MyEduSolve

Kampus Merdeka
INDONESIA JAYA

Numpy Group Summary #4

The Teams

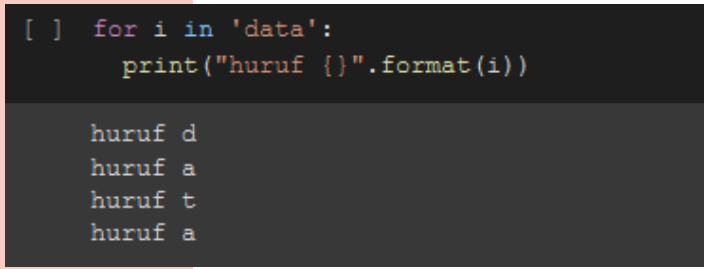
Atika Rahmah
Daril Hana Salsabila
Donny P. Tambunan
Muhammad Atthariq Ramadhan
Muhammad Iqbal Rustan

Table Of Contents

Table Of Contents

01

Looping, For, While, Break and Continue



Looping

There are two kinds of loops, namely for loops and while loops for. For is a function that can iterate over each type of variable in the form of a collection or sequence.

For Loop in List

```
color = ['merah','kuning','hijau']

for i in color:
  print("warna {}".format(i))
```

```
warna merah
warna kuning
warna hijau
```

For Loop in Range

```
[ ]  for i in range(0,3):
       print(i)
```

```
0
1
2
```

For Multilevel

```
for a in range (4):
    for b in range (4):
        if a == 0 and b <= 3:
            print("*",end = " ")
        elif a == 1 and b <= 1:
            print("*", end = " ")
        elif a == 2 and b == 1:
            print("*",end = " ")
        elif a == 3 and b == 0:
            print("*",end = " ")
    print()
```

```
* * * *
* *
*
*
```

While

While in Python is used to execute a statement as long as a given condition is met (True). When the condition becomes false, the program will continue to the line after the statement block.

```
]  angka = int(input())
   while (angka < 6):
       print("bilangan {}".format(angka))
       angka = angka + 6
```

```
2
bilangan 2
```

Break

```
for i in "Data Science":
    if i == " ":
        break
    print(i)
```

```
D
a
t
a
```

```
[ ]  for task in "Hari Senin_malam":
        if task == "_":
            break
        print(task)
```

```
H
a
r
i

S
e
n
i
n
```

In the break will stop the loop and then exit, followed by executing the statement after the loop block.

In the multilevel loop, the break will stop the loop according to the level or loop where it belongs

Continue

The continue statement will continue to the iteration when stopped and continue to the next iteration

```
[ ]  for i in "Data":
         if i == "a":
             continue
         print(i)
```

```
D
t
```

```
[ ]  for task in "Jam belajar malam":
         if task == "a":
             continue
         if task == "e":
             continue
         print(task)
```

```
J
m

b
l
j
r

m
l
m
```

02

Else After For,Else After While, Pass, List Comprehension

Else After For

The "else" after the "for" function is preferred in search loops - to provide a program exit when the search is not found

```
for i in range(12,20):
    if i % 4 == 0 :
        print("{} is number that divisible by 4".format(i))
        continue
    else:
        print("{} is not number that divisible by 4".format(i))
```

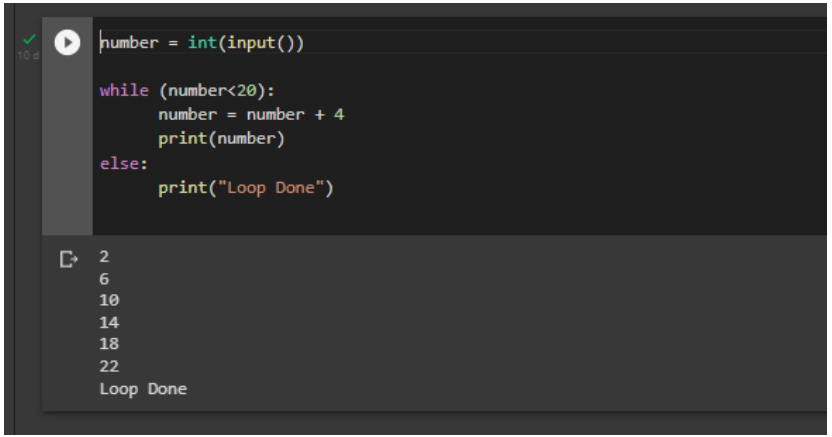
```
12 is number that divisible by 4
13 is not number that divisible by 4
14 is not number that divisible by 4
15 is not number that divisible by 4
16 is number that divisible by 4
17 is not number that divisible by 4
18 is not number that divisible by 4
19 is not number that divisible by 4
```

```
[21] for i in range(12,20):
        if i % 4 == 0 :
            print("{} is number that divisible by 4".format(i))
            break
        else:
            print("{} is not number that divisible by 4".format(i))
```

```
12 is number that divisible by 4
```

