第一章用户交互

本文是Python通用编程系列教程，已全部更新完成，实现的目标是从零基础开始到精通Python编程语言。本教程不是对Python的内容进行泛泛而谈，而是精细化，深入化的讲解，共5个阶段，25章内容。所以，需要有耐心的学习，才能真正有所收获。虽不涉及任何框架的使用，但是会对操作系统和网络通信进行全局的讲解，甚至会对一些开源模块和服务器进行重写。学完之后，你所收获的不仅仅是精通一门Python编程语言，而且具备快速学习其他编程语言的能力，无障碍阅读所有Python源码的能力和对计算机与网络的全面认识。对于零基础的小白来说，是入门计算机领域并精通一门编程语言的绝佳教材。对于有一定Python基础的童鞋，相信这套教程会让你的水平更上一层楼。

一 编程入门

1. 编程的概念

我们学习一门编程语言需要先了解清楚，什么是编程，为什么要编程，最后才学习怎么编程。
计算机的发明就是为了用机器取代人力，来帮助人类进行无休止的工作，还不给他工资，这就是编程的目的，因为计算机听不懂人话，那就是只好人来说计算机的话，来传达给计算机这个工作应该如何进行，人在说“机话”的过程其实就是编程，是人类把我们需要工作的内容通过某种指令传达给计算机。

2. 编程语言的划分

你说“机话”就行了，难道说“机话”还有不同的。
是的，人话都各有千秋，“机话”自然也是风情万种。上面所说的能被计算机所识别的表达方式就是编程语言，语言是人与人之间沟通的介质，编程语言是程序员与计算机沟通的桥梁。

总结一下：编程就是程序员按照某种编程语言我自己的工作流写下来，结果就是一堆包含有字符，数字或者英文字母的文件。需要注意的是，在程序运行之前和普通的文件没有区别，只有当程序运行了，计算机才会按照该编程语言的语法格式读取里面的内容，这样程序的内容才会生效，才会有计算机工作的效果。

> 编程语言发展到如今经历了三个过程，分别是：机器语言，汇编语言和高级语言。

机器语言可以理解为人完全说“机话”，我们要先知道计算机的原理是二进制，计算机他也就能看懂二进制了（关于二进制，后面的内容到字符编码会有详细的讲解），那就用二进制编程吧：比如我想让计算机计算一个100加200，我要这样操作：

```
1100100,101011,11001000
```

注释：二进制1100100 代表十进制的100，二进制11001000代码十进制200，二进制101011代码“+”这个符号，100+200的计算用机器语言编写大概就是上面的代码，我太年轻，没出生在那个美好的时代。显而易见，用这种方式编写程序对于程序员来说是十分痛苦的，开发效率非常低下。相反，对于计算机来说，他一看到100100000这种东西就会心想，“这个对我的胃口，我就喜欢这样的”，但是作为程序员的你，并不能很轻松的写出这种东西。所以总结一下，机器语言是人站在计算机的角度去编程，编程开发效率低，程序运行效率高。

汇编语言可以理解为人说“机人话”，就是人用简写的英文标识符去取代二进制，这样的话人编写的效率肯定会提升了，但是计算机的有点不乐意了，不是直接的二进制，要先把这个英文标识符转化成二进制，进而才能执行代码。所以，相比较机器语言来说，汇编语言开发效率提升，程序运行效率略微降低。

高级语言可以理解为人说“人话”(人：终于能好好说话了)，之所以称他为高级，是因为他与以上两种语言有本质的区别，高级语言是建立在操作系统的基础之上，而以上两种编程语言都是不需要操作系统，直接操作硬件的。操作系统是人创造的，他的出现目的就是提高人们的工作效率，让人们更好的使用计算机，由于操作系统的封装，把操作的硬件的悲伤的工作留给了自己，而让程序员的编程变得非常友好，使用和自己的语言非常相似的语法格式去编程，大大的提升了程序员的开发效率。相反，计算机对于这件事有意见。

```
# 计算机：你们这些愚蠢的人类，搞了个什么破操作系统，这样的我要先翻译成二进制才能执行，执行的很慢，我很不高兴，后果很严重。
# 程序员：我管你高兴不高兴，你能拿到我怎么样，还后果很严重，我好怕怕啊，你怎么不上天啊
# 计算机：喔，行。。。。。。你行。。。。。。
```

接下来用一句话总结一下，机器语言到汇编语言再到高级语言，程序员开发效率提升的同时，程序执行效率也降低了，那么这是进步还是退步呢?

