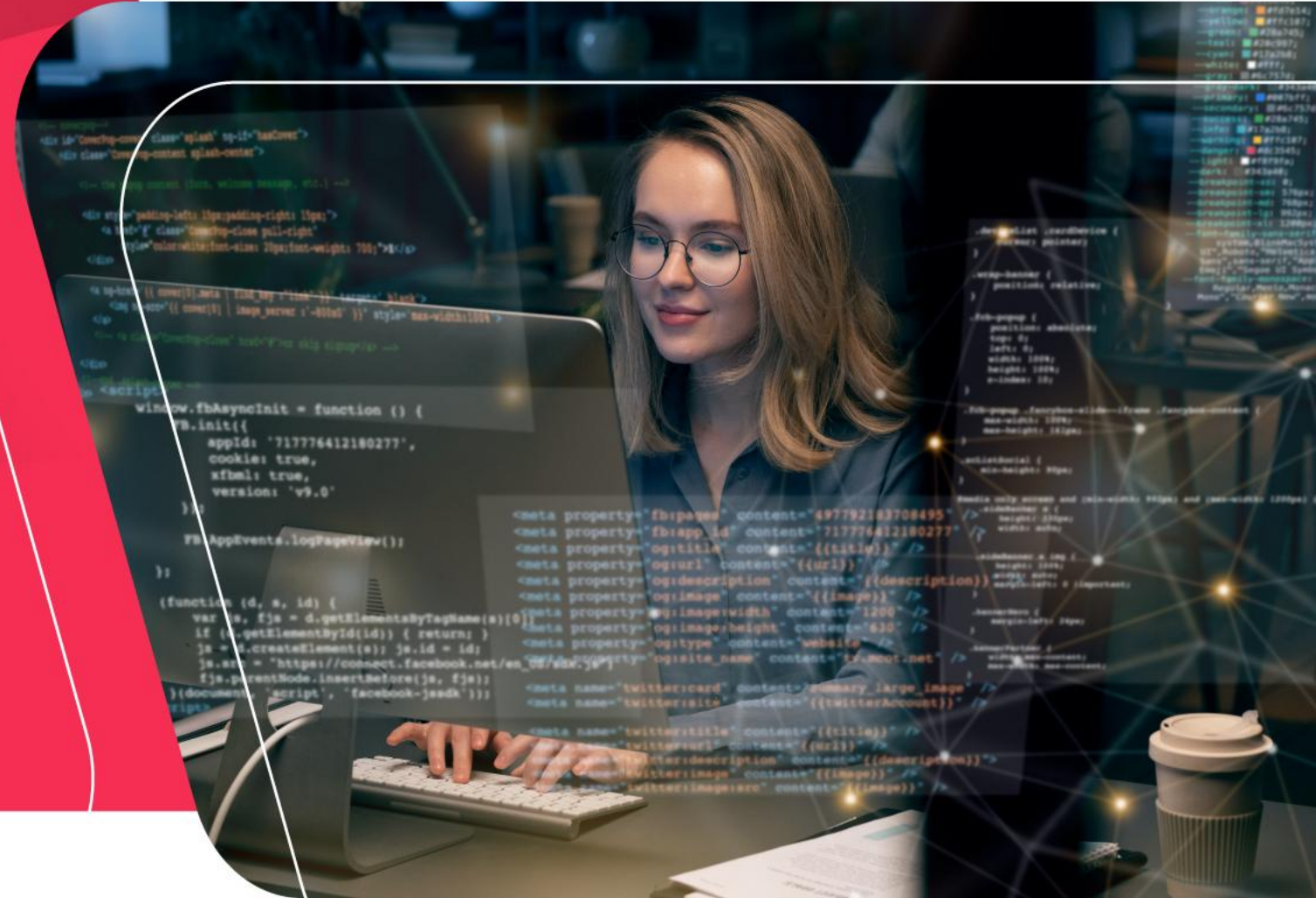


Python for Data Science

Day 01



Python

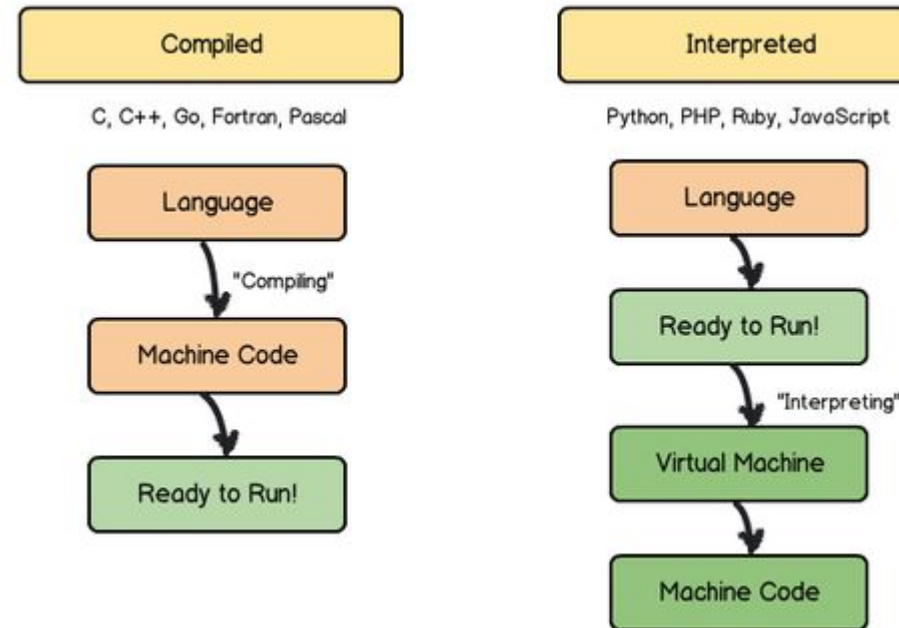
What is Python?



- Python is interpreter language **Guido van Rossum**, released in 1991.
- Python is a programming language that lets you work more quickly and integrate your systems more effectively. (python.org)

What is Python?

Python is interpreted language, what is interpreter language?



Scripting vs Compiling Language

Applications of Scripting Languages :

- To automate certain tasks in a program
- Extracting information from a data set
- Less code intensive as compared to traditional programming languages

Applications of Programming Languages :

- They typically run inside a parent program like scripts
- More compatible while integrating code with mathematical models
- Languages like JAVA can be compiled and then used on any platform

Why we learn to Python?

C++ "Hello World"

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

Java "Hello World"

