Python Programming

NCTU Network Administration 2014

Created by darkgerm.

http://www.python.org/

Hello World

```python
#!/usr/bin/env python3
print('Hello World')
```

Outline

1. Introduction

Python Introduction

- General-purpose, high-level programming language.
- Dynamic typing, strong typing.
- Object-oriented, imperative and functional programming styles.
- Automatic memory management.
- Large and comprehensive standard library.
- 3rd package repository: PyPI (the Python Package Index)
    - 40357 packages now.
- `Readability` is important.

Who use Python

- Python is widely used in many domains, including:
  - Scientific and Math. (numpy, scipy)
  - Web programming. (Django, Pyramid)
  - Cloud computing. (Openstack)
  - Multimedia, animation, and graphics. (SimpleCV)
  - Game programming. (PyGame)
  - GUI programming. (PyQt, wxPython)
  - Hardware/Embedded system design. (raspberryPi)
  - Network Programming. (Twisted)
  - System tools. (yum, many gentoo tools)
  - …
- Heavy usage of Python at Google, Dropbox, …

The Zen of Python

There should be one obvious way to do it.

```
>>> import this
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 wa
Although that way may not be obvious at first unless you'r
Now is better than never.
Although never is often better than *right* now.
```

Python Versions

- Python 2.7.6
- Python 3.3.4
- 2 and 3 are not compatible.
    - 3 is now!!! We should learn 3.
    - `2to3` can help you convert code from 2 to 3.

Python Installation

- FreeBSD
  - with ports system:
    - `cd /usr/ports/lang/python3`
    - `make install clean`
  - with pkgng:
    - `pkg install python3`
- Built-in in Modern Linux
- `python --version`

Use Python Interpreter

```
$ python3
Python 3.3.2 (default, Jul  4 2013, 17:20:25)
[GCC 4.2.1 20070831 patched [FreeBSD]] on freebsd9
Type "help", "copyright", "credits" or "license" for more
>>>
```

Execute the Python Script

```
$ cat demo.py
#!/usr/bin/env python3
print('Hello World!')

$ python3 demo.py
Hello World!

$
```

Python document and help functions.

- **http://docs.python.org/3/**
- In Python Interpreter !!

```
>>> help()

Welcome to Python 3.3!  This is the interactive help

If this is your first time using Python, you should 
the tutorial on the Internet at http://docs.python.c

Enter the name of any module, keyword, or topic to g
Python programs and using Python modules.  To quit t
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or top
"keywords", or "topics".  Each module also comes wit
of what it does; to list the modules whose summaries
such as "spam", type "modules spam".

help>
```

Python document and help functions.

Example 1/3: `help()` functions.

```
>>> help('if')
The ``if`` statement
********************

The ``if`` statement is used for conditional executi

   if_stmt ::= "if" expression ":" suite
               ( "elif" expression ":" suite )*
               ["else" ":" suite]

It selects exactly one of the suites by evaluating t
by one until one is found to be true (see section *B
for the definition of true and false); then that sui
(and no other part of the ``if`` statement is execut
If all expressions are false, the suite of the ``els
present, is executed.

Related help topics: TRUTHVALUE
```

Python document and help functions.

Example 2/3: `help()` functions.

```
>>> help(str.split)
Help on method_descriptor:

split(...)
    S.split(sep=None, maxsplit=-1) -> list of string

    Return a list of the words in S, using sep as th
    delimiter string.  If maxsplit is given, at most
    splits are done. If sep is not specified or is N
    whitespace string is a separator and empty strin
    removed from the result.
```

Python document and help funcions.

Example 3/3: `dir()` functions.

```
>>> dir([])
['__add__', '__class__', '__contains__', '__delattr_
'__dir__', '__doc__', '__eq__', '__format__', '__ge_
'__getitem__', '__gt__', '__hash__', '__iadd__', '__
'__iter__', '__le__', '__len__', '__lt__', '__mul__'
'__reduce__', '__reduce_ex__', '__repr__', '__revers
'__setattr__', '__setitem__', '__sizeof__', '__str__
'append', 'clear', 'copy', 'count', 'extend', 'index
'remove', 'reverse', 'sort']
```

2. Python Data Type

Built-in Data Types

- Boolean Type - `bool`
- Numeric Types - `int`, `float`, `complex`
- Sequence Types - `list`, `tuple`
- Text Sequence Type - `str`
- Binary Sequence Type - `bytes`
- Mapping Type - `dict`
- Null Object - `None`
- Functions - `function`
- More Types - `bytearray`, `set`, ...

---

http://docs.python.org/3/library/stdtypes.html

Boolean Type - bool

- `True`, `False`
- `and`, `or`, `not`

Null Object

- `None`

Numeric Types - int, float, complex

- `1`, `2.17`, `3+4j`, `1e4`, `0xFF`, `0b1010`
- Operations:
  - `+`, `-`, `*`, `/`, `//`
  - bitwise: `|`, `^`, `&`, `<<`, `>>`
  - power: `**`
  - absolute: `abs()`
  - comparisons: `<`, `>`, `<=`, `>=`, `==`, `!=`
  - `round()`, `math.floor()`, `math.ceil()`
- All numbers are big number.

