**编译型语言与解释型语言的区别及各自的优缺点**

编译型语言在程序执行之前，有一个单独的编译过程，将程序翻译成机器语言就不用再进行翻译了。

解释型语言，是在运行的时候将程序翻译成机器语言，所以运行速度相对于编C/C++ 等都是编译型语言，而Java，C#等都是解释型语言。

虽然Java程序在运行之前也有一个编译过程，但是并不是将程序编译成机器语言，而是将它编译成字节码（可以理解为一个中间语言）。  
在运行的时候，由JVM将字节码再翻译成机器语言。  
注：脚本语言一般都有相应的脚本引擎来解释执行。 他们一般需要解释器才能运行。JAVASCRIPT,ASP,PHP,PERL,Nuva都是脚本语言。C/C++编译、链接后，可形成独立执行的exe文件。

### 编译型语言：

1. 编译型语言最大的优势之一就是其执行速度。用C/C++编写的程序运行速度要比用Java编写的相同程序快30%-70%。
2. 编译型程序比解释型程序消耗的内存更少。
3. 不利的一面——编译器比解释器要难写得多。
4. 编译器在调试程序时提供不了多少帮助——有多少次在你的C语言代码中遇到一个“空指针异常”时，需要花费好几个小时来明确错误到底在代码中的什么位置。
5. 可执行的编译型代码要比相同的解释型代码大许多。例如，C/C++的.exe文件要比同样功能的Java的.class文件大很多。
6. 编译型程序是面向特定平台的因而是平台依赖的。
7. 编译型程序不支持代码中实现安全性——例如，一个编译型的程序可以访问内存的任何区域，并且可以对你的PC做它想做的任何事情（大部分病毒是使用编译型语言编写的）
8. 由于松散的安全性和平台依赖性，编译型语言不太适合开发因特网或者基于Web的应用。

### 解释型语言：

1. 解释型语言提供了极佳的调试支持。一名Java程序员只需要几分钟就可以定位并修复一个“空指针异常”，因为Java运行环境不仅指明了异常的性质，而且给出了异常发生位置具体的行号和函数调用顺序（著名的堆栈跟踪信息）。这样的便利是编译型语言所无法提供的。
2. 另一个优势是解释器比编译器容易实现
3. 解释型语言最大的优势之一是其平台独立性
4. 解释型语言也可以保证高度的安全性——这是互联网应用迫切需要的
5. 中间语言代码的大小比编译型可执行代码小很多
6. 平台独立性，以及严密的安全性是使解释型语言成为适合互联网和Web应用的理想语言的2个最重要的因素。
7. 解释型语言存在一些严重的缺点。解释型应用占用更多的内存和CPU资源。这是由于，为了运行解释型语言编写的程序，相关的解释器必须首先运行。解释器是复杂的，智能的，大量消耗资源的程序并且它们会占用很多CPU周期和内存。
8. 由于解释型应用的decode-fetch-execute（解码-抓取-执行）的周期，它们比编译型程序慢很多。
9. 解释器也会做很多代码优化，运行时安全性检查；这些额外的步骤占用了更多的资源并进一步降低了应用的运行速度。