Python Programming

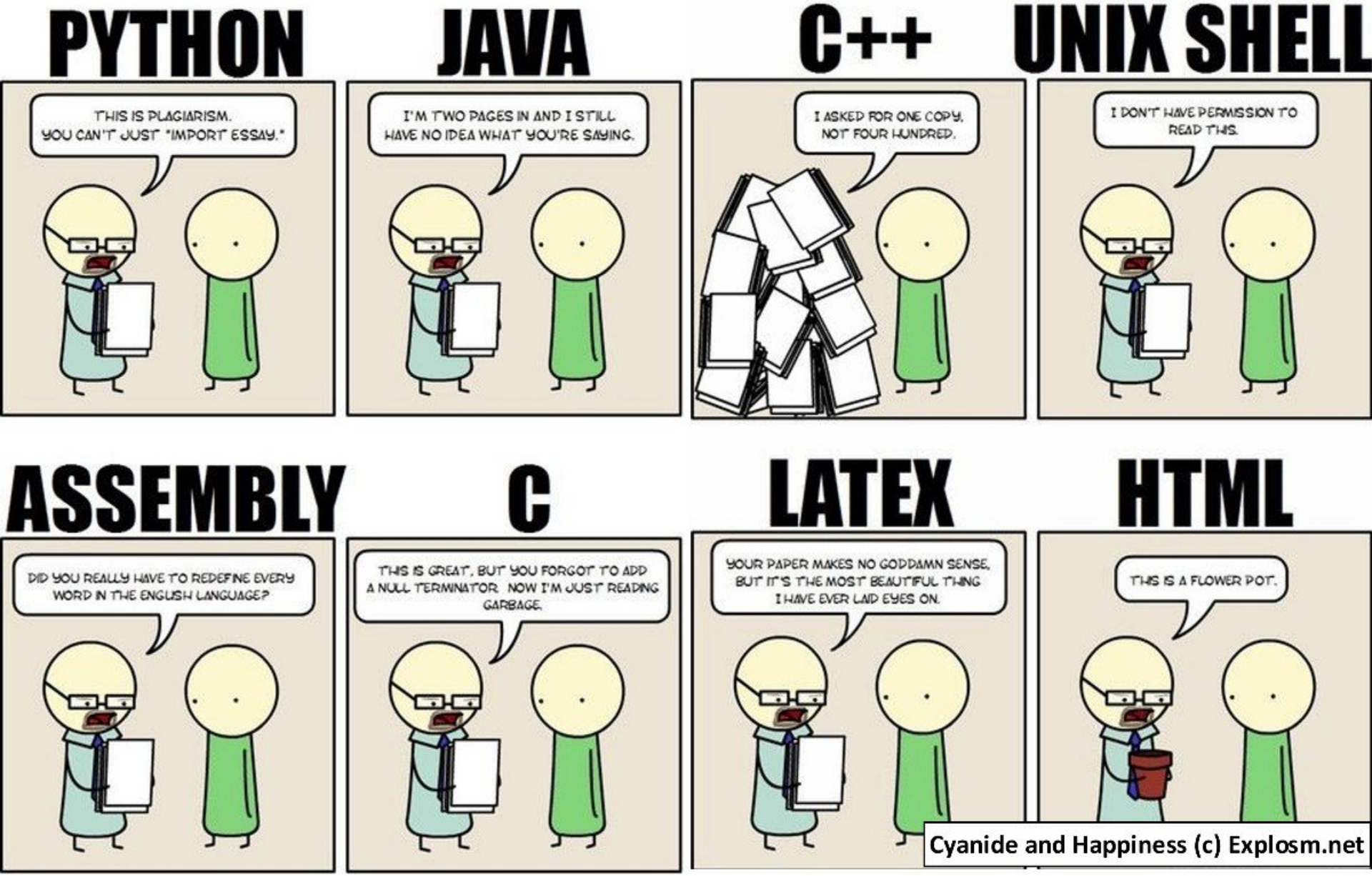
weicc

Outline

- Introduction
- Overview
  - Python
  - Why use Python
  - Python 3 or Python 2 ?
- Python data type
  - boolean, int, float, str
  - list, tuple
  - dictionary, set
- Python
  - for, if, while, try
  - function, class
- Python class
  - super()
  - self
  - property
  - __init__
- Generator & Decorator
- Python with PEP8
  - What is PEP8
  - How use

