**freeggache**

python

basic

目录

[1 序言 3](#_Toc407625793)

[2 我学Python用来干什么？ 4](#_Toc407625794)

[3 编写运行“Hello, world!” 6](#_Toc407625795)

[4 Python对象类型 10](#_Toc407625796)

[Python的核心数据类型 10](#_Toc407625797)

[数字 10](#_Toc407625798)

# 1 序言

作为一只编程菜鸟，每次学一门语言，我都要反反复复，对自己的记忆力感到无比崩溃。于是乎，蠢爆了的我只能仔仔细细老老实实的专研（本人实在是找不到任何捷径）。幸运的是，菜鸟做久了，也琢磨出点道道来了，在这里，想将自己的一些心得和学习笔记分享与大家，但想强调一下，我写的这些渣文仅适用于和我一样初学的菜鸟（承认吧，孩子），谢谢。

# 2 我学Python用来干什么？

《Python学习手册（第4版）》一书中对Python的优缺点做了很多详细的说明，在基础篇里我们不太关心，因为道行还不够深，所以只关心学了Python后可以干些啥，在提高篇里再对其进行补充。

* 系统编程

Python对操作系统服务的内置接口，使其成为编写可移植的维护操作系统的管理工具和部件（有时也称shell工具）的理想工具。

* 用户图形接口

Python的简洁以及快速的开发周期十分适合开发GUI程序。Python内置了TKinter的标准面向对象接口Tk GUI API，使Python程序可以生成可移植的本地观感的GUI。

* Internet脚本

Python提供了标准的Internet模块，它使得Python程序能够广泛地在多种网络任务中发挥作用，无论是在服务器端还是在客户端都是如此。

* 组件集成

Python可以通过C/C++系统进行扩展，并能够嵌套C/C++系统的特性，使其能够作为一种灵活的黏合语言，可以脚本化处理其他系统和组件的行为。

* 数据库编程

对于传统的数据库需求，Python提供了对所有主流关系数据库系统的接口。

* 快速原型

由于使用Python或C编写的组件看起来一样，所以可以在一开始利用Python做系统原型，之后再讲组件移植到C或C++这样的变异语言上。

* 数值计算和科学计算编程
* 游戏、图像、人工智能、XML、机器人等

最后两点，这里暂时不提，感兴趣可以私下研究。

貌似有些复杂，好了，总之就是能干很多事了，刚开始学，大概了解下就行，知道些名词能吹吹牛就好。（没前途…）但有一个优点对于我们这些懒癌患者相当重要，那就是Python编程所需代码比Java、C++少太多了！

在java中输出字符串代码如下：

public class A

{

public static void main(String args[]){

System.out.print(“haha”);

}

}

C++中输出字符串代码如下：

#include<iostream>

using namespace std;

int main(){

cout<<”haha”;

}

然而在Python中，代码仅仅如下：

print(“haha”)

是不是格外的简单？

# 3 编写运行“Hello, world!”

无论学什么语言，永远的第一个程序都是输出“Hello, world!”，之所以如此，无非是让我们去了解该语言是如何运行程序的。所以，按照国际惯例。我们也来学习一下Python的“Hello, world！”程序该如何编写（基础篇暂时只研究Windows）。

* 首先，要安装Python。
  + 下载Python（这里就以当前最新版3.4.2为例）

<https://www.python.org/ftp/python/3.4.2/python-3.4.2.amd64.msi>

* + 安装

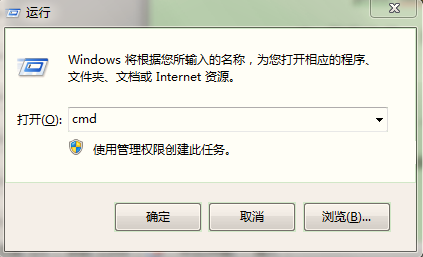
按照默认路径，一直点确认即可。

* + 修改系统Path

右击“我的电脑”——属性——高级系统设置——环境变量——双击系统变量“Path”——在变量值的最后加上“;C:\Python34”如下图：



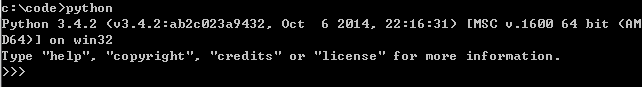
* 进入命令控制行窗口（windows键+R，输入cmd）



* 检验python是否安装好

输入：python

如有显示以下画面说明已经安装好

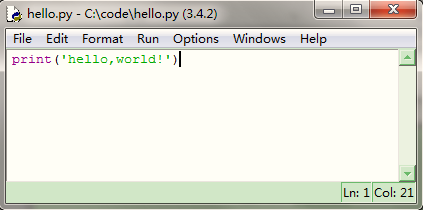


* 编写“Hello，world”程序
  + 首先在C盘下新建一个code文件夹，接着在code文件夹中新建文本文档hello.txt，然后手动把文件名改成hello.py，如下图：

* + 选中该文件，然后右击选择Edit with IDLE，在文本框内

输入：print (‘hello,world!’)



* 保存文件，进入命令控制行，切换目录

输入：cd c:\code



* 运行hello.py

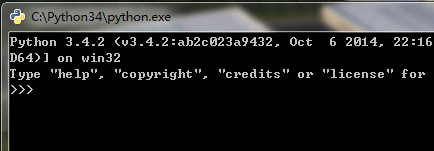
输入：python hello.py



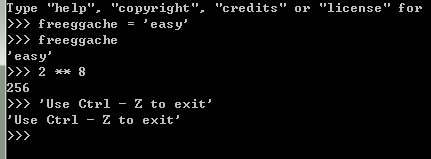
好了，“hello,world!”就这样完成了!