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

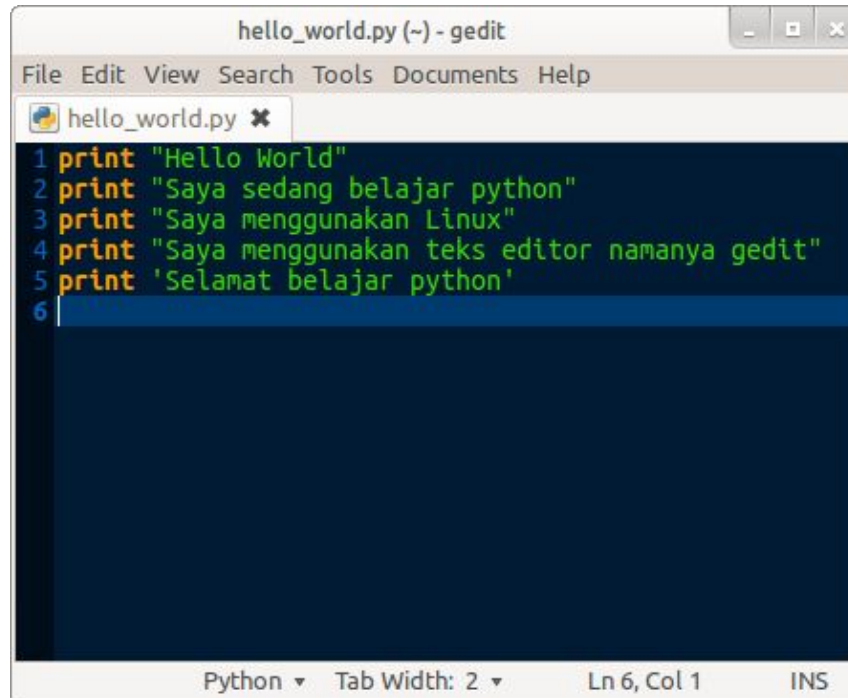
Python

```
print "Hello world"
```

1. Easy to develop
2. Can create to develop many products (Website, Desktop, and Machine Learning Model)



How to create Python Script



```
hello_world.py (~) - gedit
File Edit View Search Tools Documents Help
hello_world.py x
1 print "Hello World"
2 print "Saya sedang belajar python"
3 print "Saya menggunakan Linux"
4 print "Saya menggunakan teks editor namanya gedit"
5 print 'Selamat belajar python'
6
Python Tab Width: 2 Ln 6, Col 1 INS
```

- We can write in editor like notepad, and etc to save python script.
- To create python script, you can save using notepad as script_name.py (all extensions) or run a command **python script_name.py**.

5 Rule to Write Python

- How to Write Statement
- String
- Naming Convention
- Programming Block of Statement
- Comment

```
def fib(n):  
    print 'n = ', n  
    if n > 1:  
        return n * fib(n-1)  
    else:  
        print 'end'  
        return 1
```


Write a statement

```
penulisan.py > ...  
1 print("Hello world")  
2 print ("Belajar Python Dasar")  
3 nama = "Hartono"
```

```
4  
5 print ("Hello"); print ("World")  
6 print("Belajar Python Dasar.")  
7 nama = "Hartono"; nick = "Cimiko"
```

- Statement is instruction or command that will be executed by computer.
- Python use semicolon (;) to divide two statements in one line.

String

```
10  ## Penulisan String Pada Python ##  
11  judul = "Belajar Penulisan String"  
12  penulis = 'Hartono'
```

```
13  
14  judul = """Belajar Penulisan String"""  
15  penulis = '''Hartono'''
```

- String is compound of character.
- String is written by using double quote (" ") or single quote (' ').
- You can enter new line on string by using triple of double quote (""" """)

Naming Convention

```
17  ## Penulisan Case Pada Python ##  
18  judul = "Belajar Penulisan Case"  
19  Judul = "Variable judul dan Judul itu dibedakan"  
20
```

```
21  # Snake Case digunakan pada:  
22  module_name, package_name, method_name, function_name,  
23  global_var_name, instance_var_name,  
24  function_parameter_name, local_var_name  
25  
26  # CamelCase digunakan Pada:  
27  ClassName, ExceptionName  
28  
29  # ALL CAPS digunakan Pada:  
30  GLOBAL_CONSTANT_NAME
```

- Python is case sensitive means you need to comply on capital or lower case (e.g: judul, Judul)
- If you defined a variable name as judul, and you call your variable as Judul, it will show up an error (variable is not found)
- Python is using snake case to define variable means every words must split by underscore (_) (e.g: angka_pertama, sisa_bagi)

Naming

```
17  ## Penulisan Case Pada Python ##  
18  judul = "Belajar Penulisan Case"  
19  Judul = "Variable judul dan Judul itu dibedakan"  
20
```

```
21  # Snake Case digunakan pada:  
22  module_name, package_name, method_name, function_name,  
23  global_var_name, instance_var_name,  
24  function_parameter_name, local_var_name  
25  
26  # CamelCase digunakan Pada:  
27  ClassName, ExceptionName  
28  
29  # ALL CAPS digunakan Pada:  
30  GLOBAL_CONSTANT_NAME
```

- Python use CamelCase to define class (will be explained in bootcamp) and exception.
- To define constant variable (variable that will be used in one file) Python use ALL CAPS and snake_case combined (e.g: PI)

Block statement



```
32  ### Penulisan Blok Program Python ###
33  username = 'cimiko'
34
35  if username == 'cimiko':
36      print "Hallo", username
37      print("Penulisan yang Benar")
38
39  for i in range(10):
40      print i
```



```
42  if username == 'cimiko':
43      print("Jangan Ditiru")
44      print["Penulisan Yang Salah"]
```

