SANS
SEC573: Automating Information Security with Python 課程內容介紹

課程網頁： https://www.sans.org/course/automating-information-security-with-python

目錄

- Essentials Workshop with pyWars
- Essentials Workshop with MORE pyWars
- Defensive Python
- Forensics Python
- Offensive Python
- Capture the flag

Essentials Workshop with pyWars

- About Python
- Python 語法規則
- 變數
- 運算符號
- 字串
- 函式
- 條件式流程
- 模組
- 內部自省模式

About Python

- A kind of Interpreted language
- Created by Guido van Rossum
- It has two versions,Python2 and Python3
- Python 2.7 will retire in January , 2020

About Python

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!

優美勝於醜陋

明了勝於晦澀

簡潔勝於複雜

複雜勝於凌亂

扁平勝於嵌套

間隔勝於緊湊

可讀性很重要

即便假借特例的實用性之名，也不可違背這些規則

不要包容所有錯誤，除非你確定需要這樣做

當存在多種可能，不要嘗試去猜測 而是儘量找一種，最好是唯一一種明顯的解決方案

雖然這並不容易，因為你不是 Python 之父

做也許好過不做，但不假思索就動手還不如不做

如果你無法向人描述你的方案，那肯定不是一個好方案；

反之亦然

命名空間是一種絕妙的理念，我們應當多加利用

Python 語法規則

- 由上而下

```
a=10
b=2
a,b=b,a
print(a,b)
```

- 注重縮排

```
a=10
if(a==10):
…..a=a+1
print(a,b)
```

變數

Variable Types	Example
Integers int()	1,2,5,10…...
Longs long()	Python 2 only -Integer larger than the CPU word size(32,64bit)
Floats float()	3.14159
String str()	“this is a string” ‘and so is this’
List list()	[“this”,”is”,123,3.14]
Tuples tuple()	(“Group”,”of”,”values”)
Dictionary dict()	{“key1”:”value1”,“key2”:”value2”}

NOTE: 可使用 type() 察看變數類型

變數

```
>>> a = 0xff00
>>> a = int("ff00",16)
>>> a
65280
>>> hex(a)
'0xff00'
>>> b=0b11000101
>>> b=int("11000101",2)
>>> b
197
>>> bin(b)
'0b11000101'
```

指定 16 進制變數

轉換 10 進制變數為 16 進制變數

指定 2 進制變數

轉換 10 進制變數為 2 進制變數

運算符號 (數學運算)

Consider when x=5

Operation	Example	Result in x
Addition	x = x+5	10
Subtraction	x = x-10	-5
Multiplication	x = x*5	25
Division*	x = x/2	2.5
Floor	x = x//2	2
Modulo	x = x%2	1
Exponent	x = x**2	25

* 這裡我們使用 Python 3 除法運算

運算符號（邏輯運算）

x	y
False	True
True	False

not

x	y	z
False	False	False
False	True	False
True	False	False
True	True	True

and

x	y	z
False	False	False
False	True	True
True	False	True
True	True	True

or

```
>>> True or True and False
True
>>> (True or True) and False
False
```

運算符號 (位元運算)

Operation	Example	Result in x
bitwise and	11111110 & 101	00000100
bitwise or	11111010 | 110	11111110
bitwise exclusive or	11111011 ^ 110	11111101
bitwise complement	~00001111	11110001
shift bits left	01110001 << 1	11100010
shift bits right	11100011 >> 1	01110001

運算符號 (優先權)

```
>>> (1+2)*3
9
>>> 1+2*3
7
```

```
>>> True or True and False
True
>>> (True or True) and False
False
```

- 數學運算符號具有優先權 (( ) > ** > * > / > + > -)
- 邏輯運算符號亦具有優先權 (and > or)

