**Python环境建立过程指导手册**

Python是一种面向对象、解释型计算机程序设计语言，由Guido van Rossum于1989年发明，第一个公开发行版发行于1991年。Python是纯粹的自由软件， 源代码和解释器CPython遵循 GPL(GNU General Public License)协议 。Python语法简洁清晰，特色之一是强制用空白符(white space)作为语句缩进。Python具有丰富和强大的库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。常见的一种应用情形是，使用Python快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如3D游戏中的图形渲染模块，性能要求特别高，就可以用C/C++重写，而后封装为Python可以调用的扩展类库。

Python解释器、标准库及扩展包构成Python的基本开发环境，满足学习和软件开发的要求。Python解释器及其标准库可从Python官网下载；其他扩展包可用pip命令等方法安装。

大型软件开发宜采用集成开发工具，常用的有: Eclipse，VS2015等，其中，Eclipse加PyDev插件的方式， 为较多Python程序开发人员使用。此外，还有使用编辑器(如Atom)构建开发环境的方法。

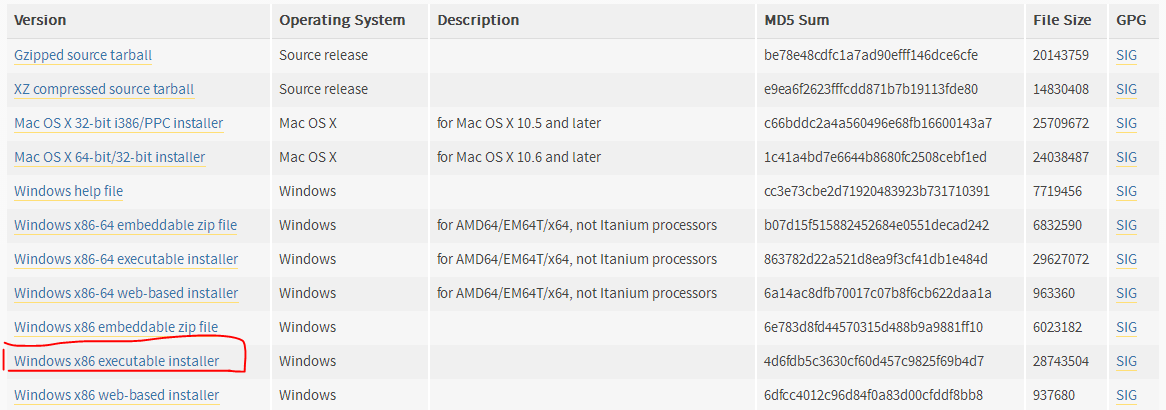
1. **Python基本开发环境建立**

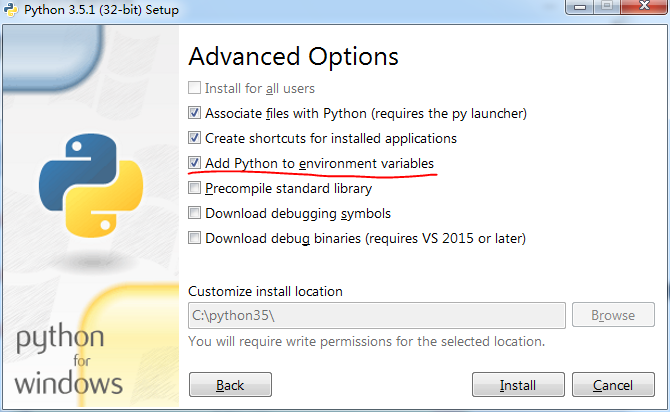
我的电脑是win7系统32位的，从官网：

<http://www.python.org>

下载32位版的Python3软件。下面以Python3.5.1版本安装为例做简单说明

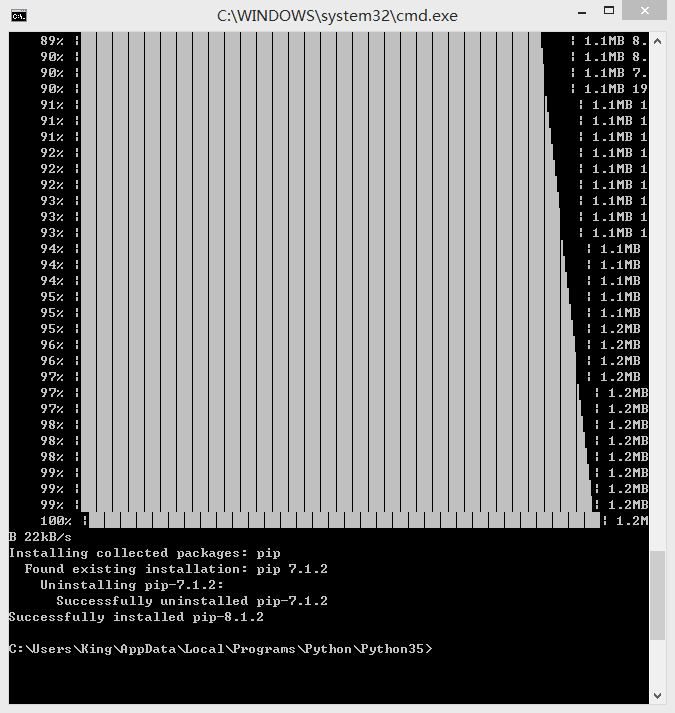
**Windows：** 建议使用定制方式安装: 1) 加环境变量PATH; 2) 自定义安装目录： C:\python35