速度不是关键，开发效率才是王道。因为速度有瓶颈，机器语言到高级语言，虽然速度降了，但是依然快到人类的大脑意识无法捕捉出来差距，0.000000001秒和0.00001秒对你来说应该没什么区别吧，虽然他们相差了1万倍。开发效率可以大幅提升，所以这样的发展自然是进步。

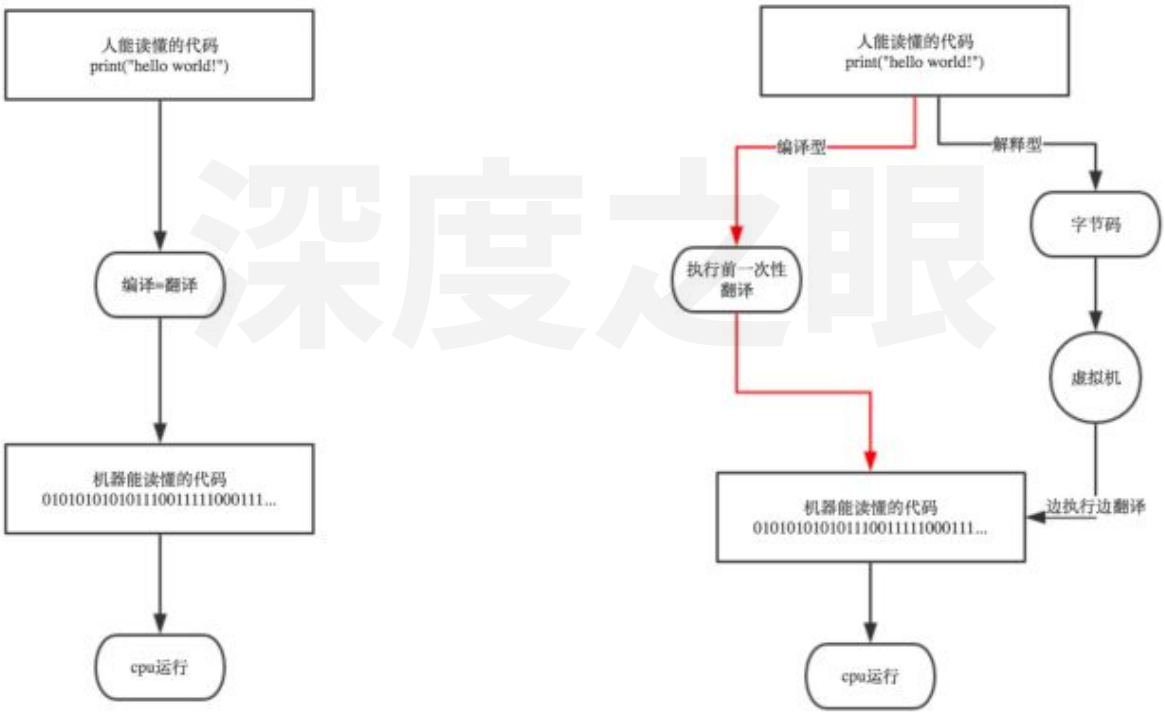
二 高级语言的分类

高级语言更贴近人类的语言，他必须先被转换成能计算机能读懂的二进制后，才能被执行，按照转换的方式分为：

编译型语言和解释型语言

编译型语言需要编译器，就像谷歌翻译一样，你写的代码就是一个程序，在程序执行之前要先编译（就像是把一本英文的书先翻译成中文再给你看），编译的结果就转化成为了二进制之后，计算机才可以运行。这样就带来了一个问题，当程序需要修改的时候，不能直接修改编译之后的目标文件，而是要修改源文件，再次编译生成目标文件，这个时候计算机再运行，查看修改的结果。由于不同的计算机的操作系统的不同，编译器生成的目标文件也不相同（不同的操作系统它所提供的硬件接口不同，所以不同的硬件接口就需要不同的二进制文件来执行）。和解释型语言相比较，他的执行速度快，要依赖编译器，重点是调试很麻烦。像C语言，C++都是编译型语言。

解释型语言需要解释器，就像口译一样，需要一句一句的翻译，应用程序源代码一边由相应语言的解释器“翻译”成目标代码（机器语言），一边执行，因此效率比较低，而且不能生成可独立执行的可执行文件，应用程序不能脱离其解释器(想运行，必须先装上解释器，就像跟老外说话，必须有翻译在场)，但这种方式比较灵活，可以动态地调整、修改应用程序。像Python，Php都是解释型语言。