練習題 1

- 實作 Fermat's little theorem

- 令 $g^p \equiv a \left(mod\ p \right) \forall\ g \in \Re \wedge g \perp p$

  Then $g^{p-1} \equiv 1 \left(mod\ p \right)$

練習題 2

- 請設計一個 Swap Function ，令 Swap(a=1,b=2) output 為 a=2,b=1

- 函式設計條件如下
  - 僅能使用 a,b 兩變數
  - 請以 C 語言整數範圍做參考， output 不能 Overflow

字串

- 字元的集合，例如 text 或 HTML source code
- 在 python 可以使用單引號（’）或雙因號（”） enclosed 字串

```
>>> mystring = 'hello'
>>> mystring
'hello'

>>> mystring = "hello"
>>> mystring
'hello'

>>> mystring = """hello"""
>>> mystring
'hello'
```

字串

- 格式字串（ python 2,python 3 通用）

Format string	Description
'%d'	Integer decimal
'%10d'	Integer decimal that is 10 digits wide
'%010d'	Integer decimal 10 wide with leading zero
'%x'	Hexadecimal(lowercase)
'%X'	Hexadecimal(uppercase)
'%f'	Floating-point decimal format
'%6.2f'	Floating-point 6 wide with decimal in l00ths place
'%s'	Make it a string with str()
'%%'	No seriously, print a percent sign

字串

- 格式字串 (範例)

```
>>> a=10
>>> print("%d"%a)
10
>>> print("%10d"%a)
        10
>>> print("%010d"%a)
0000000010
>>> print("%x"%a)
a
>>> print("%X"%a)
A
>>> print("%f"%a)
10.000000
>>> print("%6.2f"%a)
 10.00
>>> print("%06.2f"%a)
010.00
```

```
>>> print("%s"%a)
10

>>> print("%%",a)
('%%', 10)
>>>
```

字串

- 輸入 (python 2)

```
>>> def uses_input():
...     x=input("what is your name")
...

>>> uses_input()
what is your name __import__("os").system("ls")
```

輸出

```
output.txt
output-w3af.txt
Pictures
Public
PycharmProjects
studios.applab.bikestationstoronto.apk
Templates
Videos
w3af
```

非常危險阿，嚇死寶寶了 QQ

字串

- 解決方式

```
>>> try:
...     input = raw_input
...   except:
...     pass
...
>>> input("what is your name?")
```

可以在 Python 2 與 python3 執行此段語法

字串 (Methods)

Consider when x=”pyWars rocks!”

Description	Example	Result in x
Upper Case	x.upper()	PYWARS ROCKS!
Lower Case	x.lower()	pywars rocks!
Title	x.title()	Pywars Rocks!
Replace Substring	x.replace(‘cks’,’x’)	pyWars rox!
Is substring in x?	“War” in x	True
Is substring in x?	“Peace” in x	False
Convert to list	x.split()	[‘pyWars’,’rocks!’]
Count substrings	x.count(‘r’)	2

字串 (slicing)

Consider when x=

P	y	t	h	o	n		r	o	c	k	s
0	1	2	3	4	5	6	7	8	9	10	11
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Example	Result in x
x[0]	P
x[2]	y
X[0:3] or x[:3]	Pyt
X[0:-1] or x[:-1]	Python rock
X[3:]	hon rocks
X[0::2] or x[::2]	Pto ok
X[::-1]	Skcor nohtyP
X[:6][::-1]	nohtyP

字串 (Encode/Decode)

- Common codecs

codecs	Description
bz2	Bzip2 encoding/decoding
rot13	Rotates letters 13 ASCII places
base64	Uses Base64 to encode/decode bytes()
zip	Creates a compressed zip version
hex	Makes a hex string of chars ‘41414141’
utf-16	A 2-byte(16-bit)Unicode version

字串 (Encode/Decode)