然后，命令行下更新pip到新版本：

>pip install --upgrade pip



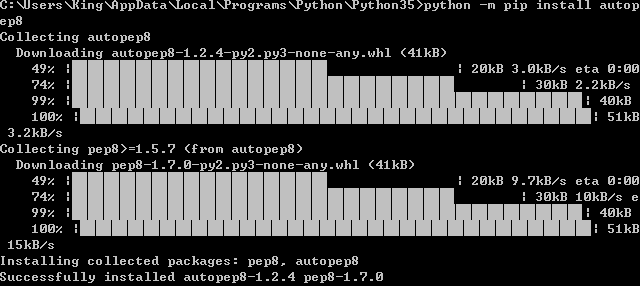
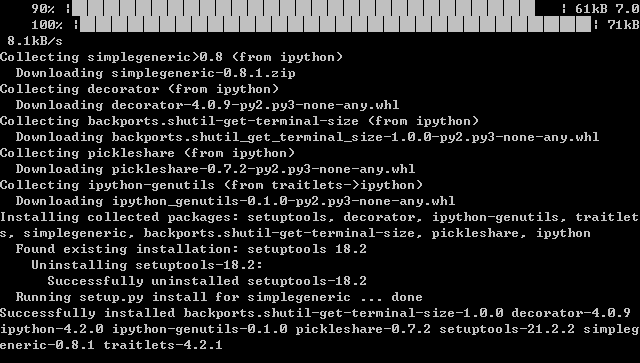
1. **Python扩展包安装**

**2.1 编码规范**

**2.1.1 autopep8**

Autopep8是一个将Python代码自动排版为PEP8风格的小工具。它使用pep8工具来决定代码中的哪部分需要被排版。Autopep8可以修复大部分pep8工具中报告的排版问题。

安装命令：pip install autopep8



**2.1.2 pylint**

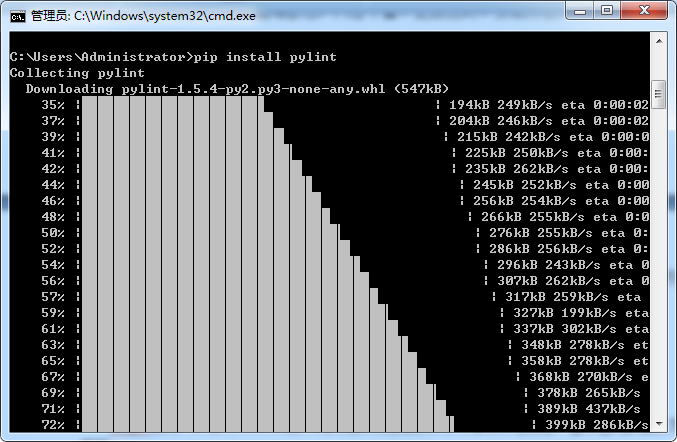
Pylint 是一个 Python 工具，除了平常代码分析工具的作用之外，它提供了更多的功能：如检查一行代码的长度，变量名是否符合命名标准，一个声明过的接口是否被真正实现等等。

Pylint 的一个很大的好处是它的高可配置性，高可定制性，并且可以很容易写小插件来添加功能。

如果运行两次 Pylint，它会同时显示出当前和上次的运行结果，从而可以看出代码质量是否得到了改进。

目前在 eclipse 的 pydev 插件中也集成了 Pylint。

安装命令：pip install pylint



**2.2 科学计算软件包**

**2.2.1 SciPy**

Numpy

Scipy

Matplotlib

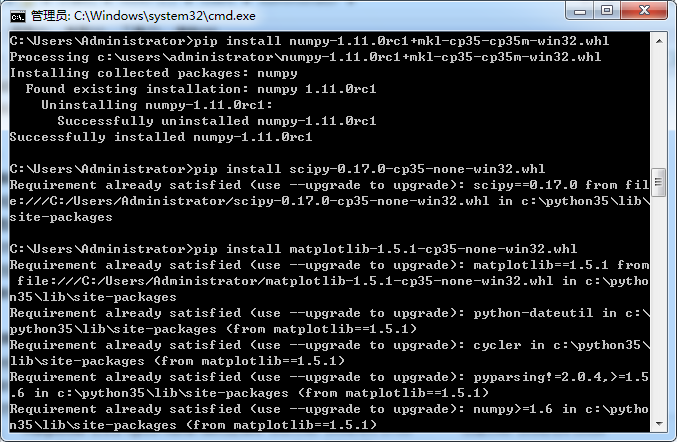
**Window**环境下Scipy 软件包的 安装，建议从加州大学欧文分校

<http://www.lfd.uci.edu/~gohlke/pythonlibs/>

下载编译好的\*.whl，然后，用pip逐个本地安装：

>pip install \*.whl

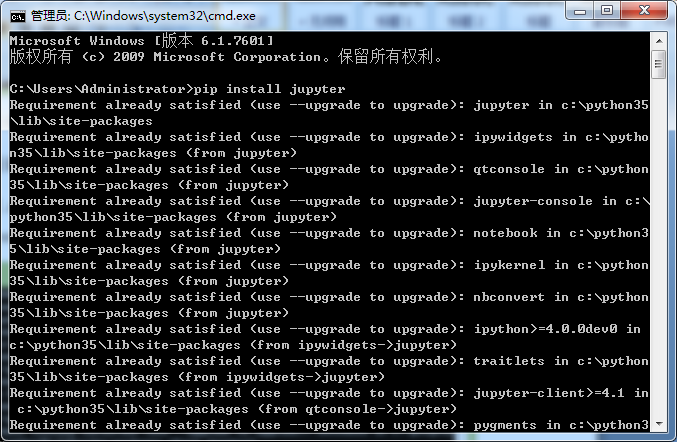
安装过程中发现.whl文件必须放到C:\Users\Administrator文件夹中才能成功安装。



