

#python

# WORKSHOP PYTHON DASAR

# WORKSHOP PYTHON DASAR



[Nanang Suryadi]

# WORKSHOP PYTHON DASAR



[Nanang Suryadi]



# WORKSHOP PYTHON DASAR



[Nanang Suryadi]



# WORKSHOP PYTHON DASAR



[Nanang Suryadi]



Komunitas python surabaya [surabaya.py](http://surabaya.py)

# Perkenalkan

- Hi, Saya Nanang Suryadi
- Full Stack Developer
- Pembina dan Pengurus Surabaya.py

# Perkenalkan

- Hi, Saya Nanang Suryadi
- Full Stack Developer
- Pembina dan Pengurus Surabaya.py
- LinkenIn <http://intip.in/Rpbi>
- Github <http://intip.in/BTna>
- Twitter @suryakencana007

# Agenda

## Hari ke-1

- Zen of Python
- Syntax dan Output
- Whitespace
- Comments
- Variable
- Tipe Data
- Operator Aritmatika
- Operator Logika
- Control Flow
  - if Statements
  - while
  - for Statements

# Agenda

## Hari ke-2

- Tuple
- List
- Dictionary
- Function
- class
- Fungsi - Fungsi list
- Manipulasi String
- Fungsi-fungsi String
- Install dan Setup Virtual enviroment
- Instalasi Library module Third-party

# Hari ke - 2

# Persiapan Workshop

Berdoa



python 3.5+

#perkenalan peserta

# Zen of Python

# Zen of Python

**import** this

The Zen of Python, by Tim Peters

Beautiful **is** better than ugly.

Explicit **is** better than implicit.

Simple **is** better than complex.

Complex **is** better than complicated.

Flat **is** better than nested.

Sparse **is** better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now **is** better than never.

Although never **is** often better than \*right\* now.

If the implementation **is** hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

# Tuple

# Tuple

# Tuple

Tuple adalah tumpukan dalam python yang berisi beberapa element yang bisa berbeda jenis tipe data yang bersifat immutable. Tuple mirip seperti sekali dengan list, hanya yang berbeda tuple tidak dapat dirubah dan menggunakan parentheses "(-)".

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
tuple_beda_tipe = ("satu", 2, "tiga", 3.0, "lima")
print(tuple_sama_tipe)
print(tuple_beda_tipe)
```

# Tuple

# Tuple

Tuple adalah tumpukan dalam python yang berisi beberapa element yang bisa berbeda jenis tipe data yang bersifat immutable. Tuple mirip seperti sekali dengan list, hanya yang berbeda tuple tidak dapat dirubah dan menggunakan parentheses "(-)".

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
tuple_beda_tipe = ("satu", 2, "tiga", 3.0, "lima")
print(tuple_sama_tipe)
print(tuple_beda_tipe)
```

## Pencarian element dalam tuple

kata kunci in bernilai True jika element ditemukan.

```
tuple_beda_tipe = ("satu", 2, "tiga", 3.0, "lima")
print("satu" in tuple_beda_tipe)
print(2 in tuple_beda_tipe)
print("2" in tuple_beda_tipe and 3.0 in tuple_beda_tipe)
print("Satu" in tuple_beda_tipe and 2 in tuple_beda_tipe)
```

## Penggabungan tuple

Dengan menggunakan operator +, dua buah atau lebih tuple bisa dijadikan dalam satu tuple

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
tuple_beda_tipe = ("satu", 2, "tiga", 3.0, "lima")
tuple_total = tuple_sama_tipe + tuple_beda_tipe
print(tuple_total)
```

## Index Tuple

Setiap element dalam tuple mempunyai alamat index, index disini berfungsi sebagai posisi / urutan element.

| Format penulisan tuple[ **index** ]

```
tuple_sama_tipe = (satu, "dua", "tiga", "empat", "lima")  
  
print(tuple_sama_tipe[2])  
print(tuple_sama_tipe[3])  
print(tuple_sama_tipe[0])
```

## Index ranges

| Format penulisan tuple[ start : end ]

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[2:4])
print(tuple_sama_tipe[0:3])
print(tuple_sama_tipe[3:4])
```

## Index ranges

| Format penulisan tuple[ **start : end** ]

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[2:4])
print(tuple_sama_tipe[0:3])
print(tuple_sama_tipe[3:4])
```

Tanpa **start**

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[:3])
```

## Index ranges

| Format penulisan tuple[ **start : end** ]