Introduction

- 直譯式語言
  - Python, JavaScript, Perl, Ruby, PHP
  - But not: Go, C, C++, JAVA
- 自動記憶體管理
  - Python, JavaScript
- 具有大量函式庫
  - sys, os, ftplib, smtp
  - requests, ldap3, NumPy, Pandas, matplotlib
  - PyPy, Cpython
  - Django (framework)
- 膠水程式


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
Cyanide and Happiness (c) Explosm.net

Overview

- Python
  - Beautiful, Explicit, Simple
  - Complex, Flat, Sparse
  - Readability
- 創作者 Guido van Rossum
  - 仁慈的獨裁者
- ABC (programming language) 的繼承

```
HOW TO RETURN words document:
    PUT {} IN collection
    FOR line IN document:
        FOR word IN split line:
            IF word not.in collection:
                INSERT word IN collection
     RETURN collection
```

Overview

- Python
  - Beautiful, Explicit, Simple
  - Complex, Flat, Sparse
  - Readability
- 創作者 Guido van Rossum
  - 仁慈的獨裁者
- ABC (programming language) 的繼承
- 開放語言
  - 與 C 語言結合能力非常出色

Python >>> import this

>>> 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 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!

Why use Python

- 誰在使用

Why use Python

- 對剛接觸程式語言的人友善
- Python 的使用範圍廣泛
  - Web, GUI application, Chatbot
- ML 與 AI 的應用，雲端運用
  - Spark
  - TensorFlow
- 敏捷 (Agile) 開發
- Python 優雅而高捷
  - 優秀的開發速度
  - 簡單的交接時間

Python2 or Python3

- 目前在 Linux 中還有大量 Python2 語言
- 目前在眾多商用 solution 中 Python2 仍然存在
  - Instagram Makes a Smooth Move to Python 3
  - https://thenewstack.io/instagram-makes-smooth-move-python-3/
- Python2 轉移 Python3 成本高
- Python 2.7 將於 2020 年退役
  - https://pythonclock.org/

Python2 or Python3

- print
  - Python 2 : print “I am good guy!”
  - Python 3 : print(“I am good guy!”)
- Utf-8
  - Python 2: ASCII str() and unicode()
  - Python 3: Utf-8
- Range
  - range and xrange -> range

Python2 or Python3

```
# weikevin @ Kevin-MacBook in ~/Documents [21:47:51]
$ cat test.py
from platform import python_version

print('Python', python_version())
print('strings are now utf-8 \u03BCnico\u0394é!')


# weikevin @ Kevin-MacBook in ~/Documents [21:47:56]
$ python3 test.py
Python 3.6.0
strings are now utf-8 μnicoΔé!

# weikevin @ Kevin-MacBook in ~/Documents [21:48:06]
$
```

How to use Python

- Text editor
  - Atom, VS code, Sublime
  - 以上都支援 PEP8
- IDE
  - PyCharm, Anaconda
  - Jupyter

PYTHON DATA TYPE

Python Data type

- int
  - 在 Python 2 整數為 32 bit
    - –2,147,483,648 到 2,147,483,647
    - 具有 long 型態
  - 在 Python 3 下沒有邊界

```
# weikevin @ Kevin-MacBook in ~/Documents [13:49:59]
$ cat test.py
from platform import python_version

print('Python', python_version())
data = 100000000000000000000000000000000000000000000000000000000000000000000000000000000000
print(data*data)


# weikevin @ Kevin-MacBook in ~/Documents [13:50:04]
$ python3 test.py
Python 3.6.0
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000

# weikevin @ Kevin-MacBook in ~/Documents [13:50:10]
$ 
```

Python Data type

- float
  - float(True)
    - 1.0
  - float(False)
    - 0.0

Python Data type

- str
  - PEP8
    - “Iamgoodguy” – 多用在給人看的字串
    - ’Iamgoodguy’ – 多用在給機器的字串
  - str + str
    - “Iamgoodguy” + “Iambadguy”
  - str * int
    - “Iamgoodguy” * 4

Python Data type

- list
  - ['pen', 'pineapple', 'apple']
    - append()
    - extend()
    - insert()
    - remove()
    - pop()
    - index()
    - count()
    - join()
    - sort()
    - len()