Else After While

The "else" after the “while" function is preferred in search loops - to provide a program exit when the search is not found



Pass

Pass statement is a Null (empty) operation, nothing happens when it is called. Pass is used if you want a statement or block of statements, but do nothing or continue execution in order.

This control is widely used when not implementing (or preparing a place for implementation), as well as letting the program run when for example we experience a failure or exception

```
import sys

mes = ""

while (mes != "DST"):
  try:
    mes = (input("Peroleh : "))
    print('dapat angka {}'.format(int(mes)))
  except:
    if mes == 'DST':
      pass
    else:
      print("dapat error {}".format(sys.exc_info()[0]))
```

```
Peroleh : 1
dapat angka 1
Peroleh : 4
dapat angka 4
Peroleh : 7
dapat angka 7
Peroleh : DST
```

List Comprehension

Somethimes we need to create a new list from the previous list operation. List Comprehension is a way to generate new lists based on pre-existing lists or iterables.

```
DST = [3,4,9]

[37] square = []

for i in DST:
    square.append(i**2)
print(square)

[9, 16, 81]
```

Writing in another way :

```
new_list = [expression for_loop_one_or_more conditions]
```

03

List Comprehension,Syntax Errors and Exceptions

List Comprehensions

```
abc = ['a','b','c']
cde = ['c','d','e']

duplikat = [i for i in abc for j in cde if i == j]

print(duplikat)
```

```
['c']
```

```
[ ]  data = ["data", "science"]

     upper = [_.upper() for _ in data]

     print(upper)
```

```
['DATA', 'SCIENCE']
```

List Comprehension

Create a list with inline loop and if

```
[ ]  angka = range(1,8,2)
     kuadrat_kubik = [[_**2, _**3] for _ in angka]
     print(kuadrat_kubik)

     [[1, 1], [9, 27], [25, 125], [49, 343]]
```

```
[ ]  number_list = [ x for x in range(5) if x % 1 == 0]
     number_list_1 = [ x for x in range(5) if x // 1 == 0]
     print(number_list +number_list_1)

     [0, 1, 2, 3, 4, 0]
```

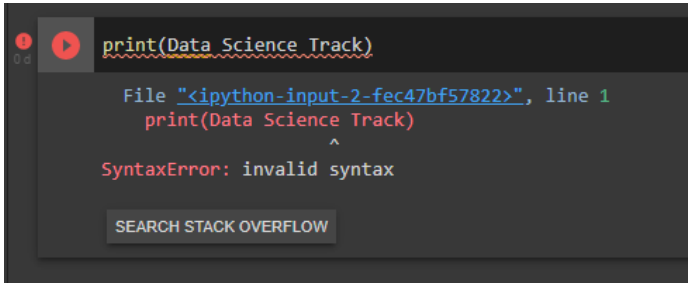
Error Handling (Error and Exception Handling)

Syntax Errors

Two types of errors based on their occurrence:

1. Syntax Errors
2. Exceptions

Syntax errors occur when Python can't understand what it's being told.



