Python基础学习

一.环境安装与虚拟环境

> 安装部分不做过多讲解，每个人的电脑环境相差很大，具体情况单独处理，只给出参考文档

windows系统

https://blog.csdn.net/liang19890820/article/details/51068914

mac 系统

默认有python2，python3的安装参考下面的文章 https://www.cnblogs.com/1009-smile/p/8005524.html

centos/ubuntu

既然会用linux，安装python应该不是问题

```
虚拟环境：
安装：pip3 install virtualenv
创建：virtualenv demo --no-site-packages  --python=python2.7
进入：source demo/bin/activate
退出：deactivate
```

```
pip包管理：

pip install six -i https://pypi.douban.com/simple
```

二.字符串/列表/元组/字典

> Python是动态语言，变量不需要声明。每个变量在使用前都必须赋值，变量赋

> 值以后该变量才会被创建。在Python中，变量就是变量，它没有类型，我们所说的“类型”是变量所指的内存中对象的类型。

Python 3中有六个标准的数据类型：

1. Numbers（数字）
2. String（字符串）
3. List（列表）
4. Tuple（元组）
5. Sets（集合）
6. Dictionaries（字典）

Numbers（数字）

> Python 3支持int、float、bool、complex（复数）。数值类型的赋值和计算都是很直观的，就像大多数语言一样。内置的type()函数可以用来查询变量所指的对象类型。

数值运算：

```
>>> 5 + 4  # 加法
9
>>> 4.3 - 2 # 减法
2.3
>>> 3 * 7  # 乘法
21
>>> 2 / 4  # 除法，得到一个浮点数
0.5
>>> 2 // 4 # 除法，得到一个整数
0
>>> 17 % 3 # 取余
2
>>> 2 ** 5 # 乘方
32
```

要点：

- 1、Python可以同时为多个变量赋值，如a, b = 1, 2。
- 2、查看 数据类型：type 或者isinstance

- 3、数值的除法（/）总是返回一个浮点数，要获取整数使用//操作符。

字符串（String）

```
最简单的：
a = "hello world"
print (a)
```

要点：

- 1、编码问题：2存在编码问题，3不存在 utf-8
- 2、字符串可以用+运算符连接在一起，用*运算符重复。{} %s做格式化
- 3、Python中的字符串有两种索引方式，从左往右以0开始，从右往左以-1开始。
- 4、Python中的字符串不能改变 ，值类型 不可变
- 5、str.count("c")字符串的数量
- 6、字符串的替换：replace
- 7、字符串的长度：len()
- 8、是否在：if in
- 9、翻转字符串：[::-1]
- 10、string转list split

List（列表）

> list是 Python中使用最频繁的数据类型。列表是写在方括号之间、用逗号分隔开的元素列表。列表中元素的类型可以不相同

```
>>> a = ['him', 25, 100, 'her']
>>> print(a)
['him', 25, 100, 'her']
和字符串一样，列表同样可以被索引和切片，列表被切片后返回一个包含所需元素的新列表。详细的在这里就不赘述了。

列表还支持串联操作，使用+操作符：
>>> a = [1, 2, 3, 4, 5]
>>> a + [6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]
与Python字符串不一样的是，列表中的元素是可以改变的：
```

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> a[0] = 9
>>> a[2:5] = [13, 14, 15]
>>> a
[9, 2, 13, 14, 15, 6]
>>> a[2:5] = []   # 删除
>>> a
[9, 2, 6]
添加元素：append
删除元素：pop
列表相加：+ 、 extend
列表属于引用类型
```

值类型： 包含：字符串、元组、数值，本身不允许被修改

引用类型： 包含：列表、字典，本身允许修改

copy deepcopy

要点：

- 1、List写在方括号之间，元素用逗号隔开。
- 2、和字符串一样，list可以被索引和切片。
- 3、List可以使用+操作符进行拼接。
- 4、List中的元素是可以改变的。
- 5、list转string :join
- 6、添加：append
- 7、删除：pop，remove，del
- 8、其他操作：index()、count()、reverse()、clear()、insert、sort()、extend()
- 9、enumerate循环，得到index
- 10、extend和+=的区别

Tuple元组

> 元组（tuple）与列表类似，不同之处在于元组的元素不能修改。元组写在小括号里，元素之间用逗号隔开。元组中的元素类型也可以不相同：