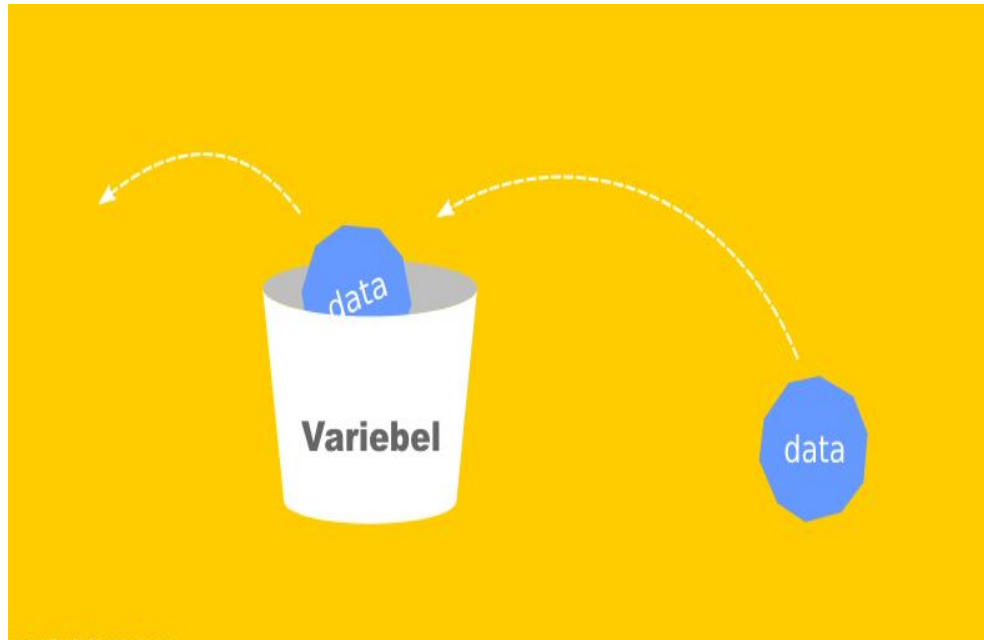
- Block is list of instructions that will be run in one place.
- Block is written by using tab or four times of blank space (to make a different of code).

Comment

```
46  ### Cara Penulisan Komentar Pada Python ###
47
48  # Komentar menggunakan tanda pagar
49  " Komentar menggunakan tanda petik dua"
50  ' Komentar menggunakan tanda petik satu'
51  """ Komentar menggunakan triple tanda petik"""
52
```