Python Data type

- tuple
  - (‘pen’, ‘pineapple’, ‘apple’)
    - 無法任意修改內容
    - 沒有 append, insert
  - 內容保存完整
  - 佔用空間小

Python Data type

- dict
  - {‘a’: ‘aaa’, ‘b’: ‘bbb’, ‘c’: ‘ccc’}
    - keys()
    - values()
    - items()
  - 一般 json 會從這個形態轉錄

Python Data type

- set
  - {‘pen’, ‘pineapple’, ‘apple’}
  - in
    - ‘pen’ in demoSet
      - True
    - ‘kevin’ in demoSet
      - False
  - 基本運算
    - 交集 & intersection()
    - 聯集 | union()
    - 差集 - difference()
    - 互斥 ^ symmetric_difference()

PYTHON OPERATOR

Python operator

- if
  - is 左右兩邊變數指向相同物件
  - == 左右兩邊變數內容是否相等

```
$ python3
Python 3.6.0 (v3.6.0:41df79263a11, Dec 22 2016, 17:23:13)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1000 is 10**3
False
>>> 1000 == 10**3
True
>>>
```

Python operator

- for
  - break 跳出迴圈
  - continue 跳過

```
>>>
>>> for letter in 'Python':
...     print(letter, end=' ')
...
P y t h o n >>>
>>>
```

Python operator

- while

```
1  numbers = [12, 37, 5, 42, 8, 3]
2  even = []
3  odd = []
4  while len(numbers) > 0 :
5      number = numbers.pop()
6      if(number % 2 == 0):
7          even.append(number)
8      else:
9          odd.append(number)
```

www.penjee.com

Python operator

- try

```
>>> short_list = [1, 2, 3]
>>> position = 5
>>> try:
...     short_list[position]
... except:
...     print('error: {}'.format(position))
...
error: 5
>>>
```

Python operator

- try
  - 沒有
    「要就好好做，要不然就不要做」
  - try 開頭
  - except 判別例外
  - raise
    - 不在 except 中 > 出現以下例外
      NameError, TypeError
      Concrete exceptions
    - 在 except 中 > 以例外回傳

Python operator

- func
  - def demoFunc(arg1='kevin', arg2):
  - *args tuple
  - **kwargs dictionary
  - 內部函示

```
>>> def outer(a, b):
...     def inner(c, d):
...         return c + d
...     return inner(a, b)
...
>>> outer(4, 7)
11
>>>
```

Python class

- Python 是類別語言
- num = 7
  - 建立一整數類別
- 物件分兩部分
  - 變數 – 屬性
  - 函式 – 方法

Python class

```
$ cat test.py
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart

x = Complex(3.0, -4.5)
print(x.r, x.i)
```

```
$ python3 test.py
3.0 -4.5
```

Python class

- 繼承

```
# weikevin @ Kevin-MacBook in ~/Documents [21:24:55]
$ cat test.py
class demo01():
    def addInt(self, a, b):
        return a + b

class demo02(demo01):
    def printDeom():
        return True

x = demo02()
print(x.addInt(1, 2))

# weikevin @ Kevin-MacBook in ~/Documents [21:25:00]
$ python3 test.py
3
```

Python class

- 複寫

```
# weikevin @ Kevin-MacBook in ~/Documents [21:08:54]
$ cat test.py
class demo01():
    def addInt(self, a, b):
        return a + b

class demo02(demo01):
    def addInt(self, a, b):
        print("{} add {}".format(a, b))
        return a + b

    def printDemo(self):
        return "I am good guy"


x = demo02()
print(x.addInt(1, 2))

# weikevin @ Kevin-MacBook in ~/Documents [21:09:00]
$ python3 test.py
1 add 2
3
```

Python class

- 添加

```
# weikevin @ Kevin-MacBook in ~/Documents [21:02:25]
$ cat test.py
class demo01():
    def addInt(self, a, b):
        return a + b

class demo02(demo01):
    def printDemo(self):
        return "I am good guy"


x = demo02()
print(x.addInt(1, 2))
print(x.printDemo())

# weikevin @ Kevin-MacBook in ~/Documents [21:04:25]
$ python3 test.py
3
I am good guy
```

Python class

- super

```
$ cat test.py
class demo01():
    def __init__(self, a):
        self.inta = a

class demo02(demo01):
    def __init__(self, ad, op):
        super().__init__(ad)
        self.intb = op

x = demo02(3, 5)
print(x.inta)
print(x.intb)

# weikevin @ Kevin-MacBook in ~/Documents [21:57:36]
$ python3 test.py
3
5
```

