

## 复盘总结

### 1、首先认识了 python 的框架结构



```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # @author: caramel
4
5 def main():
6
7
8
9 if __name__ == '__main__':
10     main()
```

Python 的运行脚本, 也可以省略不写

编码模式

自己添加自己的名字

这是 python 代码的运行框架, 其中可以添加不同的函数函数代码块, 用来实现不同的功能。

### 2、第一天的作业中其中一个作业示例:



```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # @author: caramel
4
5 def main():
6     who = '焦糖的妈妈'
7     good_price = 6 #小贩的价格
8     good_description = '西双版纳的大白菜' #小贩的招牌
9     is_cheap = False #是否便宜
10    reasonable_price = 5 #老妈能接受的最高价格
11    buy_amount_1 = 2 #准备买2斤
12    buy_amount_2 = 3 #准备买3斤
13    buy_amount_3 = 4 #准备买4斤
14    #开始秀
15    print "%s上街看到了 %s, 卖 %d 元/斤" % (who, good_description, good_price)
16
17    if good_price <= reasonable_price:
18        print '她认为便宜'
19        is_cheap = True
20        print '她买了 %d 斤' % (buy_amount)
21    else:
22        print '她认为贵了'
23        is_cheap = False
24        print '她和小贩开始砍价'
25
26        if good_price == 5:
27            print '她买了 %d 斤' % (buy_amount_1)
28
29 if __name__ == '__main__':
30     main()
```

用 # 来标示代码所表示的作用; = 用来赋值, 把右面的东东赋值给左面 (自定义的名字); “ ” 收用来记录字符串。

### 3、对 python 的一些简单的概念有了基本的了解;

- 1) 数字 (Number)
- 2) 字符串

- 3) 运算符
- 4) If 条件语句
- 5) for 循环, while 循环, 循环嵌套
- 6) break 语句, continue 语句, pass 语句

4、几个重要的函数:

- 1) 列表 list: 列表的引入, 即与简答的代码块比较来引入

```
5 ▼ def main():
6     good_1 = 'cabbage'
7     good_2 = 'tomato'
8     good_3 = 'cucumber'
9     good_4 = 'bean'
10    good_5 = 'carrot'
11
12    #-----One hundred items are omitted.
13    good_100 = 'potato'
14
15    print 'My mum saw %s in the vegetable market.' % (good_1)
16    print 'My mum saw %s in the vegetable market.' % (good_2)
17    print 'My mum saw %s in the vegetable market.' % (good_3)
18    print 'My mum saw %s in the vegetable market.' % (good_4)
19    print 'My mum saw %s in the vegetable market.' % (good_5)
20
21 ▼ def main2():
22     goods = 'cabbage, tomato, cucumber, bean, potato'
23     print 'My mum saw %s in the vegetable market.' % (goods)
24
25 ▼ def main3():
26     print '-----'
27     lst = ['cabbage', 'tomato', 'cucumber', 'bean', 'potato']
28     for lst_item in lst:
29         print 'My mum saw %s in the vegetable market.' % (lst_item)
30
31 ▼ if __name__ == '__main__':
32     main()
33     main2()
34     main3()
```

创建一个列表, 只要把逗号分隔的不同的数据项使用方括号括起来即可, for 用来遍历列表。

序列都可以进行的操作包括索引, 取值, 长度, 添加, 删除, 切片, 需要注意格式。

```
▼ #记录下标
    index = 0
    for lst_item in lst:
        index = index + 1
#迭代访问
    for index, lst_item in enumerate(lst):
#取值
    print lst[2]
#长度
    print len(lst)
#加
    lst.append('猪肉')
#删除
    del lst[2]
▼ #切片
    lst2 = lst[0:3] #左开右闭区间, 从第二个到第四个记为[2:5]
    print_list(lst2)
```