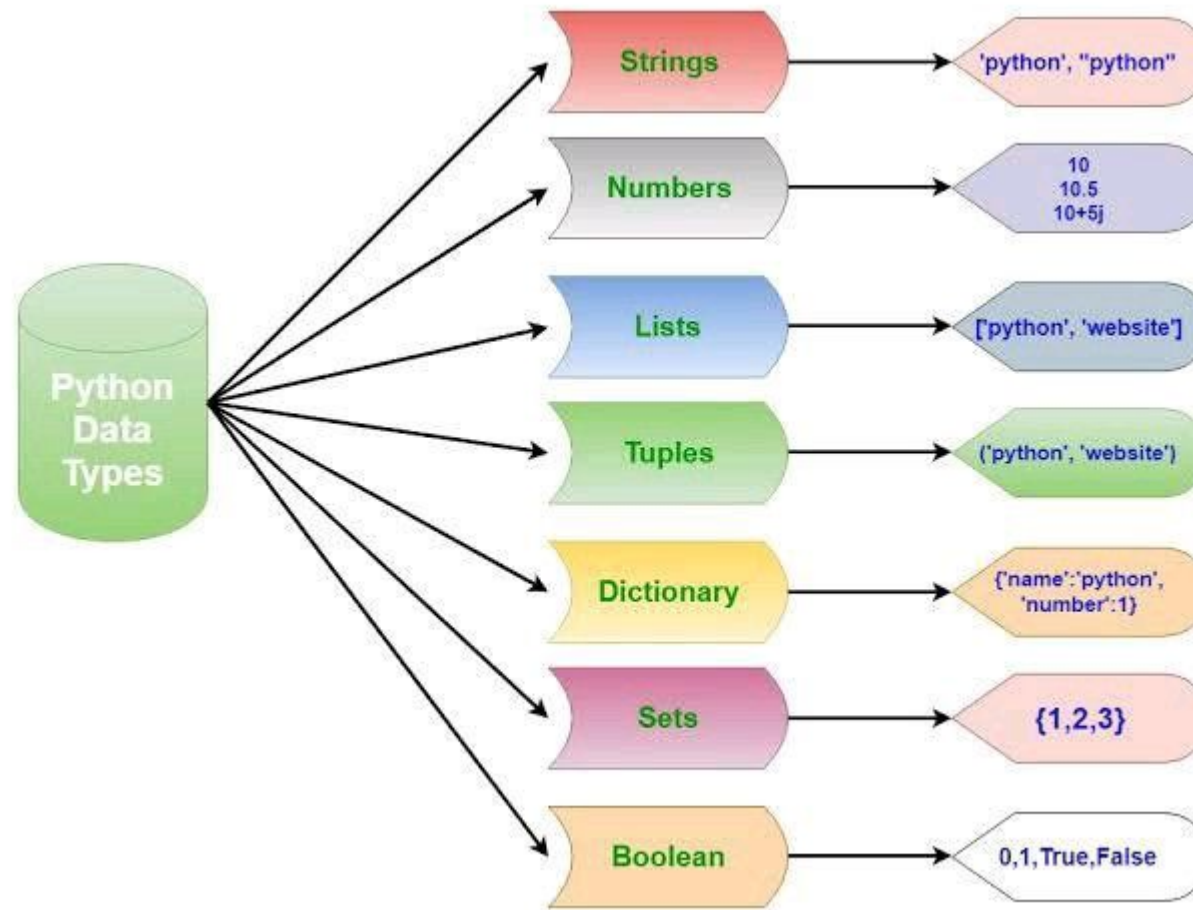
- Comment is one or many lines that will be not execute to give any knowledge about what is python script file using for to another programmer.
- Comment also can be used for skip any instructions in Python.
- There are many ways to write a comment in python:
 1. Use number sign/hash symbol (#).
 2. Use triple of number sign/hash symbol (###).
 3. Use double quote or single quote (" ") atau (' ').
 4. Use triple of double quote (""" """).

Variable and Data Type in Python



- Variable is a place to store a data, meanwhile data type is content that we store.
- Variable is mutable means its value can be changed.

Data Type in Python



List

- Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end-1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

List

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
```

```
tinylist = [123, 'john']
```

```
print list      # Prints complete list
```

```
print list[0]   # Prints first element of the list
```

```
print list[1:3] # Prints elements starting from 2nd till 3rd
```

```
print list[2:]  # Prints elements starting from 3rd element
```

```
print tinylist * 2 # Prints list two times
```

```
print list + tinylist # Prints concatenated lists
```

Output:

```
['abcd', 786, 2.23, 'john', 70.2]
```

```
abcd
```

```
[786, 2.23]
```

```
[2.23, 'john', 70.2]
```

```
[123, 'john', 123, 'john']
```

```
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

Tuple

- A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]), and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as **read-only** lists.

Tuple

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
```

```
tinytuple = (123, 'john')
```

```
print tuple      # Prints complete list
```

```
print tuple[0]   # Prints first element of the list
```

```
print tuple[1:3] # Prints elements starting from 2nd till 3rd
```

```
print tuple[2:]  # Prints elements starting from 3rd element
```

```
print tinytuple * 2 # Prints list two times
```

```
print tuple + tinytuple # Prints concatenated lists
```

OUTPUT:

```
('abcd', 786, 2.23, 'john', 70.2)
```

```
abcd
```

```
(786, 2.23)
```

```
(2.23, 'john', 70.2)
```

```
(123, 'john', 123, 'john')
```

```
('abcd', 786, 2.23, 'john', 70.2, 123, 'john')
```

Dictionary

- Python 's dictionaries are hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs.
- Keys can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Dictionary

```
dict = {}  
dict['one'] = "This is one"  
dict[2] = "This is two"  
tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}  
print dict['one']    # Prints value for 'one' key  
print dict[2]        # Prints value for 2 key  
print tinydict       # Prints complete dictionary  
print tinydict.keys() # Prints all the keys  
print tinydict.values() # Prints all the values
```

OUTPUT:

This is one

This is two

{'dept': 'sales', 'code': 6734, 'name': 'john'}

['dept', 'code', 'name']

['sales', 6734, 'john']

Define a variable