Python class

- 特殊使用
  - __eq__():
  - 官方文件

```
# weikevin @ Kevin-MacBook in ~/Documents [15:52:12]
$ cat test.py
class demo01():
    def __init__(self, a):
        self.inta = a

    def __eq__(self, classB):
        return self.inta == classB.inta


x = demo01(3)
b = demo01(3)
print(x == b)

# weikevin @ Kevin-MacBook in ~/Documents [15:52:16]
$ python3 test.py
True
```

Python class

- 特殊用法
  - __ne__ !=
  - __lt__ <
  - __gt__ >
  - __le__ <=
  - __ge__ >=

Python class

- Property
  - Python 無需使用 getter 以及 setter
  - 不過仍可以達成相同功能
  - @ property
  - @<func>.setter
  - 可以達成唯讀效果

Python class

```
# weikevin @ Kevin-MacBook in ~/Documents [17:48:11]
$ cat test.py
class demo01():
    def __init__(self, radius):
        self.radius = radius
        self.meter = 2 * radius

    @property
    def diameter(self):
        return self.meter

    @diameter.setter
    def diameter(self, dia):
        self.meter = dia


x = demo01(3)
print("diameter {}".format(x.diameter))
x.diameter = 50
print("diameter {}".format(x.diameter))
```

```
# weikevin @ Kevin-MacBook in ~/Documents [17:48:15]
$ python3 test.py
diameter 6
diameter 50
```

Python class

```
# weikevin @ Kevin-MacBook in ~/Documents [17:51:58]
$ cat test.py
class demo01():
    def __init__(self, radius):
        self.radius = radius
        self.meter = 2 * radius

    @property
    def diameter(self):
        return self.meter

x = demo01(3)
print("diameter {}".format(x.diameter))
x.diameter = 50
print("diameter {}".format(x.diameter))
```

Python class

```
# weikevin @ Kevin-MacBook in ~/Documents [17:51:36]
$ python3 test.py
diameter 6
Traceback (most recent call last):
  File "test.py", line 12, in <module>
    x.diameter = 50
AttributeError: can't set attribute
```

Generator & Decorator

- Python tool for special case
- Generator 產生器
  - Python 序列建立物件
  - range()
- Decorator 裝飾器
  - 接收一個函式，並回傳另一個函式

Generator

- 用途
  - Function 重複呼叫，紀錄上一次動作
  - 減少記憶體用量
- 使用
  - 不使用 return
  - 利用 yield
    參考: **What does the “yield” keyword do?**

Generator

```
$ python3
Python 3.6.0 (v3.6.0:41df79263a11, Dec 22 2016, 17:23:13)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> def demoFunc():
...     demoA = 1
...     demoB = 2
...     yield demoA
...     demoC = 3
...     yield demoB
...     yield demoC
...
>>> generator = demoFunc()
>>> print(generator.__next__())
1
>>> print(generator.__next__())
2
>>> print(generator.__next__())
3
>>>
```

Decorator

- Decorator 可以是 Function 也可以是 Class
- Decorator 可以帶入參數
  - 可以參考
  - Python Decorator 四種寫法範例 Code
- 做 Function 的修飾用途
  - 預先處理函式
  - 以及後續處理
  - 包裝 Python 函式

Decorator

```
def decorateDemo(func):
    def demoFunc(*args, **kargs):
        print("Running function:{}".format(func.__name__))
        print("Positional arguments:{}".format(args))
        print("Keyword arguments:{}".format(kargs))
        result = func(*args, **kargs)
        print("Result:{}".format(result))
        return result
    return demoFunc

@decorateDemo
def printDemo(a, b):
    return a + b

printDemo(3, 5)
```

```
$ python3 test.py
Running function:printDemo
Positional arguments:(3, 5)
Keyword arguments:{}
Result:8
```

Python PEP8

- Coding style in Python
- PEP 8 is set that contains rules of best coding practices that needs to be followed. If those practices and conventions are not followed, PEP 8 gives errors
  - Quora
- 程式的閱讀次數遠高於撰寫次數
  - 因此程式更應該專注於閱讀容易度

What is PEP8

PEP8 is SHIT !!

