KOMINFO digitalent

PRO Professional Academy

Digital Talent Scholarship 2022

Python - 1

Lead a sprint through the Machine Learning with Tensorflow Track

Agenda

- Pengenalan Python
- Demo Coding
- Next Step

Are your students ML-ready?

KOMINFO digitalent

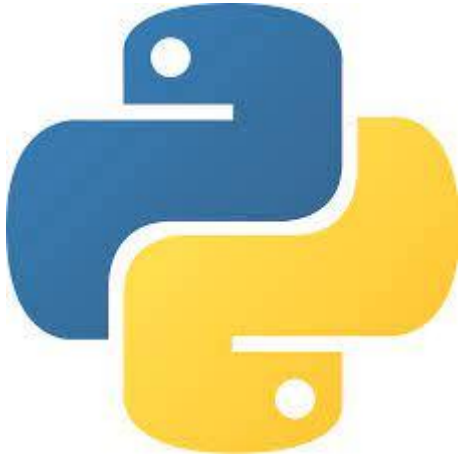
Intro to Python

Python

1. Expression, variables, function, conditionals
2. String, List, Dictionary
3. Loop, While

Python

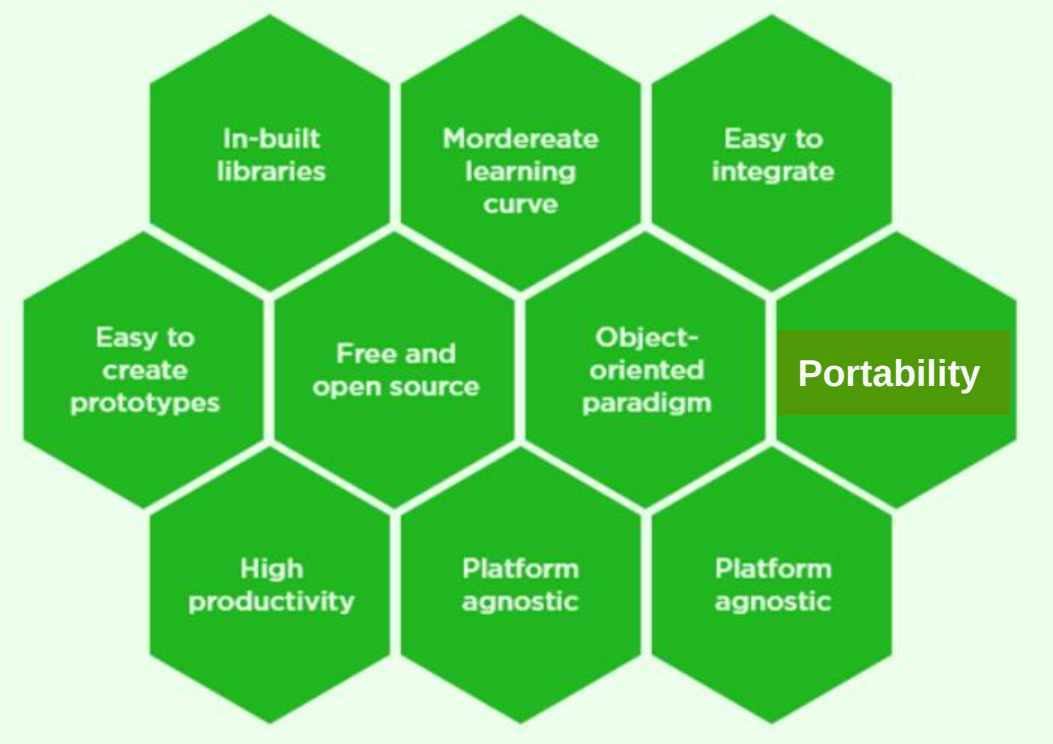
Python adalah bahasa pemrograman yang sederhana dan mudah dimengerti. Python bertujuan untuk menghasilkan kode yang lebih jelas dan lebih logis untuk berbagai keperluan. Dalam perkembangannya, Python tak hanya digunakan dalam dunia teknologi, namun juga dalam hal lain khususnya analisis.



