细说Python

刘英

2018年07月15

目录

第一部分

Python基础语法

第一章　Python介绍

这里我们介绍一些很杂乱而又及其不重要的内容，比如计算机语言，Python的历史，发展现状等等。其中N多内容只是我的一家之言，并不具有任何学术价值，我的意思是说，我不保证本章内容的严谨性甚至正确性，姑且听之，姑妄言之。

1.1　语言和编译器

语言其实是交流的一种媒介，抑或是一个交流的工具。

一般一门语言包括语法和语义两个方面的内容，我们不做具体讨论，我们提起计算机语言来，首先想到的是一套符合某些个则的代码。我们人跟人交流需要说人话，狗跟狗交流相互汪汪就可以，但人跟计算机交流怎么办呢？人身上肯定不能插上根电线，而计算机也很难张口说话，于是在人跟计算机交流的过程中，我们需要一个交流的工具，于是诞生了我们称之为计算机语言的东西。

虽然借助于计算机语言我们可以跟计算机交流，但是很遗憾计算机只懂得机器语言，而我们人类却觉得机器语言比较难懂，最终大家各退一步，发明另一种语言，这些语言我们人觉得容易理解，虽然计算机不懂，但我们可以给计算机在发明一个翻译官，于是这个翻译官我们叫做解释器。

我们通俗的说某某计算机语言，一般指的是一套语法规则和一套解释器(编译器)的组合。

1.2 语言的大致分类

语言按照发展历程大致可以分为机器语言，汇编语言，高级语言。

机器语言（二进制语言）：计算机本质上是一台机器，机器哪里懂得什么语言，它只是懂得两个不同的电压-高电压和低电压（其实我们可以做出N多不同的电压值，只不过最终只选择了两个），我们于是尝试用两个不同的电压值来通过不同的组合顺序来表达一些复杂的事情，这个方法类似摩尔斯电报码。在计算机语言中，用数字0和1来分别表示高电压和低电压。0和1通过不同的组合最终形成了只有0，1数字构成的机器语言。

汇编语言： 有了机器语言之后，人们发现二进制其实并不合适做交流的工具，因为人们觉得这个语言作为计算机的母语，我们的学习成本太高，可以想象一下一连串血(Xie)长血长的0，1组成的东西，这东西基本人类很难很难理解的，聪明的人们于是想到另一套东西，大概是这样的，我们把机器语言某些固定的内容用某些相应的英文单词来代替，按照动词+宾语的形式表示一定的语义，于是就出现了汇编语言，具体详情请自行百度。

高级语言：汇编语言严格上说只是机器语言的一个简化版，我们需要的其实是人话，或者一种更高级的语言，最好能直接使用我们的语言，计算机科学家干脆发明一种全新的语言，这种语言要尽可能符合我们人类高级语言的特性，让我们写起来很容易，计算机执行的时候我们再想办法翻译给计算机就好了，于是在此思想下诞生了高级计算机语言，其中包含著名的C语言，Java，C等语言，当然还包括我们的主角-Python。

1.3 Python的一些问题

1.3.1 读音问题

中国一般读作排森，美国读音为排桑，许老师有时候读排孙... ...

1.3.2 速度问题

语言这个玩意儿，基本是解决问题的工具，趁手就好。Python的速度

确实不快，但也基本够用，真需要唯快不破了，我们有别的解决办法。

1.3.3 能不能做大项目问题

欧美国家其实用Python做大项目的不少，据说Instagram全部使用Python作为编程语言，当然如果你家的访问量分分钟超过Instagram的话，当我没说。

1.3.4 到底是学Python还是学其他语言

这个问题别问了，我就一句话，你似不似撒？别的语言在中国火了有的几十年了，别说你未必学会，就算你学会了，大牛都愁去哪儿找工作呢，你还能成精？Python刚火起来，学了Python你未必成精，但你至少跟将来的Python大牛在一起懵逼过。

1.4 Python简史

Python的历史不算太短，以前在我们国家不太出名有其特殊愿意，但Python的大事记大概如下:

- 1989: Python诞生了
- 2008: Python3.0 诞生
- 2014: 宣布2.7将成为2.x系列最后版本，被支持到2020年
- 2018: AI元年，中国Python培训遍地开花

1.5 Python的特点

Python语言是很有个性的一门计算机语言，可以说特点鲜明，大概的几点:

- 简单易学不罗嗦，导致经常被用作青少年编程入门语言

- 语法简洁明了，层次分明，格式严格
- 功能强悍且严重不偏科，基本各个领域都有涉及，均衡发展
- 执行速度相对不快，但开发速度嗷嗷快
- 丰富的库任君挑选，俗称“轮子”多
- 可移植性高，Python代码移植在编写的时候稍微注意下，基本没有移植问题

1.6 Python运行环境

第二章　注释，基础变量和运算符

2.1　Python注释

注释是怕别的程序员看不懂自己的牛叉代码而人为添加的一些解释性内容，计算机不会执行，对于计算机而言，遇到注释内容他会直接忽略。

我们知道Python代码有两个作用:

- 人用来控制计算机进行工作
- 程序员之间相互交流思想的工具

程序员之间进行交流，一般情况下，Python代码作为程序员的一个主要产出物，是可以充当交流工具这样一个角色的，但是，代码再利于交流，毕竟也不是我们人类的自然语言，所以有时候经常会出现别人写的代码自己看不懂的情况，于是，程序员经常为复杂代码或者算法提供一个额外的解释性说明，我们称之为注释。

计算机在在执行代码的时候，发现注释后会自动忽略，因为它知道那是给别人看到东西，不是给计算机的指令。

Python的注释分为两种:

- 行注释：一般只有一行或者一句话，简明扼要的解释你的代码。用#开头，程序在执行的时候，#后面内容自动忽略
- 块注释：如果你觉得一两句话不能阐述你那深邃的思想，一般会七嗦

八的写一大堆话，此时用块注释，通常块注释用来说明一个函数或者类。

在下面例子中，真正执行输出的代码只有一行，即打印出"Hello world"来，但我为了向大家解释这行伟大的代码，我给它写了好多注释：

```
#这是一行注释呀，程序不会看到的
#这又是一行注释呀
print("Hello world") #程序看到井号后就不读了
'''
这个是块注释，一下可以写一大堆啦
上面写了这么多
其实输出也就一句话
没什么大不了的
反正程序是忽略注释的
'''
```

以后我们的代码中会添加很多注释，同学们在看代码的时候前往不要忽略注释，因为那是专门给你们看的东东呦。

2.2 Python变量的命名和规范

2.2.1 变量的基本使用

变量就是在代码中为我们指代某些值的一些临时性标记。

有一种提法就是，程序即使数据结构加上算法，这是很经典的一个说法，或者通俗的解释程序就是数据和数据被按计划操作的集合，这里面的数据包含很多内容，比如数字，比如文字信息，比如一个结构体，我们为了对我们的数据进行操作，需要给我们的数据起一个名字，这个名字很可能是临时性的，比如我们班的班长，班长其实就是一个指代，某一天我们发现它不合格了，完全可以把他罢免，此时班长这个代号就代表别人了。

在Python中，给一个变量命名很简单，只要需要一个变量的时候之间起一

个名字并给这个名字一个值就好，以后使用某个值的时候可以直接用这个代号表示就可以了，例如下面代码：

```
#生成一个变量，这个变量叫a，它指的是数字100，那么以后我们就可以在用到100的时候直接写一个a就可以
a = 100
# 我想把100打印到屏幕上，这个时候我直接用变量就可以了
print(a)
```

2.2.2 变量的命名规范

起名字是个大学问，我们变量的命名也需要有一个规范，此处规范的含义是，你必须这么做，否则，Python 会拒绝你的变量，他会给你报错。每个语言变量命名规范可能不太一样，但大体规则类似，Python变量命名规则如下：

- 变量命名可以包含数字，大小写字母，下划线甚至汉字，但是我们不推荐除了前三种内容之外的符

```
# 变量Teacher包含大写字母，我们允许
Teacher = "刘大拿"
print(Teacher)

# 下面变量定义合法
student_1 = 19
print(student_1)

# 用下划线开头是可以的，但在Python中，下划线开头变量一般具有特殊含义
_me = 10
print(_me)

```

```
13          # 其实中文也是可以做变量名称的，只是我们超级不推荐
14          老师 = "刘大拿"
15          print(老师)
```

- 数字不可以作为开头，下划线开头一般有特殊含义，中文可以变量名，但不提倡

```
1           # 下面的代码大家可以把代码行的注释一个一个打开，然后运行下看看报错内容
2           # 用数字开头是不行滴
3           #4person = "Five persons"
4
5           # 但是，包含数字是可以的
6           a4 = "This is a a a paper"
7           zhao4 = "This is a person"
```

- 一般在python中，以下划线开头的内容具有特殊含义，只限具体环境使用

```
1           # 下划线开头的内容包括但不限于一个下划线开头，两个下划线开头的内容
2           # 需要注意的是，下面四个变量并不相同
3           _age = 18
4           _name = "Liu Dana"
5
6           # 请注意以下两个变量用两个下划线开头
7           __age = 10
8           __name = "Liu Ying"
9
10          print(_age)
11          print(_name)
```

```
12
13          print(__age)
14          print(__name)
15
16          #上面代码都是可以运行的，但是我们平时不要这么写
```

- 大小写不一样，俗称大小写敏感

```
1           # 变量大小写是不一样的
2           forMan = 19
3           ForMan = "WangXiaojing is my girlfriend"
4           forman = "哈哈，刘大拿你崔留逼了"
5
6
7           print(forMan)
8           print(ForMan)
9           print(forman)
```

- 不允许使用保留字或者关键字作为变量名称。我们有时候会说计算机语言是人类给计算机的指令，在使用这些指令的时候不可避免使用一些约定好的命令，比如开始，结束，判断等命令，这些都是系统使用的，不允许个人作为变量使用，这些命令俗称关键字或者保留字。需要注意的是，从计算机科学的角度讲，关键字和保留字有着不同的含义，此处不作区分。

```
1           # 不能使用关键字作为变量名
2           # 系统中有多少关键字或者保留字呢，下面代码演示了查看系统关键字的方法
3
4           # keyword中包含了所有的系统关键字
5           import keyword
```

```
6           # 把系统关键字打印出来
7           print(keyword.kwlist)
8
9
10          # 下面代码，如果你用关键字作为变量名，会报错的，把注释取消掉，看看会出现啥错误呀
11          #class = 9
12          #print(class)
```