Exception Error

This type of error occurs whenever syntactically correct Python code results in an error. The last line of the message indicated what type of exception error you ran into.

instead of showing the message exception error, Python details what type of exception error was encountered

Some type of exception error is ZeroDivisionError, KeyError, FileNotFoundError, and others

ZeroDivisionError

```
[1] a=0

divide=1/a

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-d5104a6ea908> in <module>()
----> 1 divide=1/a

ZeroDivisionError: division by zero

SEARCH STACK OVERFLOW
```

KeyError

```
[ ] d = {'ratarata': '10.0'}

[ ] print('rata-rata : {}'.format(d['rata_rata']))

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-19-f51b98098c56> in <module>()
----> 1 print('rata-rata : {}'.format(d['rata_rata']))

KeyError: 'rata_rata'

SEARCH STACK OVERFLOW
```

FileNotFoundError

```
open('todoroki_shouto.py')

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-6-94d2cf242df8> in <module>()
----> 1 open('todoroki_shouto.py')

FileNotFoundError: [Errno 2] No such file or directory: 'todoroki_shouto.py'

SEARCH STACK OVERFLOW
```

04

Exception Handling

Exception Handling

The exception handling process uses a try statement which pairs quickly.

```
a = 0

try:
  bagi = 1/a
  print(bagi)
except ZeroDivisionError:
  print("tidak bisa membagi angka dengan nol")
```

Exception Handling

ZeroDivisionError is an exception that occurs when program execution results in a mathematical calculation of division by zero.

Exception Handling for ZeroDivisionErrror is done so that the application no longer exits execution due to an error, but is replaced by printing a message to the layer.

```
try:
  bagi = 0/0
  print(bagi)
except ZeroDivisionError:
  print("tidak bisa dibagi dengan nol")
```

```
tidak bisa dibagi dengan nol
```

Unsupported Cell Type

```
[ ]  d = {'ratarata': '10.0'}

     try:
       print('rata-rata: {}'.format(d['rata_rata']))
     except KeyError:
       print('kunci tidak ditemukan di dictionary')
     except ValueError:
       print('nilai tidak sesuai')
```

Error Using Tuple

```
d = {'ratarata': '10.0'}

try:
  print('pembulatan rata-rata'.format(int(d['ratarata'])))

except (ValueError, TypeError) as e:
  print("penangan error {}".format(e))
```

```
penangan error invalid literal for int() with base 10: '10.0'
```

05
Function

Complete Form of “Try” Statement

```
try:

   pass #gantikan
```

Statements that may occur exceptions

```
except:

    pass #gantikan
```

Statements are operated if an exception occurs

```
else:

    pass #gantikan
```

The statement is operated if no exception occurs

```
finally:

    pass #gantikan
```

The statement is operated after all the statements above occur

Function

Functions are defined with the keyword "def" followed by the function name and parameters in brackets ()
The function stops when there is a return [expression] statement that returns [expression] to the caller

```
[ ]  def pangkat(x):
         kuadrat = x**2 #fungsi pangkat dua
         print (kuadrat)

[ ]  pangkat(3)

     9

[ ]  def pangkat(x):
         kuadrat = x**2 #fungsi pangkat dua
         return kuadrat

[ ]  pangkat(3)

     9
```

Return

The return [expressions] statement will make program execution exit the current function, and return a specified value.

```
def tambah(x,y):
  plus = x + y #fungsi penjumlahan
  print("Hasil penjumlahan adalah {}".format(plus))
  return plus
```

```
[ ] tambah(2,3)
```

```
Hasil penjumlahan adalah 5
5
```

```
[ ] output = tambah(2,3)
    print("Hasil dari return adalah {}".format(output))
```

```
Hasil penjumlahan adalah 5
Hasil dari return adalah 5
```

Pass by reference vs value

All parameters (arguments) in Python are "Passed by reference" meaning that when changing a variable, the data that references it will also change unless an assignment operation changes the reference parameter.

```
daftar_list = [1, 3, 5]
ubah(daftar_list)
print("Nilai di luar fungsi adalah {}".format(daftar_list))
```

```
Nilai di dalam fungsi adalah [2, 4, 6]
Nilai di luar fungsi adalah [1, 3, 5]
```

Function arguments and parameters

- Parameter is the definition of the input the function accepts
- Arguments are what we put in when we call the function

```
[ ]  def jadwal (tgl, bln, thn):
         berangkat = "Jadwal keberangkatan adalah tgl {} bln {} thn {}".format(tgl,bln,thn)
         return berangkat

[ ]  jadwal(tgl=24, bln='feb',thn=2022)

     'Jadwal keberangkatan adalah tgl 24 bln feb thn 2022'
```

Thank you!