---

http://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex

Numeric Types - int, float, complex

Examples 1/3

```
>>> type(1)         # <class 'int'>
>>> type(2.3)       # <class 'float'>
>>> type(4 + 5j)    # <class 'complex'>

>>> 1 + 2 * 3       # 7
>>> 5 / 2           # 2.5
>>> 5 // 2          # 2
>>> 2 ** 31         # 2147483648
>>> 2 ** 100        # 126765060022822940149670320537
>>> 2 ** 0.5        # 1.4142135623730951
>>> abs(3 + 4j)     # 5.0

>>> import math
>>> math.e ** (math.pi * 1j)    # (-1+1.224646799147
```

Numeric Types - int, float, complex

Examples 2/3

```
# More about float
>>> round (3.14159, 2)        # 3.14
>>> import math
>>> math.floor(1.5)           # 1
>>> math.ceil(1.5)            # 2

# More about bitwise
>>> 0b101 | 0b10          # 0b111 = 7
>>> 1 << 10               # 1024

# More about complex
>>> x = 1 + 2j
>>> x.imag            # 2.0
>>> x.real            # 1.0
>>> x.conjugate()     # 1 - 2j
```

Numeric Types - int, float, complex

Examples 3/3

```
# More about comparison
>>> 3 > 2 > 1      # True

>>> a = 0.1 + 0.2
>>> b = 0.3
>>> a == b                  # False  (It's computer)

>>> allowed_error = 1e-6
>>> abs(a - b) < allowed_error   # True

# More about convert
# int(x, base=10), hex(x), oct(x), bin(x)
>>> int('3')          # 3
>>> int('0xF', 16)    # 15
>>> int('F', 16)      # 15
>>> int('10', 2)      # 2
>>> hex(254)          # '0xfe'
>>> bin(224)          # '0b11100000'
```

---

http://docs.python.org/3/library/functions.html#int

Sequence Types - list, tuple

- list: Mutable, `[1, 2.3, 'abc', [4, 5]]`
- tuple: Immutable, `(255, 255, 128)`
- Operations:
  - `in`, `not in`
  - extend: `+`
  - repeat: `*`
  - length: `len()`
  - index: `[start:end:step]`
  - `.append()`, `.extend()`, `.insert()`, `.sort()`
- List Comprehension.

---

http://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Sequence Types - list, tuple

Examples 1/5

```
>>> type([])          # <class 'list'>
>>> type(())          # <class 'tuple'>

# Common Sequence Operations  (list, tuple)
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> len(a)            # 3
>>> a + b             # [1, 2, 3, 4, 5, 6]
>>> [1] * 5           # [1, 1, 1, 1, 1]
>>> 2 in a            # True
>>> 4 in a            # False

>>> c = [5, 6, 6, 7, 8]
>>> c.count(6)        # 2
```

Sequence Types - list, tuple

Examples 2/5

```
# More about index  (list, tuple)
>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# index: 0  1  2  3  4  5  6  7  8  9
#      -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
>>> a[3]              # 3
>>> a[-1]             # 9
>>> a[100]            # IndexError: list index out of
>>> a[2:4]            # [2, 3]    Different from Perl!

>>> i = 3
>>> a[i:i+5]          # [3, 4, 5, 6, 7]
>>> len(a[i:i+5])     # 5

>>> a[:5]             # [0, 1, 2, 3, 4]
>>> a[6:]             # [6, 7, 8, 9]
>>> a[::2]            # [0, 2, 4, 6, 8]
>>> a[::-1]           # [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

Sequence Types - list, tuple

Examples 3/5

```
# More about Mutable Sequence Type (list)
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a                       # [1, 2, 3, 4]
>>> a.append([5])
>>> a                       # [1, 2, 3, 4, [5]]
>>> a.extend([6, 7])
>>> a                       # [1, 2, 3, 4, [5], 6, 7]
>>> a.pop()
>>> a                       # [1, 2, 3, 4, [5], 6]
>>> a.insert(0, 5566)
>>> a                       # [5566, 1, 2, 3, 4, [5], 6]
```

Sequence Types - list, tuple

Examples 4/5

```
# More about Immutable Sequence Type (tuple)
>>> hash([])         # TypeError: unhashable type: 'l
>>> hash(())         # 3527539

# More about tuple
>>> (x, y) = (4, 5)      # x = 4 and y = 5
>>> x, y = 4, 5          # same
>>> x, y = y, x          # swap

# Conversion
>>> list((1, 2, 3))      # [1, 2, 3]

# Sort
>>> a = [2, 0, 9, 3, 6, 1, 8, 4, 5, 7]
>>> a.sort()
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Sequence Types - list, tuple