```
>>> a = (1991, 2014, 'physics', 'math')
```

```
>>> print(a, type(a), len(a))
(1991, 2014, 'physics', 'math')
```

元组与字符串类似，可以被索引且下标索引从0开始，也可以进行截取/切片（看上面，这里不再赘述）。其实，可以把字符串看作一种特殊的元组。

```
>>> tup = (1, 2, 3, 4, 5, 6)
>>> print(tup[0], tup[1:5])
1 (2, 3, 4, 5)
>>> tup[0] = 11  # 修改元组元素的操作是非法的
```

虽然tuple的元素不可改变，但它可以包含可变的对象，比如list列表。

要点：

- 1、与字符串一样，元组的元素不能修改。
- 2、元组也可以被索引和切片，方法一样。
- 3、元组也可以使用+操作符进行拼接。
- 4、和list的区别：不可变，用于常量定义

Sets集合

> 集合（set）是一个无序不重复元素的集。基本功能是进行成员关系测试和消除重复元素。可以使用大括号 或者 set()函数创建set集合，注意：创建一个空集合必须用 set() 而不是 { }，因为{ }是用来创建一个空字典。

```
>>> student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}
>>> print(student)   # 重复的元素被自动去掉
{'Jim', 'Jack', 'Mary', 'Tom', 'Rose'}
>>> 'Rose' in student  # membership testing（成员测试）
True
>>> # set可以进行集合运算
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a
{'a', 'b', 'c', 'd', 'r'}
>>> a - b     # a和b的差集
{'b', 'd', 'r'}
>>> a | b     # a和b的并集
{'l', 'm', 'a', 'b', 'c', 'd', 'z', 'r'}
>>> a & b     # a和b的交集
{'a', 'c'}
>>> a ^ b     # a和b中不同时存在的元素
```

```
{'l', 'm', 'b', 'd', 'z', 'r'}
```

要点：

- 1、set集合中的元素不重复，重复了它会自动去掉。
- 2、set集合可以用大括号或者set()函数创建，但空集合必须使用set()函数创建。
- 3、set集合可以用来进行成员测试、消除重复元素。

Dictionary字典

> 字典（dictionary）是Python中另一个非常有用的内置数据类型。字典是一种映射类型（mapping type），它是一个无序的键：值对集合。关键字必须使用不可变类型，也就是说list和包含可变类型的tuple不能做关键字。在同一个字典中，关键字还必须互不相同。

```
>>> dic = {}  # 创建空字典
>>> tel = {'Jack':1557, 'Tom':1320, 'Rose':1886}
>>> tel
{'Tom': 1320, 'Jack': 1557, 'Rose': 1886}
>>> tel['Jack']   # 主要的操作：通过key查询
1557
>>> del tel['Rose']  # 删除一个键值对
>>> tel['Mary'] = 4127  # 添加一个键值对
>>> tel
{'Tom': 1320, 'Jack': 1557, 'Mary': 4127}
>>> list(tel.keys())  # 返回所有key组成的list
['Tom', 'Jack', 'Mary']
>>> sorted(tel.keys()) # 按key排序
['Jack', 'Mary', 'Tom']
>>> 'Tom' in tel       # 成员测试
True
>>> 'Mary' not in tel  # 成员测试
False

>>> dict(sape=4139, guido=4127, jack=4098)
{'jack': 4098, 'sape': 4139, 'guido': 4127}
另外，字典类型也有一些内置的函数，例如clear()、keys()、values()等。
```

要点：

- 1、字典是一种映射类型，它的元素是键值对。

- 2、字典的关键字必须为不可变类型，且不能重复。
- 3、创建空字典使用{ }。
- 4、增 demo["a"] = ""
- 5、删：pop，clear
- 6、改：重新赋值
- 7、查：get(),setdefault(),has_key()
- 8、其他：items()、keys(),values()、iteritems(),iterkeys(),itervalues()

三.运算、判断、循环

```
运算符：

绝对值：abs()
转为int： int(x)
转为float： float(x)
x的y次幂：pow(x, y)
```

#写程序将温度从华氏温度转换为摄氏温度。转换公式为C=5/9*(F-32)