**2.2.2 交互计算 Jupiter**

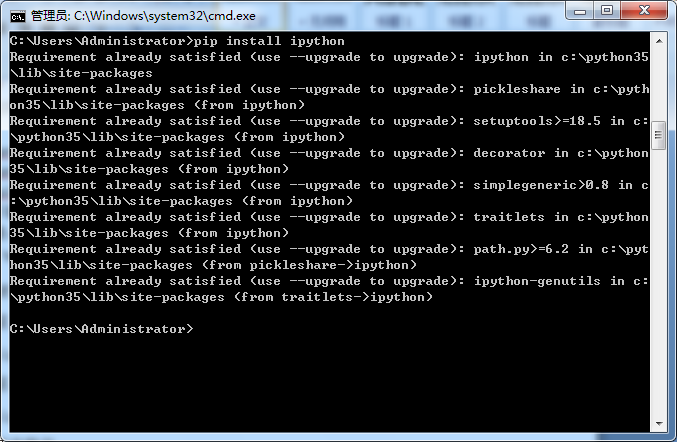
1） 安装Jupiter ：

>pip install jupyter 在线安装



2） 安装Python语言内核

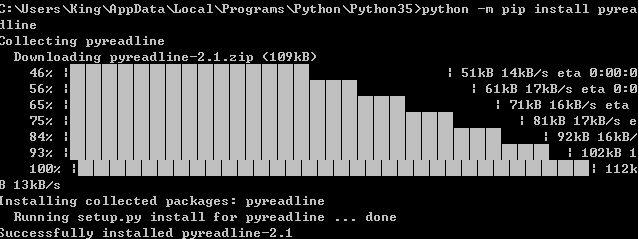
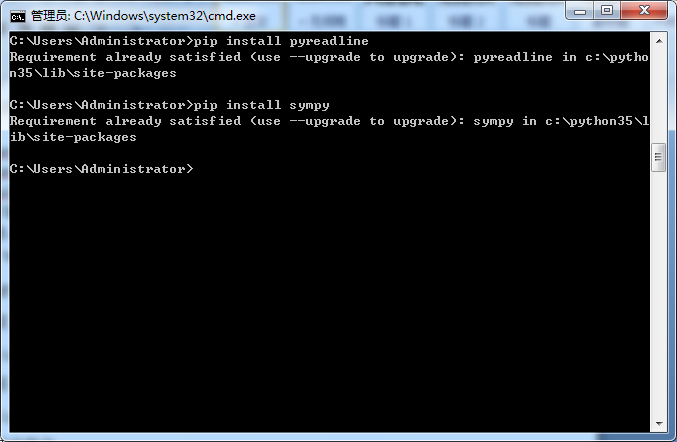
>pip install ipython ，支持Python语言



3） 安装依赖包

>pip install pyreadline

>pip install sympy



4）支持显示数学符号、公式，安装MathJax：

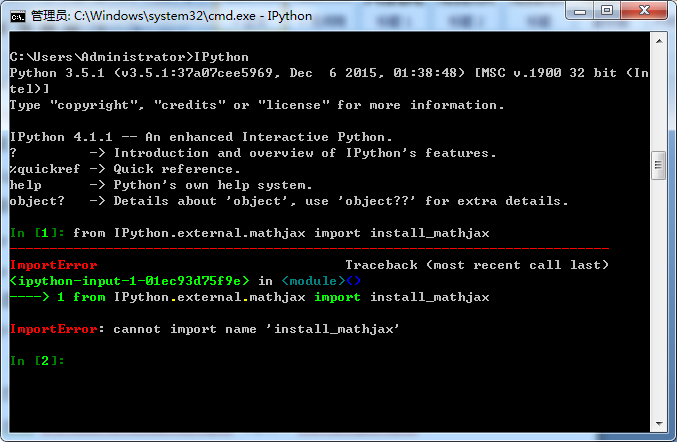
命令行下，

>IPython

打开一个IPython的shell，然后，在其中键入如下代码：

from IPython.external.mathjax import install\_mathjax

install\_mathjax()



结果出现如上问题，待解决。

百度可得该网站

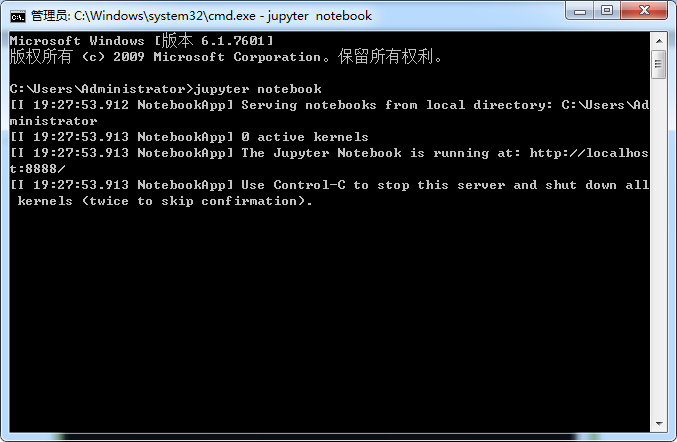
http://www.throwexcept.com/article/1418122387791.html

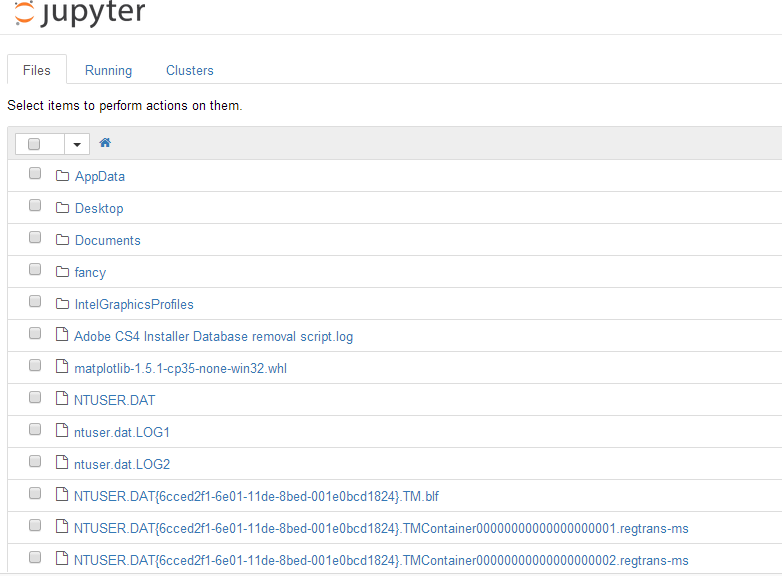
用离线安装法，需要下载mathjax，没能在官网成功下载，因此问题尚未解决。

5） 运行notebook:

在iPython notebook文件所在目录下，打开命令行窗口：

>jupyter notebook





**2.2.3 IF97 for Python**

Windows 32/64位版: 从

<https://github.com/Py03013052/SEUIF97>

下载：SEUIF97.dll和 seuif97.py，或者直接从老师所给的U盘文件夹里找到，然后：

1) SEUIF97.dll拷贝到 c:\windows\system

2) seuif97.py拷贝到 c:\python35\Lib

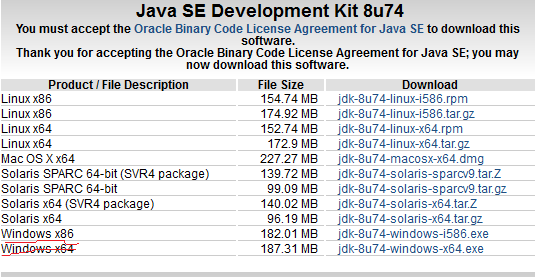
1. **基于Eclipse的Python开发环境**

**3.1 安装Eclipse IDE**

**3.1.1安装JavaSDK**

Eclipse IDE是使用Java开发的，电脑中需要预先安装好Java JRE/JDK软件包，Java JDK的Oracle官方下载地址如下：

[**http://www.oracle.com/technetwork/java/javase/downloads/index.html**](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

****

下载时需要注意操作系统32/64位，选择相应版本下载（\*是Java版本号）：

Windows 64位：\*-windows-i64.exe.

运行下载Oracle的Java安装包（安装时注意配置环境变量），即可安装好Java环境。

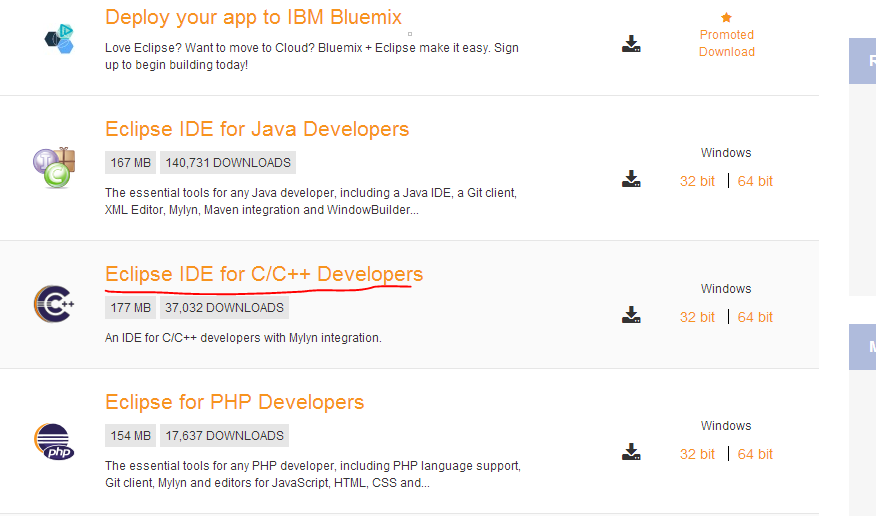


**3.1.2 Eclipse IDE**

　　Eclipse IDE是插件型开发环境，有很多版本可以下载。本课程主要使用Python语言，会涉及C/C++开发，所以，建议下载Eclipse CDT(Eclipse IDE　for　C/C++)版.

Eclipse CDT官方下载地址：

http://www.eclipse.org/downloads/



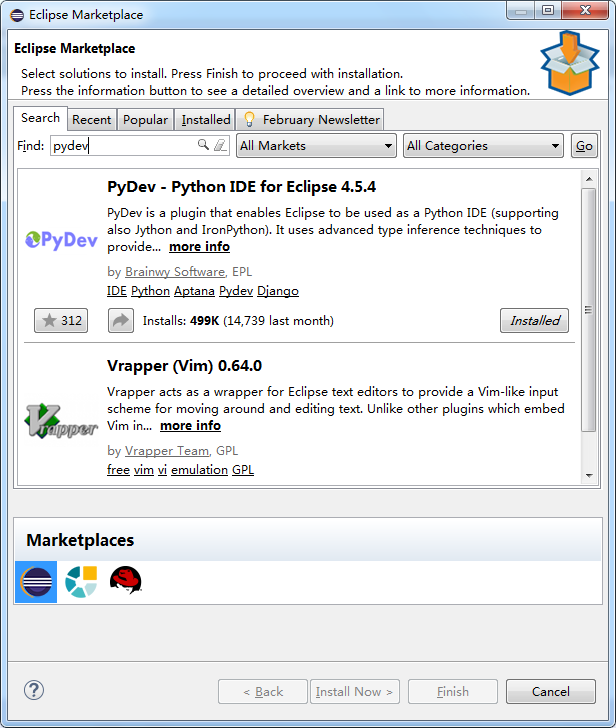
根据操作系统32/64位，下载相应的版本，然后将下载的Eclipse CDT解压到指定目录下，运行解压目录下的：eclipse.exe即可。