Examples 5/5

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

>>> b = [i*i for i in a]
>>> b
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> c = [i for i in a if i % 2 == 0]
>>> c
[0, 2, 4, 6, 8]
```

Text Sequence Type - str

- Single quotes: `'"one" obvious way to do it.'`
- Double quotes: `"It's true."`
- Triple quoted: `'''multi-line string'''`, `"""same effect"""`
- Operations:
    - str is type of Immutable Sequence Type. Operations are the same :)
    - `str.split()`, `str.strip()`, `str.replace()`
    - `str.join()`
    - `str.format()`
    - `re` module. (Mention it later) **5. Built-in Modules**
    - `str.encode()` (Mention it later) **Binary Sequence Type - bytes**

---

http://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Text Sequence Type - str

Examples 1/2

```
>>> type('')            # <class 'str'>

>>> s = 'NA is very interesting.'
>>> 'NA' in s               # True
>>> s[6:10]                 # 'very'
>>> s[::-1][1:7:2].upper()  # 'GIS'
>>> s.replace('NA', 'Network Administration')
'Network Administration is very interesting.'

>>> s.split()
['NA', 'is', 'very', 'interesting.']

>>> '  bsd  \n\n  '.strip()
'bsd'

>>> ', '.join('abc')
'a, b, c'

>>> ''.join(['a', 'b', 'c'])
```

Text Sequence Type - str

Examples 2/2

```
>>> '{0} {1} {2}'.format('a', 'b', 'c')     # 'a b c
>>> '{} {} {}'.format('a', 'b', 'c')        # 'a b c
>>> '{2} {1} {0}'.format('a', 'b', 'c')     # 'c b a

>>> 'draw a circle at ({x}, {y}) r={radius}'.format(
...     x=3, y=4, radius=5
... )
'draw a circle at (3, 4) r=5'

>>> 'align right "{:>10}"'.format(345)
'align right "       345"'

>>> 'align center "{:-^20}"'.format('我是分隔線')
'align center "-------我是分隔線--------"'

>>> "int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:
'int: 42;  hex: 2a;  oct: 52;  bin: 101010'
```

http://docs.python.org/3/library/string.html#format-string-syntax

http://docs.python.org/3/library/string.html#format-examples

Binary Sequence Type - bytes

- Single quotes: `b'"one" obvious way to do it.'`
- Double quotes: `b"It's true."`
- Triple quoted: `b'''multi-line bytes'''`, `b"""same effect"""`
- `bytes` is immutable (same as `str`)
- The differences between `bytes` and `str` are:
  - `str` is unicode, while `bytes` is raw character array.`
  - We can encode `str` to `bytes`.
  - We can decode `bytes` to `str`.
  - `str` to C++ `string`, as `bytes` to C++ `char[]`.

---

http://docs.python.org/3/library/stdtypes.html#binary-sequence-types-bytes-bytearray-memoryview

Binary Sequence Type - bytes

Examples 1/2

```
>>> type(b'')            # <class 'bytes'>

>>> b'安安你好'     # SyntaxError: bytes can only con
>>> '安安你好'      # '安安你好'

>>> len('大中天')          # 3
>>> len('大中天'.encode())  # 9

# encode default is UTF-8
>>> '大中天'.encode()       # b'\xe5\xa4\xa7\xe4\xb8\
>>> '大中天'.encode('big5') # b'\xa4j\xa4\xa4\xa4\xd1
>>> '大中天'.encode('sjis') # b'\x91\xe5\x92\x86\x93V
```

Binary Sequence Type - bytes

Examples 2/2

```
# More about encode, decode

# You will receive bytes() from socket, system call,
# You should tell the program how to translate it.
>>> received = b'\x83n\x83\x8b\x83q'
>>> print(received)
b'\x83n\x83\x8b\x83q'

>>> print(received.decode())
UnicodeDecodeError: 'utf-8' codec cannot decode byte

>>> print(received.decode('sjis'))
ハルヒ
```

Mapping Type - dict

- `key: value` pairs
- `{ 'name': 'Yotsuba', 'age': 5, 'hair-color': 'green'}`
- Operations:
  - length: `len()`
  - get value: `d[key]`, `d.get(key[, default])`
  - delete key: `del d[key]`
  - find: `key in d`, `key not in d`
  - `d.keys()`, `d.values()`, `d.items()`

---

http://docs.python.org/3/library/stdtypes.html#mapping-types-dict

Mapping Type - dict

Examples 1/2

```
>>> type({})                # <class 'dict'>

>>> d = { 'name': 'Yotsuba', 'age': 5, 'hair-color':
>>> len(d)                  # 3, means 3 items.
>>> d['name']               # 'Yotsuba'
>>> d['height']             # KeyError: 'height'
>>> d['height'] = 107
>>> d
{'hair-color': 'green', 'name': 'Yotsuba', 'age': 5,

>>> 'age' in d        # True
>>> del d['age']
>>> d
{'hair-color': 'green', 'name': 'Yotsuba', 'height':
```

Mapping Type - dict

Examples 2/2

