Python 快速上手

基礎語法

2020/09/23

變數 Variables

Variables

```
>>> x = 2
>>> y = 5
>>> xy = “Hey”
>>> print(x+y, xy)
7 Hey
```

可同時給多個變數賦值

```
>>> x = y = 1
>>> print(x, y)
1 1
```

Python 保留字

- 這些字已被 Python 保留特定意義，不能用來當作變數名稱。

and	continue	yield	for	in
as	def	True	from	Is
assert	del	with	global	not
break	elif	except	if	pass
class	else	False	import	while

運算 Calculator

Python 運算符號

Symbol	Task performed
+	加
-	減
*	乘
/	除以
%	求餘數
//	相除後去除小數點後的值
**	指數運算

Python 數學運算

```
>>> 17 / 3
5.666666666666667
>>> 17 // 3 # 去除小數點後的數值
5
>>> 17 % 3 # 算餘數
2
>>> 2*2*2*2*2
32
>>> 2**5 # 2 的 5 次方
32
```

Numbers - XOR

```
>>> 2**10
1024
>>> 2^10 # XOR operation
8
```

XOR	1	0
1	0	1
0	1	0

Decimal	Binary
2	0010
10	1010
?	1???

Numbers - XOR

```
>>> 2**10
1024
>>> 2^10 # XOR operation
8
```

XOR	1	0
1	0	1
0	1	0

Decimal	Binary
2	0010
10	1010
?	10??

Numbers - XOR

```
>>> 2**10
1024
>>> 2^10 # XOR operation
8
```

XOR	1	0
1	0	1
0	1	0

Decimal	Binary
2	0010
10	1010
8	1000

Python 比較符號

Symbol	Task Performed
==	相等回傳 True
!=	不相等回傳 True
<	小於回傳 True
>	大於回傳 True
<=	小於等於回傳 True
>=	大於等於回傳 True

內建函數 Built-in Functions

type

```
>>> print(type(100), 100)
<class  'int'  > 100

>>> print(type(3.14), 3.14)
<class  'float'  > 3.14

>>> print(type(1+2j), 1+2j)
<class  'complex'  > (1+2j)

>>>print(type(1e-10), 1e-10)
<class  'float'  > 1e-10 #Scientific notation but float type
```

int

```
>>> print(int( “010” ), 8)
8
>>> print(int( “0xaa” ,16))
170
>>> print(int( “1010” , 2))
10
>>> print(int(7.7))
7
>>> print(int( “7” ))
7
```

round, divmod

```
>>> print(round(5.6231))
6.0

>>> print(round(4.55892, 2))
4.56

>>> divmod(9, 2)
(4, 1)
```

isinstance

```
>>> print(isinstance(1, int))
True

>>> print(isinstance(1.0, int))
False

>>> print(isinstance(1.0, (int, str, float)))
True
```

range

```
>>> print(list(range(3)))
[0, 1, 2]
>>> print(list(range(2, 9)))
[2, 3, 4, 5, 6, 7, 8]
>>> print(list(range(2, 27, 8)))
[2, 10, 18, 26]
>>> print(list(range(10, 0, -1)))
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

format

- 透過 format 加上 {}，把變數放進字串中

```
>>> name = “Adam”
>>> age = 18
>>> score = 87

>>> print(“name={} age={}”.format(name, age))
name=Adam age=18

>>>print(“name={n} age={a}”.format(a=age, n=name))
name=Adam age=18
```

format

- 透過 format 加上 {}，把變數放進字串中

```
>>> name = “Adam”
>>> age = 18
>>> score = 87

>>> print(“name:\t{}\nscore:\t{:.2f}”.format(name, score))
name:	Adam
score:	87.00
```

Containers 容器

Containers

- tuple : 不可新增、刪除或替換 tuple 內的元素
  - (3, 5, 6)
- list : 可新增、刪除或替換元素
  - [3, 5, 6, “dog”, False]
- set : 元素不可重複
  - {3, 5, 6, “dog”, False}
- dictionary : 為 {key : value} 的集合，key 不可重複
  - {“Name” : “Jim”, “Gender” : “Male”, “Age” : 27}

Method on tuple

```
>>> tuple = (3, 5, 6)
```

- tuple.count(item) 計算 item 的個數
- tuple.index(item) 尋找 item 的索引值

list indexing

```
>>> x = [ ‘apple’ , ‘orange’ ]
>>> x[0]
apple
>>> x[-1]
orange

>>> y = [ ‘carrot’ , ‘potato’ ]
>>> z = [x, y]
[[ ‘apple’ , ‘orange’ ], [ ‘carrot’ , ‘potato’ ]]
```

list indexing

```
>>> z[1]
[‘carrot’, ‘potato’]

