Tutorial 1
Basic Python

Kai CHEN

Python (part I)

What can Python do

Tasks		Libraries
General	Text processing	Standard libraries
	Web service	Flask, django, …
	Web crawler	requests, BeautifulSoup, Scrapy, ...
	GUI	Tkinter, wxWidgets, PyQt, …
Academic	Scientific computation	numpy, scipy, matplotlib, …
	Image processing	Pillow, OpenCV, …
	Natural language processing	NLTK, spaCy, CoreNLP, …
	Machine learning	PyTorch, scikit-learn, …

Python (part I)

- Easy to get started
- Rich libraries
- Productive

- Speed

Python (part I)

Interactive shell

```
$ python3
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
[Clang 9.1.0 (clang-902.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1.2
>>> b = 3 / 2.0
>>> max(a, b)
1.5
>>> a ** b
1.3145341380123985
>>>
```

Python (part I)

Python script

test.py

```
 1 # -*- coding: utf-8 -*-
 2
 3 import os
 4
 5
 6 def read_file(filename):
 7     if os.path.isfile(filename):
 8         with open(filename, 'r') as fin:
 9             content = fin.read()
10         print(content)
11     else:
12         print('file not exist')
13
14
15 if __name__ == '__main__':
16     read_file('in.txt')
17
```

Encoding statement

Import modules/packages

Method definition

A little complex top-level script

Call a method

```
$ python test.py
hello world!
```

Python (part I)

Built-in types – numeric types

int, long, float, complex

```
>>> a = 5
>>> a / 2
2.5
>>> a % 2
1
>>> b = -a * 2.0
>>> b
-10.0
>>> abs(b)
10.0
>>> int(b)
-10
>>> max(a, b)
5
>>> round(2.7)
3
```

Different behavior in Python 2/3

+, -, *, /, //, %, **, bitwise operations

Python (part I)

Built-in types – sequence types

str

```
>>> a = 'hello'
>>> len(a)
5
>>> a[0]
'h'
>>> a[2:4]
'll'
>>> a[-1]
'o'
>>> a.upper()
'HELLO'
>>> a + ' world'
'hello world'
```

list

```
>>> a = ['ok', 3.0, [1]]
>>> len(a)
3
>>> a[1]
3.0
>>> a[2].append('string')
>>> a
['ok', 3.0, [1, 'string']]
>>> a.reverse()
>>> a
[[1, 'string'], 3.0, 'ok']
>>> 'ok' in a
True
```

tuple

```
>>> a = ('lily', 10)
>>> a[1]
10
>>> a[1] = 12
Traceback (most recent call
last):
  File "<stdin>", line 1, in
 <module>
TypeError: 'tuple' object do
es not support item assignme
nt
>>> len(a)
2
>>> list(a)
['lily', 10]
```

immutable vs mutable

Python (part I)

Built-in types – mapping types

dict

```
>>> a = {'name': 'MyName', 'age': 18, 'scores': [100, 90]}
>>> len(a)
3
>>> a['name']
'MyName'
>>> 'scores' in a
True
>>> a['good'] = True
>>> a
{'age': 18, 'good': True, 'name': 'MyName', 'scores': [100, 90]}
>>> b = dict(test=a)
>>> b['test']
{'age': 18, 'good': True, 'name': 'MyName', 'scores': [100, 90]}
>>> a.keys()
['age', 'good', 'name', 'scores']
>>> a.items()
[('age', 18), ('good', True), ('name', 'MyName'), ('scores', [100, 90])]
```

Python (part I)

Built-in types – others

- Set
- Ordereddict
- defaultdict
- …

Python (part I)

Control flow

If statement

```
if a > 0:
    print('a is positive')
elif a < 0:
    print('a is negative')
else:
    print('a is 0')
```

Conditional operator?

```
a = b if b >= 0 else -b
```

switch-case?

