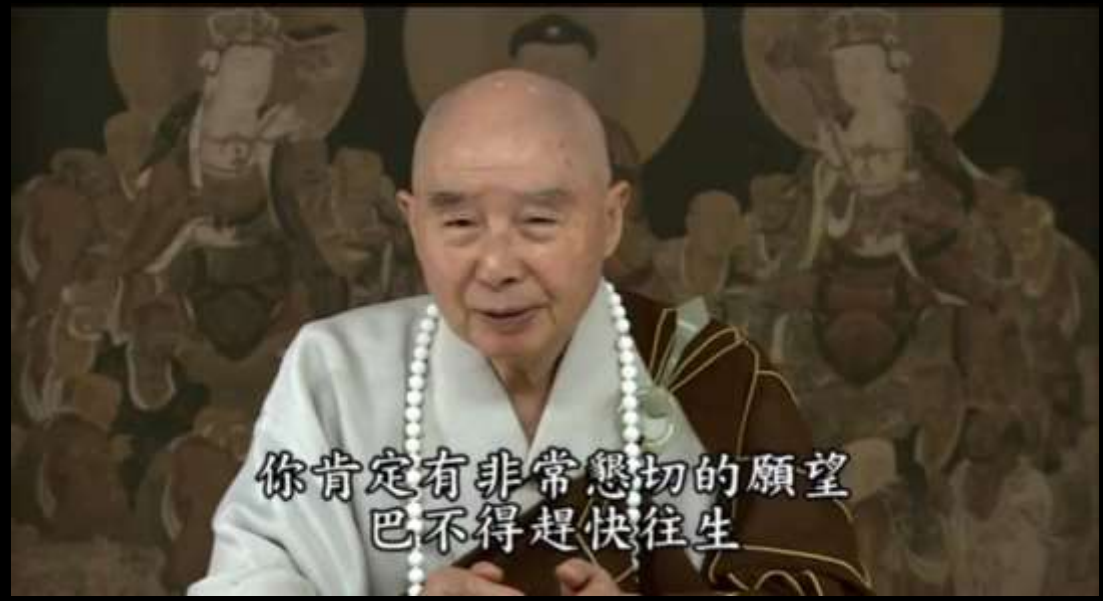
Python

bananaapple

Who am I?