What is PEP8

- 繁瑣且要求高
  - 空行 空白
  - 斷行
  - 一行字數限制
  - import 規格

How use PEP8

- pip3 install pycodestyle

```
$ pip3 install pep8
Collecting pep8
  Downloading pep8-1.7.1-py2.py3-none-any.whl (41kB)
    100% |████████████████████████████████| 51kB 386kB/s
Installing collected packages: pep8
Successfully installed pep8-1.7.1
```

- pycodestyle << python file >>

```
# weikevin @ Kevin-MacBook in ~/Documents [17:13:53]
$ pycodestyle test.py
test.py:6:1: E305 expected 2 blank lines after class or function definition, found 1
test.py:8:1: W391 blank line at end of file
```

How use PEP8

- 空行
  - 兩行
    - 頂層物件
    - 頂層函式
  - 一行
    - 分隔類別內的 method
- Import
  - 應該分開寫
  - 官方函式庫 > 第三方函式庫 > 專案內函式

How use PEP8

- 每行最大長度
  - 限制 79 個字符長度
  - 文檔字符 或 註解 限制 72 個字符長度
- 換行方式
  - 利用 Python 小括號、中括號和大括號
  - 使用反斜槓

Python Library

- Standard Library
  - https://docs.python.org/3/library/ (英文 3.6.4)
  - https://docs.python.org.tw/3/library/ (中文 3.5.2)
- Standard library
  - os, sys, configparser, logging, getopt, json, unittest
- The third part library
  - requests, numpy

PYTHON LIBRARY

Python3 -v

- 詳細模式
- demo
  - 找 this module
  - python3 –v
  - import this
  - Find
    # /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/__pycache__/this.cpython-36.pyc matches /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/this.py
  - Cat
    /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/this.py

The Zen of Python

```
Hayrff rkcyvpvgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)

print("".join([d.get(c, c) for c in s]))
```

Outline

- os
- configparser
- logging
- getopt
- json
- requests
- numpy
- Install python3
- pipenv

os

- os.path.exists(*path*)
- 確認檔案或目錄是否存在
  - 其可以是相對或絕對路徑
  - 官方文件

- os.path.isfile(*path*)
- os.path.isdir(*path*)
- 檢查 path 下是否是檔案或是目錄
  - 其可以是相對或絕對路徑
  - 官方文件

os

- os.rename(*src, dst*)
- 將 src 改名為 dst
  - 官方文件

- os.remove(*path*)
- 刪除檔案
  - 官方文件

os

- os.chmod(*path, mode*)
- 變更權限
  - *mode* 八進位(0o400)，或是 stat.S_IRUSR
  - 官方文件
- os. chown(*path, uid, gid*)
- 變更擁有者
  - Unix/Linux/Mac 獨有
  - *uid, gid* 使用者ID數字，群組ID數字
  - 官方文件

os

- os.mkdir(*path*)
  - 建立目錄
- os.rmdir(*path*)
  - 刪除目錄
- os.listdir(*path*)
  - 列出所有目錄
- os.chdir(*path*)
  - 變更目前目錄

configparser

- 解析 INI 檔案
  - INI file
    Informal standard for configuration files
    Simple text files with sections, properties, and values
- 內建函式庫
- 官方文件

configparser

```
; last modified 1 April 2001 by John Doe
[owner]
name=John Doe
organization=Acme Widgets Inc.

[database]
; use IP address in case network name resolution is not working
server=192.0.2.62
port=143
file="payroll.dat"
```

configparser

```python
import configparser

cfg = configparser.ConfigParser()
cfg.read('config.ini')
print(cfg)
print(cfg['owner'])
print(cfg['owner']['name'])
```

```
# weikevin @ Kevin-MacBook in ~/Documents/demo [18:03:32]
$ python main.py
<backports.configparser.ConfigParser object at 0x1033daad0>
<Section: owner>
John Doe
```

logging

- 不要使用 print()
- Logging Level
  - CRITICAL
  - ERROR
  - WARNING
  - INFO
  - DEBUG
  - NOTSET