Python (part I)

Control flow

Iterate over a list

```
word_list = ['a', 'b', 'c']

for word in word_list:
    print(word.upper())
```

Iterate over a dict

```
word_freq = {'hello': 2, 'food': 1}

for word in word_freq:
    print(word, word_freq[word])
```

Iterate over a range

```
for i in range(10):
    if i < 5:
        print(i)
    elif i < 8:
        print(i+1)
    else:
        break
```

Python (part I)

Method

```
def sum(nums, bias=0):
    total = bias
    for num in nums:
        total += num
    return total
```

Method name

Required arguments

Optional arguments with default value

Return value

Python (part I)

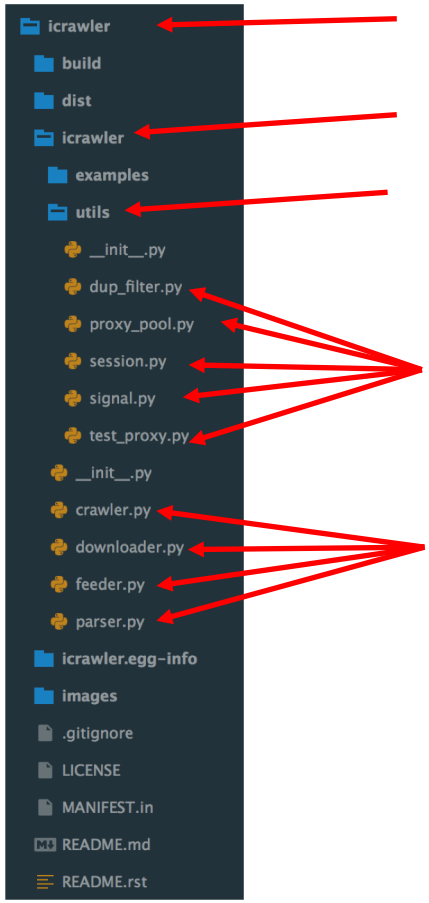
Class

```
class Test(object):

    def __init__(self, base=''):
        self.base = base
        self.string = ''

    def concat(self, extra_str):
        if isinstance(extra_str, str):
            self.string += extra_str
        elif isinstance(extra_str, list):
            self.string += ''.join(extra_str)
        else:
            pass
```

```
test = Test('origin')
test.concat('string')
test.concat(['string', 'list'])
print(test.string)
```

Python (part I)

Module/Package and imports



Project folder

Package

Package

Module

Module

```
import package1
import package2.module1
from module2 import ClassA, func1
from package2.module2 import ClassB, func2
```

Python (part I)

Code style

https://www.python.org/dev/peps/pep-0008/

Python 2/3

- https://wiki.python.org/moin/Python2orPython3
- http://pythonhosted.org/six/

Virtual environment

- https://virtualenv.pypa.io/en/stable/
- https://docs.python.org/3/library/venv.html

Python (part II)

Jupyter notebook

Features

- In-browser editing for code
- execute code from the browser
- Displaying the result using rich media representations
- …

Python (part II)