**3.2 安装Python开发插件PyDev**

使用Eclipse IDE作为Python开发环境，需要：1） 安装PyDev插件；2）配置解释器为Python3.5；

**3.2.1 安装PyDev插件**

在线安装法：通过Windows->Eclipse Marketplaces进入市场，输入Pydev，找到Pydev安装/更新项目，在线安装即可：



离线安装法：从pydev官网（<http://www.pydev.org/>）提供的下载地址：

http://sourceforge.net/projects/pydev/files/

下载插件包，然后，通过：

Help->Install->New Software->Add->Archive

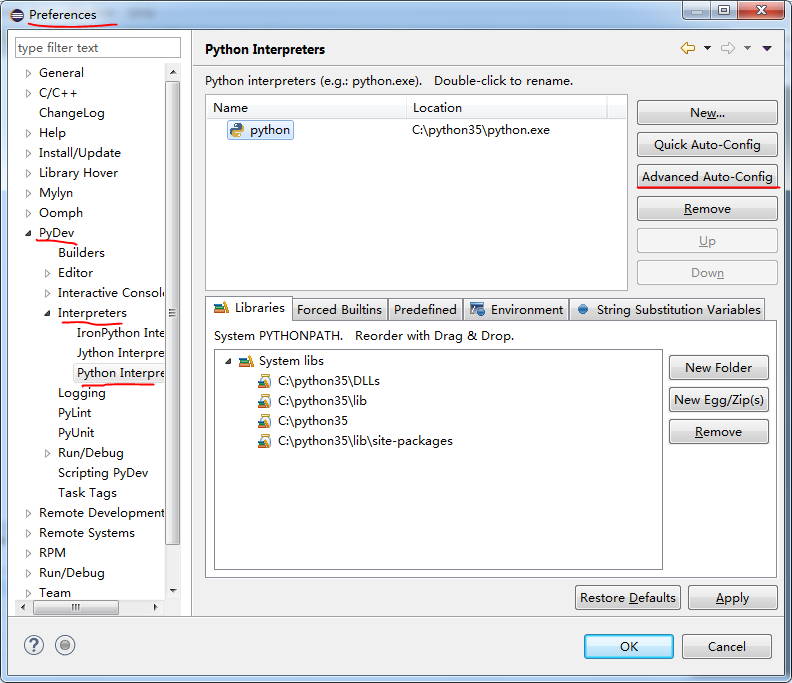
离线安装.



**3.2.2 配置 Python解释器**

安装好后，重新启动。通过：

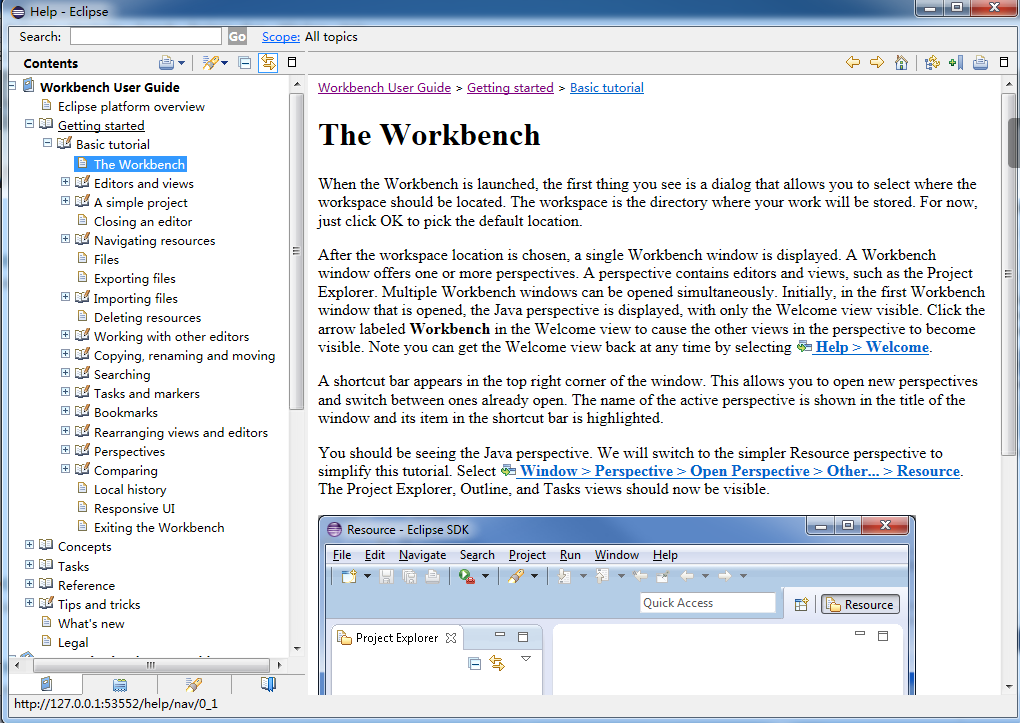
Windows->Preference->Pydev->Interperters->Python Interperter点其中的：Advanced Auto-config配置开发使用的 Python解释器版本:



配置好后切换到Python场景，就可以进行Python开发了。

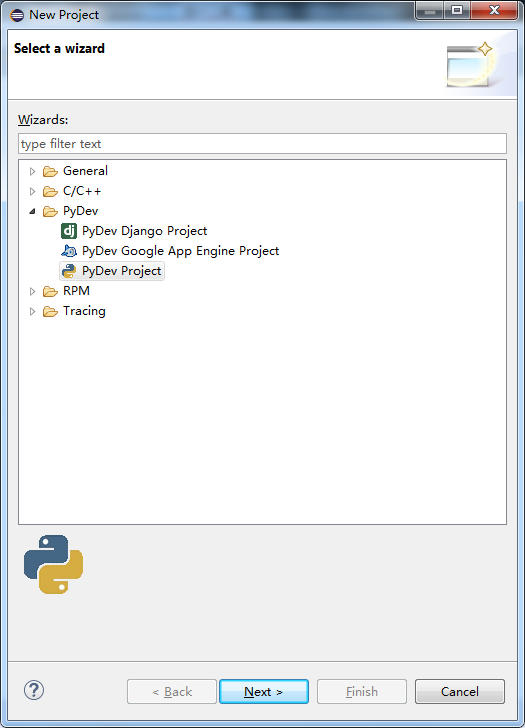
**3.3 联机指南**

**Help** -> **Help Contents**

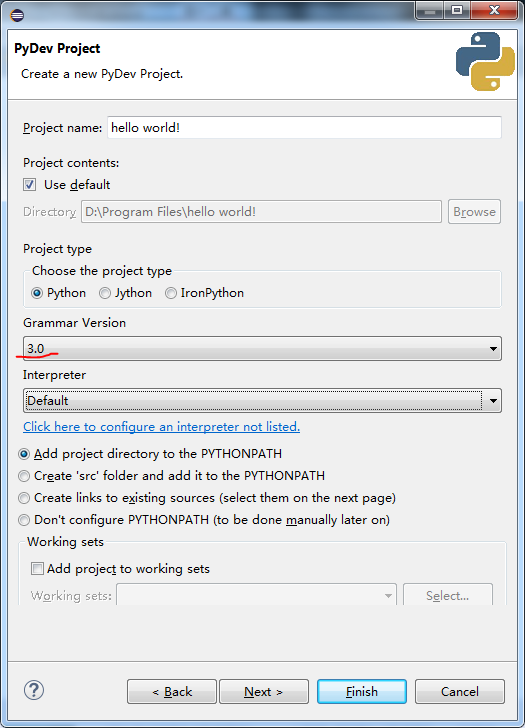


**3.4 创建一个简单的Python工程**

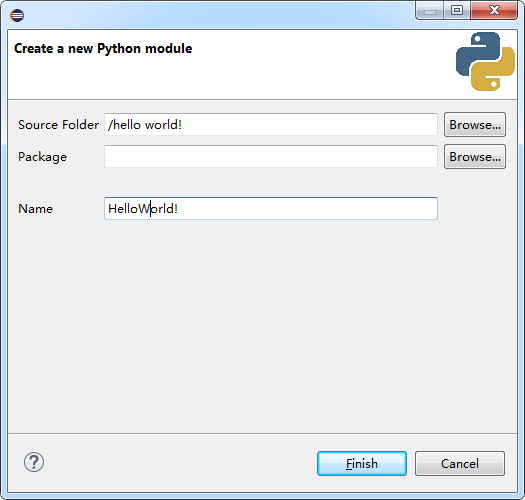
通过File->New->Project，在New Project中选择Pydev->Pydev Project，建立新的工程。

****

选择语法版本为3.0

****

在左侧workplace中选择新建的工程，File->New->Pydev Module以建立可编写窗口。（注：文件名Name中不能有空格）

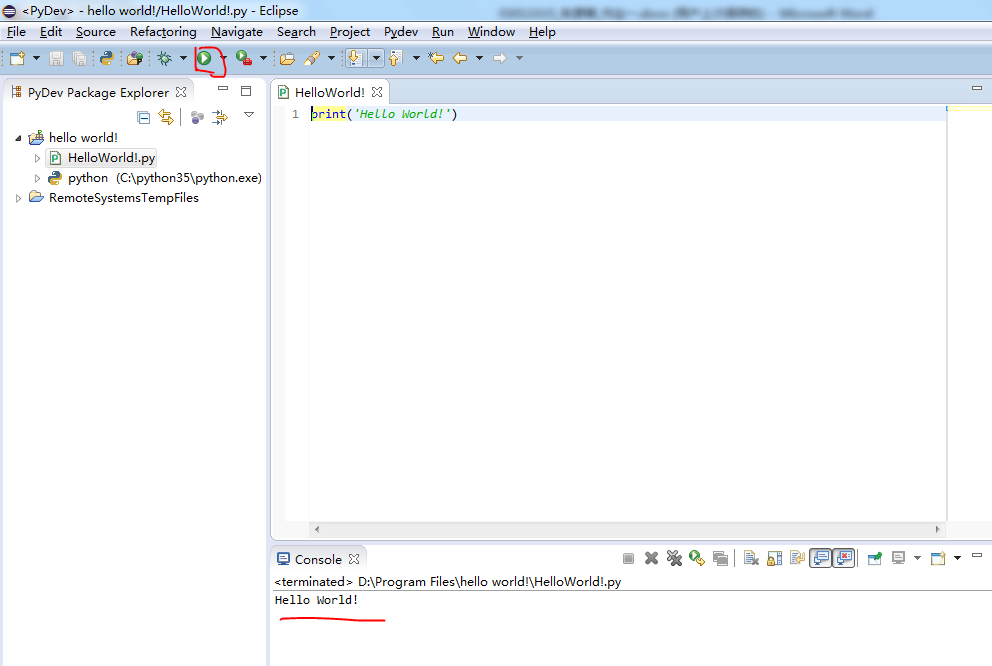
****

将已有的模版内容删除即可做为空工程使用。编写最简单的工程示例Hello World!作为示范

代码内容：

print(*'Hello World!'*)

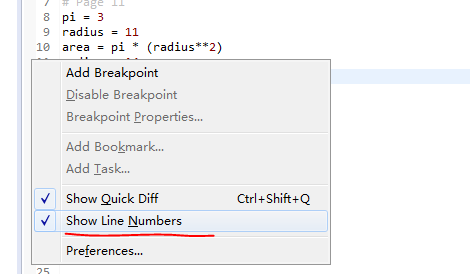
于右上角选择PyDev，并点击左侧的运行按钮运行代码，运行结果显示于下方的console中。（注：命令必须区分大小写，若输入大写Print则无法识别）