```
variabel.py > ...  
1  ### Membuat Variabel di Python ###  
2  variabel_string = "ini isi variabel"  
3  variabel_integer = 100  
4  
5  print variabel_string  
6  print variabel_integer  
7
```

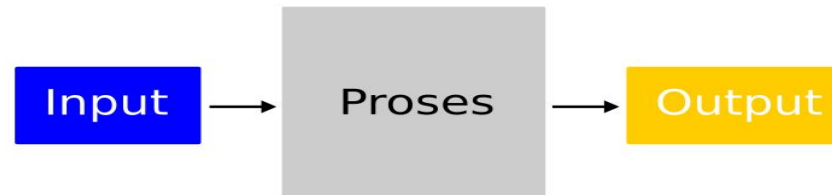
- Variable can be defined with format (**nama_variabel = <nilai>**).
- We can see what is inside its variable with using command **print**.

How to convert data type

Function	Description
<code>int(x [,base])</code>	Converts x to an integer. base specifies the base if x is a string.
<code>long(x [,base])</code>	Converts x to a long integer. base specifies the base if x is a string.
<code>float(x)</code>	Converts x to a floating-point number.
<code>complex(real [,imag])</code>	Creates a complex number.
<code>str(x)</code>	Converts object x to a string representation.
<code>repr(x)</code>	Converts object x to an expression string.
<code>eval(str)</code>	Evaluates a string and returns an object.
<code>tuple(s)</code>	Converts s to a tuple.
<code>list(s)</code>	Converts s to a list.
<code>set(s)</code>	Converts s to a set.
<code>dict(d)</code>	Creates a dictionary. d must be a sequence of (key,value) tuples.
<code>frozenset(s)</code>	Converts s to a frozen set.
<code>chr(x)</code>	Converts an integer to a character.
<code>unichr(x)</code>	Converts an integer to a Unicode character.
<code>ord(x)</code>	Converts a single character to its integer value.
<code>hex(x)</code>	Converts an integer to a hexadecimal string.
<code>oct(x)</code>	Converts an integer to an octal string.

Input - Output

- This is how we are processing tasks



- Input is prerequisites of doing task
- Process is how to do task
- Output is result of task

Input – Output (Example)



- Eating:
 - Input: Food
 - Process: Human Digestion System
 - Output: Energy

Input – Output (Example)



- Calculator:
 - Input: Numbers
 - Process: Math Operations (Addition, Substraction, Multiply, Divison)
 - Output: Number

How to get input from keyboard

```
input.py > ...
1  ### Cara Mengambil Input dari Keyboard ###
2  nama_variabel = input("Sebuah Teks")
3
4  # Mengambil Input
5  nama = raw_input("Siapa Nama Kamu: ")
6  umur = input("Berapa umur kamu: ")
7
8  # Menampilkan output
9  print "Hello",nama,"umur kamu adalah",umur,"tahun."
```

- Python use `input()` and `raw_input()` to take inputs from keyboard.
- Python3 not using `raw_input()` anymore meanwhile are input will be treated as string.
- If we define variable to store input means its variable will store anything that we type on keyboard.

Output

```
[>>> nama = "Hartono"  
[>>> print("Hallo",nama)  
( 'Hallo', 'Hartono' )  
[>>> print("Hallo " + nama)  
Hallo Hartono  
>>> █
```

- To show teks in our console use **print()** command.
- **print()** command must insert with string. To combine between string and variable, python must use **(+)**.

Use string `format()`

- `format()` is use to concat between variable and text menggabungkan isi variabel dengan teks.
- `{}` on sentence will be replaced by value of variable `nama`.

```
17  ### Menggunakan Fungsi Format() ###
18  nama = raw_input("Nama: ")
19  print "Hello {} apa kabar?".format(nama)
20
21  nama_mu = raw_input("Nama kamu: ")
22  nama_dia = raw_input("Nama dia: ")
23  print "{} dengan {} sepertinya teman baik :)".format(nama_mu, nama_dia)
24
```