Jupyter notebook

- Installation

  pip3 install jupyter

- Run

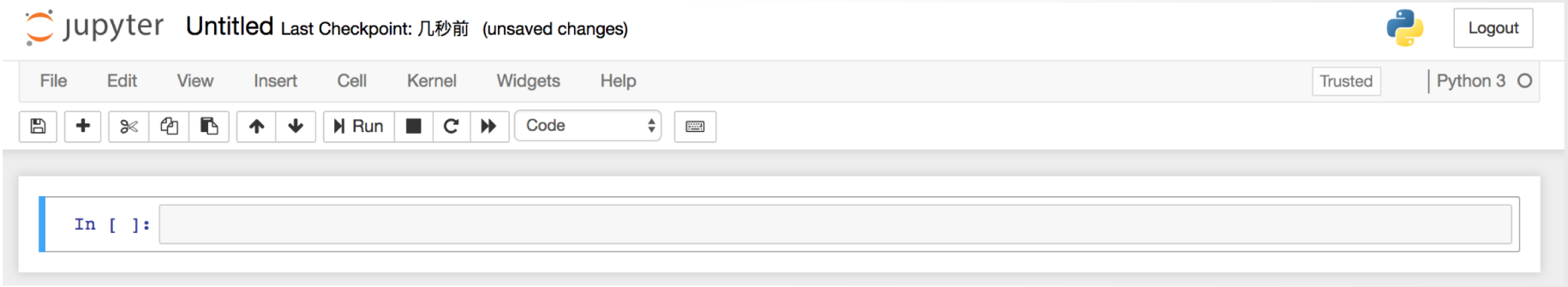
  jupyter notebook

Python (part II)

Jupyter notebook

Python (part II)

Jupyter notebook



Python (part III)

More topics for those who are interested

- List comprehensions
- Exception handling
- Decorator

Python (part III)

List comprehensions

```
filtered_words = []
for word in words:
    if len(word) > 3:
        filtered_words.append(word)

filtered_words = [word for word in words if len(word) > 3]
```

```
freq = {}
for key, val in word_freq.items():
    if val > 2:
        freq[key] = val

freq = {key: val for key, val in word_freq.items() if val > 2}
```

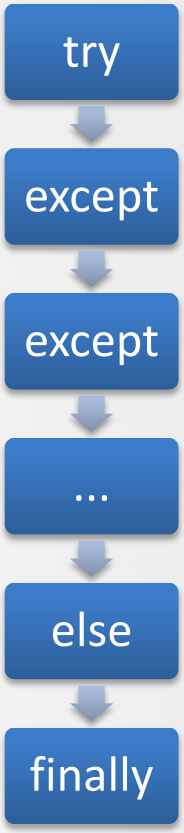
Python 2: items(), iteritems()
Python 3: items()

Python (part III)

Exception handling

Example from official documentation

```
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except IOError as e:
    print "I/O error({0}): {1}".format(e.errno, e.strerror)
except ValueError:
    print "Could not convert data to an integer."
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```



Python (part III)

Exception handling

Raise exceptions

```
def birth_year(year):
    if not isinstance(year, int):
        raise TypeError('year must be an integer')
    elif year > 2016 or year < 1900:
        raise ValueError('birth year must be between 1900 and 2016')
    else:
        print('birth year {}'.format(year))
```

Python (part III)

Decorator

- Widely used in Flask (and other libraries)
- Make simple modifications to callable objects

Python (part III)

Decorator

```
def hello_world():
    print('hello world')


func = hello_world
func()
```

Print calling information

```
def hello_world():
    print('hello world')


func = hello_world
print('call function {}'.format(func.__name__))
func()
```

Python (part III)

Decorator

```
def hello_world():
    print('hello world')


def log(func):
    def wrapper(*args, **kwargs):
        print('call function {}()'.format(func.__name__))
        return func(*args, **kwargs)
    return wrapper


new_hello_world = log(hello_world)
new_hello_world()
```

A method that returns a method

Python (part III)

Decorator

```
def log(func):
    def wrapper(*args, **kwargs):
        print('call function {}()'.format(func.__name__))
        return func(*args, **kwargs)
    return wrapper


@log
def hello_world():
    print('hello world')


hello_world()
```

Python (part III)

Decorator

hello_world.__name__ # outputs: warpper

```
import functools


def log(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        print('call function {}()'.format(func.__name__))
        return func(*args, **kwargs)
    return wrapper
```

Python (part III)

Decorator

Decorator with parameters?
like @app.route('/', mehtods=['GET', 'POST'])

Python (part III)

And more topics

- Iterators / Generators
- Lambda Functions
- Context manager
- Logging
- Map, filter, reduce
- Write a package
- Conda
- …

Thank you!