编译型语言与解释型语言的执行过程

总结：编译型语言和解释型语言各有其优缺点，当我们需要编写的程序不需要经常改动或者升级的时候，我们一般会选用编译型语言，比如写操作系统或者其他的一些比较底层的应用；当我们编写的程序需要经常改动升级以此来满足用户的新的需求的时候，我们一般会选用解释型语言，比如一些应用软件，一些网站项目。值得一提的是，Python语言由于他的简洁性与高度封装，在解释型语言里面依然是属于调试最方便，使用最灵活的，所以当项目需要快速开发和迭代的时候，公司里面很多新项目一般会选用Python语言，既可以写移动应用的接口，也可以写Web项目，还可以非常方便地进行比较复杂的机器学习，数据挖掘的运算等等，所以这也是Python语言比较火热的原因之一。

三 Python介绍

1. Python语言说明

> _python的创始人为吉多·范罗苏姆（Guido van Rossum）。1989年的圣诞节期间，Guido开始写能够解释Python语言语法的解释器。Python这个名字，来自Guido所挚爱的电视剧Monty Python's Flying Circus。他希望这个新的叫做Python的语言，能符合他的理想：创造一种C和shell之间，功能全面，易学易用，可拓展的语言。最新的TIOBE排行榜，Python赶超C++占据第4， Python崇尚优美、清晰、简单，是一个优秀并广泛使用的语言。Python可以应用于众多领域，如：数据分析、组件集成、网络服务、图像处理、数值计算和科学计算等众多领域。目前业内几乎所有大中型互联网企业都在使用Python，如：Youtube、Dropbox、BT、Quora（中国知乎）、豆瓣、知乎、Google、Yahoo!、Facebook、NASA、百度、腾讯、汽车之家、美团等。

2. Python应用领域

1. 网站开发：能够最快上手的Web框架Django，短小精悍的Flask框架，能够适用于创业型公司快速建立产品业务逻辑。
2. 网络编程：支持高并发的Twisted网络框架，使异步编程变的非常简单。
3. 爬虫开发：爬虫领域，Python几乎是霸主地位，Scrapy\Request\BeautifuSoap\urllib等，可以轻松爬取几乎网上所有的资源。
4. 云计算开发：目前最知名的开源云计算框架OpenStack就是用Python语言编写的，这也极大的推动了云计算行业的发展。
5. 人工智能：谁会成为AI 和大数据时代的第一开发语言？这已是一个不需要争论的问题。过去人们会使用MATLAB，Octave等语言来做科学计算，但是落地到应用开发上还是用使用C++，Java，R或者Python，但是现在Python有开源的 PyTorch 和TensorFlow，它作为 AI 时代头牌语言的位置基本确立，未来的悬念仅仅是谁能坐稳第二把交椅。
6. 自动化运维：问问中国的每个运维人员，运维人员必须会的语言是什么？就算你问10个人，相信他们都会给你一个相同的答案，它的名字叫Python
7. 金融分析：金融分析公司写的很多分析程序、高频交易软件就是用的Python,到目前,Python是金融分析、量化交易领域里用的最多的语言
8. 科学运算：你知道么,97年开始，NASA就已在大量使用Python在进行各种复杂的科学运算，随着NumPy, SciPy, Matplotlib, Enthought librarys等众多程序库的开发，使得Python越来越适合于做科学计算、绘制高质量的2D和3D图像。和科学计算领域最流行的商业软件Matlab相比，Python是一门通用的程序设计语言，比Matlab所采用的脚本语言的应用范围更广泛
9. 游戏开发：在网络游戏开发中Python也有很多应用。相比Lua or C++,Python 比 Lua 有更高阶的抽象能力，可以用更少的代码描述游戏业务逻辑，与 Lua 相比，Python 更适合作为一种 Host 语言，即程序的入口点是在 Python 那一端会比较好，然后用 C/C++ 在非常必要的时候写一些扩展。Python

非常适合编写 1 万行以上的项目，而且能够很好地把网游项目的规模控制在 10 万行代码以内。另外据我所知，知名的游戏<文明> 就是用Python写的

3. Python在公司的应用

- 谷歌：Google App Engine 、code.google.com 、Google earth 、谷歌爬虫、Google广告等项目都在大量使用Python开发
- CIA: 美国中情局网站就是用Python开发的
- NASA: 美国航天局(NASA)大量使用Python进行数据分析和运算
- YouTube:世界上最大的视频网站YouTube就是用Python开发的
- Dropbox:美国最大的在线云存储网站，全部用Python实现，每天网站处理10亿个文件的上传和下载
- Instagram:美国最大的图片分享社交网站，每天超过3千万张照片被分享，全部用python开发
- Facebook:大量的基础库均通过Python实现
- Redhat: 世界上最流行的Linux发行版本中的yum包管理工具就是用python开发
- 豆瓣: 公司几乎所有的业务均是通过Python开发
- 知乎: 国内最大的问答社区，通过Python开发(国外Quora)
- 春雨医生：国内知名的在线医疗网站是用Python开发
- 除上面例举之外，还有搜狐、金山、腾讯、盛大、网易、百度、阿里、淘宝 、土豆、新浪、果壳等公司都在使用Python完成各种各样的任务。