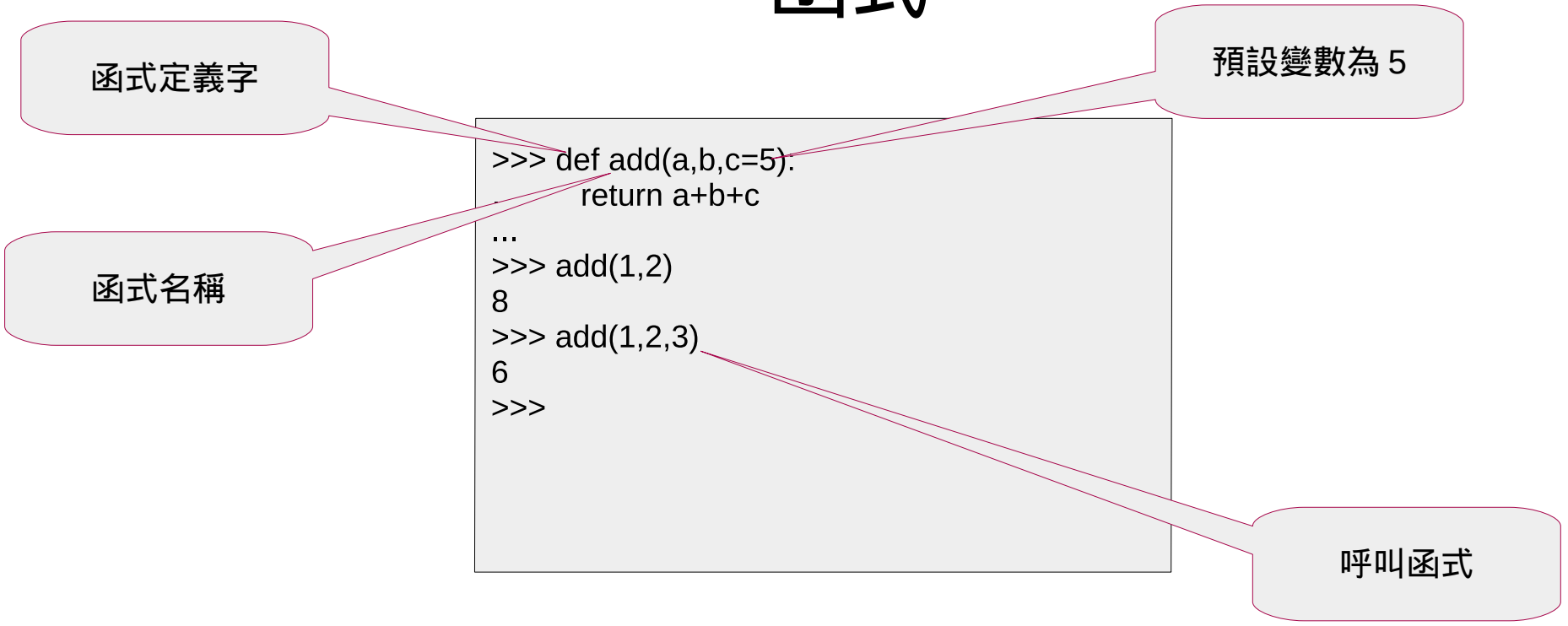
- Example

```
Python 2.7.15+ (default, Aug 31 2018, 11:56:52)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a="hello"
>>> k=a.encode('rot13')
>>> k
'uryyb'
>>> k.decode('rot13')
u'hello'
>>>
```

```
Python 3.6.7 (default, Oct 21 2018, 08:08:16)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import codecs
>>> a='hello'
>>> k=codecs.encode(a,'rot13')
>>> k
'uryyb'
>>> codecs.decode(k,'rot13')
'hello'
>>>
```

- >=2.7:codecs.encode(“ENCODE THIS”,”rot13”)
- <=2.7:“ENCODE THIS”.encode(“rot13”)

函式


函式定義字
預設變數為 5
函式名稱
呼叫函式
>>> def add(a,b,c=5):
... return a+b+c
...
>>> add(1,2)
8
>>> add(1,2,3)
6
>>>

函式

- 函式回傳多個數值範例

```
>>> def add(a,b,c=5):
...       return "sum=",a+b+c

>>> a,b=add(5,6)

>>> print(a,b)
('sum=', 16)
```

- 一個函式可以回傳一個或多個數值
- 銜接函式回傳值時，變數數量需與函式回傳數量一致
- 從最左邊之變數銜接函式回傳值

函式

- 函式內定義函式範例

```
>>> def sum(a,b,c=5):
...     def sub(a,b,c=5):
...             return a-b-c
...     total=sub(a,b,c)
...     return 100+total
...
>>> k=sum(5,6)
>>> k
94
```

