**软件工程基础及实践**

**Python学习和开发环境的建立**

03014323 陈子聿

2017

目录

[**1.** **Python的简介** 1](#_Toc477108002)

[**2.** **Python开发环境的建立** 1](#_Toc477108003)

[**2.1.** **Python 3.5.2安装过程** 1](#_Toc477108004)

[**2.2.** **交互计算环境Jupyter Notebook的建立** 3](#_Toc477108005)

[**2.3.** **交互计算Jupyter Notebook的程序事例** 5](#_Toc477108006)

[**2.4.** **Python3.5.2的使用事例** 6](#_Toc477108007)

[**2.5.** **问题及解决** 8](#_Toc477108008)

[**a)** **问题1：安装Python3.5.2过程中，安装程序显示安装失败** 8](#_Toc477108009)

[**3.** **扩展模块的建立** 11](#_Toc477108010)

[**3.1.** **安装科学计算包** 11](#_Toc477108011)

[**3.2.** **安装规范格式扩展包** 12](#_Toc477108012)

[**3.3.** **安装IF97物性计算** 12](#_Toc477108013)

[**3.4.** **问题及解决** 13](#_Toc477108014)

[**b)** **输入 >pip install \*.whl时，显示失败** 13](#_Toc477108015)

[**4.** **集成开发环境Eclipse** 13](#_Toc477108016)

[**4.1.** **Java SDK的安装和配置** 13](#_Toc477108017)

[**4.2.** **安装Eclipse及配置工作环境** 15](#_Toc477108018)

[**4.2.1.** **Eclipse的安装** 15](#_Toc477108019)

[**4.2.2.** **Eclipse的工作空间的配置** 16](#_Toc477108020)

[**4.2.3.** **Python 开发插件PyDev的配置** 17](#_Toc477108021)

[**4.3.** **配置PyDev以提高效率** 19](#_Toc477108022)

[**4.3.1.** **修改编辑器配色提高可读性** 19](#_Toc477108023)

[**4.3.2.** **配置任务标签** 20](#_Toc477108024)

[**4.4.** **基于PyDev的语言规范静态检查** 23](#_Toc477108025)

[**4.4.1.** **Pep8的检查和修改** 23](#_Toc477108026)

[**4.4.2.** **Pylint的使用** 26](#_Toc477108027)

[**4.5.** **Eclipse的配置** 28](#_Toc477108028)

[**4.5.1.** **文件字符编码方式** 28](#_Toc477108029)

[**4.5.2.** **空间编码方式** 28](#_Toc477108030)

[**4.5.3.** **工程编码方式** 29](#_Toc477108031)

[**4.6.** **Eclipse使用事例** 29](#_Toc477108032)

[**4.7.** **问题及解决** 30](#_Toc477108033)

[**c)** **Eclipse安装失败** 30](#_Toc477108034)

[**5.** **个人小结** 32](#_Toc477108035)

[**6.** **参考文献** 33](#_Toc477108036)

