

#python

# WORKSHOP PYTHON DASAR

# WORKSHOP PYTHON DASAR

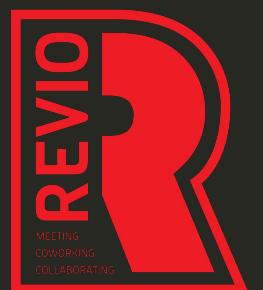


[Nanang Suryadi]

# 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 [in/nanangask] <http://intip.in/Rpbi>
- Github [suryakencana] <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 - 1

# 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!

# Syntax dan Output

### Menampilkan output pada terminal / cli

```
print('Hello, Surabaya.py')
```

## Menampilkan output pada terminal / cli

```
print('Hello, Surabaya.py')
```

## Multiline output

```
print('Hello, Surabaya.py\n I\'m pythonista\n Better Code with Python')
```

### Menampilkan sesuai data tipe

#### Data tipe String

```
print('Hello, %s' % 'World')
```

### Menampilkan sesuai data tipe

#### Data tipe String

```
print('Hello, %s' % 'World')
```

#### Data tipe integer

```
print('Sevendust %d' % 7)
```

### Menampilkan sesuai data tipe

#### Data tipe String

```
print('Hello, %s' % 'World')
```

#### Data tipe integer

```
print('Sevendust %d' % 7)
```

#### Data tipe float

```
print('Terbagi dalam %f' % 0.5)
```

## Whitespace

Dalam python untuk blok statement hanya di batasi oleh whitespace atau indent

## Break Line / Enter

```
Foo = 'foo'  
Bar = 'bar'
```

# Whitespace

## Syntax dan Output

Dalam python untuk blok statement hanya di batasi oleh whitespace atau indent

## Break Line / Enter

```
Foo = 'foo'  
Bar = 'bar'
```

## Indent

Dalam PEP8 disarankan untuk menggunakan 4 spasi / whitespaces

```
for bar in bars:  
    print(bar)  
    if 'wine' in bar:  
        print('%s ditemukan' % bar)
```

## Class Identifier

- Format CamelCase

```
class SurabayaPy():  
    ...
```

## Variable Identifier

- Format Word joined by \_ (underscore)

```
surabaya_dot_py = 'surabaya.py'
```

## Function Identifier

- Format Word joined by \_ (underscore)

```
def surabaya_dot_py():
    ...
```

## Constant Identifier

- Format All uppercase

```
SURABAYA_DOT_PY = 6543
```

## Comments

Comments adalah teks apapun yang diawali dengan tanda #, """" digunakan untuk memberikan catatan kepada pembaca kode

Comments adalah teks apapun yang diawali dengan tanda #, """" digunakan untuk memberikan catatan kepada pembaca kode

## Comments single-line

```
# komentar untuk catatan fungsi dari kode kita
```

### Comments multi-line

```
"""komentar untuk catatan fungsi dari kode kita  
agar mudah dibaca dan dimengerti baik fungsi dari argument atau  
hasil pengembalian nilai yg diinginkan.  
"""
```

### Apa itu?

digunakan untuk penyimpanan nilai / informasi

# Variable

## Syntax dan Output

Apa itu?

```
surabaya_py = 'Hello, Surabaya'  
print(surabaya_py)
```

Assignment

# Variable

Syntax dan Output

Apa itu?

```
variable = "..."  
camelCase = "..."  
with_underscores = "..."  
number1 = "..."
```

Assignment

Format  
Penulisan

### Apa itu?

### Assignment

### Format Penulisan

### Permasalahan

```
baris pertama = "Lorem ipsum dolor sit amet"  
2_baris = "consectetur adipiscing elit"  
baris3 = "sed do eiusmod tempor incididunt"  
print(baris pertama)  
print(2_baris)  
print(baris3)
```

### Apa itu?

### Assignment

### Format Penulisan

### Permasalahan

### Solusi

```
baris_pertama = "Lorem ipsum dolor sit amet"  
ke_2_baris = "consectetur adipiscing elit"  
baris3 = "sed do eiusmod tempor incididunt"  
print(baris_pertama)  
print(ke_2_baris)  
print(baris3)
```

