[返回主页](https://www.cnblogs.com/jin-xin/)[**python基础一**](https://www.cnblogs.com/jin-xin/articles/7459977.html)

**一，Python介绍**

**1. python的出生与应用**

　　python的创始人为吉多·范罗苏姆（Guido van Rossum）。1989年的圣诞节期间，吉多·范罗苏姆（中文名字：龟叔）为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序，作为ABC语言的一种继承。

（龟叔：2005年加入谷歌至2012年，2013年加入Dropbox直到现在，依然掌握着Python发展的核心方向，被称为仁慈的独裁者）。



2017年7月的TIOBE排行榜，Python已经占据第四的位置， Python崇尚优美、清晰、简单，是一个优秀并广泛使用的语言。



由上图可见，Python整体呈上升趋势，反映出Python应用越来越广泛并且也逐渐得到业内的认可！！！

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。

**目前Python主要应用领域：**

* **云计算**: 云计算最火的语言， 典型应用OpenStack
* **WEB开发**: 众多优秀的WEB框架，众多大型网站均为Python开发，Youtube, Dropbox, 豆瓣。。。， 典型WEB框架有Django
* **科学运算、人工智能**: 典型库NumPy, SciPy, Matplotlib, Enthought librarys,pandas
* **系统运维**: 运维人员必备语言
* **金融**：量化交易，金融分析，在金融工程领域，Python不但在用，且用的最多，而且重要性逐年提高。原因：作为动态语言的Python，语言结构清晰简单，库丰富，成熟稳定，科学计算和统计分析都很牛逼，生产效率远远高于c,c++,java,尤其擅长策略回测
* **图形GUI**: PyQT, WxPython,TkInter

**Python在一些公司的应用：**

* 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实现的
* 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开发的
* 豆瓣: 公司几乎所有的业务均是通过Python开发的
* 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开发的
* 除上面之外，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。

**python发展史**

* 1989年，为了打发圣诞节假期，Guido开始写Python语言的编译器。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。
* 1991年，第一个Python编译器诞生。它是用C语言实现的，并能够调用C语言的库文件。从一出生，Python已经具有了：类，函数，异常处理，包含表和词典在内的核心数据类型，以及模块为基础的拓展系统。
* Granddaddy of Python web frameworks, Zope 1 was released in 1999
* Python 1.0 - January 1994 增加了 [lambda](https://en.wikipedia.org/wiki/Lambda_calculus), [map](https://en.wikipedia.org/wiki/Map_%28higher-order_function%29), [filter](https://en.wikipedia.org/wiki/Filter_%28higher-order_function%29) and [reduce](https://en.wikipedia.org/wiki/Fold_%28higher-order_function%29).
* Python 2.0 - October 16, 2000，加入了内存回收机制，构成了现在Python语言框架的基础
* Python 2.4 - November 30, 2004, 同年目前最流行的WEB框架Django 诞生
* Python 2.5 - September 19, 2006
* Python 2.6 - October 1, 2008
* Python 2.7 - July 3, 2010
* In November 2014, it was announced that Python 2.7 would be supported until 2020, and reaffirmed that there would be no 2.8 release as users were expected to move to Python 3.4+ as soon as possible
* Python 3.0 - December 3, 2008
* Python 3.1 - June 27, 2009
* Python 3.2 - February 20, 2011
* Python 3.3 - September 29, 2012
* Python 3.4 - March 16, 2014
* Python 3.5 - September 13, 2015
* Python 3.6 - December 16,2016

**2. python是什么编程语言。**

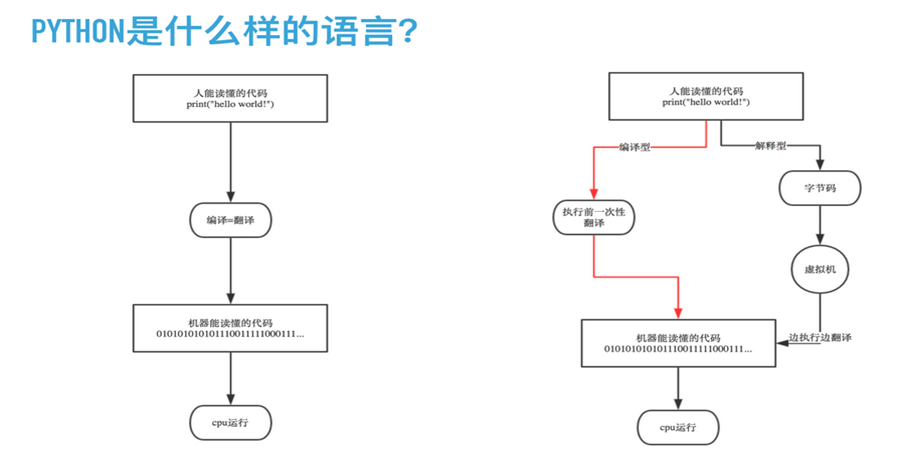
*编程语言主要从以下几个角度为进行分类，编译型和解释型、静态语言和动态语言、强类型定义语言和弱类型定义语言，每个分类代表什么意思呢，我们一起来看一下。*