**Python 的学习和开发环境的建立**

——03014323 陈子聿

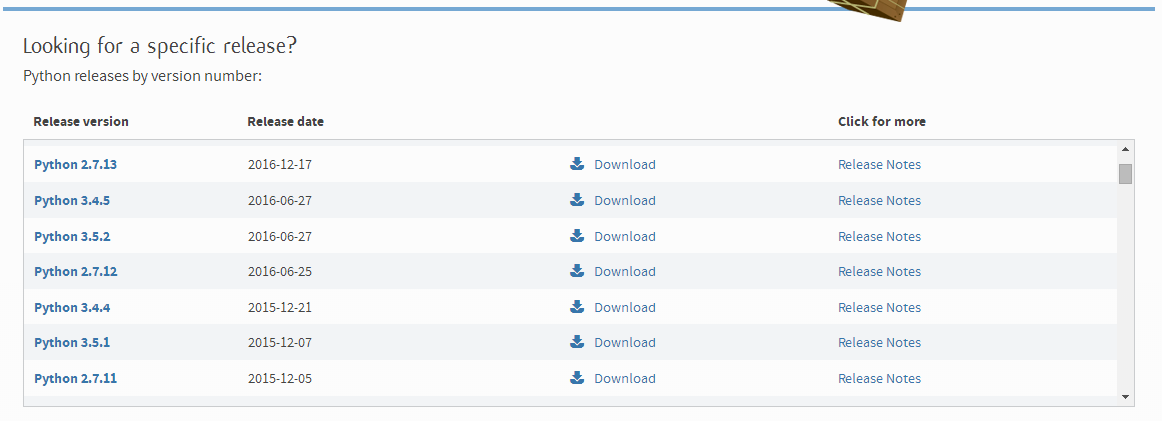
1. **Python的简介**

Python是一门通用编程语言，笔者认为Python相比其他计算机语言，更为简洁，容易入门。更为重要的是，Python是一门高级的编程语言，有着非常完善的基础代码库，这也为用户的使用提供了方便。用一个简单的例子来说明Python的简洁性：在Python语言中，交换两个变量的值只需要用这样简单的语言就可以完成：x,y=y,x.但是在其他编程语言中，需要引入中间变量。程序繁琐是次要，每引入一个中间变量，都大大降低了程序的可读性，所以由此可知，Python语言具有更好的可读性。但是同时，Python语言也有其弊端，Python语言运行速度较慢，在解释性语言编译的过程中，需要先对代码进行翻译，这就大大降低了程序的运行速度。

笔者在这里介绍的就是建立Python开发环境，并能够进行正常的软件开发所必要的步骤。

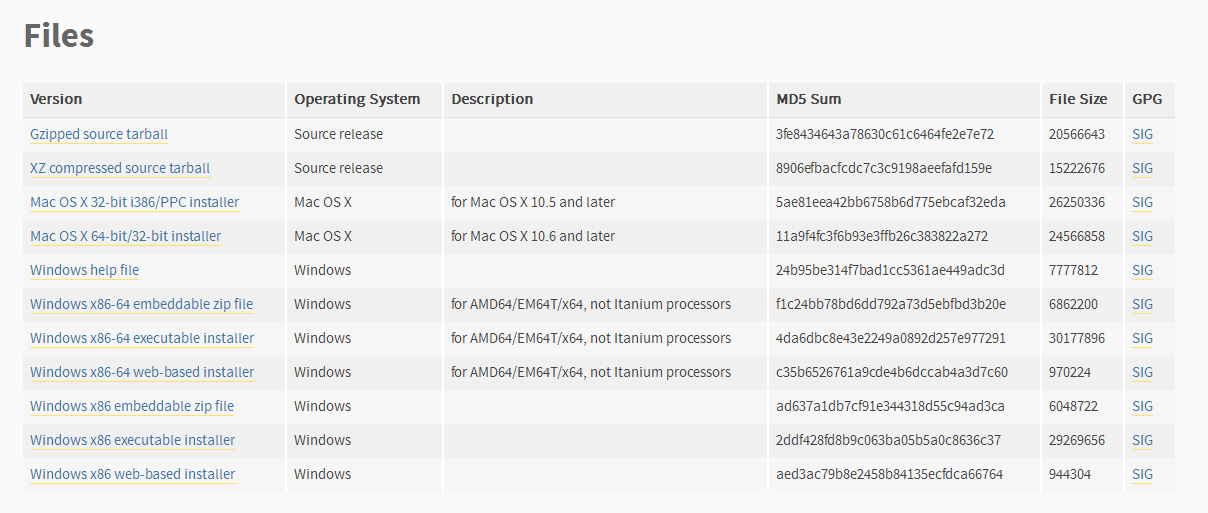
1. **Python开发环境的建立**
   1. **Python 3.5.2安装过程**

首先从Python官网上找到最新版本下载地址<https://www.python.org/downloads/>，考虑最新的Python3.6.0选择2017年次新版Windows 64位Python3.5.2（版本视系统而定，作者为win7 64位操作系统）



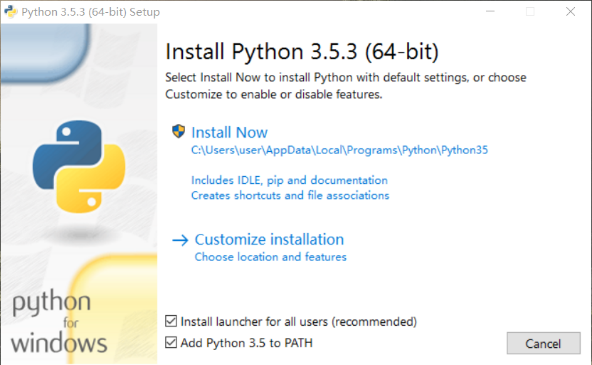
**图 1 Python3.5.2版本的选择**

选择版本Python并点击下载Download.转到画面如下，选择64位操作系统的可执行安装文件。保存到地址 C:\python35.