### Kombinasi nilai variabel

Kombinasi nilai variabel bisa menggunakan operator +

```
q = "Hello"  
w = "Surabaya.py"  
print(q + w)
```

### Kombinasi nilai variabel

### Menggubah nilai variabel

```
nama = "Chris"  
alamat = "Pakuwon"  
print(nama + " tinggal di " + alamat)  
alamat = "Singapore"  
print(nama + " tinggal di " + alamat)
```

### Kombinasi nilai variabel

### Menggubah nilai variabel

### Latihan

Buat sebuah kalimat dengan menggunakan variabel ini:

```
subjek = "Code Bender"
```

```
tempat = "Revio"
```

```
aktivitas = "Coding"
```

### Multiple assigntment

#### Harmful

```
a = 'foo'  
s = 'foo'  
d = 'foo'  
w = 'foo'
```

#### Idiomatic

```
a = s = d = w = 'foo'
```

## Swap assignment

menghindari menggunakan temporary variabel

## Harmful

```
foo = 'foo'  
bar = 'bar'  
temp = foo  
foo = bar  
bar = temp
```

## Idiomatic

```
foo = 'foo'  
bar = 'bar'  
(foo, bar) = (bar, foo)
```

# Tipe Data

# String

Tipe Data

## Apa itu?

String adalah rangkaian karakter. String bisa berupa kata, kalimat atau sebuah paragraph.

# String

Tipe Data

Apa itu?

Single Quote

Single Quote

```
print('Halo, Surabaya.py')  
print('Hari ini hari jum\'at')
```

# String

Tipe Data

Apa itu?

## Single Quote

Single Quote

```
print('Halo, Surabaya.py')  
print('Hari ini hari jum\'at')
```

Double Quote

Perhatikan tanda quote ' harus di escape pada single quote. Selain itu tidak ada perbedaan antara single quote dan double quote, anda bebas untuk memilih

## Double Quote

```
print("Halo, Surabaya.py")  
print("Hari ini hari jum'at")
```

# String

Tipe Data

Apa itu?

## Triple Quote

Python mendukung multi-line string atau string dengan baris lebih dari satu. Anda dapat dengan bebas menuliskan single quote ' dan double quote " dalam string literal yang diapit dengan triple quote.

```
triple_string = """Ini adalah contoh multi-line string  
saya tambahkan single quote ' dan double  
quote ", tanpa perlu meng-escape \\ terlebih dahulu"""  
  
print(triple_string)
```

```
triple_string = '''Ini adalah contoh multi-line string  
saya tambahkan single quote ' dan double  
quote ", tanpa perlu meng-escape \\ terlebih dahulu'''  
  
print(triple_string)
```

# String

Tipe Data

## Apa itu?

Coba buat kotak 5x5 dengan memakai karakter '\*' didalam fungsi print()

## Single Quote

## Double Quote

## Triple Quote

## Latihan

# String

Tipe Data

Apa itu?

Coba buat kotak 5x5 dengan memakai karakter '\*' didalam fungsi print()

Single Quote

Solusi

```
print(' * * * * *\n * * * * *')
```

Double Quote

Triple Quote

Latihan

## Apa itu?

Integer adalah bilangan atau angka. untuk penulisan tidak memerlukan quote untuk operasi matematika.

# Integer

Tipe Data

Apa itu?

```
bilangan = 17
```

Contoh

# Integer

Tipe Data

Apa itu?

Hitung total jumlah buah yang ada dalam keranjang?

Contoh

```
apel = 13
```

```
mangga = 17
```

```
semangka = 46
```

Latihan

## Apa itu?

Kombinasi dua tipe berbeda, Integer dengan String

## Contoh

```
buah = 'mangga'  
mangga = 17  
print(buah + mangga)
```

## Latihan

## Kombinasi

# Integer

Tipe Data

## Apa itu?

Hasil tadi pasti akan menimbulkan error:

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

## Contoh

## Latihan

supaya bisa kombinasi string dengan integer diperlukan fungsi str()

## Kombinasi

```
bah = 'mangga'
mangga = 17
print(bah + str(mangga))
```

# Operator Matematika

# Penambahan dan Pengurangan

## Operator Matematika

Penambahan integer menggunakan operator + dan Pengurangan integer menggunakan operator -

```
a = 3  
b = 5  
  
print(3 + 5)  
  
print(a + b)  
  
print(a - b)
```

# Penambahan dan Pengurangan

## Operator Matematika

Penambahan integer menggunakan operator + dan Pengurangan integer menggunakan operator -

```
a = 3  
b = 5  
  
print(3 + 5)  
  
print(a + b)  
  
print(a - b)
```

## Test

Rumah budi berada tepat diantara rumah alex dan lita, jarak rumah alex dan lita 550m, dan jarak rumah budi dengan alex 150m. Hitung jarak rumah budi dengan lita?