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[2:4])
print(tuple_sama_tipe[0:3])
print(tuple_sama_tipe[3:4])
```

Tanpa **start**

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[:3])
```

Tanpa **end**

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(tuple_sama_tipe[3:])
```

Tips: Penggunaan \* operator untuk mengembalikan sisa nilai dari tuple

## Harmful

```
pengurus = ['Chris', 'Nanang', 'Robi', 'Putri', 'Tia', 'Edy']

(first, second, rest) = pengurus[0], pengurus[1], pengurus[2:]

print(rest)

(first, middle, last) = pengurus[0], pengurus[1:-1], pengurus[-1]

print(middle)

(head, penultimate, last) = pengurus[:-2], pengurus[-2], pengurus[-1]

print(head)
```

## Idiomatic

```
pengurus = ('Chris', 'Nanang', 'Robi', 'Putri', 'Tia', 'Edy')

(first, second, *rest) = pengurus
print(rest)

(first, *middle, last) = pengurus
print(middle)

(*head, middle, last) = pengurus
print(head)
```

\*Fungsi ini berlaku hanya di Python 3+

## Merubah Element

Karena sifat dari tuple adalah immutable, maka sebuah tuple tidak dapat dirubah.

## Length

Untuk mengetahui jumlah element dalam sebuah list dapat menggunakan fungsi len

| Format penulisan **len( tuple )**

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(len(tuple_sama_tipe))
```

## Length

Untuk mengetahui jumlah element dalam sebuah list dapat menggunakan fungsi len

| Format penulisan **len( tuple )**

```
tuple_sama_tipe = ("satu", "dua", "tiga", "empat", "lima")
print(len(tuple_sama_tipe))
```

Python mempunyai beberapa fungsi built-in yang digunakan untuk perhitungan statistik dalam tuple

```
print(max((4, 5, 7, 9, 6, 3)))
print(min((4, 5, 7, 9, 6, 3)))
print(sum((4, 5, 7, 9, 6, 3)))
```

# Dictionary

# Dictionary

Sebuah Dictionary mempunyai hubungan key dengan value. Setiap key harus unik untuk mendapatkan nilai dari element dalam Dictionary.

```
{  
    key: value,  
    key: value  
}
```

# Dictionary

Sebuah Dictionary mempunyai hubungan key dengan value. Setiap key harus unik untuk mendapatkan nilai dari element dalam Dictionary.

```
{  
    key: value,  
    key: value  
}
```

## definisi dan Assigment

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
print(dict_assign)
```

### Akses Nilai dari Dict

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
print(dict_assign["chris"])  
  
print(dict_assign["doni"])
```

### Akses Nilai dari Dict

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
print(dict_assign["chris"])  
  
print(dict_assign["doni"])
```

menambahkan element (a key-value pair)

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
dict_assign["surya"] = "a child of surabaya"  
  
print(dict_assign)  
  
dict_assign["doni"] = "a child of Surabaya"  
  
print(dict_assign)
```

### Iteration dalam Dictionary

Dalam setiap iterate for loop di dictionary mengeluarkan key dalam perulangan.

```
for key in dictionary:  
    statement
```

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
for person in dict_assign:  
  
    print(person)  
  
    print(dict_assign[person])
```

## Keys dan Values

Dictionary mempunyai fungsi untuk menampilkan keys dan values dalam bentuk list

| dict.keys() or dict.values()

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
print(dict_assign.keys())  
  
print(dict_assign.values())
```

# Dictionary

## Dictionary

Setiap keys dan values bisa juga di iterate dengan for loop

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
for key in dict_assign.keys():  
    print(key)  
  
for value in dict_assign.values():  
    print(value)
```

# Dictionary

## Dictionary

Setiap keys dan values bisa juga di iterate dengan for loop

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
for key in dict_assign.keys():  
    print(key)  
  
for value in dict_assign.values():  
    print(value)
```

menghapus element dalam Dictionary

```
dict_assign = {  
    "chris": "a child of pakuwon",  
    "doni": "a child of Perak",  
    "juang": "a child of Kediri"  
}  
  
del dict_assign["chris"]  
print(dict_assign)  
del dict_assign  
print(dict_assign)
```

# Function

## Definisi Function

## Function

A function is a sequence of instructions that performs a specific computation. A function can take inputs and produce outputs. Functions are useful for organizing your code and are essential for building complex programs.