- ID : bananaapple
- 學校科系 : 交通大學資工系
- 年級 : 大四
- 目前為 Bamboofox 中的一員
- 學習資安約一年

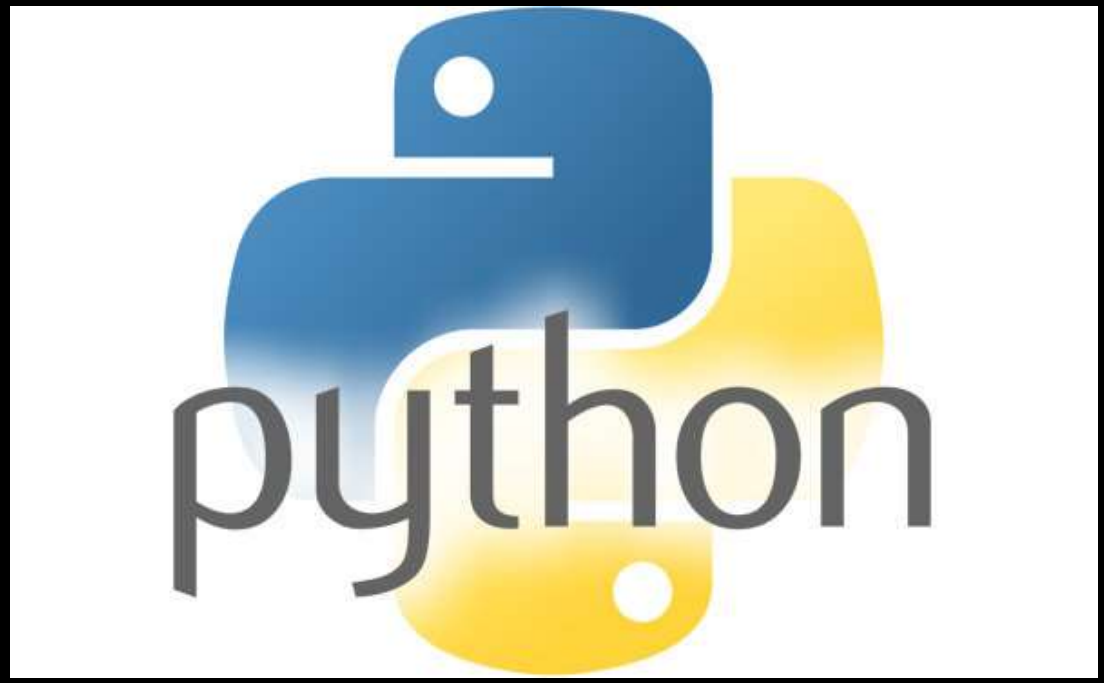


Outline

- Introduction
- Installation
- Getting Started
- Version
- Print
- Input
- Object
- Integer
- String
- List
- Arithmetic
- Conditional and Comment
- Loop and function
- Module
- Socket
- Struct
- Pwntools
- Vulnerable
- Practice
- Reference

Introduction

- Easy
- Swift
- Grace
- Object-oriented
- Strong module support
- Default built in most environment
- Script language



Installation

- Debian GNU / Linux

apt-get install python

apt-get install python-dev

// install additional header file and static library

- Windows

Sorry

Getting Started

- From terminal type python

python

- Save file with file extension .py and type python print.py

python print.py

- Add first line #!/usr/bin/env python
- Add executable privilege to file and ./filename execute it

./print.py

Version

- Python2 or Python3?
- We recommended use Python3
- Almost the same
- Except for print

Print

- End with newline character
- Format output

print "%d" % (100)

print "{0}{1}".format('hello', 'world')

If you want to manually control output use sys.stdout.write() instead

- Python2

```
>>> print "Hello world!"
Hello world!
```

- Python3

```
>>> print("Hello world!")
Hello world!
```

Input

- raw_input()

Read a line from stdin and strip a trailing newline

```
>>> s = raw_input()
apple
>>> s
'apple'
>>>
```

- Python2

raw_input()

- Python3

input()

Difference:

Python3 will run eval(input())

and return

Object

- Everything in Python is object
  - an identity (use id to observe it)
  - a value (immutable and mutable)
- Immutable: Integer, String, Tuple
- Mutable: List , Dictionary
- When immutable value change

id will be different

- When mutable value change

id will be the same

```
>>> a = 1
>>> id(a)
139693565145432
>>> a = 2
>>> id(a)
139693565145408
```

Integer

- Declare a variable

i = 1 or i = 0x5566

- Print integer as hex

```
i = 0x5566
hex(i)
# '0x5566'
chr(0x61)
# 'a'
```

- Change hex string to integer

```
s = '0x5566'
i = int(s,16)
print str(i)
# 21862
```

- Convert character to integer

```
ord('a')
# 97
```

String

- s.strip()

將字串頭尾的newline和space去掉

- s.find(‘string’)

return找到string的index

- s.replace('old', 'new', [max])

將old字串取代成new

最多取代max次

- s[0:len(s)]

```
s='abcde'
len(s) # 5
s=s[0:2] # s= 'ab'
s='abcde'
s[::2] # 'ace'
s[:-1] # 'abcd'
s[::-1] # 'edcba'
s[:] # 'abcde'
```

List

- Declare with []

lis =[]

- lis.append(element)

lis = [element]

- lis.remove(element)
- lis.sort()
- lis.reverse()

- Split string include spaces

s = 'a b c d e'

lis = s.split(' ')

lis = ['a', 'b', 'c', 'd', 'e']

- map(function_name, sequence)

```
def f(x):
        return x**2
map(f,range(10))
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

arithmetic

- Add

+

- Minus

-

- Multiply

*

- Divide

/

- Power

**

Ex: 2**3 = 8

- Modulo

%

Ex : 8 % 3 = 2

Conditional and Comment

```
if condition:
        statement
elif condition:
        statement
else:
        statement
```

- Single line comment begins with # character

```
#Code to be commented out
```

- Multiple line comment

```
"""
Code to be commented out
Code to be commented out
"""
```

Loop and function

for i in range(N):

    print I

will print 0 to N-1

for x in string:

    print x

will print every character in the string appended with newline

While condition:

    statement

in the loop we could use break or continue to control the loop

def function_name (parameter):

    statement

    return

Module

- import module
  - module.name
  - module.attribute