# Penambahan dan Pengurangan

## Operator Matematika

Penambahan integer menggunakan operator + dan Pengurangan integer menggunakan operator -

```
a = 3  
b = 5  
  
print(3 + 5)  
  
print(a + b)  
  
print(a - b)
```

## Test

Rumah budi berada tepat diantara rumah alex dan lita, jarak rumah alex dan lita 550m, dan jarak rumah budi dengan alex 150m. Hitung jarak rumah budi dengan lita?

```
print(550 - 150)
```

# Pembagian dan Perkalian

## Operator Matematika

Perkalian integer menggunakan operator \* dan Pembagian integer menggunakan operator /

```
a = 10  
b = 2  
  
print(2 * 10)  
  
print(a * b)  
  
print(a / b)
```

# Pembagian dan Perkalian

## Operator Matematika

Perkalian integer menggunakan operator \* dan Pembagian integer menggunakan operator /

```
a = 10  
b = 2  
  
print(2 * 10)  
  
print(a * b)  
  
print(a / b)
```

## Soal Operator Matematika

Sabtu besok Alex berencana akan ke rumah orang tua lita dengan berkendara memakai sepeda motor, jarak tempat tinggal Alex ke kost lita 10 Km. Alex dan lita saat ini tinggal di surabaya, sedangkan orang tua lita sabtu besok berada di sidoarjo yang berjarak 19 Km dg tempat tinggal Alex. Berapa jam yang diperlukan alex untuk tiba di tempat tinggal ortu lita, jika kecepatan sepeda motor 60 Km/jam dan dalam keadaan macet kecepatan turun 30% dari normal yang kondisi macet ditempuh 2 Km?

### Solusi

```
alex_lita = 10
alex_sda = 19
macet = 2
kecepatan = 0.017 # 1.0 / 60 dalam jam
kecepatan_macet = 0.055 # 1.0 / (60 * (30.0/100))

print(((alex_lita + alex_sda) - macet) * kecepatan) + (macet * kecepatan_macet)
```

## Perpangkatan

```
x = 3  
  
# x squared  
print(x ** 2)  
  
# 5 to the power of 3  
print(5 ** 3)
```

## Perpangkatan

```
x = 3  
  
# x squared  
print(x ** 2)  
  
# 5 to the power of 3  
print(5 ** 3)
```

## Sisa Bagi

```
print(5 % 3)  
  
print(10 % 5)  
  
print(12 % 13)
```

### Urutan dalam operasi aritmatika

Dalam python urutan aritmatika di kenal dengan PEMDAS

- Parentheses
- Exponents
- Multiplication, Modulus, and Division
- Addition and Subtraction

```
print(2 + 2 * 2)  
print(2 - (6 + 8))  
print(32 / 4 ** 2)
```

# Operator Logika

Boolean merupakan tipe data yang mempunyai nilai pengembalian True dan False

Catatan: Dalam python, true dan false berupa huruf Capitalized True dan False

```
right = True
left = False

print(right)
print(left)
```

### Equality / persamaan

Tanda operator menggunakan dua sama dengan == yang kedua sisi bernilai Boolean

```
print(2 == 2)  
print("foo" == "foo")  
print(10 + 3 == 13)
```

### Inequality

Tanda operator menggunakan tanda seru dan sama dengan != yang mempunyai nilai tidak sama dengan

```
print(5 != 2)  
print(10 + 3 != 13)
```

### and

Operator and menghasilkan nilai kembalian True jika sisi kanan dan kiri bernilai True

```
happy = True
sad = False
coding = True
print( happy and coding )
print( sad and coding )
```