---

## Definisi Function

Function didefinisikan dengan keyword def. Setelah keyword def nama fungsi yang bersifat unik.

```
def nama_fungsi():
    instruksi
```

```
def surabaya_py():
    print("Workshop Basic Python")
    print("Tempat di Revio")

surabaya_py()
```

## Parameter Fungsi

```
def nama_fungsi( parameter ):
    instruksi

    # cara memanggil
    nama_fungsi( argument )

    # multiple parameter
def nama_fungsi( parameter1, parameter2, parameter3 ):
    instruksi

    # cara memanggil
    nama_fungsi( arg1, arg2, arg3 )
```

## Returning Value

Fungsi dapat mengembalikan nilai dengan keyword return

| return value

## Returning Value

Fungsi dapat mengembalikan nilai dengan keyword return

| return value

```
def nama_fungsi( parameter ):
    return instruksi

# cara memanggil
print( nama_fungsi( argument ) )

def surabaya_py(nama, lokasi):
    return ''.join([nama, lokasi])

print(surabaya_py("doni", "Perak"))
```

## Default Parameter

```
def nama_fungsi( a, b, input = default):
    instruksi

def surabaya_py(nama, lokasi, statement=""):
    return ''.join([nama, lokasi, statement])

print(surabaya_py("doni", "Perak"))

print(surabaya_py("doni", "Perak", "sedang kuliah di ITS"))
```

# Class

# Class

Python sudah dikenal sejak kemunculan sebagai bahasa pemrograman object-oriented. Sehingga, untuk pembentukan sebuah class dalam python sangat mudah dan sederhana.

## Sekilas komponen OOP

- Class
- Class Variable
- Data member
- Function overloading
- Inheritance
- Method

referensi lebih detil

| <http://intip.in/pythonCl45>

# Class

## Struktur Class

```
class SurabayaPy():
    class_variabel = 0

    def __init__(self, nama, lokasi):
        self.nama = nama
        self.lokasi = lokasi

    def print_nama(self):
        print(self.nama)

    def print_lokasi(self):
        print(self.lokasi)
```

pemanggilan class dan pembuatan object instance

```
object_inst = SurabayaPy("doni", "Perak")

# akses attribut class

object_inst.print_nama()
```

# Class

## Inheritance

### Pewarisan Class

```
class SurabayaPy():
    class_variabel = 0

    def __init__(self, nama, lokasi):
        self.nama = nama
        self.lokasi = lokasi

    def print_nama(self):
        print(self.nama)

    def print_lokasi(self):
        print(self.lokasi)

class OodoSurabaya(SurabayaPy):

    def __init__(self, nama, lokasi):
        super(OodoSurabaya, self).__init__(nama, lokasi)
        self.nama = nama
        self.lokasi = lokasi

    def print_nama(self):
        print(self.nama + ' :Oodo Surabaya')

    def print_lokasi(self):
        print(self.lokasi + ' :Oodo Surabaya')
```

# Fungsi-Fungsi list

### Counting Elements

Menghitung jumlah total dari element didalam sebuah list

```
# Fungsi Count
list.count( value )

coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]

#Number of quarters
print ( coin_jar.count(25) )

#Number of pennies
print ( coin_jar.count(1) )
```

### Pencarian Index

Mencari posisi index elements dalam sebuah list

```
# Fungsi index
list.index( value )

coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]

#Number of quarters
print ( coin_jar.index(25) )

#Number of pennies
print ( coin_jar.index(1) )
```

### Menambahkan Element

Penambahan element di sebuah list menggunakan fungsi append

```
# Fungsi append
list.append( value )

coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]

coin_jar.append(25)
print ( coin_jar )

coin_jar.append(1)
print ( coin_jar )
```

### Menghapus Element

Penghapusan element di sebuah list menggunakan fungsi remove

```
# Fungsi remove
list.remove( value )

coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]

coin_jar.remove(25)
print ( coin_jar )

coin_jar.remove(1)
print ( coin_jar )
```

### Membalik posisi Element

Pembalikan posisi element di sebuah list menggunakan fungsi reverse

```
# Fungsi reverse  
list.reverse()  
  
coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]  
  
coin_jar.reverse()  
print ( coin_jar )
```

### Mengurutkan nilai Element

Pengurutan nilai element di sebuah list menggunakan fungsi sort dimana pengurutannya bersifat ascending

```
# Fungsi sort
list.sort()

coin_jar = [25, 10, 10, 1, 1, 5, 1, 5, 5, 25]

coin_jar.sort()
print ( coin_jar )
```