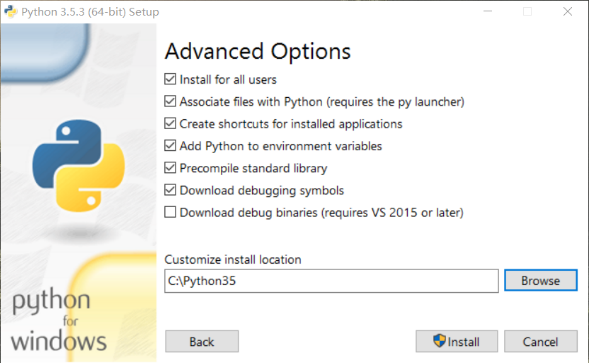


**图 2 64位系统Python程序下载**

在 C:\python35中找到安装程序，打开并进行安装，看到如图3所示画面，勾选“Add Python3.5 to Path”选项，点击Install Now进行安装。跳出如图所示窗口，在下列选项中根据提示，倘若未安装软件VS 2015，则不勾选debug binaries的选项。存放地址到当前目录下，不进行修改。

****

**图 3 Python安装程序初始界面**

****

**图 4 安装选项的设置**

成功安装后，在DOS命令窗口中键入>python -m pip install -U pip指令, 更新pip到新版本，完成Python 3.5.2版本的安装。

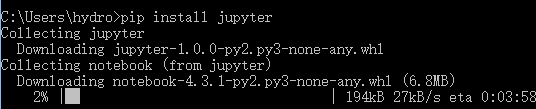
****

**图 5 pip更新操作**

出现图示指令，表示当前pip已更新完成，可进行下一步操作。

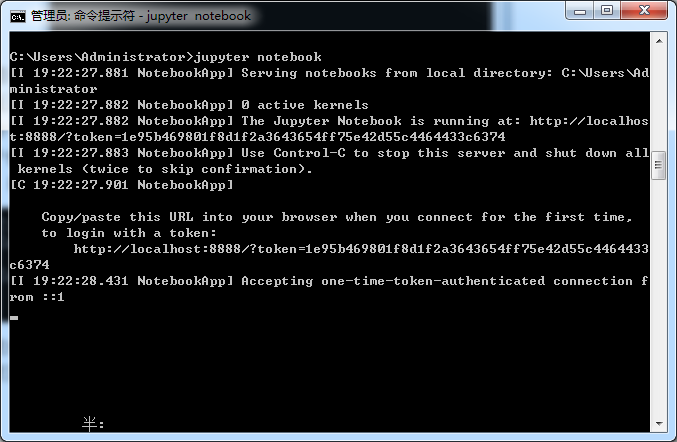
* 1. **交互计算环境Jupyter Notebook的建立**

DOS命令窗口下，输入指令>pip install jupyter，表示安装Jupyter软件包的过程，其费时较长，用户也可以在加州大学欧文分校下载相应软件包的whl文件，然后，用pip安装.出现窗口如图，表示Jupyter软件包正在安装。