### and

Operator and menghasilkan nilai kembalian True jika sisi kanan dan kiri bernilai True

```
happy = True
sad = False
coding = True
print( happy and coding )
print( sad and coding )
```

Operator and bisa dipadukan dengan operasi matematika lain nya

```
x = 10
# Is x between 5 and 15?
print(x >= 5 and x <= 15)

# Is x not 2, 3, or 7?
print(x != 2 and x != 3 and x != 7)
```

### or

Operator or menghasilkan nilai kembalian True jika salah satu bernilai True

```
happy = True
sad = False
coding = True
angry = False

print( happy or coding )
print( sad or coding )
print( sad or angry )
```

### not

Operator not kembalikan dari nilai boolean True ke False atau dari False ke True

```
happy = True
angry = False

print( not happy )
print( not angry )
print( happy and not angry )
```

# List

## List Index

List adalah Element dalam python yang berisi beberapa element yang bisa berbeda jenis tipe data.

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
list_beda_tipe = ["satu", 2, "tiga", 3.0, "lima"]
print(list_sama_tipe)
print(list_beda_tipe)
```

## List Index

List adalah Element dalam python yang berisi beberapa element yang bisa berbeda jenis tipe data.

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
list_beda_tipe = ["satu", 2, "tiga", 3.0, "lima"]
print(list_sama_tipe)
print(list_beda_tipe)
```

## Pencarian element dalam list

kata kunci `in` bernilai True jika element ditemukan.

```
list_beda_tipe = ["satu", 2, "tiga", 3.0, "lima"]
print("satu" in list_beda_tipe)
print(2 in list_beda_tipe)
print("2" in list_beda_tipe and 3.0 in list_beda_tipe)
print("Satu" in list_beda_tipe and 2 in list_beda_tipe)
```

## Penggabungan list

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

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
list_beda_tipe = ["satu", 2, "tiga", 3.0, "lima"]
list_total = list_sama_tipe + list_beda_tipe
print(list_total)
```

## Index List

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

| Format penulisan list[ **index** ]

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[2])
print(list_sama_tipe[3])
print(list_sama_tipe[0])
```

## Index ranges

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

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[2:4])
print(list_sama_tipe[0:3])
print(list_sama_tipe[3:4])
```

## Index ranges

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

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[2:4])
print(list_sama_tipe[0:3])
print(list_sama_tipe[3:4])
```

Tanpa **start**

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[:3])
```

## Index ranges

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

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[2:4])
print(list_sama_tipe[0:3])
print(list_sama_tipe[3:4])
```

Tanpa **start**

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[:3])
```

Tanpa **end**

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]

print(list_sama_tipe[3:])
```

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

## 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+

# List Index

## Merubah Element

Karena sifat dari list adalah mutable, maka berdasarkan index dapat dirubah nilai dari element tersebut

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
list_sama_tipe[2] = 2
print(list_sama_tipe)
```

## Length

Untuk mengetahui jumlah element dalam sebuah list dapat menggunakan fungsi len

| Format penulisan **len( list )**

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
print(len(list_sama_tipe))
```

## Length

Untuk mengetahui jumlah element dalam sebuah list dapat menggunakan fungsi len

| Format penulisan **len( list )**

```
list_sama_tipe = ["satu", "dua", "tiga", "empat", "lima"]
print(len(list_sama_tipe))
```

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

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

# Control Flow

# if Statement

## Control Flow

Perintah percabangan yang menjalankan baris instruksi jika kondisi dalam keadaan True

```
x = 3

if x == 3:
    print("X bernilai tiga")

print("Variabel x")
```

# if Statement

## Control Flow

Perintah percabangan yang menjalankan baris instruksi jika kondisi dalam keadaan True

```
x = 3

if x == 3:
    print("X bernilai tiga")

print("Variabel x")
```

## Percabangan

```
pilih = input("garam atau gula")

if pilih == "garam":
    print("pilihan rasamu asin")

if pilih == "gula":
    print("pilihan rasamu manis")
```

## else statement

Perintah percabangan yang akan dijalakan jika kondisi tidak sesuai dengan if statement

```
x = 3

if x == 3:
    print("X bernilai tiga")
else:
    print("X bukan tiga")

print("Variabel x")
```

