# Python 学习笔记

郑华

2018 年 5 月 29 日

# 目录

# 第一章　基本

## 1.1　参考

菜鸟教程：http://www.runoob.com/python3/python3-basic-syntax.html

## 1.2　关键字

常量　True False None

对象与容器　class import from del

逻辑操作　and or not

函数　def return

判断与循环’:’　if elif else is in assert for while continue break

异常　raise try except finally with as

作用域　global nonlocal

匿名函数与协程　yield lambda

## 1.3 容器与字符串

### 1.3.1 list 数组

用法示例 ->

```
# 定义
li = [1,2,3,4,5]

#遍历
for i in li:
    print(i)

#使用range
#range(x) -> [0, x-1]
#range(x,y) -> [x, y-1]
#range(x,y,z) ->[x, y-1]间隔为z -> [x, x+z, x+2z, ..]
for i in range(len(li))
    print(i)

#负数索引: 倒数第多少
print(li[-1])

li2 = []
#添加元素
li2.append(1)
li2.append('123')
li2.append(['d','ef','a'])

#合并两个数组
li.extend(li2)

#删除元素
li2.pop()
li2.pop(2) #删除索引位置

#排序
li2.sort()
def item_key(x):
    return x[0]
li2.sort(key = item_key)
li2.sort(key = lambda x:x[0]) #与上述等价
```

### 1.3.2 tuple 只读数组

```
tp = (1,2,3)
```

### 1.3.3 set 没用重复元素的数组

```
# 使用list 初始化
st = set([1,2,3,4,2,3,5])
# 自动去重并排序
# 使用tuple 初始化
st = set((2,2,3,4,5))
```

### 1.3.4 dict 字典 map

```
# 空子典
di = {}
di = {'k1':'v1','k2':'v2'}
di['k3'] = 'v3'

for k in di:
    print(di[k])

for k,v in di.items():
    print(k,v)
```

### 1.3.5 数组切片

```
#[start, end, step] -> [start, start+step, ...<end]
#start 默认等于0 #end 默认等于-1 #step 默认等于1
li = [1,2,3,4,5]
li_0_2 = li[0:3] #-> li[0] li[1] li[2] ->i>=start < end

# 直接用切片反转数组,切片是复制操作
print(li[::-1])
```

### 1.3.6 字符串与数组的关系

默认的字符串相当于 c++ 里面的字符串常量，需要利用 list(str) 将其转化成字符数组进行操作。

然后通过''.join(list) 将 list 转化为 string

切割 -

```
s = 'abc,def,ghi'
p1,p2,p3 = s.split(',')
print(p1,p2,p3)
```

切片 同list

## 1.4 函数

定义函数 ->

```
def func():
    pass
```

默认参数

名字参数

函数可以当成变量

函数式编程 **map/reduce**

## 1.5 面向对象

**type(parameter)** 查看对象类型

### 1.5.1 定义

**绑定属性** `__init__` 绑定属性, 并且第一个参数是 self 且不能忘记

```
class Clazz(object):
    def __init__(self, x, y)
        self.x = x
        self.y = y
```

**实例化** `clz = Clazz(100,200)`

### 1.5.2 this -> self

self 等价于 this 指针

### 1.5.3 多态与继承

```
class Base:
    def run(self):
        print('Base:run')

class Tom(Base):
    def run(self):
        print('Tom:run')

t = Tom()
t.run()
```

## 1.6 文件读写

### 1.6.1 文本文件读写

```
f = open('text.txt','r')
# 打印整个文件
print(f.read())
# 按行打印
for line in f.readlines():
    print(line)
```

```
# 异常 处理
with open('text.txt') as f:
    for line in f.readlines():
        print(line)
```

### 1.6.2 二进制文件读写

### 1.6.3 string 与 bytes

### 1.6.4 文件和目录操作

## 1.7 多线程

### 1.7.1 多线程

线程与进程消耗的是一样的在 Python 里

```
# 引入线程包
import threading

# 定义线程函数
def thread_func(x):
    print('%d\n',%(x*100))

# 添加线程
threads = []
for i in range(5)
    threads.append( threading.Thread(target = thread_func, args = (100,)) )

# 执行线程
for thread in threads:
    thread.start()

# 等待线程结束
for thread in threads:
    thread.join()
```

### 1.7.2 多进程

### 1.7.3 线程同步

## 1.8 错误与异常

基础使用 ~

```
try:
    r = 10/0
# 捕获具体类型的异常
except ZeroDivisionError as e:
    print(type(e))
    print(e)
finally:
    # 防止资源泄露
    print('always␣come␣here')
```

**logging 库** ~

```
import logging


logging.info('xx')
logging.debug('xx')
```

# 第二章　爬虫

## 2.1　基础

**安装第 3 方包**　`pip install package`

### 2.1.1　HTTP 简介

**概念**

- Http = hyperText Transfer Protocol
- URI = Uniform Resource Identifier
- URL = Uniform Resource Locator
- URI 和 URL 的区别：URI 强调的是资源，URL 强调的是资源的位置。

**请求类型**

- `OPTIONS` ：返回服务器针对特定资源所支持的 http 请求方法
- `HEAD` ：向服务器索要与 get 请求相一致的响应，只不过响应体将不会返回。只是确定文件是否存在。
- `GET` ：向特定资源发出请求
- `PUT` ：向指定资源位置上传其最新内容
- `POST` ：向指定资源提交数据进行处理请求
- `DELETE` ：请求服务器删除特定 URI 所标识的资源
- `PATCH` ：用于将局部修改应用于某一资源

常见状态码

- 200/OK ：请求成功
- 201/Created: 请求已被实现，且一个新资源已根据请求被建立，URI 跟随 Location 头信息返回。
- 202/Accepted: 服务器已接受请求，但尚未处理
- 400/Bad Request: 请求无法被服务器理解
- 401/Unauthorized: 当前请求需要用户验证
- 403/Forbidden: 服务器已理解请求，但是拒绝执行
- 404/Not Found: 未发现

### 2.1.2 HTML/XML/Json 简介

Html: 标签

Xml: 树结构

Json(JavaScript Object Notation): 类似与 XML，但是更小、更快、解析更容易。

### 2.1.3 MySQL/SqlLite

### 2.1.4 爬虫框架

- 将种子 URL 放入队列
- 从队列中获取 URL，抓取内容
- 解析抓取内容，将需要进一步抓取的 URL 放入工作队列，存储解析后的内容

抓取策略

- 深度优先
- 广度优先
- PageRank

- 大站优先

如何去重

- Hash 表
- bloom 过滤器

## 2.2 相关库的使用

## 2.3 Scrapy 及相关应用

## 2.4 爬虫设计实战

## 2.5 高级内容-并发编程

## 2.6 分布式爬虫框架设计