- 一個函式內定義可以多個函式
- 每一個函式內定義函式，只能給該函式使用

條件式

- 判斷什麼狀況下該做什麼事
- Python 使用 if/else/elif
  - if<logic expression>:logic expression 為 true 時成立
  - else: 當 if 不成立時執行
  - elif<logic expression>: 也就是 else if 縮寫，表示 else 裡頭還有 if 判斷式

```
>>> a=100
>>> if a==100:
...         print("ok")
ok
>>> if a%2==1:
...         print("odd")
...     elif a==100:
...         print("100")
...     else:
...         print("even")
100
```

模組

- 一種可重複使用的程式區塊
- python 本身具有一些模組（如 sys ），也支援第三方模組
- Python 使用 **import** 加入模組至程式中
- 可從 https://pypi.org/ 搜尋所需模組

練習題 3

- 實作 KMP(Knuth-Morris-Pratt) Algorithm

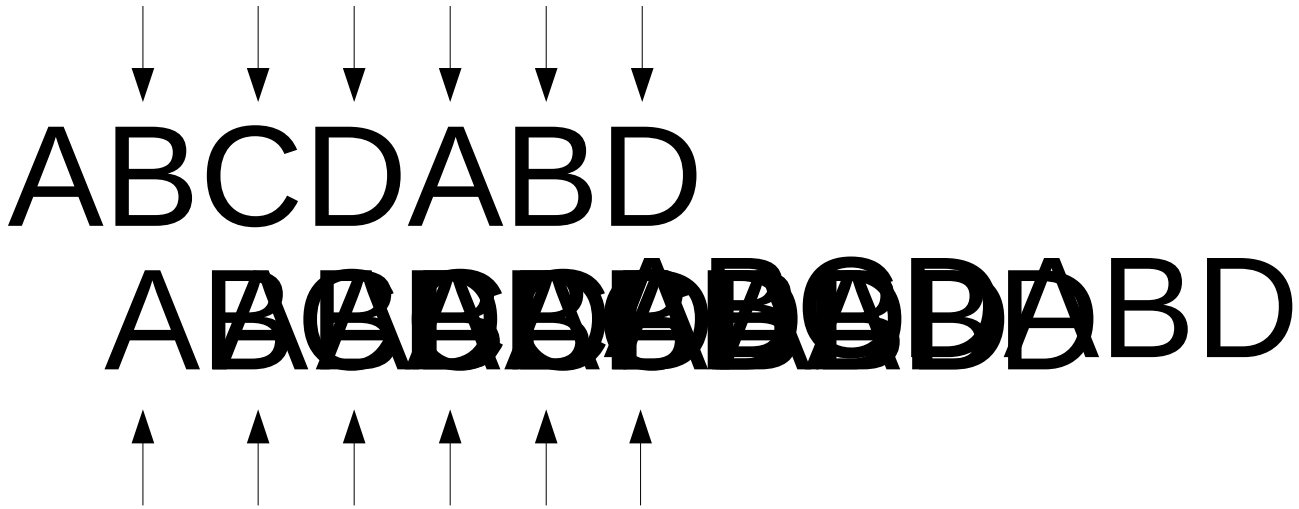
  時間複雜度： O(m+n)

  流程如下：
  - 建立 Partial match table
  - 字串比對

練習題 3(演算法講解)