```
>>> def FtoC(degree):
...     return float("%.2f" %(5.0*100/9*(degree-32)/100))
...
>>> print FtoC(100)
37.78
```

python计算中7*7*7.可以有多少种写法

```
第一种：
>>> 7*7*7*7
2401
第二种：
>>>pow(7,4)
2401
第三种：
>>>7**4
2401
```

```
第四种：
 >>> a=7
>>> for i in range(3):
...     a*=7
...
>>> a
2401
```

判断和循环：

```
if "页面":
    "前端工程师"
elif "后台":
    "后端工程师"
else:
    "全栈"
```

Python中用elif代替了else if，所以if语句的关键字为：if - elif - else。

要点：

- 1、每个条件后面要使用冒号（:），表示接下来是满足条件后要执行的语句块。
- 2、使用缩进来划分语句块，相同缩进数的语句在一起组成一个语句块。
- 3、在Python中没有switch - case语句。

```
x = int(input("Please enter an integer: "))
if x < 0:
    print('Negative.')
elif x == 0:
    print('Zero.')
else:
    print('Positive.')
```

while语句

while 判断条件： statements

在Python中没有do..while循环。

```
a, b = 0, 1
while b < 10:  # 循环输出斐波纳契数列
    print(b)
    a, b = b, a+b
```

for语句

> Python中的for语句与C语言中的for语句有点不同：C语言中的for语句允许用户自定义迭代步骤和终止条件；而Python的for语句可以遍历任何序列（sequence），按照元素在序列中的出现顺序依次迭代。一般形式为：

```
for variable in sequence:
    statements
else:
    statements


words = ['cat','love','apple','python','friends']
for item in words:
    print(item, len(item))
```

range函数

如果你要遍历一个数字序列，那么内置的range()函数就可以派上用场了。函数range()常用于for循环中，用于产生一个算术数列：

```
>>> list(range(10))   # 默认从0开始
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))  # 从1到11，前闭后开
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))  # 5表示步长，每隔5取一个数
[0, 5, 10, 15, 20, 25]


for i in range(2, 11):
    print(i)
```

break、continue、pass及else子句

break break语句与C语言中的一样，跳出最近的for或while循环。

continue continue语句同样是从 C 语言借用的, 它终止当前迭代而进行循环的 下一次迭代。

pass pass语句什么都不做，它只在语法上需要一条语句但程序不需要任何操作时使用。pass语句是为了保持程序结构的完整性。

四.异常处理

1.Python中常见的异常类型

AttributeError 试图访问一个对象没有的树形，比如foo.x，但是foo没有属性x IOError 输入/输出异常；基本上是无法打开文件 ImportError 无法引入模块或包；基本上是路径问题或名称错误 IndentationError 语法错误（的子类）；代码没有正确对齐 IndexError 下标索引超出序列边界，比如当x只有三个元素，却试图访问x[5] KeyError 试图访问字典里不存在的键 KeyboardInterrupt Ctrl+C被按下 NameError 使用一个还未被赋予对象的变量 SyntaxError Python代码非法，代码不能编译(个人认为这是语法错误，写错了） TypeError 传入对象类型与要求的不符合 UnboundLocalError 试图访问一个还未被设置的局部变量，基本上是由于另有一个同名的全局变量， 导致你以为正在访问它 ValueError 传入一个调用者不期望的值，即使值的类型是正确的

```
try:
    msg = input(">>")
    int(msg)
except Exception as e:
    print("异常的类型是:%s"%type(e))
    print("异常对象的内容是:%s"%e)
```

2.Python中的异常处理机制

在Python当中，若一个程序在运行的时候出错，Python解释器会自动的在出错的地方生成一个异常对象，而后Python解释器会自动的在出错地方的附近寻找有没有对这个异常对象处理的代码，所谓异常处理代码就是try……except语句，如果没有，Python解释器会自动的将这个异常对象抛给其调用函数，就这样层层抛出，如果在main当中也没有对这个异常对象处理的代码，Python解释器（实际上是操作系统）最后会做一个简单粗暴的处理，将整个程序给终止掉，并将错误的信息在显示屏上输出。 （三）Python中的异常处理方法