## 2) 字典 dictionary

```
def homework1():
    dictionary = {'caramel': 'firm chewy candy made from caramelized sugar and butter and milk',
                  'is': 'the third person singular of the present tense of be ',
                  'handsome': 'very beautiful'}

    print len(dictionary) #长度

    #获取keys
    print dictionary.keys() #获得keys的列表,所有单词

    #获得values
    print dictionary.values() #获得values的列表,所有解释

    #是不是在
    print dictionary.has_key('caramel') #有, 输出结果为True
    print dictionary.has_key('bad') #没有, 输出结果为False

    #增加
    dictionary['bad'] = 'not very good'
    print dictionary
    print len(dictionary)

    #修改modify
    dictionary['bad'] = 'failing to reach an acceptable standard'

    #删除delete
    del dictionary['bad']
    print dictionary
    print len(dictionary)
```

字典的每个键值(key=>value)对用冒号(:)分割, 每个对之间用逗号(,)分割, 整个字典包括在花括号({})

字典可进行的操作: 长度, 过去 keys, 获取 value, 判断是否在字典中, 增加, 修改,

```
#for循环
print '-----'
print dictionary['caramel']
if dictionary.has_key('bad'):
    print dictionary['bad'] #这一段不执行
else:
    print '没有bad这个单词'
print '-----'

#for遍历列表
for key in dictionary.keys():
    print key #单个单词
    print dictionary[key] #相对应的解释
```

还可以进行 for 循环。

## 3) 元组 tuple

```
def main():
    tup = (1,2,3,4)
    #取值
    print tup[1]
    #切片
    print tup[0:3]
    print tup[2:]
    #是否存在某值
    print (1 in tup)
    print (5 in tup)
    #赋值
    a,b,c,d = tup
    print a
    print b
    print c
    print d
    #遍历
    for item in tup:
        print item
    #enumerate
    print '===== '
    for index, item in enumerate(tup):
        print index
        print item
    #花式报错
    #不支持 1)插入 2)修改 3)删除
```

Python 的元组与列表类似, 不同之处在于元组的元素不能修改。元组使用小括号, 列表使用方括号。元组创建很简单, 只需要在括号中添加元素, 并使用逗号隔开即可。

元组可进行的操作有: 取值, 切片, 是否存在某值, 赋值, 遍历, 但是不支持插入, 修改, 删除

#### 4、python 异常处理，try 一下

```
try:
    tup.append(9) #插入失败 'tuple' object has no attribute 'append'
    del tup[0]    #删除失败 'tuple' object doesn't support item deletion
    tup[0] = 'aaa' #修改失败 'tuple' object does not support item assignment
except Exception, e:
    print e
```

其中还有很多错误类型，需要在以后的学习过程中慢慢识别，积少成多。

#### 5、工具：

命令行工具 windows 的 power shell

一些 shell 的常用命令：

cd 改变路径

pwd 获取当前位置信息

ls 获取一个或多个指定位置的项目列表

man 寻求帮助

clear 清空屏幕

#### 6、类的引入：以 turtle 为例

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # @author: xxx
4  import turtle
5
6  def main():
7      #设置一个画面
8      windows = turtle.Screen()
9      #设置背景
10     windows.bgcolor('pink')
11     #生成一个黄色乌龟
12     bran = turtle.Turtle()
13     bran.shape('turtle')
14     bran.color('blue')
15     #设置速度
16     bran.speed(1)
17     bran.lt(60)
18     bran.fd(50)
19     bran.rt(60)
20     bran.fd(10)
21
22 if __name__ == '__main__':
23     main()
```

引入类，即站在已有函数的基础上方便快捷的调用函数，而不用再次自己编写代码块。

进入网站 <http://devdocs.io/> ，学会查询 turtle 的其他用法

## 7、面向对象

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # 什么是面向对象
4
5  class Car():
6      def __init__(self, brand, speed):
7          self.brand = brand
8          self.speed = speed
9
10     def be_drived(self, distance):
11         hour = float(distance)/self.speed
12         print ' 驾驶 %s 开 %dkm 去b地开了 %.2f 小时' % (self.brand, distance, hour)
13
14     def main():
15         car1 = Car('捷达',60)
16         car1.be_drived(300)
17
18         car2 = Car('捷豹', 120)
19         car2.be_drived(400)
20
21
22     if __name__ == '__main__':
23         main()
24
```

对于重复操作或函数可采取以上方法，可以方便快捷地使用。

## 8、两个重要的原则：

单一职责原则

**Don't repeat yourself**

## 9、Git, Github 的使用