```
>>> d = { 'name': 'Yotsuba', 'age': 5, 'hair-color':
>>> d.keys()
dict_keys(['hair-color', 'name', 'height'])

>>> d.values()
dict_values(['green', 'Yotsuba', 107])

>>> d.items()
dict_items([('hair-color', 'green'), ('name', 'Yotsu

>>> for key, value in d:
...     print('key = {:<10}, value = {:<10}'.format(
...
key = hair-color, value = green
key = name      , value = Yotsuba
key = height    , value = 107
```

Functions

- Mention it later!
  - **4. Syntax and Control Flows**

3. Input and Output

Input and Output

- Standard I/O - `print()`, `input()`
- File I/O - `open()`

---

http://docs.python.org/3/tutorial/inputoutput.html

Standard I/O - print(), input()

Example: `print()`

```
>>>> print('hello', 'world')
hello world

>>> print('hello', 'world', sep=', ')
hello, world

>>> print('hello', 'world', sep=', ', end='$$$\n')
hello, world$$$

>>> print([1, 2, 3])         # convert to string usin
[1, 2, 3]

>>> import sys
>>> print('This line will be printed to stderr.', fi
This line will be printed to stderr.
```

---

http://docs.python.org/3/library/functions.html#print

Standard I/O - print(), input()

Example: `input()`

```
>>> name = input('What is your name: ')
What is your name: darkgerm

>>> print('your name is', name)
your name is darkgerm
```

---

http://docs.python.org/3/library/functions.html#input

File I/O - open()

Example: open a file for read.

```
>>> f = open('/etc/resolv.conf')
>>> print(f.read())
search cs.nctu.edu.tw
nameserver 140.113.235.103
nameserver 8.8.8.8
nameserver 140.113.1.1

>>> open('/etc/resolv.conf', 'r').readlines()
['search cs.nctu.edu.tw\n', 'nameserver 140.113.235.
'nameserver 8.8.8.8\n', 'nameserver 140.113.1.1\n']

>>> f = open('/etc/resolv.conf', 'rb')
>>> f.read(6)
b'search'

>>> f.close()
```

http://docs.python.org/3/library/functions.html#open

File I/O - open()

Example: open a file for write.

```
>>> f = open('/etc/hosts.allow', 'w')
>>> f.write('ALL : ALL : deny')
>>> f.close()

>>> open('/etc/hosts.allow').read()
ALL : ALL : deny
```

4. Syntax and Control Flows

Python Syntax 1/2

- Use `#` for inline comments.
- Use multi-line string for block comments.
  - `'''This is a comment'''`
  - `"""Me too"""`

```python
# I am a comment.
def add(a, b):
    """add two number."""
    return a + b
```

---

http://legacy.python.org/dev/peps/pep-0008/#comments

Python Syntax 2/2

- Use `indentation` to delimit program blocks.
  - `tab`, `any number of spaces` are OK, but only use one in a file.
  - Suggestion: `4 spaces` (PEP8)

```
def fibonacci(n):
    if n <= 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

---

http://en.wikipedia.org/wiki/Python_syntax_and_semantics#Indentation

PEP8: http://legacy.python.org/dev/peps/pep-0008/

Control Flows

- `if` statement
- `for` statement
- `while` statement
- `def` statement (functions)
- More control flows:
  - `try`, `except`, `raise` statements
  - `with` statement
  - `lambda` expression
  - ...

---

http://docs.python.org/3/tutorial/controlflow.html

http://docs.python.org/3/reference/compound_stmts.html

`if` statement

`if`, `elif`, `else`

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

---

http://docs.python.org/3/reference/compound_stmts.html#the-if-statement

`for` statement 1/2

`for` var `in` iterable_object `:` statement

```
for animal in ['cat', 'dog', 'fish', 'bird']:
    print(animal)

'''output:
cat
dog
fish
bird
'''


for char in 'abcd':
    print(char, end=' ')
print()

'''output:
a b c d
'''
```

---

http://docs.python.org/3/reference/compound_stmts.html#for

`for` statement

Useful function for iteration: `range()`

```
'''
range(stop)
range(start, stop[, step])
'''
square_numbers = []
for i in range(10):
    square_numbers.append(i*i)

print(square_numbers)   # [0, 1, 4, 9, 16, 25, 36, 49, 64,


odd_numbers = []
for i in range(1, 12, 2):
    odd_numbers += [i]

print(odd_numbers)      # [1, 3, 5, 7, 9, 11]
```

---

http://docs.python.org/3/library/functions.html#func-range

`while` statement

`while` condition `:` statement

```
#!/usr/bin/env python3
''' 3n+1 Problem '''
step = 0
n = int(input('give me a number: '))

while n != 1:
    if n % 2 == 0:  n //= 2
    else:           n = 3 * n + 1
    step += 1

print('{} steps to 1.'.format(step))

'''sample run
$ python3 3n+1.py
give me a number: 10
6 steps to 1.
'''
```