```
try:
    可能出现异常的代码块
except Exception as e:
    print("异常的类型是:%s"%type(e))
    print("异常的内容是:%s"%e)
else:
    print('如果代码块不抛出异常会执行此行代码!')
finally:
    print('不管代码块是否抛出异常都会执行此行代码!')

raise
traceback
```

5.函数：

> 函数（function）是组织好的、可重复使用的、具有一定功能的代码段。函数能提高应用的模块性和代码的重复利用率，Python中已经提供了很多内建函数，比如print()，同时Python还允许用户自定义函数。

```
def 函数名(参数列表):
    """文档字符串"""
    函数体
    return [expression]

def fun(name, age, gender):
    print('Name:',name,'Age:',age,'Gender:',gender,end=' ')
    print()

fun('Jack', 20, 'man') # call
```

6.类

创建和使用类 创建Dog类，赋予dog蹲下（sit()）和打滚（roll_over()）的能力

```
class Dog():
    def __init__(self,name,age):
        self.name=name;
        self.age=age;
    def sit(self):
        print(self.name.title()+" is now sitting.")
    def roll_over(self):
```

```
        print(self.name.title()+" rolled over!")
```

方法_init_()是一个特殊的方法，相当于构造方法，每当创建新实例时，Python都会自动运行它 *init*()中的形参self必不可少，还必须位于其他形参的前面。创建实例时不用给self传递值 后面的两个方法由于不需要额外的信息，因此只有一个形参self，变量都有前缀self，以self为前缀的变量 可供类中的所有方法使用。self.name=name获取存储在形参name中的值，并将其存储到变量name中，然后 该变量被关联到当前创建的实例。

根据类创建实例

```
my_dog=Dog('stupy',3)
这里使用上面的Dog类创建了一个名为 ‘stupy’ 、年龄为3的my_dog实例
访问属性
my_dog.name获取名字，my_dog.age获取年龄
调用方法
my_dog.sit()     my_dog.roll_over()
```

给属性指定默认值

比如上例默认为公狗，则在_init_()函数中创建一个性别属性并设置初始值，self.sex='gong'

修改属性的值

1.直接修改属性的值 接上例，使用句点表示法来直接访问并设置小狗的属性name。my_dog.name='clever'

2.通过方法修改属性的值 相当于java中的set方法 例：def update_name(self,newname) self.name=newname my_dog.update_name('wuwu') 这样就无需直接访问属性，而可以将值传递给一个方法，由它在内部进行更新

导入类

导入单个类

将Car类存储在一个名为car.py的模块中，在另一个文件中导入Car类并创建其实例：from car import Car

在一个模块中存储多个类

例如类Battery和ElectricCar都在模块car.py中，若只要导入一个ElectricCar类：
from car import ElectricCar

若两个类都导入：

from car import Car,ElectricCar

导入整个模块

可以直接import module_name，再用句点表示法访问需要的类。这种导入方法简单，代码也易于阅读

导入模块中的所有类

from module_name import * 不推荐这种导入方式。 一、如果只要看一下文件开头的import语句，就能清楚地知道程序使用了哪些类，将大有裨益。 二、如果程序文件中有其他同名的类，运行时将引发难以诊断的错误

静态方法/类方法

https://www.cnblogs.com/revo/p/7381101.html

习题：

- 1.写一个英镑/公斤之间的转换函数
- 2.请将字符串反转并输出。例：'abc'的反转是'cba'
- 3.下节课：先自己注册一个微信公众号，了解下管理功能

备注：

- 1.流行趋势：python为什么容易学，java python php go
- 2.说在前面的话，python的国内公司，豆瓣，饿了么等，python2.x，还没有兼容3，所以关于版本，不要太排斥，你进了这些公司，仍然是要维护老代码，学习python3，和2的区别也要了解。我最后有一个demo，之前是用python2写的，我会尽量改成2和3通用的方式。
- 3.window中的命令行工具：Babun

- 参考：https://www.cnblogs.com/haochuang/p/5593411.html
- 4.查看python版本：python --verison
- 5.变量和常量：python中没有规定的常量，大小写区分 NAME='' 常量
- 6.不指定数据类型，动态语言
- 7.python是大小写识别的，True