjang sembarang ini berbeda dari fungsi dengan argumen wajib dan default. Karena argumennya tidak disebutkan saat pendefinisian fungsi. Tanda asterisk (*) ditempatkan sebelum nama variabel yang menyimpan

nilai dari argumen yang tidak didefinisikan. Tuple ini akan kosong bila tidak ada argumen tambahan pada saat pemanggilan fungsi.

```
def funct1( arg1, *vararg ):  
    print ("Outputnya adalah: ")  
    print (arg1)  
    for var in vararg:  
        print (var) |  
funct1(20)
```

16.3 Scope Variabel

Di Python, tidak semua variabel bisa diakses dari semua tempat. Ini tergantung dari tempat di mana kita mendefinisikan variabel. Variabel ada 2 jenis, yaitu variabel global (didefinisikan di luar fungsi) dan variabel local (didefinisikan di dalam fungsi).

Maka variabel lokal hanya bisa diakses dari dalam fungsi di mana ia didefinisikan, dan variabel global diakses dari seluruh tempat dimanapun di dalam program.

```
total = 0 #variabel global  
def jumlah(a,b):  
    total = a + b; #variabel local  
    print ("Local variabel berjumlah : ", total)  
    return total  
  
# Pemanggilan fungsi sum  
jumlah( 4, 7 )  
print ("Global variabel berjumlah : ", total )
```

Local variabel berjumlah : 11
Global variabel berjumlah : 0

17. Python Exception Handling

17.1 Exception

Seringkali error pada program menyebabkan program berhenti sendiri. Error yang terjadi akibat kesalahan struktur program disebut syntax error. Contohnya :

```
a = 10  
if a>10  
    print("a bernilai lebih dari 10")
```

if a>10
 ^
SyntaxError: invalid syntax

Penyebab dari kesalahan di atas adalah kurangnya titik dua pada syntax if. Error juga dapat terjadi pada saat runtime (saat program berjalan). Error jenis ini disebut eksepsi. Contoh eksepsi adalah saat kita membagi bilangan dengan nol, maka akan muncul `ZeroDivisionError`, dan lain sebagainya.

Saat terjadi eksepsi, python akan menampilkan traceback dan detail di mana kesalahan terjadi.

```
a = 10/0  
print(a)
```

```
a = 10/0  
ZeroDivisionError: division by zero
```

17.2 Build in Exception in Python

Eksepsi	Penyebab Error
AssertionError	Muncul pada saat pernyataan assert gagal
AttributeError	Muncul pada saat penugasan terhadap attribute atau referensi gagal
EOFError	Muncul saat fungsi input() mendapatkan kondisi akhir file (end-of-file)
FloatingPointError	Muncul saat operasi terhadap bilangan float gagal
GeneratorExit	Muncul saat metode close() generator dipanggil
ImportError	Muncul saat modul yang hendak diimpor tidak ditemukan
IndexError	Muncul saat indeks dari sequence berada di luar range
KeyError	Muncul saat suatu key tidak ditemukan di dalam dictionary
KeyboardInterrupt	Muncul saat user menekan tombol interupsi (Ctrl + C)
MemoryError	Muncul saat operasi kehabisan memori

NameError	Muncul saat variabel tidak ditemukan
NotImplementedError	Muncul oleh metode abstrak
OSError	Muncul saat sistem operasi bersangkutan mengalami error
OverflowError	Muncul saat hasil operasi perhitungan terlalu besar untuk direpresentasikan
ReferenceError	Muncul saat weak reference digunakan untuk mengakses referensi sampah program
RuntimeError	Muncul saat error yang terjadi di luar semua kategori eksepsi lain
StopIteration	Muncul oleh fungsi next() untuk menunjukkan bahwa tidak ada lagi item yang tersisa pada iterator
SyntaxError	Muncul oleh parser saat terjadi kesalahan sintaks
IndentationError	Muncul saat ada indentasi yang salah
TabError	Muncul saat indentasi memiliki jumlah spasi atau tab yang tidak konsisten
SystemError	Muncul saat interpreter mendeteksi kesalahan internal
SystemExit	Muncul oleh fungsi sys.exit()
TypeError	Muncul saat melakukan operasi pada tipe data yang tidak sesuai

UnboundLocalError	Muncul saat referensi dibuat untuk variabel lokal dari fungsi, tapi tidak ada nilainya.
UnicodeError	Muncul saat terjadi kesalahan berkenaan dengan encoding dan decoding unicode
UnicodeEncodeError	Muncul saat terjadi kesalahan pada proses encoding
UnicodeDecodeError	Muncul saat terjadi kesalahan pada proses decoding
UnicodeTranslateError	Muncul saat terjadi kesalahan berkenaan dengan penerjemahan unicode
ValueError	Muncul saat fungsi menerima argumen yang tipe datanya salah
ZeroDivisionError	Muncul saat terjadi operasi pembagian bilangan dengan nol