---

http://docs.python.org/3/reference/compound_stmts.html#while

`def` statement (functions) 1/2

`def` function_name `(` argument_list `)` `:`

```
#!/usr/bin/env python3
""" fibonacci """
def fib(n):
    a, b = 1, 1
    for i in range(n-2):
        a, b = b, a+b
    return b

while True:
    print(fib(int(input('n = '))))

"""sample run
n = 10
55
'''
```

http://docs.python.org/3/tutorial/controlflow.html#defining-functions

http://docs.python.org/3/reference/compound_stmts.html#def

`def` statement (functions) 2/2

functions are objects.

```
#!/usr/bin/env python3
rng = range                 # alias of built-in function range

def print_pow_of_2(n):
    for i in rng(n):
        print(2**i, end=' ')
    print()

ppo2 = print_pow_of_2
# ppo2 and print_pow_of_2 are the same function

ppo2(10)                    # 1 2 4 8 16 32 64 128 256 512
print_pow_of_2(10)          # 1 2 4 8 16 32 64 128 256 512

print(type(ppo2))           # <class 'function'>

print(ppo2 == print_pow_of_2)   # True
```

5. Built-in Modules

The Python Standard Library

- Python’s standard library is very extensive.
  - Regular Expression. (`re`)
  - Date and Time. (`datetime`)
  - Data Structure. (`heapq`)
  - Filesystem. (`os.path`, `stat`, `glob`)
  - Database. (`sqlite3`)
  - Compression and Archiving. (`zlib`, `gzip`, `zipfile`)
  - Concurrent Execution. (`threading`, `subprocess`)
  - Networking. (`socket`, `ssl`)
  - Internet Protocols. (`http`, `urllib`, `telnetlib`, `smtpd`)
  - Multimedia. (`audioop`, `wave`)
  - ...

The Python Standard Library

- It's impossible to introduce them all.
- Here I will introduce the following common modules.
    - Regular Expression. (`re`)
    - System call. (`subprocess`)
    - HTTP. (`urllib`)
    - Socket Programming. (`socket`)
    - Other modules. (`os`, `sys`)

---

http://docs.python.org/3/library/index.html

How to use modules?

Use `import` statement.

```
# import a module
import math
print('pi =', math.pi)                  # 3.14159265358979
print('log10(2) =', math.log10(2))      # 0.30102999566398

# only import a function/class/variable
from itertools import combinations
print(list(combinations('ABC', 2)))
# [('A', 'B'), ('A', 'C'), ('B', 'C')]

# import a module and make an alias
import random as rnd
rnd.randint(0, 10)          # 8  (or a random number in [(

# import a function/class/variable and make an alias
from datetime import date as dt
print(dt.today())           # 2014-03-02
```

Regular Expression (`re`)

- Provide regular expression matching operations similar to those found in Perl.
- Match the string at any location: `re.search()`
- Split the string by pattern: `re.split()`
- Find all the matched pattern: `re.findall()`

---

http://docs.python.org/3/library/re.html

http://docs.python.org/3/howto/regex.html#regex-howto

Regular Expression (`re`)

`re.search()` Example

```python
import re
''' re.search(pattern, string) '''

string = 'Sun Mar  2 21:33:29 CST 2014'

r = re.search('\d+:\d+:\d+', string)
print('{}-{} matched: {}'.format(r.start(), r.end(),
# 11-19 matched: 21:33:29

r = re.search('(\d+):(\d+):(\d+)', string)
print('hour = {}  minute = {}  second = {}'.format(
    r.group(1), r.group(2), r.group(3)
))
# hour = 21  minute = 33  second = 29
```

Regular Expression (`re`)

`re.split()`, `re.findall()` Example

```
>>> import re

>>> re.split('[: ]', 'Sun Mar  2 21:23:09 CST 2014')
['Sun', 'Mar', '', '2', '21', '23', '09', 'CST', '20

>>> re.findall('\w+', 'regexp is very important')
['regexp', 'is', 'very', 'important']

>>> re.findall('\w+s', 'raining cats and dogs')
['cats', 'dogs']
```

System call. (`subprocess`)

- `subprocess` allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.
- `subprocess.call()`
- `subprocess.check_output()`
- `subprocess.Popen`

---

http://docs.python.org/3/library/subprocess.html

System call. (`subprocess`)

`subprocess.call()` Example

```
#!/usr/bin/env python3
from subprocess import call

# call() is just like system() in C.
return_code = call('ls')        # 0

# The output will display on screen, but you can't g
# To get the output,
# you should use the more powerful one: check_output
```

http://docs.python.org/3/library/subprocess.html#subprocess.call

System call. (`subprocess`)

`subprocess.check_output()` Example

```
#!/usr/bin/env python3
from subprocess import check_output

# check_output() is just like backquote in Perl
stdout = check_output(['ls', '-al'])

# or convenient way (but not safe)
stdout = check_output('ls -al', shell=True)

# You can't get stderr and can't give the stdin.
# If you want to get control of stderr and stdin,
# you should use the more powerful one: Popen()
```