4. 使用Python2还是Python3

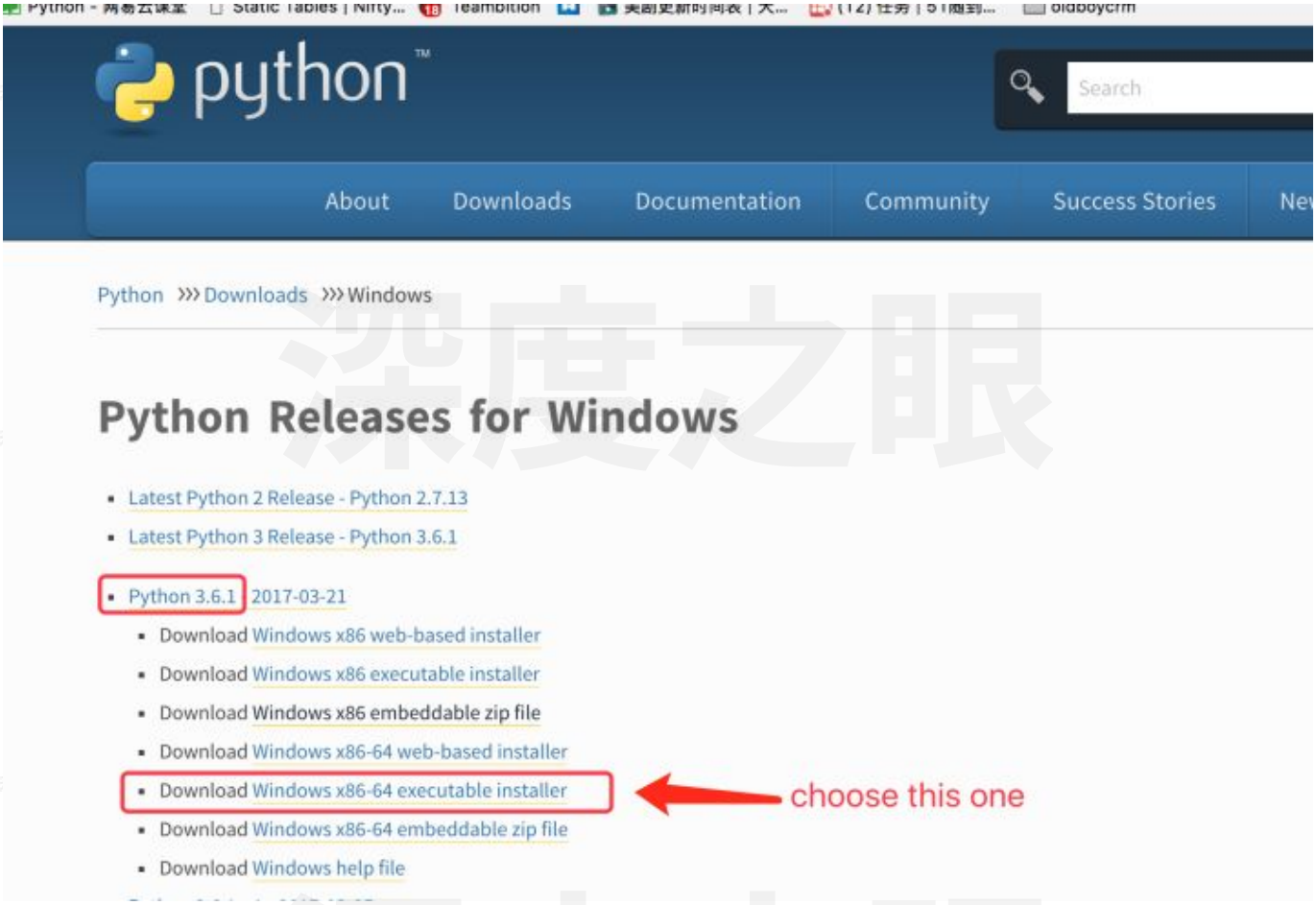
> 先来看一段官方文档
>
> In summary : Python 2.x is legacy, Python 3.x is the present and future of the languagePython 3.0 was released in 2008. The final 2.x version 2.7 release came out in mid-2010, with a statement ofextended support for this end-of-life release. The 2.x branch will see no new major releases after that. 3.x isunder active development and has already seen over five years of stable releases, including version 3.3 in 2012,+3.4 in 2014, and 3.5 in 2015. This means that all recent standard library improvements, for example, are onlyavailable by default in Python 3.x.Guido van Rossum (the original creator of the Python language) decided to clean up Python 2.x properly, with less regard for backwards compatibility than is the case for new releases in the 2.x range. The most drastic improvement is the better Unicode support (with all text strings being Unicode by default) as well as saner bytes/Unicode separation.Besides, several aspects of the core language (such as print and exec being statements, integers using floor division) have been adjusted to be easier for newcomers to learn and to be more consistent with the rest of the language, and old cruft has been removed (for example, all classes are now new-style, "range()" returns a memory efficient iterable, not a list as in 2.x).