```
>>> import sys
>>> import os.path
>>> sys.modules['os']
<module 'os' from '/usr/lib/python3.4/os.py'>
>>> sys.modules['os.path']
<module 'posixpath' from '/usr/lib/python3.4/posixpath.py'>
>>> globals()['os']
<module 'os' from '/usr/lib/python3.4/os.py'>
>>> globals()['os.path']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'os.path'
>>> os
<module 'os' from '/usr/lib/python3.4/os.py'>
>>> os.path
<module 'posixpath' from '/usr/lib/python3.4/posixpath.py'>
>>>
```

Imports the module X, and creates a reference to that module in the current namespace. Then you need to define completed module path to access a particular attribute or method from inside the module (e.g.: X.name or X.attribute)

Module

- from module import *
  - name
  - attribute

```
>>> import sys
>>> from os import path
>>> globals()['path']
<module 'posixpath' from '/usr/lib/python3.4/posixpath.py'>
>>> locals()['path']
<module 'posixpath' from '/usr/lib/python3.4/posixpath.py'>
>>> globals()['os']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'os'
>>> path
<module 'posixpath' from '/usr/lib/python3.4/posixpath.py'>
>>> os.path
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>>
```

Imports the module X, and creates references to all public objects defined by that module in the current namespace (that is, everything that doesn't have a name starting with _) or whatever name you mentioned.

This makes all names from the module available in the local namespace.

Socket

```
from socket import *
from telnetlib import *
ip = '140.113.209.24'
port = 10000
s = socket(AF_INET, SOCK_STREAM)
s.connect((ip,port))
t = Telnet()
t.sock = s
t.interact()
```

Socket

- s.recv(buf_size)

收 buf_size 長度的字串

buf = s.recv(4096)

- s.send(string)

將 string 送過去

s.send(payload)

- s.close()

關閉 socket

Struct

- Pack the integer into little-indian or big-indian

```
import struct
address = 0x0804aabb
payload = struct.pack('<I', address)
#payload = "\xbb\xaa\x04\x08"
address = struct.unpack('<I', payload)[0]
hex(address)
# address = 0x804aabb
```

Pwntools

- Installation

sudo apt-get install python-dev

sudo apt-get install python-setuptools

sudo easy_install pip

sudo pip install pwntools

- Getting started

from pwn import *

Pwntools

- Context - Setting runtime variables

context.update(arch='i386', os='linux')

i386 is 32bits, amd64 is 64bits

- If you don’t want to see the notice

context.log_level = 'error'

Pwntools

~~ip = '140.113.209.24'~~

~~port = 10000~~

~~s = socket(AF_INET, SOCK_STREAM)~~

~~s.connect((ip,port))~~

- s = remote(ip, port)

~~t = Telnet()~~

~~t.sock = s~~

~~t.interact()~~

- s.interactive()

Pwntools

- Packing integer

~~address = 0x0804aabb~~

~~payload = struct.pack('<I', address)~~

- Payload = p32(0x0804aabb)
- 8 bytes?
- Payload = p64(0x0804aabb)

- Unpack string to integer

payload = "\xbb\xaa\x04\x08"

~~address = struct.unpack('<I', payload)[0]~~

- address = unpack(payload)

hex(address)

address = 0x804aabb

Pwntools

- Too much to list
- Shellcode
- Working with elf
- Working with gdb
- Memory leak
- Rop chain
- Translate assembly to string
- Shellcode

Vulnerable

- Pickle

```
import pickle
import os
class Exploit(object):
        def __reduce__(self):
                comm="sh"
                return (os.system, (comm,))
a = pickle.dumps(Exploit())
b = pickle.loads(a)
```

shell跑出來啦!!!

Practice

- Hackerrank

https://www.hackerrank.com/

- Combination

http://ctf.cs.nctu.edu.tw/problems/31

- Pickle

http://140.113.194.85:3000/problems/8

Reference

- 90% of Python in 90 Minutes

http://www.slideshare.net/MattHarrison4/learn-90

- From import vs import

http://stackoverflow.com/questions/9439480/from-import-vs-import

- Angelboy's CTF note

http://angelboy.logdown.com/posts/245988-ctf-notes

- Pwntools document

https://pwntools.readthedocs.org/en/2.2/about.html