【白话python连载（1）】一起遇见python--发现程序之美

* 笔者与编程的那些事

从最开始接触c，SQL，asp，Java，JS, Android到VB，python，MongoDB等等就这样在编程的路上摸爬滚打过了好久，从基础认知到项目实践；

或许会为一个用超简单代码解决工作量很大的项目而兴奋不已；

或许在操作一个用户体验很好且功能齐全的软件时而赞叹不已；

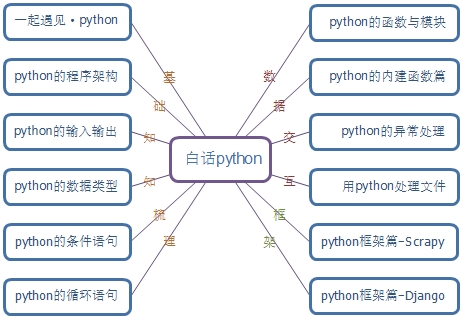
或许看到屏幕上清晰明了的分析结果和酷炫的展示而震惊不已；

或许是深深感受到被bug和编码问题所支配的恐惧与抓心挠肝；

在不断尝试和应用后，你会发现由代码构建的世界也是五彩斑斓的。奋战在一线的程序员曾说读懂代码语言比读懂人的的语言更简单一些，毕竟编程无非就是进行计算，判断，交互，展示，自学习等，并且依靠一定的规则与逻辑的设置帮助人们解决实际问题或创建相关应用，使人们从重复枯燥的工作中解放出来。最近使用python较多，因此将自己所学所感与大家分享，以期通过与大家的交流互动而更深层次的了解应用python，并和大家一起发现python之美。

* 本系列连载的框架介绍

从python加入小学课本来看，python是初学者的语言，符合“大脑思维习惯”，是易于学习与理解的。本次白话python连载将按照自己天马行空的想法外加一点逻辑来介绍python，主要分为三大部分，包括python的基础知识梳理和介绍；python与其他数据的交互；python的常用框架。本系列连载可能无法做到事无巨细或面面俱到，但是将尽可能从python语言特点出发并结合具体应用进行介绍。



* python的基本介绍

正式介绍python语言之前，先来看个小应用来初步了解一下python具体可以解决的问题。文中使用的python版本为anaconda（python3.6），代码编辑器（python的集成开发环境）为pycharm2017社区版。关于python的具体安装大家可通过以下网站自行下载安装。

https://www.python.org/getit/ （python官网）

https://www.anaconda.com/download/ （anaconda官网）

<https://www.jetbrains.com/pycharm/> （pycharm官网）

例1：#问题描述：一球从100米高度自由落下，每次落地后反跳回原高度的一半；再落下，求它在第10次落地时，共经过多少米？第10次反弹多高？

|  |
| --- |
| *#!/usr/bin/python #coding=utf-8* sum\_ten **=** 0 **for** i **in** range(10)**:** each **=** 2**\***100**/**(2**\*\***i) #上一个反弹高度与本次下落高度相等，可同时获取  each\_up **=** 50**/**(2**\*\***i) #获取每次反弹的高度  sum\_ten **=** each **+** sum\_ten #求和计算下落和反弹的总高度  print(**'第 {} 次反弹的高度为：{}'**.format(i**+**1,each\_up)) #输出每次的反弹结果 print(**'它落地第10次时，共经过 {} 米'**.format(sum\_ten**-**100)) #输出总的反弹高度 |
| #运行结果  "D:\anaconda python\python3.6.exe" D:/vernacular/coding\_ziyi/calculate.py  第 1 次反弹的高度为：50.0  第 2 次反弹的高度为：25.0  第 3 次反弹的高度为：12.5  第 4 次反弹的高度为：6.25  第 5 次反弹的高度为：3.125  第 6 次反弹的高度为：1.5625  第 7 次反弹的高度为：0.78125  第 8 次反弹的高度为：0.390625  第 9 次反弹的高度为：0.1953125  第 10 次反弹的高度为：0.09765625  它落地第10次时，共经过 299.609375 米  Process finished with exit code 0 |

上面程序只给出了其中一种处理小球下落和反弹的基本算法；除此以外你可以尝试，通过列表存值，再通过求和函数等直接计算。从上述实际应用可以初步感受到python解决基本的便捷之处，只要了解了python规则后，便可以按照处理问题的逻辑来组织程序，使其完成我们要解决的问题。

* 一般程序语言和python语言的基本特点

作为铺垫，我们需要先了解一般程序语言和python语言的基本特点。

从编写角度看，一般的程序语言可以分为面向过程和面向对象两大类，它们的主要区别在于组织程序语言时侧重点不同，面向过程是以指令流程为中心，通过指令处理信息，具体实现的是‘如何做’的问题，并且自顶向下通过各个函数进行组织；而面向对象则是以功能的封装为核心，抽象的对象都有各自的属性和方法（功能），具体实现的是‘谁去做’的问题，并且通过模块间相互交互进行组织。两者非对立关系，面向对象可以认为是在面向过程的基础上加入了封装、继承、多态等特性，更好的描述世界上的事物，更便捷的去串联方法。使在做复杂的事情的时候效率和正确率都能有所保证。

从运行角度看，程序语言可以分为编译型和解释型。以c语言，java语言和python语言为例。对于编译型语言（c语言）编译是将源程序翻译成可执行的目标代码（二进制机器语言），翻译与执行是分开的，之后运行程序不需要再翻译，而是可以直接执行一个.exe的文件；解释是对源程序的翻译与执行一次性完成，不生成可存储的目标代码。边运行边解释的特点也使其运行效率低于纯编译型语言，因此为了保证一定的运行效率，，也是需要进行预编译的。以Java程序语言为例，预编译不是直接编译成为机器语言，而是编译成为字节码，然后在Java虚拟机上用解释方式执行字节码。

Python是典型的面向对象、解释型的计算机程序语言。

创建一个类和对象是很容易的，而且python也采用了类似Java的编译模式，先将Python程序编译成Python字节码，然后由一个专门的Python字节码解释器负责解释执行。常用的python解释器有CPython、IPython、PyPy、Jython、IronPython等。CPython是官方版本的解释器，用>>>作为提示符；IPython在交互方式上较CPython有所增强，用In[序号]:作为提示符。这两种解释器是较为常用的解释器。而PyPy是使用JIT技术为提高执行速度的；Jython和IronPython则是分别运行在Java和微软.Net平台上的Python解释器。因此在实际编程中可以按照自己的需求进行选择。

python程序的运行方式有两种，交互式运行和通过命令行运行

* 看到C:\>是在Windows提供的命令行模式：可直接运行.py的python脚本文件。

|  |
| --- |
| D:\vernacular>python3.6 D:\vernacular\coding\_ziyi\quadratic.py |

* 看到>>>或In[序号]表示是在Python交互式环境下，只能输入Python代码并立刻执行。

|  |  |
| --- | --- |
| >>> a = 1  >>> print(a)  1 | In[1]a = 1  In[2]print(a)  1 |

python的特点

python语言有很多人使用是其最大的优点，可以很好地反映出其自身的特点，简单、易学、开源免费、可移植性等等，吸引了大量的应用者，创建了丰富的库以及规范的代码。可以广泛的应用于图像处理、数字处理、文本处理、多媒体应用、数据库编程和web编程等。

关于python的具体程序架构，将在下次内容中进行具体介绍。