****

1. **配置 PyDev提高效率**

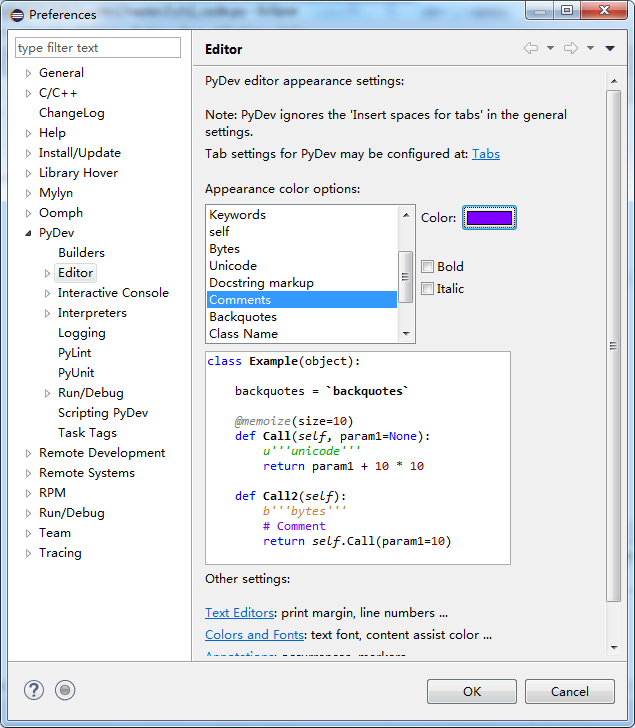
**4.1 显示源码行号**

右键源码的左边缘，选中“Show Line Number

****

**4.2 修改注释颜色提高可读行**

从Window → Preferences→PyDev→Editor,进入配置界面：

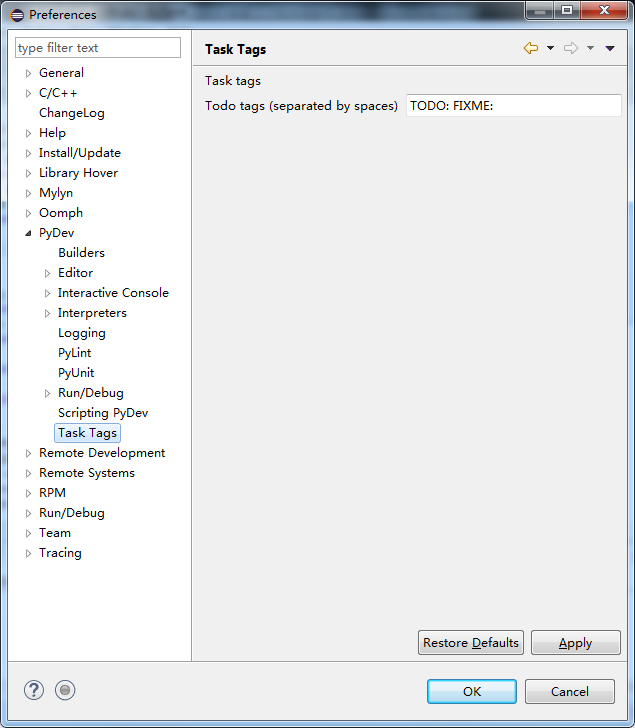
****

**4.3 配置任务标签**

在程序开发过程中可以在代码中标识，当前任务状态，计划开发工作。在代码中标识任务，可以使用任务标签，然后，让开发环境识别到工作空间的任务列表中。

1. **配置任务标签：**

通过PyDev->Task Tags中配置任务标签:



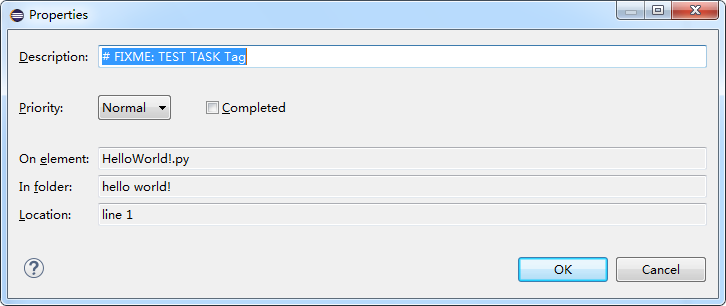
可以根据不同的任务类型，自己增加自定义标签

**2）添加任务标签注释**

在程序源码中加入使用任务标签的注释：

# FIXME: TEST TASK Tag

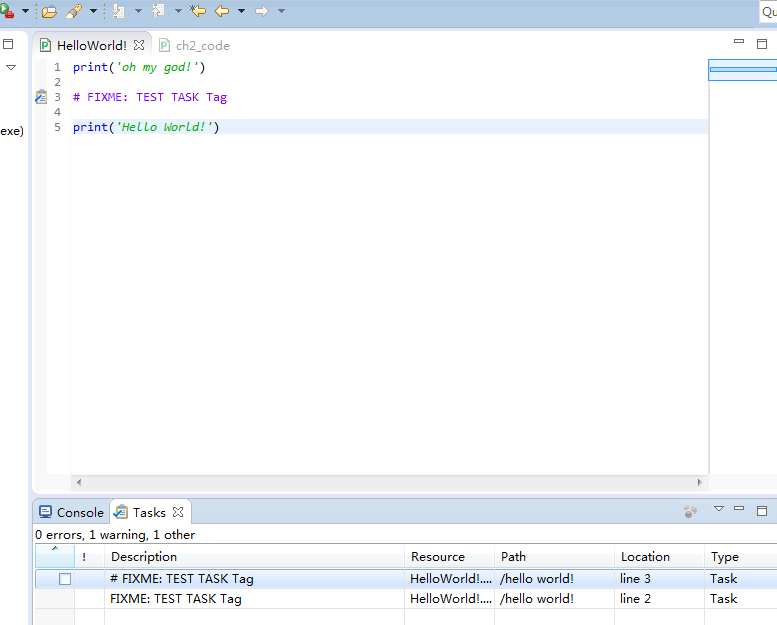
并在该行左侧右键，选择Add Task



**3）识别任务**

保存新修改、运行程序一次或者选择 Project → Clean，

就可将使用任务标签注释加入任务列表。如果当前任务窗口可视，刚加的任务，就会立刻显示在任务窗口中。



如任务窗口没有打开，，可以Window->Show View->Tasks开启任务窗口。

1. **基于PyDev的语言规范静态检查**

PyDev中集成了PEP8,AutoPEP8 和Pylint代码检查功能，这些功能默认状态都是关闭的。

程序开发过程中，要有规范意识，但不可能有很高的规范性，过分注意规范会影响开发进程。这时如果一直开启代码规范检查，经常提示不规范，会对开发形成负面影响，所以，默认关闭是合适的。在程序开发一个阶段结果出来时，进行规范性检查更好。