---

http://docs.python.org/3/library/subprocess.html#subprocess.check_output

System call. (`subprocess`)

`subprocess.Popen` Example 1/2

```
#!/usr/bin/env python3
import subprocess as sp

# Popen is the most powerful one.

# Example 1: execute `base64 -d` with stdin 'cHl0aG9

process = sp.Popen(
    ['base64', '-d'],
    stdin=sp.PIPE,
    stdout=sp.PIPE
)
stdout = process.communicate(input=b'cHl0aG9uCg==')[
# stdout = b'python\n'
```

---

http://docs.python.org/3/library/subprocess.html#popen-constructor

System call. (`subprocess`)

`subprocess.Popen` Example 2/2

```
#!/usr/bin/env python3
import subprocess as sp
import shlex

# Example 2: execute `/sbin/pfctl -t ssh_bruteforce

cmd = shlex.split('/sbin/pfctl -t ssh_bruteforce -T
# shlex.split() help you to split in shell way.
# cmd = ['/sbin/pfctl', '-t', 'ssh_bruteforce', '-T'

process = sp.Popen(cmd, stdout=sp.PIPE, stderr=sp.PI

stdout, stderr = process.communicate()
# stdout = (many ips)
# stderr = b'No ALTQ support in kernel\nALTQ relatec
```

HTTP. (`urllib`)

- `urllib` is a package, collects 4 modules.
  - `urllib.request`, `urllib.error`, `urllib.parse`, `urllib.robotparser`
- `urllib.request` defines functions and classes which help in opening URLs.
  - `urllib.request.urlopen()`
- `urllib.parse` defines a standard interface to manipulate URL (Uniform Resource Locator)
  - Parsing URL: `urllib.parse.urlparse()`
  - Parsing query string: `urllib.parse.parse_qs()`
  - String conversion: `urllib.parse.quote()`

---

http://docs.python.org/3/library/urllib.request.html#module-urllib.request

http://docs.python.org/3/library/urllib.parse.html#module-urllib.parse

HTTP. (`urllib`)

`urllib.request.urlopen()` Example

```
#!/usr/bin/env python3
""" get the google homepage. """
from urllib.request import urlopen

response = urlopen('http://www.google.com')

print(response.code)          # 200
print(response.msg)           # OK
print(response.headers)       # (the HTTP headers)
print(response.read())        # (the HTTP content)
```

---

http://docs.python.org/3/library/urllib.request.html#urllib.request.urlopen

HTTP. (`urllib`)

`urllib.parse.quote()` Example

```
#!/usr/bin/env python3
""" 取得 wiki "銀河系" 頁面 """
from urllib.reqeust import urlopen
from urllib.parse import quote

url = 'http://zh.wikipedia.org/wiki/'
keyword = '銀河系'

# urlopen(url + keyword)
# This will raise UnicodeEncodeError.
# Because '銀河系' is not valid ascii codes.

keyword_quote = quote(keyword)                # %E9%8A
response = urlopen(url + keyword_quote)       # succes

open('result.html', 'w').write(response.read())
```

---

http://docs.python.org/3/library/urllib.parse.html#urllib.parse.quote

Socket Programming. (`socket`)

- `socket` module provides access to the BSD socket interface.
- It is available on all modern Unix systems, Windows, MacOS, ...
- Common use:
  - Open a socket: `socket.socket()`
  - Connect the socket to (host, port): `socket.connect()`
  - Bind the socket to (host, port): `socket.bind()`
  - Listen the socket: `socket.listen()`
  - Accept a connection: `socket.accept()`
  - Receive/Send data from the socket: `socket.recv()`, `socket.sendall()`

---

http://docs.python.org/3/library/socket.html

http://docs.python.org/3/howto/sockets.html#socket-howto

Socket Programming. (`socket`)

Example (Echo server program)

```
# Echo server program
import socket

HOST = ''                 # Symbolic name meaning al
PORT = 50007              # Arbitrary non-privileged
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print('Connected by', addr)
while True:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
conn.close()
```

---

http://docs.python.org/3/library/socket.html#example

Socket Programming. (`socket`)

Example (Echo client program)

```
# Echo client program
import socket

HOST = 'daring.cwi.nl'    # The remote host
PORT = 50007              # The same port as used by
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM
s.connect((HOST, PORT))
s.sendall(b'Hello, world')
data = s.recv(1024)
s.close()
print('Received', repr(data))
```

http://docs.python.org/3/library/socket.html#example

Other modules. (`os`, `sys`)

- `os` module provides a portable way of using operation system dependent functionality.
  - You can find lots of unix system calls here.
  - `os.getuid()`, `os.getpid()`, `os.kill()`, ...
- `sys` module provides access to some variables used or maintained by the interpreter.
  - You can get the command line arguments passed to the script.
  - You can get the File object of stdin, stdout, stderr.
  - `sys.argv`, `sys.path`, ...