建立 Partial match table

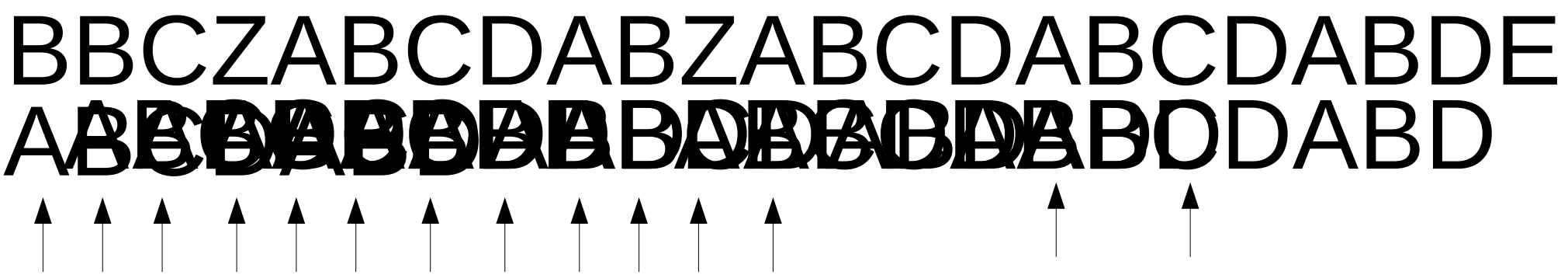
A	B	C	D	A	B	D
0	0	0	0	1	2	0



練習題 3(演算法講解)

字串比對

A	B	C	D	A	B	D
0	0	0	0	1	2	0



移動步數 = 比對相同數量 -table 紀錄數量

模組 (安裝方式)

- easy_install
- pip

模組 (easy_install)

- 是 python-setuptools package 的一部分，需用 apt-get 安裝
- easy_install <package name>

```
root@kali:~# easy_install TinyUrl
Searching for TinyUrl
Reading https://pypi.python.org/simple/TinyUrl/
Downloading https://files.pythonhosted.org/packages/c4/40/49e8af283a63c7d95a3af7
bb64ec9fa9a83826df7f7ecdd158fe5c6ba9ea/TinyUrl-0.1.0.zip#sha256=dc3210a2e4ae428d
2c601951b8c9569262fd39dfc7c62ca51f935aba5e77f09f
Best match: TinyUrl 0.1.0
```

模組 (pip)

- 另一種 python package manager
- 官方網站 www.pip-installer.org
- 使用 python get-pip.py 安裝

```
root@kali:~# pip install TinyUrl
Collecting TinyUrl
  Downloading https://files.pythonhosted.org/packages/9a/15/2c8ee3d38c816af239c3
214c3ade670f6396b9cdbf4e726e18d073ecff7f/TinyUrl-0.1.0.tar.bz2
Building wheels for collected packages: TinyUrl
  Running setup.py bdist_wheel for TinyUrl ... done
  Stored in directory: /root/.cache/pip/wheels/6a/4e/05/fcd63b806f75fd4e40cce525
59b9f9c93337ab098c42510443
Successfully built TinyUrl
Installing collected packages: TinyUrl
Successfully installed TinyUrl-0.1.0
```

模組 (pip)

- **pip<command> <command options>**
  - help Display help with pip and its command
  - install Install packages
  - uninstall Uninstall packages
  - list List installed packages
  - show Show information about installed packaged
  - search Search for packages
- Example of common usage:
  - pip help install Get help with install command
  - pip list List installed packages
  - pip show <installed packages> Get info on installed package
  - pip search <keyword> Search PyPI for packages related to keyword
  - pip install <package> Install a given package
  - pip install --upgrade <package> Upgrade existing package

內部自省模式

- 一種用來察看如何使用模組方法
- 舉例來說
  - Dir()　　列出物件裡所有方法及屬性
  - Help()　說明模組使用方式
  - Type()　說明所給的物件之類型

python 如何尋找模組？

1. 透過作業系統路徑（ PATH ）之環境變數尋找
2. 或者在程式碼所在的路徑尋找
3. 也可以使用 sys.path.append 加入新路徑

練習題 4

請將練習 3 模組化後，移到任意位置，然後透過 sys.path.append 加入新路徑後呼叫模組。

Essentials Workshop with MORE pyWars

- List、 Tuples、 Dictionaries 操作
- for and while loop
- Python Debugger
- 輸入參數設計

List、 Tuples、 Dictionaries 操作

- List 所提供的使用方法
  - list[index]=value　　改變指定位置數值
  - append(value)　　加入數值
  - insert(position,value)　　在給定的位置插入數值
  - remove(value)　　刪除指定數值（第一次比對到的）
  - sort(key,direction)　　排序
  - count(value)　　統計 list 內有多少指定的數值
  - index(value)　　察看指定的數值在 list 的位置
  - del list[index]　　刪除指定位置的數值

for and while loop

- 迴圈是一種環狀的控制結構
- python 中可使用的迴圈表達方式
  - for x in list:
  - for x in range(100):
  - for x in range(start,stop,step):
  - for index,value in enumerate(list):
  - While x:

List、 Tuples、 Dictionaries 操作

- List 屬於陣列儲存方式，可以針對元素進行新增、刪除修改

```
root@kali:~# python
Python 2.7.15+ (default, Aug 31 2018, 11:56:52)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> list=["1",2,3,"hello",[1,2,3]]
>>> list
['1', 2, 3, 'hello', [1, 2, 3]]
>>> k=tuple(list)
>>> k
('1', 2, 3, 'hello', [1, 2, 3])
```

```
>>> list[0]
'1'
>>> list[4]
[1, 2, 3]
>>> list[4][1]
2
>>> k[1]
2
```

List、 Tuples、 Dictionaries 操作

```
root@kali:~# python
Python 2.7.15+ (default, Aug 31 2018, 11:56:52)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> name=["alice","bob"]
>>> name[1]
'bob'
>>> name[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> name.append("candy")
>>> name
['alice', 'bob', 'candy']
>>> name.insert(1,"bob")
>>> name
['alice', 'bob', 'bob', 'candy']
```

```
>>> name.remove('bob')
>>> name
['alice', 'bob', 'candy']
>>> name.sort(reverse=True)
>>> name
['candy', 'bob', 'alice']
>>> name.append("bob")
>>> name.count("bob")
2
>>> name.index("bob")
1
```

List、 Tuples、 Dictionaries 操作

- slicing and math works on list

- Math!
- Slicing!

```
>>> a=["this","is"]
>>> b=["a","test"]
>>> c=a+b
>>> c
['this', 'is', 'a', 'test']
>>> c=a*2
>>> c
['this', 'is', 'this', 'is']
>>> c[1:]
['is', 'this', 'is']
>>> c[::-1]
['is', 'this', 'is', 'this']
>>>
```

List、 Tuples、 Dictionaries 操作

- Making copies of list

```
>>> alist = ["alice","bob",1,2,3]
>>> blist=alist
>>> blist
['alice', 'bob', 1, 2, 3]
>>> alist
['alice', 'bob', 1, 2, 3]
>>>
>>> clist=list(alist)
>>> clist.remove(1)
>>> clist
['alice', 'bob', 2, 3]
>>> alist
['alice', 'bob', 1, 2, 3]
>>>
```

List、 Tuples、 Dictionaries 操作

- convert strings to list with .split()

```
>>> print("THIS IS A STRING CONVERTED TO A LIST".split())
['THIS', 'IS', 'A', 'STRING', 'CONVERTED', 'TO', 'A', 'LIST']
>>> print("'comma','selimited','1.2'".split(","))
["'comma'", "'selimited'", "'1.2'"]
>>> print("THIS IS A LIST WITH IN IT".split("IS"))
['TH', ' ', ' A L', 'T WITH IN IT']
```

- convert lists to string with .join()

```
>>> " ".join(["SEC573","is","awesome"])
'SEC573 is awesome'
>>> ",".join(["SEC573","is","awesome"])
'SEC573,is,awesome'
>>> "".join(["SEC573","is","awesome"])
'SEC573isawesome'
>>>
```

List、 Tuples、 Dictionaries 操作

- Useful functions that work on list
- Sum([]) :adds all the integers in a list

```
>>> sum([2,4,6])
12
```

- Zip([],[]):creates a list of items at index 1 from each list followed by items at index 2,etc

```
>>> zip([1,2],['a','b'])
[(1, 'a'), (2, 'b')]
>>> zip([1,2],['a','b'],[4,5,6])
[(1, 'a', 4), (2, 'b', 5)]
>>>
```

List、 Tuples、 Dictionaries 操作

- More functions that work on list
- map(func(),[]) :Run function on a list or iterable

```
>>> list(map(ord,["A","B","C"]))
[65, 66, 67]
```