logging

```
# weikevin @ Kevin-MacBook in ~/Documents/logging [10:39:25]
$ cat main.py
import logging

FORMAT = '%(asctime)-15s %(user)-8s %(message)s'
logging.basicConfig(level=logging.INFO, filename='demo.log', format=FORMAT)

info = {'user': 'weicc'}
logging.warning('Hello world!', extra=info)
logging.info('Hello world again!',  extra=info)

# weikevin @ Kevin-MacBook in ~/Documents/logging [10:39:45]
$ cat demo.log
2018-03-06 10:39:15,949 weicc    Hello world!
2018-03-06 10:39:15,949 weicc    Hello world again!
```

getopt

- 外部參數
  - 可以用 sys 處理
  - getopt 表現更好
- 內建函式庫

```
import getopt
import sys

output = ''
vercode = False

print('ARGV      :', sys.argv[1:])

try:
    options, remainder = getopt.getopt(
        sys.argv[1:],
        'o:v', ['output=', 'vercode']
    )
except getopt.GetoptError as err:
    print('ERROR:', err)
    sys.exit(1)

print('OPTIONS   :', options)

for opt, arg in options:
    if opt in ('-o', '--output'):
        output = arg
    elif opt in ('-v', '--vercode'):
        vercode = True

print('OUTPUT    :', output)
print('VERCODE   :', vercode)
```

getopt

```
admin@ubuntu:~$ python3 getopts.py --vercode
ARGV      : ['--vercode']
OPTIONS   : [('--vercode', '')]
OUTPUT    :
VERCODE   : True
admin@ubuntu:~$ python3 getopts.py -oanything --vercode
ARGV      : ['-oanything', '--vercode']
OPTIONS   : [('-o', 'anything'), ('--vercode', '')]
OUTPUT    : anything
VERCODE   : True
admin@ubuntu:~$
```

json

- JavaScript Object Notation
  - human-readable text
- json and simplejson
  - json was added in 2.6

```
try:
    import simplejson as json
except ImportError:
    import json
```

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234  “
    },
    {
      "type": "fax",
      "number": "646 555-4567  “
    }
  ],
  "gender": {
    "type": "male  “
  }
}
```

json

- json.dumps

```
1   try:
2       import simplejson as json
3   except ImportError:
4       import json
5
6   demo = {
7       'name': 'kevin',
8       'lastname': 'wei'
9       }
10
11  print(json.dumps(demo, indent=4))
```

```
# weikevin @ Kevin-MacBook in ~/Documents/demo [17:57:53]
$ python3 demo.py
{
    "name": "kevin",
    "lastname": "wei"
}
```

json

- json.loads

```
1    try:
2        import simplejson as json
3    except ImportError:
4        import json
5
6    demo = json.dumps({"firstName": "John", "lastName": "Smith", "age": 25})
7
8    print(json.loads(demo))
```

```
# weikevin @ Kevin-MacBook in ~/Documents/demo [18:02:44]
$ python3 demo.py
{'firstName': 'John', 'lastName': 'Smith', 'age': 25}
```

json

- json and yaml

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234 “
    },
    {
      "type": "fax",
      "number": "646 555-4567 “
    }
  ],
  "gender": {
    "type": "male “
  }
}
```

```yaml
firstName: John
lastName: Smith
age: 25
address:
    streetAddress: 21 2nd Street
    city: New York
    state: NY
    postalCode: '10021'
phoneNumber:
  - type: home
    number: 212 555-1234
  - type: fax
    number: 646 555-4567
gender:
    type: male
```

json

- json and XML

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234 “
    },
    {
      "type": "fax",
      "number": "646 555-4567 “
    }
  ],
  "gender": {
    "type": "male “
  }
}
```

```
<person>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <age>25</age>
    <address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
    </address>
    <phoneNumber>
        <type>home</type>
        <number>212 555-1234</number>
    </phoneNumber>
    <phoneNumber>
        <type>fax</type>
        <number>646 555-4567</number>
    </phoneNumber>
    <gender>
        <type>male</type>
    </gender>
</person>
```

requests

- http protocol
  - get, post, put, delete
  - Document example: OpenStack Identity API
- Google docs API
  - Document
- 官方文件

requests

Request

Parameters

Name	In	Type	Description
nocatalog (Optional)	query	string	(Since v3.1) The authentication response excludes the service catalog. By default, the response includes the service catalog.
domain	body	object	A **domain** object, containing:
name (Optional)	body	string	The user name. Required if you do not specify the ID of the user. If you specify the user name, you must also specify the domain, by ID or name.
auth	body	object	An **auth** object.
user	body	object	A **user** object.
password	body	object	The **password** object, contains the authentication information.
id (Optional)	body	string	The ID of the user. Required if you do not specify the user name.
identity	body	object	An **identity** object.
methods	body	array	The authentication method. For password authentication, specify **password**.

Example

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "admin",
                    "domain": {
                        "name": "Default"
                    },
                    "password": "devstacker"
                }
            }
        }
    }
}
```

requests

Status Codes

Success

Code	Reason
201 - Created	Resource was created and is ready to use.