---

http://docs.python.org/3/library/os.html

http://docs.python.org/3/library/sys.html

Other modules. (`os`, `sys`)

Example

```
#!/usr/bin/env python3
# run this script by 'python3 demo.py arg1 asdf'
import os, sys

print(os.name)       # 'posix'
print(os.getuid())   # 14822
print(os.getpid())   # 10215
print(os.uname())
# posix.uname_result(sysname='FreeBSD', nodename='bs
# version='FreeBSD 9.1-RELEASE-p10 #0: Sun Jan 12 20
# root@amd64-builder.daemonology.net:/usr/obj/usr/sr

print(sys.argv)      # ['demo.py', 'arg1', 'adsf']
print(sys.path)
# ['/amd/cs/99/9917038', '/usr/local/lib/python33.zi
# '/usr/local/lib/python3.3', '/usr/local/lib/python
# '/usr/local/lib/python3.3/lib-dynload', '/usr/loca
```

6. 3rd Packages

PyPI - the Python Package Index

- **https://pypi.python.org/pypi**
- Use `pip` to install packages.
- Import it and use!

Other packages

- Download from it's website and follow its install instructions.
- Example: `BeautifulSoup`

```
# Download it from its website.
$ wget http://www.crummy.com/software/BeautifulSoup/bs4/dc

# untar it
$ tar zxvf beautifulsoup4-4.3.2.tar.gz

# Run `2to3` to make it compatible with Python3.
$ cd beautifulsoup4-4.3.2
$ 2to3 -w bs4
```

---

http://www.crummy.com/software/BeautifulSoup/

7. Examples

Examples

- `id.py`
    - control flows, functions, string, array, type conversion
- `latency.py`
    - system command, re
- `myip.py`
    - http request, re
- `parser.py`
    - file, re, dict
- `split.py`
    - file, string
- `youtube.py`
    - http request, re, argument parse, 3rd package

id.py

Code 1/3

```python
#!/usr/bin/env python3
"""Check the input is a valid id or not."""
import re

table = dict(
    A=10, J=18, S=26,
    B=11, K=19, T=27,
    C=12, L=20, U=28,
    D=13, M=21, V=29,
    E=14, N=22, W=32,
    F=15, O=35, X=30,
    G=16, P=23, Y=31,
    H=17, Q=24, Z=33,
    I=34, R=25,
)
```

id.py

Code 2/3

```python
def check(id_):
    digit = table[id_[0]]
    cks = digit // 10 + digit % 10 * 9
    cks += sum(int(id_[i]) * (9-i) for i in range(1,
    cks += int(id_[9])
    return cks % 10 == 0

# Alternative check.
def check2(id_):
    cks = int('10987654932210898765431320'[ord(id_[0
    cks += sum(int(id_[i]) * (9-i) for i in range(1,
    cks += int(id_[9])
    return cks % 10 == 0
```

id.py

Code 3/3

```python
if __name__ == '__main__':
    while True:
        id_ = input('please input id: ')

        if not re.search('^[A-Z]\d{9}$', id_):
            print('wrong format!')

        elif check(id_):
            print('valid')

        else:
            print('invalid')
```

Sample Run

```
$ python3 id.py
please input id: A123456789
valid
please input id: XDD
wrong format!
```

latency.py

Code 1/1

```python
#!/usr/bin/env python3
import re
import subprocess as sp

cmd = 'ping -c 5 linux1.cs.nctu.edu.tw | tail -n +2

ping_rst_bytes = sp.check_output(cmd, shell=True)
ping_rst = ping_rst_bytes.decode()

times = []
for line in ping_rst.split('\n'):
    reobj = re.search('time=(\d*\.\d*) ms', line)
    if reobj:
        times.append(float(reobj.group(1)))

print('sum = {:.3f} ms'.format(sum(times)))
print('max = {:.3f} ms'.format(max(times)))
print('min = {:.3f} ms'.format(min(times)))
```

latency.py

Sample Run

```
$ python3 latency.py
sum = 1.008 ms
max = 0.258 ms
min = 0.177 ms
```

myip.py

Code 1/1

```
#!/usr/bin/env python3
import re
from urllib.request import urlopen

url = 'https://www.esolutions.se/whatsmyinfo'
pattern = '<div class="col-md-8">(\d+\.\d+\.\d+\.\d+)</di

content = urlopen(url).read().decode()
reobj = re.search(pattern, content)
if reobj:
    print('my ip: {}'.format(reobj.group(1)))
else:
    print('cannot find your ip QQ.')
```