- map(func(),[],[]) :func is a custom zipper()

```
>>> def addint(x,y): return int(x)+(y)
...
>>> list(map(addint,[1,2,3],[4,5,6]))
[5, 7, 9]
>>> list(map(lambda x,y:int(x)+int(y),[1,2,3],[4,5,6]))
[5, 7, 9]
>>>
```

List、 Tuples、 Dictionaries 操作

- Tuples 屬於陣列儲存方式，但無法針對元素進行新增、刪除修改
- 是輕量化、去函式化（ less-functional ）的 list
- 因為是輕量化 list, 則排序的速度比 list 快
- list 與 tuples 之間可以互相轉換

```
>>> number=[1,2,3]
>>> number=tuple(number)
>>> number
(1, 2, 3)
>>> number=list(number)
>>> number
[1, 2, 3]
>>>
```

List、 Tuples、 Dictionaries 操作

- Dictionaries 屬於結構儲存方式
- 採用 hash_map 儲存，但搜尋速度為 O(1)
- 可以針對元素進行新增、刪除修改
- {key1:value1,key2:value2,key3:value3……}

```
>>> data={"1":"alice","2":"bob","3":"candy"}
>>> data
{'1': 'alice', '3': 'candy', '2': 'bob'}
>>>
```

無序排列

```
>>> from collections import OrderedDict
>>> data =OrderedDict()
>>> data["1"]="alice"
>>> data["2"]="bob"
>>> data["3"]="candy"
>>> data
OrderedDict([('1', 'alice'), ('2', 'bob'), ('3', 'candy')])
>>>
```

有序排列

List、 Tuples、 Dictionaries 操作

- Copies for dictionaries

錯誤方式

```
>>> dict1={1:"c",2:"b",3:"a"}
>>> dict2=dict1
>>> dict2[4]="d"
>>> dict2
{1: 'c', 2: 'b', 3: 'a', 4: 'd'}
>>> dict1
{1: 'c', 2: 'b', 3: 'a', 4: 'd'}
>>>
```

正確方式

```
>>> dict1={1:"c",2:"b",3:"a"}
>>> dict2=dict(dict1)
>>> dict2[4]="d"
>>> dict2
{1: 'c', 2: 'b', 3: 'a', 4: 'd'}
>>> dict1
{1: 'c', 2: 'b', 3: 'a'}
>>>
```

List、 Tuples、 Dictionaries 操作

- Common dictionary methods
  - Keys() returns a view of the keys
  - Values() returns a view of the values
  - Items() returns a view of tuples containing(key,value)

Python 2

```
>>> dict1={1:"c",2:"b",3:"a"}
>>> dict1.keys()
[1, 2, 3]
>>> dict1.values()
['c', 'b', 'a']
>>> dict1.items()
[(1, 'c'), (2, 'b'), (3, 'a')]
>>>
>>> dict1.viewkeys()
dict_keys([1, 2, 3])
>>> dict1.viewvalues()
dict_values(['c', 'b', 'a'])
>>> dict1.viewitems()
dict_items([(1, 'c'), (2, 'b'), (3, 'a')])
>>>
```

Python 3

```
>>> dict1={1:"c",2:"b",3:"a"}
>>> dict1.keys()
dict_keys([1, 2, 3])
>>> dict1.values()
dict_values(['c', 'b', 'a'])
>>> dict1.items()
dict_items([(1, 'c'), (2, 'b'), (3, 'a')])
>>>
```

List、 Tuples、 Dictionaries 操作

- “collections” 提供許多專門用途的字典可用
  - orderedDict : 依據放入元素的先後順序排序的字典
  - defaultdict: 遇到未定義 key 時會給予預設值的字典
  - Counter: 自動計算 key 出現次數的字典

List、 Tuples、 Dictionaries 操作

- defaultdict(factory function)

dictionaries

```
>>> dict={}
>>> dict
{}
>>> dict['1']=1
>>> dict['2']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: '2'
>>>
```

defaultdict

```
>>> from collections import defaultdict
>>> dict=defaultdict(list)
>>> dict["1"]=1
>>> dict["2"]
[]
>>> dict
defaultdict(<type 'list'>, {'1': 1, '2': []})
>>>
```