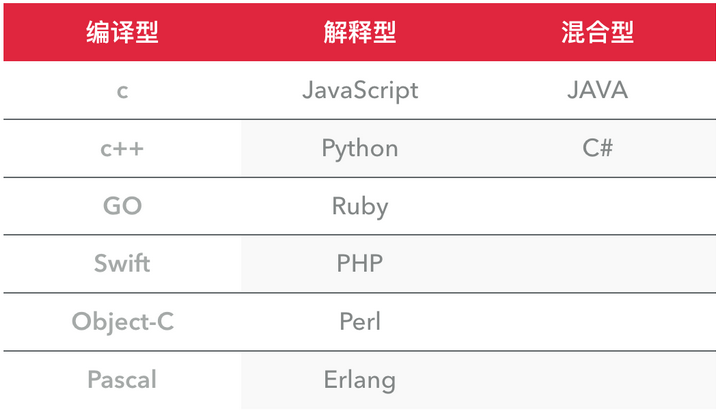
**2.1 编译型与解释型。**

**编译器**是把源程序的每一条语句都编译成机器语言,并保存成二进制文件,这样运行时计算机可以直接以机器语言来运行此程序,速度很快;

而**解释器**则是只在执行程序时,才一条一条的解释成机器语言给计算机来执行,所以运行速度是不如编译后的程序运行的快的.

这是因为计算机不能直接认识并执行我们写的语句,它只能认识机器语言(是二进制的形式)





**编译型**  
优点：编译器一般会有预编译的过程对代码进行优化。因为编译只做一次，运行时不需要编译，所以编译型语言的程序执行效率高。可以脱离语言环境独立运行。  
缺点：编译之后如果需要修改就需要整个模块重新编译。编译的时候根据对应的运行环境生成机器码，不同的操作系统之间移植就会有问题，需要根据运行的操作系统环境编译不同的可执行文件。

**解释型**  
优点：有良好的平台兼容性，在任何环境中都可以运行，前提是安装了解释器（虚拟机）。灵活，修改代码的时候直接修改就可以，可以快速部署，不用停机维护。

缺点：每次运行的时候都要解释一遍，性能上不如编译型语言。

**2.2动态语言和静态语言**  
通常我们所说的动态语言、静态语言是指动态类型语言和静态类型语言。

（1）动态类型语言：动态类型语言是指在运行期间才去做数据类型检查的语言，也就是说，在用动态类型的语言编程时，永远也不用给任何变量指定数据类型，该语言会在你第一次赋值给变量时，在内部将数据类型记录下来。Python和Ruby就是一种典型的动态类型语言，其他的各种脚本语言如VBScript也多少属于动态类型语言。

（2）静态类型语言：静态类型语言与动态类型语言刚好相反，它的数据类型是在编译其间检查的，也就是说在写程序时要声明所有变量的数据类型，C/C++是静态类型语言的典型代表，其他的静态类型语言还有C#、JAVA等。

**2.3强类型定义语言和弱类型定义语言**

（1）强类型定义语言：强制数据类型定义的语言。也就是说，一旦一个变量被指定了某个数据类型，如果不经过强制转换，那么它就永远是这个数据类型了。举个例子：如果你定义了一个整型变量a,那么程序根本不可能将a当作字符串类型处理。强类型定义语言是类型安全的语言。

（2）弱类型定义语言：数据类型可以被忽略的语言。它与强类型定义语言相反, 一个变量可以赋不同数据类型的值。

强类型定义语言在速度上可能略逊色于弱类型定义语言，但是强类型定义语言带来的严谨性能够有效的避免许多错误。另外，“这门语言是不是动态语言”与“这门语言是否类型安全”之间是完全没有联系的！  
例如：Python是动态语言，是强类型定义语言（类型安全的语言）; VBScript是动态语言，是弱类型定义语言（类型不安全的语言）; JAVA是静态语言，是强类型定义语言（类型安全的语言）。

