Python语言

欢迎来到Python语言讲义。这里我们重点讨论Python的入门学习和基本的语法，场景。

整个讲义基于`Python3`

Python的基础

`Python`是一种计算机程序设计语言。你可能已经听说过很多种流行的编程语言，比如非常难学的`C`语言，非常流行的`Java`语言，适合入门编程上手的`C#`语言，适合初学者的`Basic`语言，适合网页编程的`PHP`、`JavaScript`语言等等。

那Python是一种什么语言？

> 1、Python是一种少有的既简单又功能强大的编程语言：
>
> 由于它开源的本质，Python已经被移植到许多不同的平台，如果你小心的避免使用依赖于操作系统的特性，那么就意味着你的Python程序不需要修改就能在目前绝大多数的操作系统上运行，包括：Windows、Linux/Unix、Macintosh、OS/2、Win CE（Windows嵌入式操作系统）等等
> 2、Python是一种高级编程语言，相比其他高级语言C/C++/Java等流行开发语言，Python的代码更简单，但在执行速度上将会更慢，实际上Python本身就是由C/C++语言开发的；
> 3、Python中可以调用C/C++语言编写的代码，反过来C/C++也能调用Python代码；
> 4、Python既支持面向过程又支持面向对象编程；
> 5、Python有丰富的库文件支持，库文件你可以理解为已经写好的功能模块，在Python中可以直接调用：
> a)Python标准库:可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、GUI（图形用户界面）、Tk和其他与系统有关的操作。记住，只要安装了Python，所有这些功能都是可用的。这被称作Python的“功能齐全”理念。
> b)除标准库外，还有许多其他高质量的库，如wxPython、Twisted和Python图像库等等。

作为测试工程师学会一门语言是非常重要的，优点如下：

> 1、对被测系统的理解可以从代码级的深度，从而能更好的做好测试；
> 2、掌握一种编程语言可以做很多测试辅助工具，如：写一段程序脚本来准备测试数据；写一个辅助工具来辅助测试等；
> 3、自动化测试的需要。

在所有语言中Python是一种代表简单主义思想的语言，他的代码更像是在阅读简单的英语，它使你专注于解决问题而不是搞懂语言本身。

Python的简介

/'paɪθən/

Python的创始人为Guido Van Rossum。1989年圣诞节期间，在阿姆斯特丹，Guido为了打发圣诞节的无趣，决心开发一个新的脚本解释程序，做为ABC 语言的一种继承。之所以选中Python（大蟒蛇的意思）作为程序的名字，是因为他是一个叫Monty Python的喜剧团体的爱好者。

Python 语言除了在自动化测试领域有出色的表现外，在系统编程，网络编程，web 开发，GUI开发，科学计算，游戏开发等多个领域应用非常广泛，而且具有非常良好的社区支持。也就是说学习和掌握python 编程，其实是为你打开了一道更广阔的大门。

Python是一种相当高级的语言。比如，完成同一个任务，C语言要写1000行代码，Java只需要写100行，而Python可能只要20行。当然，代码少的代价，就是运行慢。C程序运行1秒钟，Java程序可能需要2秒，而Python程序可能就需要10秒。

对于初学者和完成普通任务，Python语言是非常简单易用的。连Google都在大规模使用Python，你就不用担心学了会没用。

Python是著名的“龟叔”Guido van Rossum在1989年圣诞节期间，为了打发无聊的圣诞节而编写的一个编程语言。

Python就为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池（batteries included）”。用Python开发，许多功能不必从零编写，直接使用现成的即可。

除了内置的库外，Python还有大量的第三方库，也就是别人开发的，供你直接使用的东西。当然，如果你开发的代码通过很好的封装，也可以作为第三方库给别人使用。

许多大型网站就是用Python开发的，例如YouTube、Instagram，还有国内的豆瓣。很多大公司，包括Google、Yahoo等，甚至NASA（美国航空航天局）都大量地使用Python。

Python的安装

因为Python是跨平台的，它可以运行在Windows、Mac和各种Linux/Unix系统上。在Windows上写Python程序，放到Linux上也是能够运行的。

Python官网：www.python.org

Python中文学习网站：http://www.runoob.com/　www.codecombat.com

要开始学习Python编程，首先就得把Python安装到你的电脑里。安装后，你会得到Python解释器（就是负责运行Python程序的），一个命令行交互环境，还有一个简单的集成开发环境。

目前，Python有两个版本，一个是2.x版，一个是3.x版，这两个版本是不兼容的。由于3.x版越来越普及，我们的教程将以最新的Python 3.4版本为基础。请确保你的电脑上安装的Python版本是最新的3.4.x，这样，你才能无痛学习这个教程。