像这样，每次编写代码（可以不用python自带的IDLE，而是使用一些工具如UltraEdit或其他）都要保存为.py的格式，然后用Python自带的python.exe去执行他。

除了上述方法，还有一种交互地运行代码的方式，可以启动IDLE得主窗口，或通过从Python的Start按钮菜单的菜单选项中选择“Python(command line)”来开始类似的交互会话。这两种方式都会产生一个具有同样功能的Python交互式命令提示符，而不必输入一条shell命令。



像这样在交互模式下工作，想输入多少Python命令就输入多少，每一个命令在输入回车后都会立即运行。此外，由于交互是对话自动打印输入表达式的结果，在这个提示模式下，往往不需要每次都刻意地输入“print”：



现在，我们对这次绘画中的代码并不是特别了解：仅仅是输入一些Python的打印语句和变量赋值的语句，以及一些表达式，这些我们都会在稍后进行深入的学习。

# 4 Python对象类型

在典型的Python程序中，部署内存结构、管理内存分配、实现搜索和读取例程这些令人头痛的大部分工作都消失了。因为Python提供了强大的对象类型作为语言的组成部分，在你开始解决问题之前往往没有必要编写对象的实现。事实上，除非你有内置类型无法提供的特殊对象要处理，最好总是使用内置对象而不是使用自己的实现。

## Python的核心数据类型

|  |  |
| --- | --- |
| 对象类型 | 例子 常量/创建 |
| 数字 | 1234,3.1415,3+4j,Decimal,Fraction |
| 字符串 | ‘aa’,”ff” |
| 列表 | [1,[2,’three’],4] |
| 字典 | {‘food’:’spam’,’taste’:’yum’} |
| 元组 | (1,’aa’,4,’U’) |
| 文件 | myfile = open(‘eggs’,’r’) |
| 集合 | set(‘abc’),{‘a’,’b’,’c’} |
| 其他类型 | 类型、None、布尔型 |
| 编程单元类型 | 函数、模块、类 |
| 与实现相关的类型 | 编译的代码堆栈跟踪 |

### 数字

Python的核心对象集合包括常规的类型：整数、浮点数以及更为少见的类型（有虚部的复数、固定精度的十进制数、带份子和分母的有理分数以及集合等）。Python中的数字支持一般的运算。例如，加号（+）代表加法，星号（\*）表示乘法，双星号（\*\*）表示乘方。

>>>123 + 222

345

>>>1.5 \* 4

6.0

>>>2 \*\* 100

1267650600228229401496703205376

注意，当需要的时候，Python3.4的整数类型会自动提供额外的精度，以用于较大的数值。例如，你可以计算2的1,000,000次幂，但打印的话得要好一会了。

>>>len(str( 2\*\* 1000000)) *# How many digits in a really BIG number*

301030

但当你刚接触浮点数时，却会遇到些奇怪的现象：

>>>3.1415 \* 2

6.2830000000000004

>>>print(3.1415 \* 2)

6.283

第一个结果并不是Bug，而是现实的问题。这里体现了两种打印方式：全精度（第一个结果显示的那样）以及用户友好的方式（第二个）。如果以后遇到有些东西显示的很奇怪，试试使用打印语句显示他们。

### 字符串

字符串是用来记录文本信息的。在Python中他们以序列的形式储存。序列中的元素包含了一个从左到右的顺序——序列中的元素根据他们的相对位置进行存储和读取，从严格意义上来说，字符串是由单个字符组成的序列，其他类型的序列还包括列表和元组（后面介绍）。

#### 序列的操作

如果我们有一个含有4个字符的字符串，我们可以通过内置的len函数验证他的长度并通过索引取得各个元素。

>>>s = ‘ABCD’

>>>len(s) *# Length*

4

>>>s[0] *# The first item in s,indexing by zero-base position*

‘A’

>>>s[1] *# The second item from the left*

‘B’

在Python中，索引也是从0开始，第二项索引为1，以此类推。另外，我们还可以进行反向索引，从最后一个开始

>>>s[-1] *# The last item in s*

‘D’

>>>s[-2] *# The second to last item from the end*

‘C’

除了简单地从位置进行索引，序列也支持一种所谓切片（slice）的操作。

>>>s *# A 4-character string*

‘ABCD’

>>>s[1:3] *# Slice of from offsets 1 through 2(****not 3****)*

‘BC’

认识切片最简单的办法就是把他们看作从一个字符串中一步就提取出一部分的方法。他们的一般形式为X[I:J]，表示“从X中取出偏移I开始，直到但不包括J的部分”。结果返回一个新的对象。

在一个切片中，左边界默认为0，并且右边界默认为切片序列的长度，于是演化出以下的用法。

>>>s[1:] # Everything past the first(1:len(s))

‘BCD’

>>>s[0:3] # Everything but the last

‘ABC’

>>>s[:3] # Same as s[0:3]

‘ABC’

>>>s[:-1] # Everything but the last again,but simpler(0:-1)

‘ABC’

>>>s[:] # All of S as a top-level copy(0:len(S))

‘ABCD’

最后，作为一个序列，字符串也支持使用加号进行合并（将两个字符串合并成为一个新的字符串），或者复制。

>>>s

‘ABCD’

>>>s + ’xyz’ # Concatenating

‘ABCDxyz’

>>>s # S is unchanged

‘ABCD’

>>>s \* 4 # Repetition

‘ABCDABCDABCDABCD’

注意，（+）对于不同的对象有不同的意义：对于数字为加法，对于字符串为合并。这个特性叫多态，在后面会提到。