目前虽然业内很多企业还在大量使用Python2.6 or 2.7，因为旧项目几十万甚至上百万行的代码想快速升级到3.0不是件容易的事，但是大家在开发新项目时几乎都会使用3.x。另外Python3 确实想比2.x做了很多的改进，直观点来讲，就像从XP升级到Win7的感觉一样，棒棒的。Python2 和Python3的具体细节区别我们在以后课程中会慢慢深入。

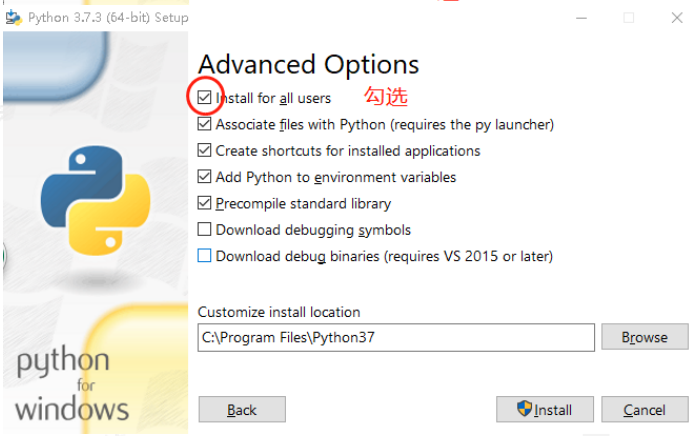
5. 安装Python解释器

Python目前已支持所有主流操作系统，在Linux,Unix,Mac系统上自带Python环境，在Windows系统上需要安装一下，超简单

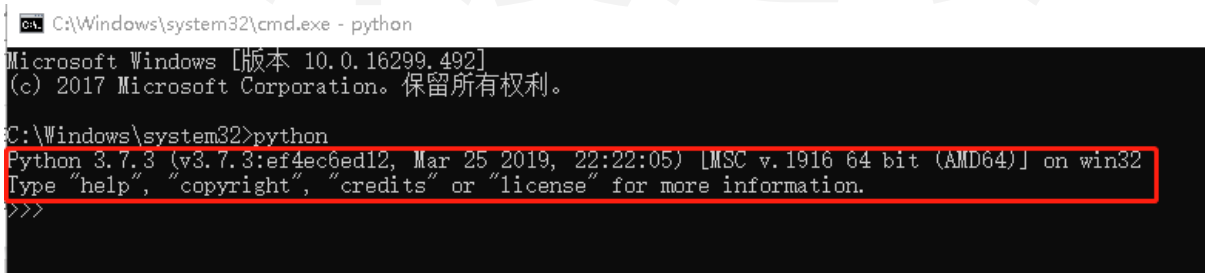
打开官网 https://www.python.org/downloads/windows/ 下载中心



安装过程中需要注意的点：



- 测试安装是否成功 windows --> 运行 --> 输入cmd ， 然后回车，弹出cmd程序，输入python,如果能进入交互环境 （见下图），代表安装成功。
-



- 多版本共存演示 注意：在安装目录下找到python.exe,拷贝一份，命名为python2.exe或python3.exe，一定要保留原版，因为pip工具会调用它。

6. 写程序的两种方式

- 进入解释器的交互式模式：调试方便，无法永久保存代码
- 脚本文件的方式（使用nodpad++演示）：永久保存代码

PS:Notepad++是 Windows操作系统下的一套文本编辑器(软件版权许可证: GPL)，有完整的中文化接口及支持多国语言编写的功能,在百度中输入Notepad即可进行下载