> Python3.5+只能安装在Windows7以及以上的操作系统。如果您的电脑是Windows XP操作系统，请选择Python3.4或者更低的版本。

Python的IDE

`IDE`，`Integrated Development Environment`，集成开发环境。

一个好的编辑器或者好的IDE将会极大的提高生产力，帮我们做很多事情，使得编码工作更加简单，编码的体验更加容易。

目前主要有以下IDE：

- `IDLE`：Python自带的IDE，功能简单，使用方便
- `Notepad++`：一个强大的开源编辑器
- `Vim`：Linux系统中最好用的编辑器之一
- `Sublime Text`：一个非常轻便好用的现代化的编辑器，推荐。
- Eclipse
- `PyCharm`：JetBrains公司提供的现代化的跨平台的Python IDE。

建议使用`Sublime Text`或`PyCharm`

Python的哲学

在控制台或者Python IDE中输入`import this`并运行，将会打印Python的哲学。

```
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!
```

Python的语句

如果使用`PyCharm`写Python，那么需要创建一个`project`，然后在项目文件夹右键，选择`Add`|`Python File`，在接下来的文件中编写即可。

首行我们往往输入以下语句，对整个文件编码进行设定`UTF-8`，避免中文出现乱码问题。

```
# -*- coding: utf-8 -*-
# -*- coding: gbk -*-
```

防止中文乱码

输入和输出

print打印

print语句，在Python2.x和Python3.x中不一样。这里的语法基于Python3.

```
# 普通打印
print("Hello Python!")
```

选择参数进行打印的时候，需要在前面的字符串中定义 `%d` | `%s` | `%r` 等，在后面用 `%` 选择参数。若超过一个参数的情况下，应该用括号 `()` 括起来。

```
# 选择一个参数 name 进行打印。s% 是指“字符串型占位符”
name = "zhangsan"

print("Hello %s, welcome to the world of python!" % name)
# % string 字符
```

```
# 选择一个参数 age 进行打印。 %d 是指“数值型占位符”
age=30
print("Oh, you are %d !" % age)
# % decimal 数值
```

```
# 选择一个参数 unknow 进行打印。 %r 是指“随机类型占位符”
unknown =100
print("Your printing is %r." %unknown)
# % random 随机的
```

```
ct = 10000 * 3
# %r random
print("xx %r" % ct)

# 选择多个参数 first_name, last_name, ct 进行打印，注意前后匹配，以及括号的使用
first_name = "Jack"
last_name = "Martin"
print("the man's name is %s %s\n%r" % (first_name, last_name, ct))
```

input输入

```
newinput = input("Please enter any content: ")
print("Your inputing is %r" %newinput)
```

引号和注释

Python中不区分单引号和双引号

单行注释：`#`

多行注释：`"""` 或者 `'''`

```
# hahaha
# eeeee
# ttttttt
# 这是单行注释
```

```
"""
这是多行注释
恩
对的
"""
'''
这也是多行注释
啊
哈哈
'''
```

分支与循环

if语句

if语句是各种语言的最基本的判断语句，对于Python来说也是一样的。

```
# 普通的分支判断语句
a = 2
b = 3
#表达式中的比较符号还可以包括：>, <, >=, <=, ==, !=
if a > b:
    print("a is bigger!")
else:
    print("b is bigger!")
```

```
# 判断字符串是否相等
employee = "haha"
if employee == "haha":
    print("haha")
else:
    print("not haha")
```

```
# 判断逻辑值 a 是否为真。 bool型
a = True
if a:
    print("a is true")
else:
    print("a is false")
```

```
# 多个判断条件， elif和else
score = 72
if score >= 90:
    print("优秀")
elif score >= 70:
    print("良好")
elif score >= 60:
    print("及格")
else:

    print("不及格")
```

for语句

for循环在Python中的应用相当广泛和简洁。

```
# 循环一个字符串
for i in "hello python! ":
    print(i)
```

```
x=1
for i in "hello python! ":
    print("第%d个字符是%s" %(x,i))
    x=x+1
```

```
# 循环一个数组
fruits = ["banana", "apple", "mango", "berry"]
for fruit in fruits:
    print(fruit)
```

```
# 从0到4 循环5次，每次步长为1
for i in range(5):
    print(i)
```

```
# 从1 到10（不包括10）循环，步长为2
for i in range(1, 10, 2):
    print(i)
```

while语句

while语句本身的应用不是很广阔，但是我们可以来借助while查看 `continue` 和 `break` 的用法。

```
count = 0
while (count < 9):
   print('The count is:', count)
   count = count + 1
print ("Good bye!")
```

```
# continue 和 break 用法
# continue 和 break 在for循环中同样适用
i = 1
while i < 10:
    i += 1
    if i%2 > 0:     # 非双数时跳过输出
        continue
    print(i)        # 输出双数2、4、6、8、10

i = 1
while 1:            # 循环条件为1必定成立
    print(i)        # 输出1~10
    i += 1
    if i > 10:      # 当i大于10时跳出循环
        break
```