Kegunaan Python

1. Pengembangan website
2. Pengembangan IoT
3. Penambangan Data
4. Pengembangan Machine Learning
5. Pengembangan Game
6. Python untuk Fintech

10 reason why Python for Machine Learning



Kelebihan dan Kekurangan Python

Kelebihan :

- Sintaks yang mudah dipahami
- Bahasa yang paling banyak dipilih untuk IT
- Banyak standard library yang dapat dipakai
- Dapat diaplikasikan pada berbagai sistem operasi

Kekurangan :

- Cukup lambat dijalankan terutama untuk pengembangan platform Android dan iOS.

Mengapa python disebut sebagai high level language ?

Tipe Data

Boolean : Menyatakan benar **True** yang bernilai **1**, atau salah **False** yang bernilai 0

String : Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')

Integer : Menyatakan bilangan bulat

Float : Menyatakan bilangan yang mempunyai koma

Hexadecimal : Menyatakan bilangan dalam format heksa (bilangan berbasis 16)

Complex : Menyatakan pasangan angka real dan imajiner

List : Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah

Tuple : Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah

Dictionary : Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

Dictionary

Python

```
a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
for key in a_dict:
    print(key)
```

Python

```
a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
d_items = a_dict.items()
print(d_items)
```

Dictionary

Python >>>

```
>>> a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
>>> keys = a_dict.keys()
>>> print(keys)
dict_keys(['color', 'fruit', 'pet'])
```

Python >>>

```
>>> a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
>>> the_values = a_dict.values()
>>> the_values
dict_values(['blue', 'apple', 'dog'])
```

Dictionary

```
a = {'a':'A','b':'B'}
a['c'] = 'C'
if 'c' in a.keys():
    print(a['c'])
```

```
a = {'a':'A','b':'B'}
del a['b']
print(a)
```

```
a = {'a' : 'A', 'b' : 'B'}
a['b'] = 'C'
print(a)
```

Float agak aneh

Python >>>

```
>>> 1.1 + 2.2 == 3.3
False
```

Python >>>

```
>>> tolerance = 0.00001
>>> abs((1.1 + 2.2) - 3.3) < tolerance
True
```

KOMINFO digitalent

Pop Quiz

Python

```python
x = -100
from math import sqrt
x > 0 and sqrt(x)
```

KOMINFO digitalent

Set

```
s = set('abc')
s = {'a', 'b', 'c'}
s = set(['a', 'b', 'c'])
print(s)
```

```
s.add('som')
print(s)
```

```
s = {'foo', 'bar', 'baz', 'qux'}
s.difference_update({'bar'})
s &= {'foo', 'baz', 'qux'}
s.discard('bar')
s -= {'bar'}
s.remove('bar') # Will raise exception if there is no bar
print(s)
```

%-formatting

Python >>>

```
>>> name = "Eric"
>>> age = 74
>>> "Hello, %s. You are %s." % (name, age)
'Hello Eric. You are 74.'
```

```
>>> first_name = "Eric"
>>> last_name = "Idle"
>>> age = 74
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> "Hello, %s %s. You are %s. You are a %s. You were a member of %s." % (first_name,
last_name, age, profession, affiliation)
```

str.format()

Python >>>

```
>>> "Hello, {}. You are {}.".format(name, age)
'Hello, Eric. You are 74.'
```

Python 0, 1 >>>

```
>>> "Hello, {1}. You are {0}.".format(age, name)
'Hello, Eric. You are 74.'
```

Python >>>

```
>>> person = {'name': 'Eric', 'age': 74}
>>> "Hello, {name}. You are {age}.".format(name=person['name'], age=person['age'])
'Hello, Eric. You are 74.'
```

str.format()