强调：python解释器执行程序是解释执行，即打开文件读内容，因此文件的后缀名没有硬性限制，但通常定义为.py结尾

用Python写一个Hello World程序

```
print('Hello World')
```

比较一下其他语言Hello World的写法

```
# C++
#include <iostream>
 int main(void)
 {
  std::cout<<"Hello world";
 }
# C
#include <stdio.h>
int main(void)
{
printf("\nhello world!");
return 0;
}
# JAVA
public class HelloWorld{
  // 程序的入口
  public static void main(String args[]){
    // 向控制台输出信息
    System.out.println("Hello World!");
  }
}
# PHP
<?php
             echo "hello world!";
?>

# GO
package main
import "fmt"
func main(){
    fmt.Printf("Hello World!\n God Bless You!");
}
# 精通各种语言的hello world
```

安装Python专用 IDE Pycharm

为何要用IDE> *到现在为止，我们也是写过代码的人啦，但你有没有发现，每次写代码要新建文件、写完保存时还要选择存放地点，执行时还要切换到命令行调用python解释器，好麻烦呀，能否一气呵成，让我简单的写代码？此时开发工具IDE上场啦，一个好的IDE能帮你大大提升开发效率。*很多语言都有比较流行的开发工具，比如JAVA 的Eclipse, C#,C++的VisualStudio, Python的是啥呢？ Pycharm，最好的Python开发IDE

安装
下载地址:https://www.jetbrains.com/pycharm/download 选择Professional 专业版
进入> Pycharm官网根据提示下载专业版即可，Comunnity社区版是免费的，但支持的功能不多，所以还是用专业版。注册完成后启动，会让你先创建一个项目，其实就是一个文件夹，我们以后的代码都存在这里面。
安装教程参考： https://blog.csdn.net/ling_mochen/article/details/79314118

四 变量与常量

1. 变量的概念

计算机工作的过程直白讲就是对数据的增、删、改、查操作，那么数据一定是变化的，我们要存储变化的数据就应该用“变量”。

什么是变量

- 变量即变化的量，核心是“变”与“量”二字，变即变化，量即衡量状态。

为什么要有变量

- 程序执行的本质就是一系列状态的变化，变是程序执行的直接体现，所以我们需要有一种机制能够反映或者说是保存下来程序执行时的状态以及状态的变化。 #比如： 英雄的等级为1，打怪升级(变)为10 ；僵尸的存活状态True，被植物打死了，于是变为False ； 人的名字为Albert，也可以修改为AlbertMa。

Python如何定义变量

- 变量名(相当于门牌号，指向值所在的空间)，等号，变量值，是从右到左赋值。

```
name='Albert'
sex='male'
age=18
level=10
```

2. Python变量的定义规范

- 变量名只能是 字母、数字或下划线的任意组合
- 变量名的第一个字符不能是数字
- 以下关键字不能成为变量名

> ['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']

3. Python变量定义方式：

驼峰体

驼峰式命名法就是当变量名或函式名是由一个或多个单词连结在一起。

驼峰体分两种：

1 大驼峰 所有单词首字母全都大写 AgeOfTeacher

2 小驼峰 除第一个单词外其他所有单词首字母大写 ageOfTeacher。

```
#大驼峰
AgeOfAlbert = 18
NumberOfStudents = 80
#小驼峰
ageOfTeacher = 28
```

下划线(推荐使用)

```
age_of_albert = 18
number_of_students = 80
```

定义变量名不好的方式

1. 变量名为中文、拼音
2. 变量名词不达意
3. 变量名用中文，Python居然支持用中文做变量名，这真是颠覆了我的世界观，也许就是为了照顾我国一些程序猿而做的优化吧

值得一提的是：在公司里项目很大的时候，我们往往会命名的变量名比较长，这是一方面是为了变量名不重复，另一方面是为了能看到变量名就知道数据的含义

有一句话给大家：变量即逻辑

定义变量会有：id（唯一标识号），type（变量类型），value（变量值）

- 等号比较的是value，
- is比较的是id

强调：

1. id相同，意味着type和value必定相同
2. value相同type肯定相同，但id可能不同,如下

```
>>x='Info Albert:18'
>>y='Info Albert:18'
>>x == y
>>True
>>x is y
>>False
```

如果自己用电脑终端或者Pycharm测试可能会出现 x is y 结果是True的情况，不用担心，这是Pycharm或者终端环境做的内存优化，当变量比较长的时候就没问题了。

4. 常量

常量即指不变的量，如pai 3.141592653..., 或在程序运行过程中不会改变的量，举例：日本首相的年龄是一个变量，加入他今天突然挂了，那么这个年龄就不会在改变了，那就是常量。在Python中没有一个专门的语法代表常量，程序员约定俗成用变量名全部大写代表常量

```
JAPANESE_PRIME_MINISTER_AGE = 56
```