数组与字典

数组

```
# 数组中的顺序是有序的
# 我们可以随意的更改任意元素
lists = [1,2,3,"a",5,"8"]
print(lists)
print(lists[0])
print(lists[4])
lists[4]="b"
print(lists[4])
print(lists)
```

字典

字典是一种全新的`键/值 对`类型，非常类似于`JSON`。

字典是一种无顺序的表达，在初始化的时候，字典的顺序会确定。

```python
dicts = {"username":"zhangsan", "password":123456}
# 打印字典的所有键
print(dicts.keys())

# 打印字典的所有值
print(dicts.values())

# 打印字典的所有项
print(dicts.items())

# 打印整个字典
print(dicts)

# 打印字典中的键为username的值
print(dicts.get("username"))

# 把字典dicts复制一份，赋值给dicts2
dicts2 = dicts.copy()
print(dicts2)

# 用for循环遍历整个字典，并按照键和值分别打印
for key, value in dicts.items():
    print("dicts keys is %r " % key)
    print("dicts values is %r " % value)

#给字典加一个项目
dicts["NewKey"] = "NewValue"
#修改一个项目
dicts["password"] = 654321
print(dicts)
#删除一个项目
del dicts["NewKey"]
print(dicts)
```

```
order = {"message": "ok",
         "nu": "401298079167",
         "companytype": "zhongtong",
         "ischeck": "1",
         "com": "zhongtong",
         "status": "200",
         "condition": "F00"
         }
for key in order.keys():
    print("order's key is %r: " % key)

print("-----")

for value in order.values():
    print("order's value is %r" % value)

print("-----")

for key, value in order.items():  # 项
    print("order's key is %r, and value is %r" % (key, value))
```

Python的面向对象

Python是一种纯面向对象的语言，函数、类和方法都是Python所擅长的。

函数

```
def add(a, b):
    print(a + b)
add(3, 5)
```

```
def add(a, b):
    return a + b

if __name__ == "__main__":
    print(add(3, 5))
```

面向对象的基础是类。这里我们尝试进行类的基本介绍，以及继承的使用。

类和方法

```
"""
创建一个类，名字为Abb，\继承object
类Abb有一个方法：add; 一个属性name
"""
class Abb(object):
     name = "这是类属性的例子"
    def add(self, a, b):
        return  a + b

# 这里是python的主方法入口
if __name__ == '__main__':
    count = Abb()
    print(count.name)
    count.name="我已经是个实例了"
    print(count.name)
    print(count.add(5, 9))
```

```
# 在上面声明了类Abb的基础上，这里继续创建类Baa
class Abb(object):

    def add(self, a, b):
        return  a + b

# 类Baa继承于Abb，具有Abb的所有方法和属性
class Baa(Abb):

    def sub(self, a, b):
        return a - b

if __name__ == '__main__':
    count = Abb()
    print(count.add(5, 9))
    count2 = Baa()
    print(count2.add(5, 9))
    print(count2.sub(5, 9))
```

```python
# 这里尝试导入模组 time
import time

class Order(object):
    def method1(self, a, b, c):
        return a * b + c

    def method2(self):
        return 100


class LittleOrder(Order):
    def method3(self, a, b):
        return a / b + 5

    def method4(self, a, b, c):
        return self.method1(a, b, c) + self.method2()


if __name__ == '__main__':
    abc = Order()
    print(abc.method1(12, 5, 8))
    ddd = Order()
    print(abc.method1(12, 8, 8))
    print(abc.method1(12, 5, 8) > ddd.method2())
    print("abc.method1 is %d: " % abc.method1(12, 5, 8))
    print("ddd.method2 is %d: " % ddd.method2())

    xiao = LittleOrder()
    print(xiao.method2())
    print(xiao.method3(500, 100))
    print("xiao's method4 %d: " % xiao.method4(12, 5, 8))

    # 这里使用了模组 time的方法
    print(time.ctime())
```

Python的高阶

模组

引入模组

上述例子中显示了导入模组的使用。在单元测试中我们继续这点的讲解

```python
import time
print(time.ctime())
print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(time.time())))
```

模块的调用

```
#把当前目录的XXX.py引入到本文件
import xxx
#从某个文件夹下面引入某个py文件
from selenium import webdriver
```

from后面加的是文件夹的名字

import的是文件的名字

异常

```
# 异常的处理显示了各种语言对于异常的方式
# 发生异常的时候，如果没有异常处理语句，系统的异常的提示将会直接显示在控制台
# 使用了异常处理语句以后，系统本身的异常将不会提示。
try:
    open("abc.txt", "r")
except Exception as e:
    print("异常了！")
    print ("错误信息是这样的：%s"%e)
```