Sample Run

```
$ python3 myip.py
my ip: 140.113.235.135
```

parser.py

Code 1/1

```python
#!/usr/bin/env python3
import re

table = {}

#Dec 21 17:07:08 nat235 pure-ftpd: (?@192.168.0.15)
for line in open('xferlog', errors='ignore'):
    if 'logged' not in line: continue

    cols = line.split(' ')
    ip, user = cols[5][3:-1], cols[7]

    if ip not in table:         table[ip] = [user]
    elif user not in table[ip]: table[ip] += [user]
    else:                       pass        # do not

for key, value in sorted(table.items()):
    print('{:20s} {}'.format(key, value))
```

parser.py

Sample Run

```
$ python3 parser.py
192.168.1.103        ['ioi23']
192.168.1.193        ['ioi16']
192.168.1.210        ['ioi28']
```

split.py

Code 1/1

```
#!/usr/bin/env python3

pass_f = open('/etc/passwd')

for line in pass_f:
    if line.strip()[0] == '#': continue
    arr = line.split(':')
    if len(arr) < 2: continue
    print('username = {:<10} uid = {}'.format(arr[0]

pass_f.close()
```

split.py

Sample Run

```
$ python3 split.py
username = root       uid = 0
username = toor       uid = 0
username = daemon     uid = 1
username = operator   uid = 2
username = bin        uid = 3
username = tty        uid = 4
username = kmem       uid = 5
username = games      uid = 7
username = news       uid = 8
username = man        uid = 9
username = sshd       uid = 22
username = smmsp      uid = 25
username = mailnull   uid = 26
username = bind       uid = 53
username = proxy      uid = 62
username = _pflogd    uid = 64
username = _dhcp      uid = 65
username = uucp       uid = 66
```

youtube.py

Code 1/4

```
#!/usr/bin/env python3
import os
import re
import sys
from urllib.request import urlopen
from urllib.parse import quote

sys.path.append(os.path.abspath('./beautifulsoup4-4.
from bs4 import BeautifulSoup
```

youtube.py

Code 2/4

```python
def youtube_search(keyword, n=6):
    url_fmt = (
        'http://www.youtube.com/results'
        '?hl=en&search_query={}'
    )
    url = url_fmt.format(quote(keyword))

    content = urlopen(url).read().decode()
    html = BeautifulSoup(content)

    base = 'http://www.youtube.com'
    for link in html.find_all(href=re.compile("watch
        if 'Watch Later' not in str(link):
            print(base + link.get('href'))
            print(link.text.strip())
            print()
            n -= 1
            if n == 0: break
```

youtube.py

Code 3/4

```python
def main1():
    import getopt

    def usage():
        print('Usage: %s [-n N] keyword.' % sys.argv
        exit(1)

    try:
        opts, args = getopt.getopt(sys.argv[1:], 'n:

    except getopt.GetoptError as err:
        usage()

    if len(args) != 1: usage()

    n = 6
    for opt, arg in opts:
        if opt == '-n': n = int(arg)
```

youtube.py

Code 4/4

```python
def main2():
    import argparse
    parser = argparse.ArgumentParser(
            description='Youtube search engine.'
    )
    parser.add_argument('-n', type=int, default=6,
            help='number of search result. default i
    parser.add_argument(
            'keyword', nargs=1,
            help='keyword to search.'
    )
    args = parser.parse_args()

    youtube_search(args.keyword[0], n=args.n)


if __name__ == '__main__':
    main2()
```

youtube.py

Sample Run 1/2

```
$ python3 youtube.py -h
usage: youtube.py [-h] [-n N] keyword

Youtube search engine.

positional arguments:
  keyword     keyword to search.

optional arguments:
  -h, --help  show this help message and exit
  -n N        number of search result. default is 6.
```

youtube.py

Sample Run 2/2

```
$ python3 youtube.py 五五六六
http://www.youtube.com/watch?v=2NqXSYfL3as
『56不能亡！』5566金曲2小時終極串燒！

http://www.youtube.com/watch?v=wfezg0ThJyo
5566-56不能亡最強串燒金曲

http://www.youtube.com/watch?v=ixt-rNDkpTU
5566 串燒歌曲 你們還記得嗎?

http://www.youtube.com/watch?v=2Ii0kpKi8kI
5566【MVP情人】我難過 MV

http://www.youtube.com/watch?v=tb9jik2cwes
20140101 TVBS 五月天vs.5566 高雄合體開唱

http://www.youtube.com/watch?v=Hqty0e-KiPE
5566 - 存在
```

The course is going to end here.

However, things in this course are only 10% of Python.

Oops, maybe 5% or less.

What's more you can learn first

- `class`
- `lambda`
- Generators, `yield` statement.
- Exceptions, `try` and `except` statements.
- `set` type.
- `json` module.
- Write your own module.
- Docstring.
- Syntax suggestion: PEP8
- Binding with C: `ctypes` module, `<Python.h>`

What's more and more you can learn

- For Development: `virtualenv`
- For GUI: `PySide`, `wxPython`
- For 3D Graph: `VPython`
- For Image Manipulate: `SimpleCV`
- For Website Design: `Django`, `jinja2`, `Flask`, `web2py`
- For Scientific Calculation: `SciPy`, `NumPy`, `metaplot2`
- For Network Programming: `Twisted`
- For Documentation: `sphinx`

Welcome to Python World !

Learning Materials and References

- **The Python Tutorial**
- **Python Standard Library**
- 良葛格學習筆記