在c语言中有专门的常量定义语法，const int count = 60;一旦定义为常量，更改即会报错

五 基本数据类型

1. 说明

Python属于强类型的动态脚本语言：不允许不同类型相加动态：定义变量不用数据类型声明，且确定一个变量的类型是第一次给他赋值的时候

2. 字符串（str）

刚才我们定义人名使用的是 name = ‘Albert’来定义的，形如 ' ' 或者 " " 或者 """ """ 单引号，双引号或者三引号中间写内容的这种定义方式就是 str类型，我们叫做字符串类型

```
print('Hello world')   # 这就是一个字符串类型
```

定义的时候是把等号右侧赋值给等号左侧的变量
我们不仅要知道他的数据类型是str，在学习一个数据类型的时候还要知道他是一个可变类型还是不可变类型。

- 不可变类型：当变量值改变的时候id也会一起变化，相当于重新开辟一块内存空间，给变量重新赋值，原来的值是不可变的，str就是一个不可变类型
- 可变类型：当变量值改变的时候，id可以保持不变，就相当于是在原来的基础上修改，位置还是没有变的

> 可变类型可以理解为一个大房子的地方不动，给这个房子装修一下就是做了修改，而不可变类型就是这个房子不能重新装修，你要改可以，你在别的地方再盖一个房子，你再住进去。

3. 整型（int）

我们定义一个人的年龄的时候 使用age = 18 来定义的，等号后面直接➕整数数字，那么这个变量的数据类型就是int类型，我们叫整型，整形是一个不可变类型

4. 浮点型（float）

当我们定一个人的身高的时候，使用height = 1.83 来定义，同理这是float类型，我们叫浮点型，浮点型是一个不可变类型，浮点型直观意义上来讲就是需要把值精确到小数的变量。

5. 列表（list）

当我们需要存储一个班级里面所有学生的名字的时候，就是用一个变量来存储多个值，以上三种数据类型都只能存储一个值，这种情况我们可以只用list这种数据类型来完成，我们叫列表。定义形式：中括号内，多个元素用逗号分割，每个元素可以是任意的数据类型，列表是一个可变类型

```
names = ['albert','james','kebo']   # 存储一个班级的多个学生
information = ['albert',18,1.83]   # 存储一个人信息
# 存储多人信息
informations = [['albert',18,1.83],['james',34,2.03],['kebo',40,1.98]]
```

6. 元组（tuple）

另外一种和list类似的数据类型是tuple，我们叫元组，他的用法和list非常相似，只是列表可以修改，而元组不能修改，只能查看，他的定义方式是小括号内，多个元素用逗号分割，每个元素可以是任意的数据类型。以上代码中括号改成小括号就可以了，注意是英文的小括号()，元组是一个不可变类型

7. 字典（dict）

上面代码存储多个人的信息的时候，你们大概的能看出来我是存储的每个人的姓名，年龄和身高，但是不明确，我们不能用猜的，要解决这个问题就是定义的时候就说明白，这个时候字典来了，定义形式：花括号内多个元素用逗号分割，每个元素按照key: value的形式，需要注意的是：字典的key只能用不可变类型，我们一般使用字符串，字典是一个可变类型。

```
information = {'name':'albert','age':18,'height':1.83}   # 存储一个人信息
informations = [
        {'name':'albert','age':18,'height':1.83},
        {'name':'james','age':34,'height':2.03},
        {'name':'kebo','age':40,'height':1.98}
]   # 存储多人信息
```

8. 集合（set）

在NBA有400多位球员，如果我们不考虑先后顺序要存储这400多位球员，肯定是不能重复的，每位球员只能算一个，上面的列表和元组是可以重复的，我们只需要知道“Albert”肯定是不在这里面的就可以了。定义形式：花括号内多个元素用逗号分割，每个元素可以是任意的数据类型，但是不能重复，集合是一个可变类型。

```
NBA_players = {'James', 'Kobe', 'Jordan', }
```

9. 布尔（Bool）

在NBA中有众多的球员，但是有的人可能一辈子也打不了NBA了，如果我们想要判断一个人是否是NBA球员就可以用布尔类型，它只有两个值，True和False。
基本的数据类型，我们就暂时先了解到这里，后面的章节会对这些数据类型的使用进行更加详细的讲解。

10. 补充(非常重要)