List、 Tuples、 Dictionaries 操作

- Counter()

example1

```
>>> from collections import Counter
>>> dcnt=Counter()
>>> dcnt.update(["a","b","c","d"])
>>> dcnt['a']
1
>>> dcnt.update("a")
>>> dcnt['a']
2
>>> dcnt['b']
1
>>>
```

example2

```
>>> import urllib2
>>> from collections import Counter
>>> c=Counter()
>>> content=urllib2.urlopen("http://www.metasploit.com").read()
>>> c.update(content.split())
>>> c.most_common(8)
[('<div', 107), ('</div>', 103), ('<a', 90), ('/></a>', 45), ('data-bio=""><img'
, 26), ('<td><a', 24), ('columns">', 23), ('class="row', 23)]
>>> for k in c.keys():
...     if "<" in k: del c[k];continue
...     if ">" in k: del c[k];continue
...     if "=" in k: del c[k];continue
...
>>> c.most_common(6)
[('the', 15), ('collapse', 13), ('medium-offset-1', 10), ('and', 9), ('security'
, 9), ('Metasploit', 8)]
```

練習題 5

給定兩個 list ，請將兩個 list 合併，並計算其中位數。

Note: 計算中位數時， list 需排序

Example:

```
A=[1,2,3]
B=[4,5,6]
            (3+4)/2=3.5

A=[1,2,3]
B=[4,5]
            3
```

練習題 6

給定 1 個 Tuples ，請將此 Tuples 轉成整數

Example:

```
Input:
A=(1,2,3)
Output:
123
Input:
A=(0,1,2)
Output:
12
```

Python Debugger

- Python 提供類似 gdb 的除錯模組 -pdb

```
root@kali:~/Desktop# python -m pdb 檔案名稱.py
```

指令	功能
b+ 數字	設置中斷點
r	繼續執行，直到當前函式返回
c	繼續執行程式
n	執行下一行程式
s	進入函式
p+ 變數名稱	印出變數
l	印出目前的程式片段
q	離開

輸入參數設計範例

參數輸入格式

```
root@kali:~/Desktop# python 檔案名稱.py [參數1] [參數2] [參數3]...
```

範例程式

```
root@kali:~/Desktop# cat sysarg.py
import sys
print("The number of arguments is %d" %len(sys.argv))
for eacharg in sys.argv:
    print(eacharg,sys.argv.index(eacharg))
```

輸出

```
root@kali:~/Desktop# python sysarg.py This is "a test"
The number of arguments is 4
('sysarg.py', 0)
('This', 1)
('is', 2)
('a test', 3)
```

注意，檔案名稱
也算一個參數

輸入參數設計範例

- python 建立輸入參數模組
- 可以解析輸入參數
- 例如 python debuger 呼叫方式

argparse

- python 提供 argparse ，是個好用的解析輸入參數套件
- 一開始需呼叫 ArgumentParser
- ArgumentParser(prog=None, usage=None, description=None, epilog=None)

參數名稱	說明
prog	program 的名字，可以把它覆寫成任何字串。
usage	告知使用者說應該怎麼使用 program。
description	通常是一段簡短的說明，用來告知使用者說這個程式在做什麼。
epilog	會出現在參數說明字串的最後面，通常是一些補充資料。

argparse

- add_argument: 定義並加入輸入參數

參數名稱	說明
name or flags	參數的名稱，可以用縮寫，但全名要有。 例如： --target, -t
default	預設的參數值
type	參數值的型態
required	參數值是否必須
help	參數的說明
dest	當 parse_args() 剖析完後的參數屬性。 若無指定此項，則為 name 的大寫全名

練習題 7

改寫練習 6 ，使用 argparse 傳遞 tuples ，並將 tuples 轉成整數

Example:

```
Input:
python xxx.py -x 123
Output:
(1,2,3)
123
```

Thank you for your attention