# Manipulasi String

# Manipulasi String

String

## String Indexing

String ialah tumpukan dari beberapa char / karakter yang mempunyai posisi index. String mempunyai sifat layaknya tuple.

```
var_string = 'workshop surabaya.py'

# Akses index
var_string[3]

# range index
var_string[0:7]

# jumlah karakter
print(len(var_string))

# immutable string
var_string[3] = 'g'
```

# Fungsi String

## Upper dan Lower case

merubah karakter string menjadi huruf besar dan huruf kecil

```
var_string = 'workshop surabaya.py'

# Upper case
var_string.upper()

# Lower case
var_string.lower()

# Capitalize case
var_string.capitalize()

# immutable string
var_string[3] = 'g'
```

# Fungsi String

String

## Count dan index

count berfungsi menghitung jumlah element dari nilai inputan,  
index berfungsi mencari posisi dari element

```
var_string = 'workshop surabaya.py'

# count case
var_string.count('o')

# index case
var_string.index('p')

var_string.index('sh')

# mencari karakter dengan in di if statement

if "surabaya" in var_string:
    print(var_string.index("surabaya"))
```

# Install dan Setup Virtual enviroment

## Install dan Setup Virtual enviroment

### Virtual enviroment

Virtual enviroment ialah sebuah alat yang digunakan untuk memisahkan dependencies library yang digunakan dalam masing-masing project.

#### virtualenv

library yang digunakan untuk membangun sebuah virtual enviroment.

```
# instalasi virtualenv  
pip install virtualenv  
  
# cara menggunakan  
  
# buat folder project  
mkdir workshop  
  
cd workshop  
  
virtualenv env
```

## Install dan Setup Virtual enviroment

### aktivasi enviroment

```
source env/bin/activate  
# tampilan cli  
(env)workshop$
```

Virtual enviroment

## Instalasi library menggunakan pip

```
# masih didalam virtual enviroment (env)workshop$  
pip install pyramid
```

## instalasi library web framework pyramid

```
# masih didalam virtual enviroment (env)workshop$  
pcreate -l  
  
# tampilan daftar pilihan scaffolding pyramid generator  
available scaffolds:  
alchemy: Pyramid SQLAlchemy project using url dispatch  
starter: Pyramid starter project  
zodb:    Pyramid ZODB project using traversal
```

## Install dan Setup Virtual enviroment

## Virtual enviroment

```
# masih didalam virtual enviroment (env)workshop$  
pcreate -s starter surabaya_py
```

tampilan hasil cli dan struktur folder pyramid

```
py/work_py/pytest.ini  
Copying setup.py_tmpl to /Volumes/RAID1/KERJAAN/Python_id/WorkshopPythonBasic/notes/webs/workshop_py  
/work_py/setup.py  
=====  
Tutorials: http://docs.pylonsproject.org/projects/pyramid_tutorials/en/latest/  
Documentation: http://docs.pylonsproject.org/projects/pyramid/en/latest/  
Twitter: https://twitter.com/trypyramid  
Mailing List: https://groups.google.com/forum/#!forum/pylons-discuss  
=====  
Welcome to Pyramid. Sorry for the convenience.  
=====  
[(env)Surya-Mac:workshop_py suryakencana$ ls -l  
total 0  
drwxr-xr-x  7 suryakencana  admin  238 Sep 29 23:22 env_Google Driv...  
drwxr-xr-x 11 suryakencana  admin  374 Sep 29 23:24 work_py  
[(env)Surya-Mac:workshop_py suryakencana$ cd work_py/  
[(env)Surya-Mac:work_py suryakencana$ ls -l  
total 56  
-rw-r--r--  1 suryakencana  admin   28 Sep 29 23:24 CHANGES.txt  
-rw-r--r--  1 suryakencana  admin  130 Sep 29 23:24 MANIFEST.in  
-rw-r--r--  1 suryakencana  admin  173 Sep 29 23:24 README.txt  
-rw-r--r--  1 suryakencana  admin 1108 Sep 29 23:24 development.ini  
-rw-r--r--  1 suryakencana  admin  930 Sep 29 23:24 production.ini  
-rw-r--r--  1 suryakencana  admin   49 Sep 29 23:24 pytest.ini  
-rw-r--r--  1 suryakencana  admin 1187 Sep 29 23:24 setup.py  
drwxr-xr-x  8 suryakencana  admin  272 Sep 29 23:24 work_py  
(env)Surya-Mac:work_py suryakencana$
```

## menjalankan web framework pyramid

```
python setup.py develop
```

```
pserve development.ini
```

# Break n Breath

**WHAT DO YOU MEAN**



**I WON'T GROW UP TO BE A TIGER?**

# Terima Kasih

The End