****

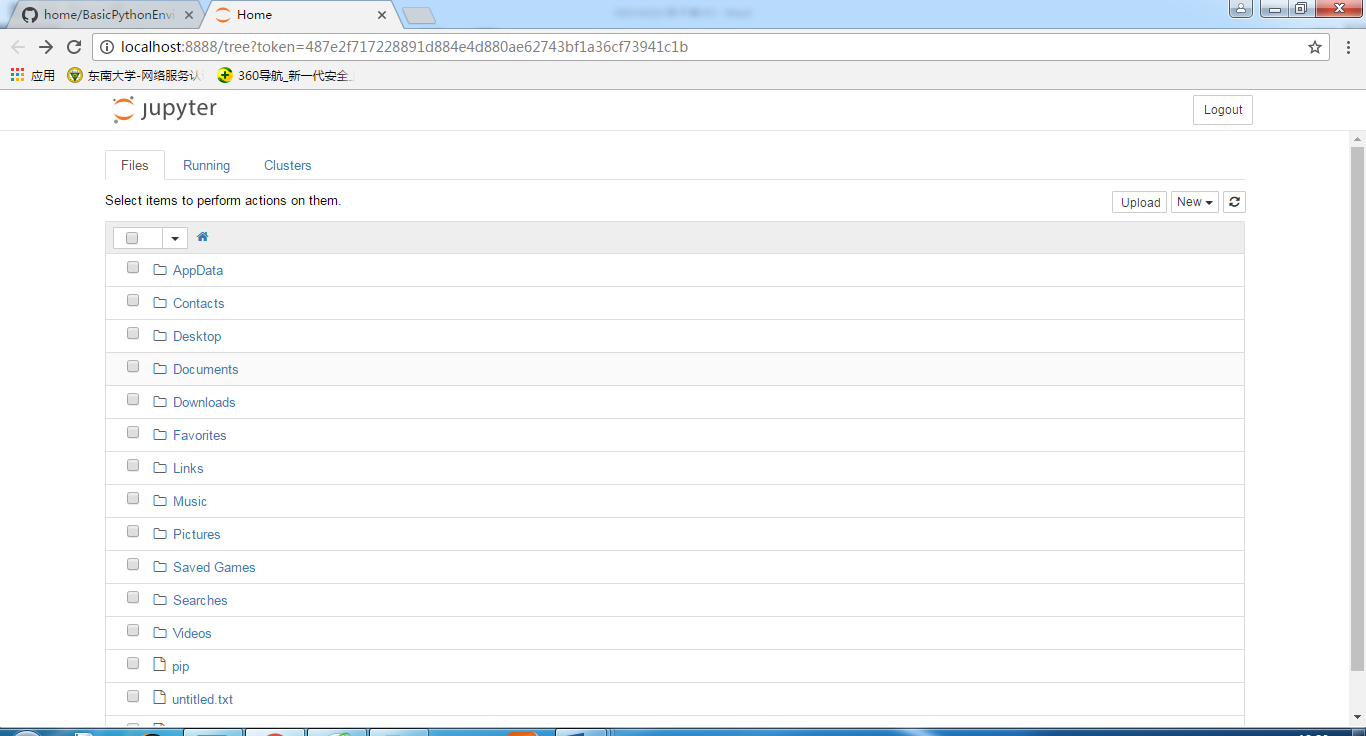
**图 6 jupyter软件包安装**

安装完成后，同样在DOS窗格下，键入>jupyter notebook，即可成功打开Jupyter Notebook的浏览器页面。

****

**图 7 打开jupyter notebook**

浏览器下的页面如图8，我们不难看出由此打开的浏览器页面其实就是用户这台计算机所生成的页面，点开Desktop就一目了然了，也即我们从网页上复现了本台计算机！

****

**图 8 jupyter notebook界面**

做到这里，我们很容易感觉到，DOS窗格在我们软件的开发中是一个被我们经常使用的“入口”，因此将其固定在我们的任务栏中。

****

**图 9 锁定DOS到任务栏**

**问题引入：**

从DOS中打开Jupyter Notebook过程的繁琐程度是显而易见的，于是考虑是否可以通过一个类似.exe的应用程序可以直接链接到我们的网页上？另外，我们很容易看出，这里我们直接在DOS命令下输入>jupyter notebook，启动以桌面作为默认目录，但是如果需要打开以其他地址作为默认目录的网页，这个方法就存在了很强的局限性。

**解决方案：**

**① 用文本编辑器建立\*.bat文件**

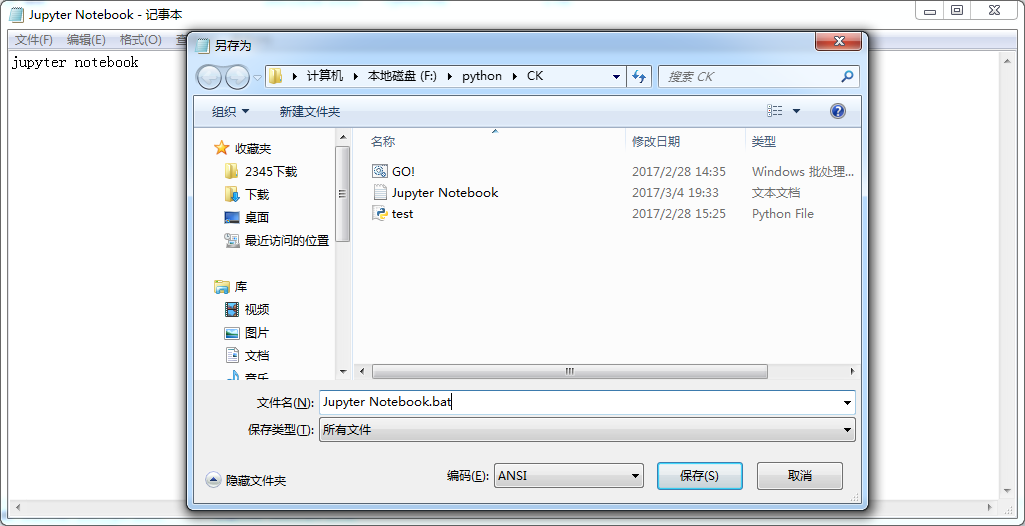
这里就引入了我们.bat格式的文件，bat是DOS下的批处理文件，通过该文件可以直接实现DOS中键入命令的效果。