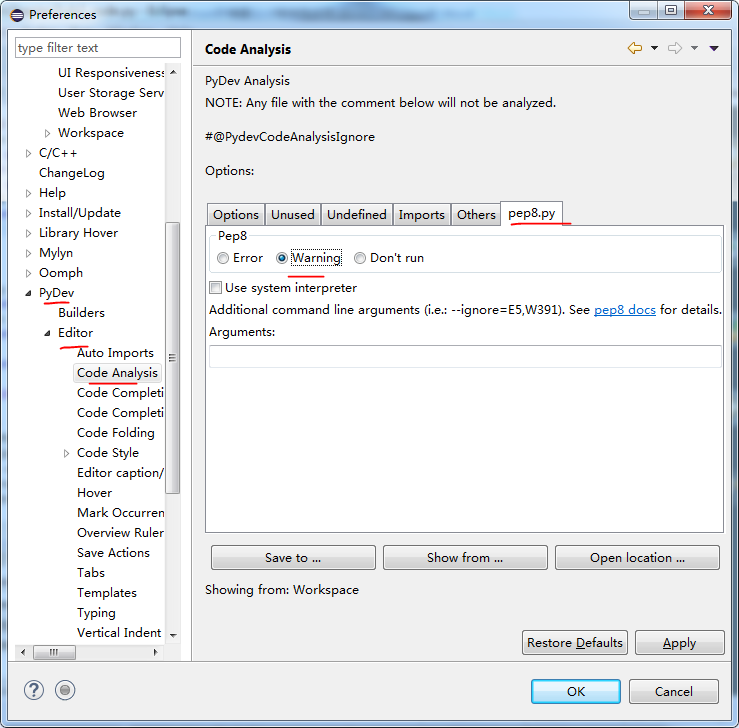
**5.1 PEP8检查和修改**

1）启动 pep8 检查：

Window > Preferences

PyDev > Editor > Code Analysis > pep8.py

选择Errors/Warnings其中之一..



右键Python工程，选择 PyDev，点 "code analysis"，即可对工程中所有Python源码进行PEP8检查。

1）启动autopep8自动修改：

点Windows -> Preferences ->输入 'autopep8' 作为搜索串.

选择（Check）: Use autopep8.py for code formatting?

|  |
| --- |
|  |

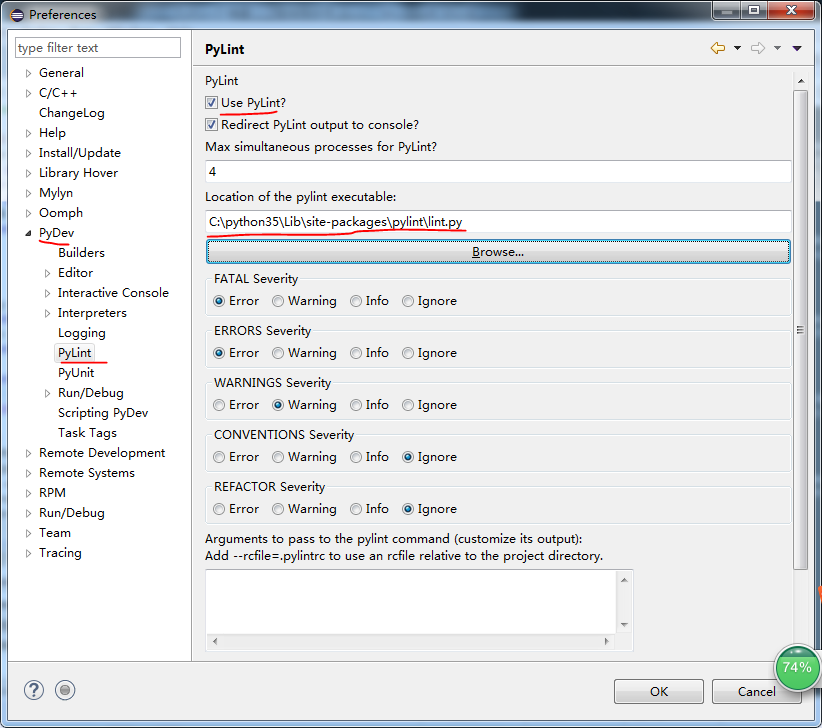
在Python源码窗口，按 CTRL-SHIFT-F 就可以自动修改代码

**5.2 使用Pylint**

PyDev 默认不开启Pylint。通过

     Window -> preferences -> Pydev -> Pylint,选中"Use pylint?",

找到安装好的lint.py的地址,例如"C:\Python34\Lib\site-packages\pylint\lint.py"



配置参数，限制Pylint的输出



选中Project->Build Automatically,这样程序修改，保存时pylint就会自动检查项目中的代码,也可用Ctrl+B手动build触发pylint。



**小结**

Python 学习和开发环境多种多样，接触的时候感觉它和之前所学的c++有很多不同。一开始并不理解老师这种教学方法，安装软件本身居然也算是一次作业，通过这一段时间的软件安装与学习的过程，我认为这样的作业没有什么不好，把这个过程记录下来是值得的。

**参考文献**

1．Brainwy Software Ltd. PyDev Manual. http://www.pydev.org/manual\_101\_root.html

2．郑伟芳. PyDev for Eclipse简介. <http://www.ibm.com/developerworks/cn/opensource/os-cn-ecl-pydev/> 2008.11

3．张颖. Python 代码调试技巧. <http://www.ibm.com/developerworks/cn/linux/l-cn-pythondebugger/> 2012.05