17.3 Exception Handling with Try, Except, and Finally

Terjadinya eksepsi pada program dapat menyebabkan program terhenti. Maka untuk mengantisipasi digunakan pernyataan try dan except. Di dalam blok try kita meletakkan baris program yang kemungkinan akan terjadi error. Apabila ada error, maka penanganan diserahkan kepada blok except.

```
lists = ['a', 0, 4]
for a in lists:
    try:
        print("Masukan:", a)
        r = 1/int(a)
        break
    except:
        print("Upps!", sys.exc_info()[0], " terjadi.")
        print("Masukan berikutnya.")
        print()
print("Kebalikan dari ", a, " =", r)
```

```
Masukan: a
Upps! <class 'ValueError'> terjadi.
Masukan berikutnya.
```

```
Masukan: 0
Upps! <class 'ZeroDivisionError'>
terjadi.
Masukan berikutnya.
```

```
Masukan: 4
Kebalikan dari 4 = 0.25
```

Program di atas digunakan untuk mencari kebalikan dari bilangan yang diinputkan. Saat bilangan 1 dibagi dengan a dan 0, maka tidak bisa dilakukan, sehingga muncul error. Apabila tidak dilakukan eksepsi, maka program akan langsung terhenti saat error terdeteksi.

Menangani Eksepsi Tertentu

Pada contoh di atas kita menangani error secara umum. Padahal ada beberapa pengelompokan error, seperti `TypeError`, `ValueError`, `SyntaxError`, dan sebagainya. Pernyataan `try`, bisa memiliki sejumlah pernyataan `except` untuk menangani jenis-jenis eksepsi secara spesifik. Contoh nya sebagai berikut :

```
try:
    # Lakukan sesuatu
    pass

except ValueError:
    # tangani eksepsi ValueError
    pass

except (TypeError, ZeroDivisionError):
    # menangani multi eksepsi
    # TypeError dan ZeroDivisionError
    pass

except:
    # menangani eksepsi lainnya
    pass
```

Pernyataan `pass` adalah pernyataan yang tidak melakukan apa-apa yaitu berisi statement kosong dan sering digunakan untuk mengisi blok fungsi yang masih kosong.

Memunculkan Eksepsi

Eksepsi muncul bila terjadi error saat runtime atau saat program berjalan. Namun, kita juga bisa memunculkan eksepsi dengan sengaja untuk maksud tertentu dengan menggunakan kata kunci `raise`. Contohnya :

```

raise KeyboardInterrupt
Traceback (most recent call last):
...
KeyboardInterrupt

try:
    a = int(input("Masukkan sebuah bilangan positif: "))
    if a <= 0:
        raise ValueError("Itu bukan bilangan positif!")
except ValueError as ve:
    print(ve)

Masukkan sebuah bilangan positif: -3
Itu bukan bilangan positif!

```

Try...finally

Pernyataan try bisa memiliki pasangan pernyataan finally yang mana statement di dalam blok finally tetap dieksekusi bagaimanapun kasusnya. Finally biasa digunakan untuk melepaskan koneksi dengan resource eksternal. Seperti misal saat kita mengedit file di internet dan koneksi tiba-tiba terputus. Maka kita perlu membersihkan resource yang digunakan. Aksi (memutus koneksi ke jaringan) dilakukan dengan menggunakan pernyataan finally untuk menjamin suatu perintah tetap dieksekusi.

Contoh try...finally :

```

try:
    f = open("C:\\test.txt")
    # melakukan operasi terhadap file
finally:
    f.close()

```

18. Python Lamda

Fungsi lamda atau biasa dikenal dengan sebutan Anonymous Function yaitu merupakan sebuah fungsi yang tidak memiliki nama fungsi, yang harus kita ketahui yaitu bahwa lambda bukanlah sebuah perintah (statement) namun lambda tersebut merupakan sebuah ekspresi (expression) yang menjalankan sebuah perintah yang terdapat pada fungsi yang akan di panggil nantinya. Lamda umumnya digunakan apabila kita punya angka yang ingin kita hitung(menggunakan operasi matematika) dengan angka yang belum kita ketahui nilainya.