>>> z[0][0]
apple

>>> num = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> num[:4]
[0, 1, 2, 3]
>>> num[:9:3]
[0, 3, 6]
```

Methods on list

list = [3, 5, 6, “dog”, False]

- list.append(item) 新增元素
- list.extent([item1, item2, …]) 新增多個元素
- list.remove(item) 移除元素
- list.pop() 取出最後一個元素
- list.reverse() 倒轉 list
- list.sort(reverse=False) 排序
    - 會直接改變 list 順序 (不會回傳結果)

Methods on set

>>> set = {3, 5, 6, “dog”, False}

- set.add(item) 新增元素
- set.update([item1, item2, …]) 新增多個元素
- set.remove(item) 移除元素
- set.pop() 取出最後一個元素
- set.difference(set2) 比較兩個 set 中不一樣的元素
- set.intersection(set2) 找出兩個 set 中都有的元素

Methods on dict

dict = {“Name”:”Jim”, “Gender”:”Male”, “Age”:27}

- dict.update({key1: val1}) 更新 key1, value1
- dict.update(dict2) 加入 dict2 所有的 key, value
- dict.pop(key) 移除 key
- dict.get(key) 取出 key 所對應的 value (無則回傳 None)
- dict.keys() 回傳所有的 key
- dict.values() 回傳所有的 value
- dict.items() 以 tuple 回傳所有的 key, value

控制流程 Control flow

if, elif, else

```
>>> if x < 0:
...       print(‘Negative’)
...    elif x == 0:
...       print(‘Zero’)
...    elif x == 1:
...       print(‘One’)
...    else:
...       print(‘More’)
```

logical operation

```
>>> if x < 0:
...     print( 'Negative' )
...   elif x == 0 or x == 1:
...     print( 'Zero or One' )
...   elif x > 1 and x <= 10:
...     print( "From 2 to 10" )
...   else:
...     print( 'More' )
```

for

iterate each element in iterable date type

```
>>> words = [  'Adam'  ,   'Brute'  ,   'Case'  ,   'Den'  ]
>>> for w in words:
...       print(w)
...
Adam
Brute
Case
Den
```

range

iterate each element with index

```
>>> words = [  'Adam'  , '  Brute'  ,   'Case'  ,   'Den'  ]
>>> # from beginning to length of words
...    for i in range(len(words)):
...        print(words[i])
...
Adam
Brute
Case
Den
```

range

iterate each element with index

```
>>> words = [ ‘Adam’ , ‘Brute’ , ‘Case’ ,’ Den’ ]
>>> # from 1 to length of words
...   for i in range(1, len(words)):
...       print(words[i])
...
Brute
Case
Den
```

range

iterate each element with index

```
>>> words = [ ‘Adam’ , ‘Brute’ , ‘Case’ ,’ Den’ ]
>>> # from 1 to length of words and skip one for each
...   for i in range(1, len(words),2):
...       print(words[i])
...
Brute
Den
```

enumerate

iterate each element with index and value

```
>>> words = [ ‘Adam’ , ‘Brute’ , ‘Case’ , ’ Den’ ]
>>> for i, w in enumerate(words):
...     print(i, w)
...
0 Adam
1 Brute
2 Case
3 Den
```

break

Breaks out the loop

Example: Fine the str i

```
>>> for x in  'string_sergksdfgsdfgsjgegjenmksbsb' :
...     if x == 'i' :
...         break
...     print(x)
...
```

pass

Doing nothing

```
>>> # infinite loop
...    while True:
...        pass
...
```

try... except...

We want to handle the exception rather than quit the process

```
>>> def divide(x, y):
...       return x/y

>>> divide(1, 0) # exception occurred, raise exception
... ZeroDivisionError: division by zero
>>> try:
...    divide(1, 0) # exception occurred, jump to except
...    except Exception as e:
...       print(‘Exception’, e)
```

List comprehension

```
>>> a = []
>>> for i in range(10):
...       a.append(i)
...
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> b = [i for i in range(10)] # the fast way to create list
>>> b
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List comprehension

Test each way in running time

```
$ python –m timeit –s    'a=[]'      'for i in range(10): a.append(i)'
500000 loops, best of 5: 803 nsec per loop

$python –m timeit    'b=[i for i in range(10)]'
500000 loops, best of 5: 494 nsec per loop
```

Define Funtcion

Example: Fibonacci series

```
>>> def fib(n):
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)
...         a, b = b, a+b
...     return result
...
>>> fib(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Function Argument

- Positional argument should place before keyword argument.

```
def test(x, y, z):
    print(x)
    print(y)
    print(z)

>>> test(1, y=2, 3)
SyntaxError!
>>>test(1, z=3, y=2)
1
2
3
```

List of positional argument

```
# sum of two variable
def add_2(x, y):
    retuen x+y

# sum of three variable
def add_3(x, y, z):
    return x+y+z
```

List of positional argument

```
def add_all(*n):
    result = 0
    for i in n:
        result += i
    return result

>>> add_all(1, 2)
>>> add_all(1, 2, 3, 4, 5, 6)
```

Anonymous Function - lambda

- Some expression and behavior of common function: Anonymous functions not bound to name

```
def add_v1(x, y):
    return x+y
>>> add_v1(1, 2)

add_v2 = lambda x, y: x+y
>>> add_v2(1, 2)
```