Python >>>

```
>>> person = {'name': 'Eric', 'age': 74}
>>> "Hello, {name}. You are {age}.".format(**person)
'Hello, Eric. You are 74.'
```

Python >>>

```
>>> first_name = "Eric"
>>> last_name = "Idle"
>>> age = 74
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> print(("Hello, {first_name} {last_name}. You are {age}. " +
>>>        "You are a {profession}. You were a member of {affiliation}.") \
>>>        .format(first_name=first_name, last_name=last_name, age=age, \
>>>                profession=profession, affiliation=affiliation))
'Hello, Eric Idle. You are 74. You are a comedian. You were a member of Monty Python.
```

f-Strings

Python >>>

```
>>> name = "Eric"
>>> age = 74
>>> f"Hello, {name}. You are {age}."
'Hello, Eric. You are 74.'
```

f-Strings

This will work:

Python >>>

```
>>> comedian = {'name': 'Eric Idle', 'age': 74}
>>> f"The comedian is {comedian['name']}, aged {comedian['age']}."
The comedian is Eric Idle, aged 74.
```

But this will be a hot mess with a syntax error:

Python >>>

```
>>> comedian = {'name': 'Eric Idle', 'age': 74}
>>> f'The comedian is {comedian['name']}, aged {comedian['age']}.'
  File "<stdin>", line 1
    f'The comedian is {comedian['name']}, aged {comedian['age']}.'
                                  ^
SyntaxError: invalid syntax
```

f-Strings

In order to make a brace appear in your string, you must use double braces:

Python >>>

```
>>> f"{{70 + 4}}"
'{70 + 4}'
```

Note that using triple braces will result in there being only single braces in your string:

Python >>>

```
>>> f"{{{70 + 4}}}"
'{74}'
```

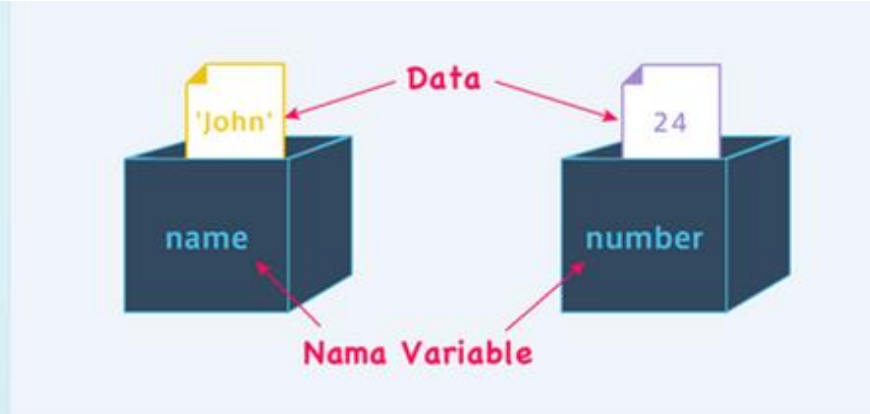
Expression

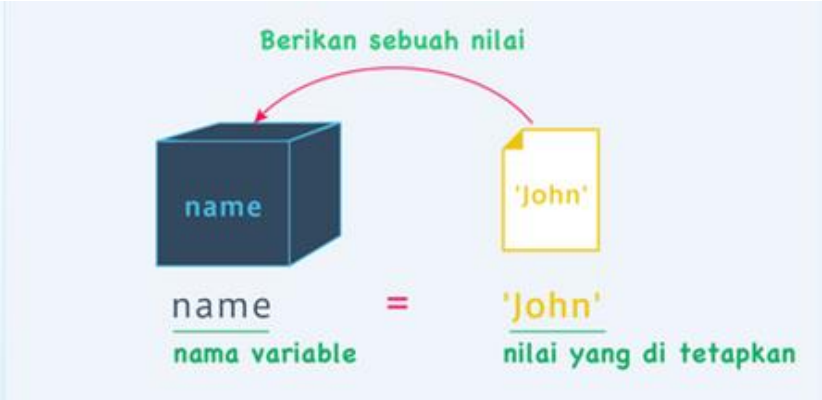
Expression adalah entitas sintaks pada bahasa pemrograman yang menghasilkan sebuah nilai. Biasanya ekspresi terdiri dari konstanta, variabel, fungsi, dan operator.