## else statement

Perintah percabangan yang akan dijalakan jika kondisi tidak sesuai dengan if statement

```
x = 3

if x == 3:
    print("X bernilai tiga")
else:
    print("X bukan tiga")

print("Variabel x")
```

```
password = input("Masukan kata kunci?")

if password == "surabayapy":
    print("Anda terdaftar")
else:
    print("Kata kunci salah")
```

## elif statements

Perintah percabangan yang mengecek jika if statement bernilai False

```
grade = int(input("What grade did you get?"))

if grade >= 90:
    print("You got an A.")

elif grade >= 80:
    print("You got a B.")

elif grade >= 70:
    print("You got a C.")

elif grade >= 60:
    print("You got a D.")

else:
    print("You got an F.")
```

### Tips: Menghindari satu baris setelah Colon (:)

#### Harmful

```
nama = 'Nanang'  
alamat = 'Keputih, SBY'  
  
if nama: print(nama)  
print(alamat)
```

#### Idiomatic

```
nama = 'Nanang'  
alamat = 'Keputih, SBY'  
  
if nama:  
    print(nama)  
print(alamat)
```

### Tips: Pengulangan satu variabel dalam 'IF' kondisi

#### Harmful

```
is_there_name = False  
  
nama = 'Chris'  
  
if nama == 'Chris' or nama == 'Nanang' or nama == 'Robi':  
    is_there_name = True  
  
print(is_there_name)
```

#### Idiomatic

```
nama = 'Chris'  
  
is_there_name = nama in ('Chris', 'Nanang', 'Robi')  
  
print(is_there_name)
```

## Tips: Ternary Operator

### Harmful

```
foo = True
value = 0

if foo:
    value = 1
print(value)
```

### Idiomatic

```
foo = True

value = 1 if foo else 0
print(value)
```

# while Statement

while adalah pengulangan blok intruksi secara terus menerus jika kondisi True.

```
n = 1  
while n < 10:  
    print(n)  
    n +=1
```

Coba tampilkan "Hello, Surabaya.py" sebanyak 13 kali

# while Statement

## Control Flow

while adalah pengulangan blok intruksi secara terus menerus jika kondisi True.

```
n = 1

while n < 10:
    print(n)
    n +=1
```

Coba tampilkan "Hello, Surabaya.py" sebanyak 13 kali

```
n = 1

while n < 13:
    print("Hello, Surabaya.py")
    n +=1
```

## Latihan

Cari bilangan faktorial dari n, jika ditemukan tampilkan bilangan tersebut

## Latihan

Cari bilangan faktorial dari n, jika ditemukan tampilkan bilangan tersebut

```
n = input("Bilangan faktorial")  
faktor = 1  
while faktor < n:  
    if n % faktor == 0:  
        print(faktor)  
    faktor += 1
```

Saat melakukan perulangan list menggunakan while, ada cara yang lebih efektif

### Harmful

```
daftar_pengurus = ['Chris', 'Nanang', 'Robi']
index = 0
while index < len(daftar_pengurus):
    print(daftar_pengurus[index])
    index += 1
```

### Idiomatic

```
daftar_pengurus = ['Chris', 'Nanang', 'Robi']
for nama in daftar_pengurus:
    print(nama)
```

# for Statement

## Control Flow

Pengunaan fungsi built-in enumerate untuk mendapatkan index dalam fungsi pengulangan

Contoh dalam pengunaan for loop di pemograman java:

```
for(int i = 0; i < 20; i++) {  
    System.out.print("value of x : " + x );  
    System.out.print("\n");  
}
```

# for Statement

## Control Flow

Pengunaan fungsi built-in enumerate untuk mendapatkan index dalam fungsi pengulangan

Contoh dalam pengunaan for loop di pemograman java:

```
for(int i = 0; i < 20; i++) {  
    System.out.print("value of x : " + x );  
    System.out.print("\n");  
}
```

Dalam python fungsi \*range termasuk built-in

```
for n in range(100):  
    print(n)
```

Format fungsi Built-in range

| Format range(**start**, end, **increment**)