Error

Code	Reason
400 - Bad Request	Some content in the request was invalid.
401 - Unauthorized	User must authenticate before making a request.
403 - Forbidden	Policy does not allow current user to do this operation.
404 - Not Found	The requested resource could not be found.
405 - Method Not Allowed	Method is not valid for this endpoint.
409 - Conflict	This operation conflicted with another operation on this resource.
413 - Request Entity Too Large	The request is larger than the server is willing or able to process.
415 - Unsupported Media Type	The request entity has a media type which the server or resource does not support.
503 - Service Unavailable	Service is not available. This is mostly caused by service configuration errors which prevents the service from successful start up.

Example


```
{
    "token": {
        "methods": [
            "password"
        ],
        "expires_at": "2015-11-06T15:32:17.893769Z",
        "user": {
            "domain": {
                "id": "default",
                "name": "Default"
            },
            "id": "423f19a4ac1e4f48bbb4180756e6eb6c",
            "name": "admin",
            "password_expires_at": null
        },
        "audit_ids": [
            "ZzZwkUflQfygX7pdYDBCQQ"
        ],
        "issued_at": "2015-11-06T14:32:17.893797Z"
    }
}
```

requests

- Code sample
- Github code
  - import json
  - import requests
  - from lxml import etree

numpy

- mathematical functions library
  - for large, multi-dimensional arrays and matrices
  - with a large collection of high-level mathematical functions
- ndarray data
  - n-dimensional array
  - ndim(維度), shape(形狀), dtype(數值類型)

numpy

```
admin@ubuntu:~/numpy$ pipenv run python3 demo.py
[0 1 2 3]
c.ndim    :1
c.shape   :(4,)
admin@ubuntu:~/numpy$ cat demo.py
import numpy as np

c = np.array(
    [0, 1, 2, 3]
)

print(c)
print("c.ndim    :{}".format(c.ndim))
print("c.shape   :{}".format(c.shape))
```

numpy

```
admin@ubuntu:~/numpy$ pipenv run python3 demo.py
[[ 0  1  2  3]
 [10 11 12 13]]
c.ndim    :2
c.shape   :(2, 4)
admin@ubuntu:~/numpy$ cat demo.py
import numpy as np

c = np.array(
    [
        [0, 1, 2, 3],
        [10, 11, 12, 13]
    ]
)

print(c)
print("c.ndim    :{}".format(c.ndim))
print("c.shape   :{}".format(c.shape))
```

numpy

- numpy and pandas and matplotlib
- Pandas

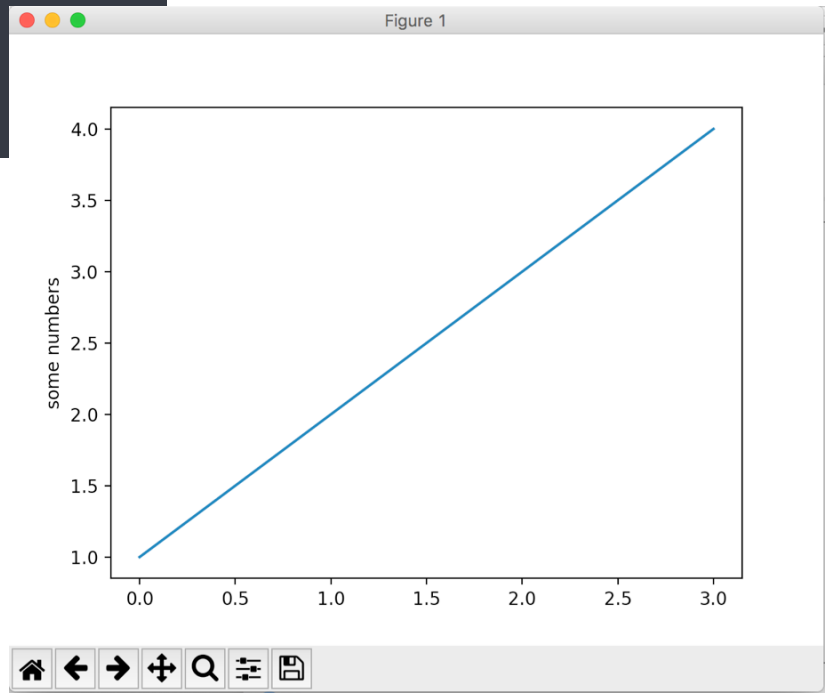
```
# 讀取 CSV File
import pandas as pd
df = pd.read_csv('shop_list.csv')
print(df)
```

```
# 讀取 HTML
import pandas as pd
dfs = pd.read_html('http://rate.bot.com.tw/xrt?Lang=zh-TW')
dfs[0]
```

