<PKU 통계스터디 두번째 수업>

파이썬 기본 어법 2

reference: 爱上 python （人民邮电出版社）

2018.03.11 SUNDAY

발표자 - 이준호

The “for” & backslash

for & for : iter num

```
k1 = 0
kk1 = 0
for k in [1,2,3,4,5]:
    k1 = k1+1
    for kk in [1,2,3,4,5]:
        kk1 = kk1+1
print(k1, kk1)
```

==========

The number of k1 & kk1 would show the iterated number

for & for : calc

```
for k in [1,2,3,4,5]:
    for kk in [1,2,3,4,5]:
        print(k+kk)    #iter num?
print(“done”)
```

==========

Result?

The backslash “\”

```
Test = 2+\
3

print(Test)
```

==========

Result?

Additional data format “set” :: {element}

Basic usage

S = set() #define a set

S.add(1) # add a element

S.add(10) # add a element

X = len(S) #length of the Set

Y = 1 in S # check whether 1 #in side of the Set “S”

Z = 2 in S # same with upper

No element repetition

Item = [1,2,3,4,1,2]

Num_list = len(Item)

Item_set = set(Item)

Num_set = len(Item_set)

print(Item_set)

==========

In the format of “set”, no repetition of element allowed.

**

But the most frequently used for our study would be “dictionary” and “list”

So now, we move to list usage

The “list”

The “sort” function

```
X = [12,521,52,25,24,124,32,42]
print(X)
Y= sorted(X)
print(Y)
print(X)
X.sort()
print(X)
```

==========

The fuction “sorted” and “sort”

A bit more..

Sorting the list in the order of absolute values:

```
X = [-1, 4, 5.34, -10]
Y = sorted(X, key=abs, reverse = False)
Z = sorted(X, key=abs, reverse = True)
```

==========

abs → absolute

The “append”

```
X = [-1, 2, -3.5, 12, -44.32, 8]
X.sort(key=abs, reverse = True)
X.append(10)
X.append(-10)
print(X)
X.sort(key=abs, reverse = True)
print(X)
```

==========

List comprehension

Basic usage 1

```
#ENum = list()
#for x in range(5):
#    if x%2 == 0:
#        ENum.append(x)

ENum = [x for x in range(5) if x%2 == 0]

Enum_Square = [x*x for x in ENum]
```

==========

Result?

Basic usage 2

```
Square_dic = {x : x*x for x in range(5)}   #define a dictionary

Square_set = {x*x for x in [-2,2]}
# define a set

Zeros = [0 for _ in [1,2,3,4,5]]
```

==========

Result?

Basic usage 3

```
Pairs1 = [(x,y) for x in range(10) for y in range(10)]
# 100 elements

Pairs2 = [[x,y] for x in range(5) for y in range(5)]

Pairs3 = [[x,y] for x in range(5) for y in range(x+1,5)]
```

==========

Result?

“Function” in python

“function” without return value

```
def Add(x,y):
    z=x+y
    print(z)

Add(3,2)
```

==========

1. “def” is used to define a function
2. Notice of Lines!
3. Invoke the function outside of definition of function

“function” with return value

```
K =0
def Add(a,b):
    z = a+b
    return z

K = Add(3,2)
```

==========

1. A function could have return value, just like math function
2. The functions would be terminated, once the process meets a “return”

Local & Global. General talks

The “z” only “survives” inside of “def”. (Local), while the “K” survives all the way from beginning till to end of the process. (Global)

A function could either have a return value or not. But recommend to have a return value to continue our working.

Be notice of inputs and outputs of a function.

import a function

import a function in general

import NameOfPythonFile

NameOfPythonFile.NameOfFunction

import a function in practice

1. On the repo "TESTs", under your own folder, there is a python file named "test_add.py"
2. If you "vi" and look into, there is a function defined with namd "Add". (we will import this fuction)
3. Get out of the python file by ":q", make a new pyton file, say "import_test.py"
4. Inside of the python file, which you just created, do : import test_add (on the first line)

import a function in practice (continue..)