Dalam python fungsi enumerate termasuk fungsi built-in

### Harmful

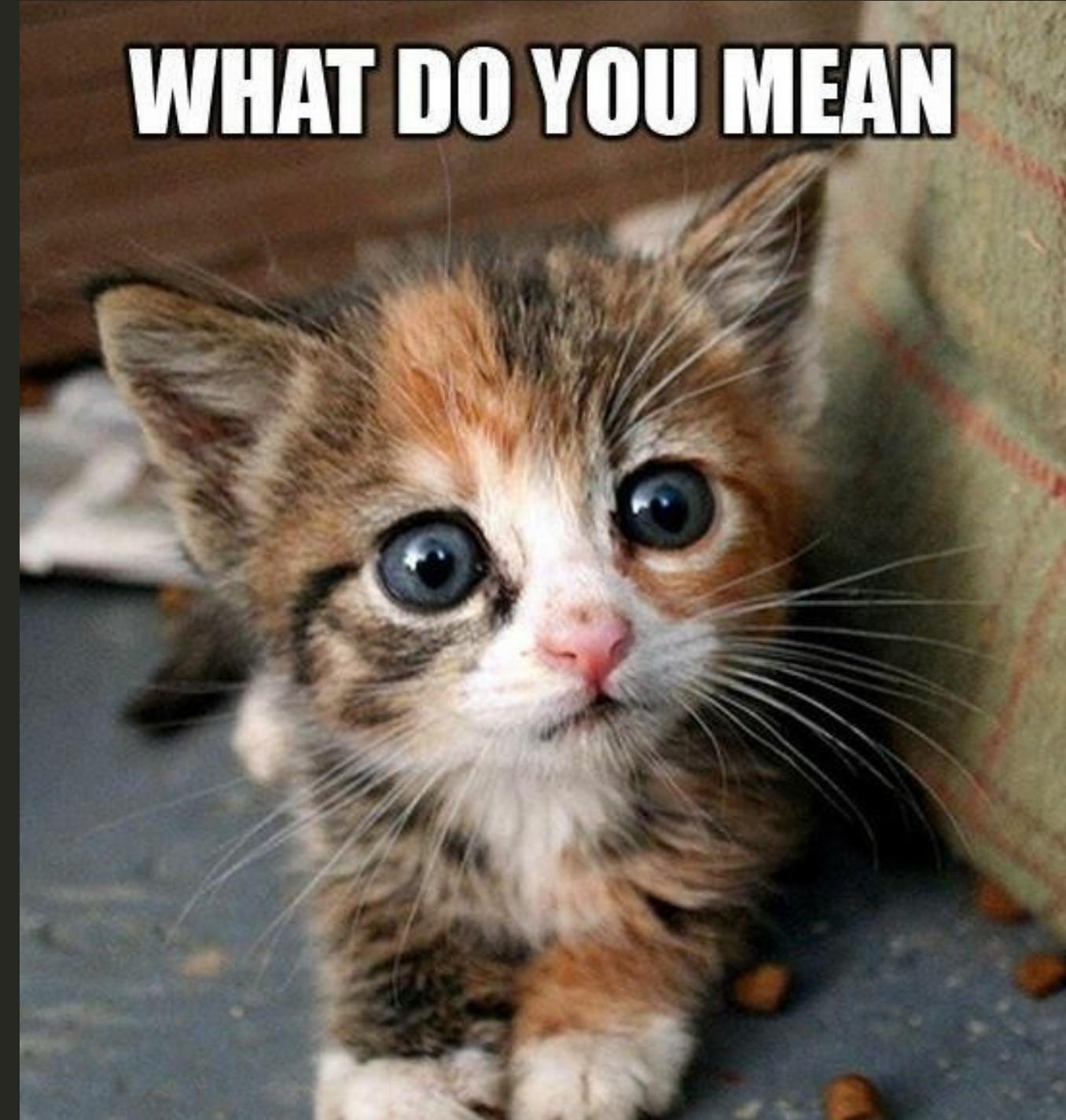
```
daftar_pengurus = ['Chris', 'Nanang', 'Robi']
index = 0
for nama in daftar_pengurus:
    print('{} {}'.format(index, nama))
    index += 1
```

### Idiomatic

```
daftar_pengurus = ['Chris', 'Nanang', 'Robi']
for index, nama in daftar_pengurus:
    print('{} {}'.format(index, nama))
```

# Break n Breath

**WHAT DO YOU MEAN**



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

# Terima Kasih

to be continue...