- Statement = melakukan tugas tertentu
- Ekspresi = menghasilkan nilai.

```
x + 1; // ekspresi menjumlahakan x dengan 1
3 * 3; // ekspresi 3 * 3
sqrt(25); // eskpresi untuk menghitung akar kuadrat
```

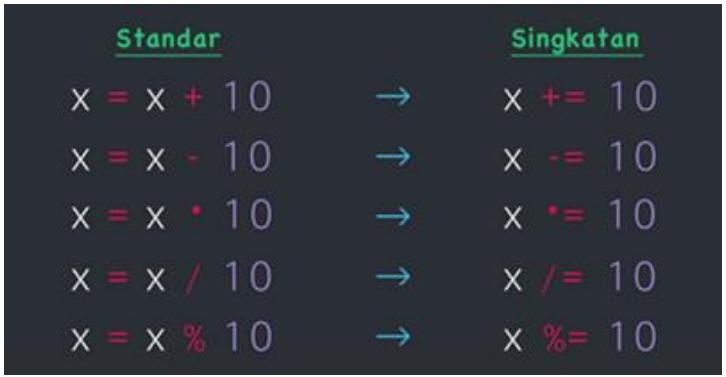
Konsep Variable





Name = ‘John’
Number = 24

Assignment Operators



Functions

fungsi (def) adalah kumpulan perintah atau baris kode yang dikelompokkan menjadi satu kesatuan untuk kemudian bisa dipanggil atau digunakan berkali-kali.

Sebuah fungsi bisa menerima parameter, bisa mengembalikan suatu nilai, dan bisa dipanggil berkali-kali secara independen.

Example

Syntax Function

```
def <nama_fungsi>(parameters):
        Statements
```

Contoh

```
def perkenalan (nama, asal):
        print(f"Perkenalkan saya {nama} dari {asal}")

perkenalan("Alvin", "Jawa Timur")
```

If, else , elif Statement

- Statement if : menulis if diikuti conditional expression dan tanda titik dua (:).
- Statement else : dapat menambahkan code yang ingin dijalankan jika kondisi statement if adalah False.
- Statement elif : untuk menambah kondisi alternatif pada control flow. Dapat ditambahkan sebanyak yang dibutuhkan.

```
a = 3
If (a>10) : #true
 return true
Elif (a>5) :
 Return true
Else #false
 return false
```

Conditional Expression

==	True jika nilai di kiri dan kanan adalah setara
!=	True jika TIDAK setara

<	True jika nilai di kanan lebih besar dari nilai di kiri
<=	True jika nilai di kanan lebih besar dari atau sama dengan nilai di kiri
>	True jika nilai di kanan lebih kecil dari nilai di kiri
>=	True jika nilai di kanan lebih kecil dari atau sama dengan nilai di kiri

Example

```
if i ==1:
        print(1)
elif i == 2:
        print(2)
else :
        print("Not 1 and 2")
i = 5  # not 1 and 2
i =1 # 1
i = 2 #2
```

Quick Recap

String

String adalah jenis yang paling populer di bahasa pemrograman. Kita bisa membuatnya hanya dengan melampirkan karakter dalam tanda kutip. Python memperlakukan tanda kutip tunggal sama dengan tanda kutip ganda. Membuat string semudah memberi nilai pada sebuah variabel.

```
print("Hello World")

“Hallo saya stevani”
‘Hallo’
```

String

Python triple quotes digunakan dengan membiarkan string untuk ditulis dalam beberapa baris, termasuk kata kerja NEWLINEs, TABs, dan karakter khusus lainnya. Sintaks untuk triple quotes terdiri dari tiga tanda kutip tunggal atau ganda ditulis berturut-turut :

```
kutipantiga = """this is a long string that is made up of
several lines and non-printable characters such as TAB (
\t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
```

List

- Tipe data list memungkinkan Anda untuk mengelola sekelompok data sekaligus. Anda dapat membuat list sebagai berikut: [element1, element2, ...]. Setiap nilai di dalam list disebut element.