Contoh penggunaan fungsi lamda:

```
10 x = lambda a : a + 10
11 y = lambda c, b : c * b
12 print(x(5))
13 print(y(5, 6))
```

19. Python Array

Array pada python memiliki fungsi yang sama dengan array pada Bahasa pemrograman lainnya, array merupakan variable yang dapat menyimpan beberapa nilai elemen di dalamnya.

Contoh Penggunaan:

```
10 bahasaPemrograman = ["Python", "Java", "C", "C++", "Kotlin"]
11
12 print(bahasaPemrograman[0], "dan", bahasaPemrograman[1])
```

Output:

```
In [1]: runfile('C:/Users/Dicky/.spyder-py3/untitled0.py', wdir='C:/Users/
Dicky/.spyder-py3')
Python dan Java
```

Menghitung panjang Array

Kita juga dapat menghitung berapa banyak elemen yang terdapat di dalam array dengan cara menggunakan fungsi `len()`

Contoh Penggunaan:

```
10 bahasaPemrograman = ["Python", "Java", "C", "C++", "Kotlin"]
11 x= len(bahasaPemrograman)
12
13 print("Jumlah elemen adalah ",x)
```

Output:

```
In [2]: runfile('C:/Users/Dicky/.spyder-py3/untitled0.py', wdir='C:/Users/
Dicky/.spyder-py3')
Jumlah elemen adalah  5
```

Method yang dapat digunakan pada Array

Method	Description
<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the first item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

20. Python Scope

Scope merupakan cakupan atau bagian yang dapat diakses oleh suatu variable. Scope dapat dibedakan menjadi 2 yaitu local scope dan juga global scope

Local Scope

Merupakan variable yang hanya dapat diakses di dalam suatu fungsi karena peninisialisasiannya berada di dalam fungsi tersebut.

Contoh Penggunaan:

```

10 def myfunc():
11     x = 300
12     def myinnerfunc():
13         print(x)
14     myinnerfunc()
15
16 myfunc()

```

Output:

```

In [1]: runfile('C:/Users/Dicky/.spyder-py3/untitled0.py', wdir='C:/Users/
Dicky/.spyder-py3')
300

```

Global Scope

Merupakan variable yang dapat diakses dari semua fungsi atau method yang terdapat pada program tersebut karena proses inisialisasinya berada di luar method apapun dan berada pada bagian paling atas dari program tersebut.

Contoh Penggunaan:

```
10 x = 300
11
12 def myfunc():
13     print(x)
14
15 myfunc()
16 print(x)
```

Output:

```
In [1]: runfile('C:/Users/Dicky/.spyder-py3/untitled0.py', wdir='C:/Users/
Dicky/.spyder-py3')
300
```

21. Python PIP

Pip merupakan sebuah tools yang dapat memudahkan programmer untuk menginstall library-library tambahan pada python

Instalasi PIP dan mengunduh package:

- Sebelum mengunduh PIP sebaiknya menentukan versi dari Python yang kita miliki dengan cara.

```
C:\Users\Your_Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version
```

- PIP dapat diunduh melalui : <https://pypi.org/project/pip/>
- Setelah mengunduh package, maka proses selanjutnya adalah membuka commad line lalu arahkan ke lokasi script directory python, lalu ketikkan :

```
C:\Users\Your_Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip install camelcase
```

Contoh diatas misalkan package yang diunduh bernama camelcase

- Setelah selesai maka package siap digunakan

Menghapus Package

Untuk menghapus package dapat dilakukan dengan cara(contohnya menghapus package camelcase):

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip uninstall camelcase
```

Tampilan setelah di run:

```
Uninstalling camelcase-0.2.1:
Would remove:
  c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camelcase-0.2-
py3.6.egg-info
  c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camelcase\*
Proceed (y/n)?
```

Setelah di run program akan meminta konfirmasi dari si user.

Cara melihat list package

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip list
```

Tampilan setelah di run:

```
Uninstalling camelcase-0.2.1:
Would remove:
  c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camelcase-0.2-
py3.6.egg-info
  c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camelcase\*
Proceed (y/n)?
```

22. Python User Input

Seperti halnya Bahasa pemrograman yang lainnya python juga memiliki method untuk memberikan keleluasaan pada user untuk menginputkan data pada saat program sedang berjalan. Ada 2 method yang dapat digunakan oleh user bergantung dari versi python yang si user gunakan. Python dengan versi 3.6 menggunakan method `input()` sedangkan Python 2.7 menggunakan method `raw_input()`.

Contoh penggunaan Python 3.6:

```
username = input("Enter username:")
print("Username is: " + username)
```

Kedua Syntax diatas memiliki hasil output an yang sama:

```
C:\Users\My Name>python demo_user_input3.py
Enter username:Chamdani
Username is: Chamdani
```