5. Remember there is a function called "Add" inside of the "test_add.py", we could invoke the function by "K = test_add.Add(2,3)"

==========

So the code inside of the "import_test.py" would be like below ::

```
import test_add

K = test_add.Add(3,4)

print(K)
```

import a function

import a function

in practice

(continue..)

6. The function from different file is just imported and used.

7. In this case, we can cooperate, since one wrote a “useful” function involved python code, others can use it simply importing it.

Another way to importing 1

import test_add as T

K = T.Add(3,2)

==========

In this case, you just imported the “test_add.py” as name of “T” .

So you can do “K = T.Add(3,2)

” without typing the whole name “test_add”

Another way to importing 2

from test_add import Add

K = Add(3,2)

==========

More specific way to importing a function from python file.

You need to specify a function name “Add” included python file.

import a function

Another way to importing 3

from test_add import Add as A

K = A(3,2)

==========

This is combination of "Another way to importing 1" & "Another way to importing 2"

Practice

Write a function, which multiples two inputs, and import the function from another python code.

import a function (from other's directory, absolute path importing)

0. locate the importing file

1. We need to locate the importing python file
2. In this example we will locate the file, in the fashion of absolute path.

Ex) my current working directory is "/Users/leejunho/Desktop/git/group_study/TESTs/junho"

The import target locates at "/Users/leejunho/Desktop/git/group_study/TESTs/functions"

1. import sys

1. "sys" is a internal module of python, provides setting path function.
2. The target importing python file is "test_add.py"
3. The target importing python file locates at "/Users/leejunho/Desktop/git/group_study/TESTs/functions"

2. The code

```
import sys

sys.path.append("/Users/leejunho/Desktop/git/group_study/TESTs/functions")

from test_add import Add

K = Add(3,2)

print(K)
```

==========

Any question ?

import a function (from other's directory, relative path importing)

0. locate the importing file

1. We need to locate the importing python file
2. In this example we will locate the file, in the fashion of relative path.

Ex) my current working directory is "/Users/leejunho/Desktop/git/group_study/TESTs/junho"

The import target locates at "/Users/leejunho/Desktop/git/group_study/TESTs/functions"

1. import sys

1. "sys" is a internal module of python, provides setting path function.
2. The target importing python file is "test_add.py"
3. The target importing python file locates at "/Users/leejunho/Desktop/git/group_study/TESTs/functions"
4. The target file is reachable from current working dirctory by "../functions"

2. The code

```
import sys
sys.path.append("../functions")
from test_add import Add
K = Add(3,2)
print(K)
```

==========

Any question ?

The “class” in python

class in general

Able to contain multiple functions as below :

```
Class ABC:
    def Add(a,b):
        z = a+b
        return z

    def Minus(x,y):
        k = x-y
        return k
```

import a function from a class

Way of importing a function from a class is similar with directly importing function.

The target importing class locates at “/Users/leejunho/Desktop/git/group_study/TESTs/class” ,

My current working directory is “/Users/leejunho/Desktop/git/group_study/TESTs/junho”

The code

```
import sys
sys.path.append("../class")
from class_test1 import ABC
KA = ABC.Add(3,2)
KM = ABC.Minus(3,2)
print(KA)
print(KM)
```

import internal function : random

1. “random” is …

1. The “random” is internal function of python, which generates random numbers in the range of indicated region
2. If the range is not set, [0,1] numbers would be generated
3. list comprehension interface available

2. Basic usage

```
import random

print(random.random())

Up_to_ten=random.randrange(
10)

print(Up_to_ten)


Ten_num = range(10)

random.shuffle(Ten_num)

print(Ten_num)

======================

random(), shuffle()
```

3. List comprehension

```
import random as r


Uni_ran = [r.random() for _ in
range(4)]


Range_ran =  [r.randrange(3,7)
for _ in range(10)]
```