numpy

- Numpy and Pandas and matplotlib
- matplotlib

```
1    import matplotlib.pyplot as plt
2    plt.plot([1, 2, 3, 4])
3    plt.ylabel('some numbers')
4    plt.show()
```



Install python3

- On FreeBSD11.1
  - pkg install python36
  - pkg install py36-pip
  - pip-3.6 install <library>
- On Linux (Ubuntu, CentOS)
  - apt install python3 (default: 3.5.2)
  - pip3 install <library>

  - sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm
  - IUS mean Inline with Upstream Stable
  - sudo yum -y install python36u
  - sudo yum -y install python36u-pip

pipenv

- Python virtualenv
- 過多專案，不同環境
  - 套件版本差異
  - 套件亂裝
  - brew install, anaconda
  - 版本釋出的時候不知道裝了哪些套件
- 虛擬環境可以開出一個乾淨的 Python 環境

pipenv

- On Linux, FreeBSD
  - pip3 install pipenv (Linux)
  - pip-3.6 install pipenv (FreeBSD)
- On MacOS
  - brew install pipenv
- Create a new project
  - mkdir <demoproject>
  - cd <demoproject>
  - pipenv install <library>
  - pipenv run python3 <file>.py

pipenv

- Pipfile

```
admin@ubuntu:~/numpy$ ls -al
total 20
drwxrwxr-x  2 admin admin 4096 Mar  8 04:23 .
drwxr-xr-x 11 admin admin 4096 Mar  8 04:23 ..
-rw-rw-r--  1 admin admin  164 Mar  8 04:05 Pipfile
-rw-rw-r--  1 admin admin 2659 Mar  8 04:05 Pipfile.lock
-rw-rw-r--  1 admin admin  183 Mar  8 04:23 demo.py
admin@ubuntu:~/numpy$
```

pipenv

- Pipfile

```
admin@ubuntu:~/numpy$ cat Pipfile
[[source]]

url = "https://pypi.python.org/simple"
name = "pypi"
verify_ssl = true


[packages]

numpy = "*"


[dev-packages]



[requires]

python_version = "3.5"
admin@ubuntu:~/numpy$
```

pipenv

- pipenv run

```
admin@ubuntu:~/numpy$ pipenv run python3 demo.py
[[ 0  1  2  3]
 [10 11 12 13]]
c.ndim     :2
c.shape    :(2, 4)
admin@ubuntu:~/numpy$
```

- pipenv install

```
admin@ubuntu:~/numpy$ pipenv install requests
Installing requests…
Collecting requests
  Using cached requests-2.18.4-py2.py3-none-any.whl
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Using cached chardet-3.0.4-py2.py3-none-any.whl
Collecting idna<2.7,>=2.5 (from requests)
  Using cached idna-2.6-py2.py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests)
  Using cached certifi-2018.1.18-py2.py3-none-any.whl
Collecting urllib3<1.23,>=1.21.1 (from requests)
  Using cached urllib3-1.22-py2.py3-none-any.whl
Installing collected packages: chardet, idna, certifi, urllib3, requests
Successfully installed certifi-2018.1.18 chardet-3.0.4 idna-2.6 requests-2.18.4 urllib3-1.22

Adding requests to Pipfile's [packages]…
Pipfile.lock (d7c22f) out of date, updating to (8704cb)…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (8704cb)!
Installing dependencies from Pipfile.lock (8704cb)…
  🐍   6/6 —
To activate this project's virtualenv, run the following:
 $ pipenv shell
```

THANK YOU !