通过上面这些介绍，我们可以得出，**python是一门动态解释性的强类型定义语言。**

**3. python的优缺点。**

先看优点

1. Python的定位是“优雅”、“明确”、“简单”，所以Python程序看上去总是简单易懂，初学者学Python，不但入门容易，而且将来深入下去，可以编写那些非常非常复杂的程序。
2. 开发效率非常高，Python有非常强大的第三方库，基本上你想通过计算机实现任何功能，Python官方库里都有相应的模块进行支持，直接下载调用后，在基础库的基础上再进行开发，大大降低开发周期，避免重复造轮子。
3. 高级语言————当你用Python语言编写程序的时候，你无需考虑诸如如何管理你的程序使用的内存一类的底层细节
4. 可移植性————由于它的开源本质，Python已经被移植在许多平台上（经过改动使它能够工 作在不同平台上）。如果你小心地避免使用依赖于系统的特性，那么你的所有Python程序无需修改就几乎可以在市场上所有的系统平台上运行
5. 可扩展性————如果你需要你的一段关键代码运行得更快或者希望某些算法不公开，你可以把你的部分程序用C或C++编写，然后在你的Python程序中使用它们。
6. 可嵌入性————你可以把Python嵌入你的C/C++程序，从而向你的程序用户提供脚本功能。

再看缺点：

1. 速度慢，Python 的运行速度相比C语言确实慢很多，跟JAVA相比也要慢一些，因此这也是很多所谓的大牛不屑于使用Python的主要原因，但其实这里所指的运行速度慢在大多数情况下用户是无法直接感知到的，必须借助测试工具才能体现出来，比如你用C运一个程序花了0.01s,用Python是0.1s,这样C语言直接比Python快了10倍,算是非常夸张了，但是你是无法直接通过肉眼感知的，因为一个正常人所能感知的时间最小单位是0.15-0.4s左右，哈哈。其实在大多数情况下Python已经完全可以满足你对程序速度的要求，除非你要写对速度要求极高的搜索引擎等，这种情况下，当然还是建议你用C去实现的。
2. 代码不能加密，因为PYTHON是解释性语言，它的源码都是以名文形式存放的，不过我不认为这算是一个缺点，如果你的项目要求源代码必须是加密的，那你一开始就不应该用Python来去实现。
3. 线程不能利用多CPU问题，这是Python被人诟病最多的一个缺点，GIL即全局解释器锁（Global Interpreter Lock），是[计算机程序设计语言](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E8%AF%AD%E8%A8%80" \o "计算机程序设计语言)[解释器](http://zh.wikipedia.org/wiki/%E8%A7%A3%E9%87%8A%E5%99%A8)用于[同步](http://zh.wikipedia.org/wiki/%E5%90%8C%E6%AD%A5)[线程](http://zh.wikipedia.org/wiki/%E7%BA%BF%E7%A8%8B)的工具，使得任何时刻仅有一个线程在执行，Python的线程是操作系统的原生线程。在Linux上为pthread，在Windows上为Win thread，完全由操作系统调度线程的执行。一个python解释器进程内有一条主线程，以及多条用户程序的执行线程。即使在多核CPU平台上，由于GIL的存在，所以禁止多线程的并行执行。关于这个问题的折衷解决方法，我们在以后线程和进程章节里再进行详细探讨。

当我们编写Python代码时，我们得到的是一个包含Python代码的以.py为扩展名的文本文件。要运行代码，就需要Python解释器去执行.py文件。

由于整个Python语言从规范到解释器都是开源的，所以理论上，只要水平够高，任何人都可以编写Python解释器来执行Python代码（当然难度很大）。事实上，确实存在多种Python解释器。

**4. python的种类。**

**CPython**

当我们从[Python官方网站](https://www.python.org/)下载并安装好Python 3.6后，我们就直接获得了一个官方版本的解释器：CPython。这个解释器是用C语言开发的，所以叫CPython。在命令行下运行python就是启动CPython解释器。

CPython是使用最广的Python解释器。教程的所有代码也都在CPython下执行。

**IPython**

IPython是基于CPython之上的一个交互式解释器，也就是说，IPython只是在交互方式上有所增强，但是执行Python代码的功能和CPython是完全一样的。好比很多国产浏览器虽然外观不同，但内核其实都是调用了IE。

CPython用>>>作为提示符，而IPython用In [序号]:作为提示符。

**PyPy**

PyPy是另一个Python解释器，它的目标是执行速度。PyPy采用[JIT技术](http://en.wikipedia.org/wiki/Just-in-time_compilation)，对Python代码进行动态编译（注意不是解释），所以可以显著提高Python代码的执行速度。

绝大部分Python代码都可以在PyPy下运行，但是PyPy和CPython有一些是不同的，这就导致相同的Python代码在两种解释器下执行可能会有不同的结果。如果你的代码要放到PyPy下执行，就需要了解[PyPy和CPython的不同点](http://pypy.readthedocs.org/en/latest/cpython_differences.html)。

**Jython**

Jython是运行在Java平台上的Python解释器，可以直接把Python代码编译成Java字节码执行。

**IronPython**

IronPython和Jython类似，只不过IronPython是运行在微软.Net平台上的Python解释器，可以直接把Python代码编译成.Net的字节码。

**小结：**

　　Python的解释器很多，但使用最广泛的还是CPython。如果要和Java或.Net平台交互，最好的办法不是用Jython或IronPython，而是通过网络调用来交互，确保各程序之间的独立性。