- Dengan menggunakan list, dapat mengelola banyak string dan integer dalam satu grup.

- Setiap element list dinomori 0, 1, 2, ...
  Ini disebut nomor indeks. Nomor indeks dimulai dari 0.

- LIst : Dapat menerima beberapa tipe data dan dapat diubah

```
List : [‘stevani’, ’akira’, ’jesslyn’ ]    => List[0] # stevani
            0          1          2
Angka : [1,2,3,4]  => Angka[0] # 1
```

List Function

Append : Menambahkan objek ke list
Count : Jumlah pengembalian berapa kali terjadi dalam list
Extend : Tambahkan isi ke list
Index : Mengembalikan indeks terendah dalam list yang muncul
Insert : Sisipkan objek ke dalam list
Pop : Menghapus dan mengembalikan objek
Remove : Remove object from list
Reverse : Membalik list objek di tempat
Sort : Urutkan objek list
Len : Memberikan total panjang list

List

```
friends = ["A", "B", "C"]

print(friends)                  Output : ["A", "B", "C"]

print(friends[0])               Output : A

print(friends[-1])              Output : C

print(friends[1:])              Output : ["B", "C"]

print(friends[0:2])             Output : ["A", "B"]
```

List Functions

X = ["A", "B", "C"]

Y = [1, 2, 3]

X.**extend**(Y)	["A", "B", "C", 1, 2, 3]
X.**append**("D")	["A", "B", "C", "D"]
X.**insert**(1, "D")	["A", "D", "B", "C"]
X.**remove**("B")	["A", "C"]
X.**clear**()	[]

List Functions

X = ["A", "B", "C"]

Y = [1, 2, 3]

X.**pop()**	["A", "B"]
X.**index(**"B"**)**	1
X.**count(**"B"**)**	1
X.**sort()**	["A", "B", "C"]
X.**reverse()**	["C", "B", "A"]

X2 = X.**copy()**

Dictionary

Dictionary adalah tipe data pada python yang berfungsi untuk menyimpan kumpulan data/nilai dengan pendekatan “key-value”.

Dictionary sendiri memiliki dua buah komponen inti:

1. Yang pertama adalah key, ia merupakan nama atribut suatu item pada dictionary.
2. Yang kedua adalah value, ia adalah nilai yang disimpan pada suatu atribut.

Example

```
days_in_a_week = {
                        "Mon": "Monday",
                        "Tue": "Tuesday",
                        "Wed": "Wednesday",
                        "Thu": "Thursday",
                        "Fri": "Friday",
                        "Sat": "Saturday",
                        "Sun": "Sunday"
                }
days_in_a_week["Mon"]
```

Output ?

Quick Recap

Perulangan (Loops)

Perulangan atau juga sering dikenal dengan looping merupakan pernyataan atau instruksi yang diberikan kepada komputer agar ia mau melakukan sesuatu entah itu memproses data, menampilkan data, atau yang lainnya secara berulang.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

- While Loop
- For Loop
- Nested Loop

For Loops

Pengulangan for pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string.

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]
for x in angka:
        print(x)
```

While Loops

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True.

Example :

```
count = 0
while (count < 9):
                print ("The count is: ", count)
                count += 1

print ("Good bye!")
```

```
# the count is 0
The count is 1
The count is 2
....
The count is 8
Good bye!
```

Nested Loops

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain.

```
#Contoh penggunaan Nested Loop

i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print(i, " is prime")
   i = i + 1

print("Good bye!")
```

Google Colab Session

Demo Colab

Q & A