关于变量的命名在这里做一点补充，这虽然简单，但是非常重要。
推荐大家使用『匈牙利命名法』，这也是在工作中最为规范的变量命名方式，有些时候看一个人写的代码，只需要看他写的变量名就能粗略判断出他的水平了。
简而言之，匈牙利命名法就是把变量的『类型』缩写，放到变量名的最前面。关键在于，这里说的变量『类型』，指的是那些和你的代码业务逻辑相关的类型。比如，在你的代码中有两个变量：students 和 teachers，他们分别代表的是用来存储学生的集合与用来存储老师的列表，使用『匈牙利命名法』后，可以把这两个名字改写成这样：

```
students -> set_students
teachers -> list_teachers
```

很多情况下，使用『匈牙利命名法』是个不错的主意，因为它可以改善你的代码可读性，尤其在那些变量众多、同一类型多次出现时，注意不要滥用就好。

六 注释

随着学习的深入，用不了多久，你就可以写复杂的上千甚至上万行的代码啦，有些代码你花了很久写出来，过了些天再回去看，发现竟然看不懂了，这太正常了。 另外，你以后在工作中会发现，一个项目多是由几个甚至几十个开发人员一起做，你要调用别人写的代码，别人也要用你的，如果代码不加注释，你自己都看不懂，更别说别人了，这是会挨打的。所以为了避免这种尴尬的事情发生，一定要增加你代码的可读性。

#号后边的任何数据在代码运行的阶段不会被输出

代码注释分为三种：

1. 单行注释，注释单独占一行，以 #开头，打一个空格，后面写注释的内容

```
a = 1
# a += 1 这就是注释的#使用方式
```

2. 行内注释，在代码尾部，打两个空格，然后打一个 #，接着再打一个空格，后面写注释的内容。

```
a += 1 # 行内注释
```

3. 多行注释， 可以用三对双引号`"""注释内容 """`，或者三对单引号中间写注释内容，和明显多行注释就是可以注释多行，当你的注释比较长的使用，写很长的一行不便于阅读，这时，我们推荐使用多行注释。

```
"""
hello, world!
"""
```

代码注释的原则：

- 不用全部加注释，只需要在自己觉得重要或不好理解的部分加注释即可
- 注释可以用中文或英文，但不要用拼音，也不要用中英文混杂

注意(非常重要)：
注释你们现在可能谁都会写，但其实在工作中，除了一些高标准的公司之外，能够规范写注释的人非常少，写注释和写规范的注释是代表一位程序员编程素养的最简单的方式。

七 文件头

注意：既然是文件头，那么自然是要放在文件顶部，这也是开发规范的一些细节

```
#!/usr/bin/env python  # 指定解释器
# -*- coding: utf-8 -*-  # 指定文件字符编码
```

如果每次写代码我们都自己去写的话有点浪费时间，我们可以自己先在Pycharm上面配置好，以后每次新建一个文件自动就会添加文件头。MacOS系统的用户可以按照先点击Pycharm左上角按照：Pycharm--Preferences--Editor--File and Code Templates--Python Script 的顺序找到一块空白的区域，把以下代码粘贴进去，Windows系统的用户可以按照：File--Settings--Editor--File and Code Templates--Python Script，执行同样的操作。注意：作者那一行代码可以改成你自己的名字。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Time    : ${DATE} ${TIME}
# @Author  : Albert
# @File    : ${NAME}.py
```

八 实现用户交互

文章写到这里才正式进入本章标题的内容，前面都是铺垫

很早以前，我们去银行取钱，需要有一个银行业务员等着我们把自己的账号密码输入给他，然后他去进行验证，成功后，我们再将取款金额输入/告诉他骄傲的现代人，会为客户提供一台ATM机（就是一台计算机），让ATM机跟用户交互，从而取代人力。然而机器是死的，我们必须为其编写程序来运行，这就要求我们的编程语言中能够有一种能与用户交互，接收用户输入数据的机制。

用户交互就是程序等待用户输入数据之后，在执行下一步的程序，我们使用 input 来完成这个操作。

```
name = input('请输入姓名:')   # 我把输入的内容赋值给name这个变量
print('你好' + name)   # 当用户输入完成自己的姓名之后就会打印出来 你好XXX
```

Python2 与Python3 的区别

- 在python3中 input：用户输入任何值，都存成字符串类型
- 在python2中 input：用户输入什么类型，就存成什么类型
- Python2 中的raw_input：等于python3的input

需要注意的是：当需要输入一个数字的时候，考虑到我们会用这个数字做计算，而Python3的input会默认转成str，所有我们需要把用户输入的数据转化成int

```
age = input('请输入年龄')
age = int(age)  # 把字符串age转化成int类型age用于计算
```