上面都是我们在使用变量的时候需要遵守的规则，为了规范变量命名，使我们的程序菜鸟也能写出高大上的代码，我们强烈建议大家在命名变量的时候遵循一下规则：

- 绝不推荐变量命名中使用汉字，时刻为你的代码全球通用做好准备，很遗憾目前全球唯一通用语言基本就是英语，不是汉语，更不是日语，日语只是全球宅男的通用语言，而且仅限于几个元音字母。
- 最好不要用汉语拼音，如果必须用，绝不允许用汉语拼音的缩写，请写全称
- 推荐使用英文单词或者单词缩写
- 如果变量太多，推荐使用驼峰命名法，或者
- 使用英文单词，各单词用下划线连接

以下的代码是比较推荐的命名方式：

```
1   # 以下的变量命名比较推荐
2
3   srv_list = []
4   teacher_name = "刘大拿"
5
6   student = "WangXiaojing"
```

```
7
8       age = 18
9
10      student_address = "北京海淀区西四环北路18号"
```

2.3 变量的声明

前面我们了解了如何命名一个变量，那么在系统中究竟怎么“创造”一个变量呢，或者说如何声明一个变量?
声明一个变量可以有三种方式，根据情况任君选择:

- 简单声明,一次声明一个变量并进行赋值
- 多个变量设置成同一内容
- 一次性声明多个变量并赋不同的值

如上例所示，上面的代码声明后就可以愉快的使用啦。

2.4 Python变量几种类型

我们一直都在说变量是用来存放数据的一个临时代号，那么我们的数据到底分几种呢？在Python中，严格意义上的数据只有一个类型，但是出于理解方便，我们把数据分为六大类，分别是:

- 数字类型-Number
- 字符串类型-str
- 列表-list
- 元组-tuple
- 字典-dictionary

- 集合-set

我们会详细介绍每一个类型，本章主要挑选两个简单的类型给大家介绍:

- 数字类型-Number
- 字符串类型-str

2.5 Python运算符

第三章　Python语句

3.1　Python语句概述

3.2　分支语句

3.2.1　if基本就够

3.2.2　if不够else凑

3.2.3　else不够elif凑

3.3　循环语句

3.3.1　for循环

3.3.2　range的作用

3.3.3　while起来无休止

3.3.4　关于循环的零七碎八

第四章　聊聊函数

4.1　函数基础

4.2　参数和返回值

4.2.1　形参和实参

4.2.2　特殊形式的参数

4.3　变量的作用域

4.4　递归函数

4.5　常见系统内置函数

第五章　Python内建数据类型

5.1　再聊聊字符串

5.2　list是个筐

5.3　字典是一对对出现的

5.4　元组就是不能改变的list

5.5　set就是集合

第二部分

Python面向对象编程

第六章　类的基本实现

6.1　面向对象概述

6.2　找个对象恋爱

6.3　关于类的命名规范

第七章　类或实例成员分析

7.1　类或对象的构成

7.2　类或实例的三类成员函数

7.3　self

第八章　OOP的三大特性

8.1　继承

8.2　多态

8.3　封装

第九章　Python类的内置操作或属性

第十章　Python类的魔术方法

10.1　什么是魔术方法

10.2　常用魔术方法举例

第十一章　Python类的高级用法

11.1　抽象类

11.2　自定义类

第三部分

Pytho常用包

第十二章　调试基本技术

12.1　软件测试流程

12.2　调试基本术语

12.3　pdb调试

12.4　pycharm调试

12.5　单元测试

第十三章　Python包

13.1　模块

13.2　包

13.3　常用包介绍

13.3.1　time

13.3.2　calendar

13.3.3　datetime

13.3.4　timeit

13.3.5　os

13.3.6　shutil

13.3.7　zip

13.3.8　math

13.3.9　string

13.3.10　zip

13.3.11　enumerate

13.3.12　collections

13.3.13　namedtuple

13.3.14　dequeue

13.3.15　defaultdict

第十四章　异常处理

14.1　错误和异常

14.2　异常处理

14.3　手动引发异常

14.4　自定义异常

第十五章　高级函数

第十六章　信息持久化

16.1　Python文件使用

16.2　Pickle

16.3　shelve

第十七章　日志系统

17.1　日志介绍

17.2　Logging的处理流程

第十八章　多线程

第十九章　协程

19.1　迭代器和生成器

19.2　asyncio

19.3　async and wait

19.4　aiohttp

19.5　futures

第二十章　信息格式化存储

20.1　数据存储之XML

20.2　数据存储之JSON

20.3　格式化数据查找之XPath

20.4　格式化数据查找之正则表达式

第二十一章　Net编程

21.1　sockets编程

21.2　Ftp编程

21.3　Mail编程

21.4　Http协议