具体步骤为：①新建一个后缀为.txt的纯文本文件，命名为Jupyter Notebook.txt ②在文件中输入DOS下命令：jupyter notebook ③将文本文件另存为.bat文件格式。

这样以后，每次想要打开Jupyter Notebook时，只需双击DOS下的该.bat文件即可！

****

**图 10 创建jupyter notebook.txt**

****

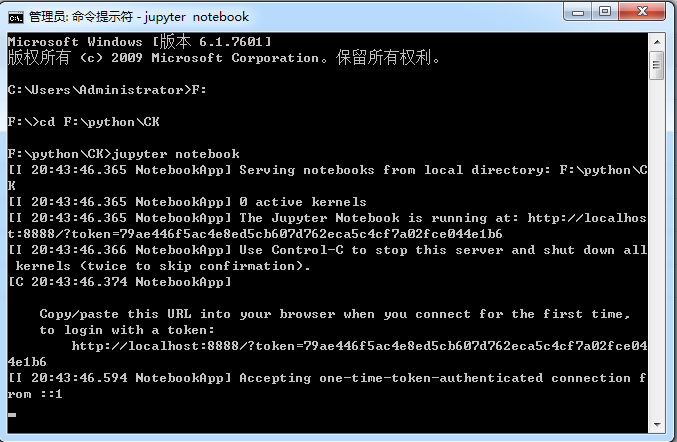
**图 11 修改格式为.bat**

****

**图 12 bat文件**

**② 在命令提示符内修改指向目录**

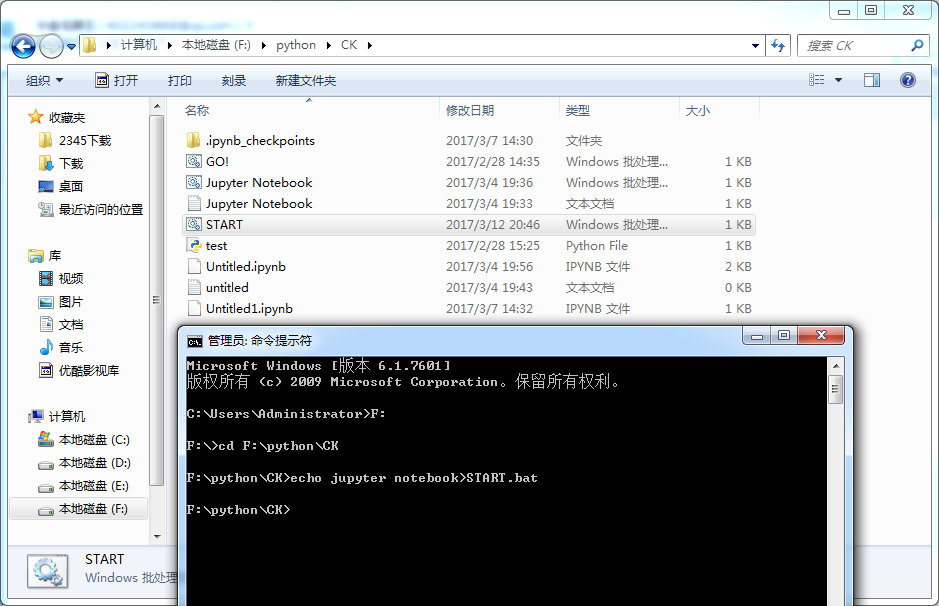
在DOS命令符下，通过修改DOS指向地址来打开以目标文件为默认地址的Jupyter服务。首先在DOS窗口下，将默认地址改为>F:；其次指向需要打开的文件目录>F:\python\CK；可以看到DOS默认的地址已经改为F:\python\CK>，这里再次键入>jupyter notebook.在这里打开的便是以F:\python\CK为目录的Jupyter服务：



**图 13 修改DOS默认地址**

**③ 用重定向的方式建立\*.bat文件**

对于内容较少的\*.bat文件，不仅可以通过文本编辑器进行创新，同时也可以用重定向的方式进行创建：修改命令提示符指向地址（作为\*.bat文件的创建地址），键入所需要的命令>echo jupyter notebook>START.bat即可在目标地址下创建所需文件。

****

**图 14 重定向创建\*.bat文件**

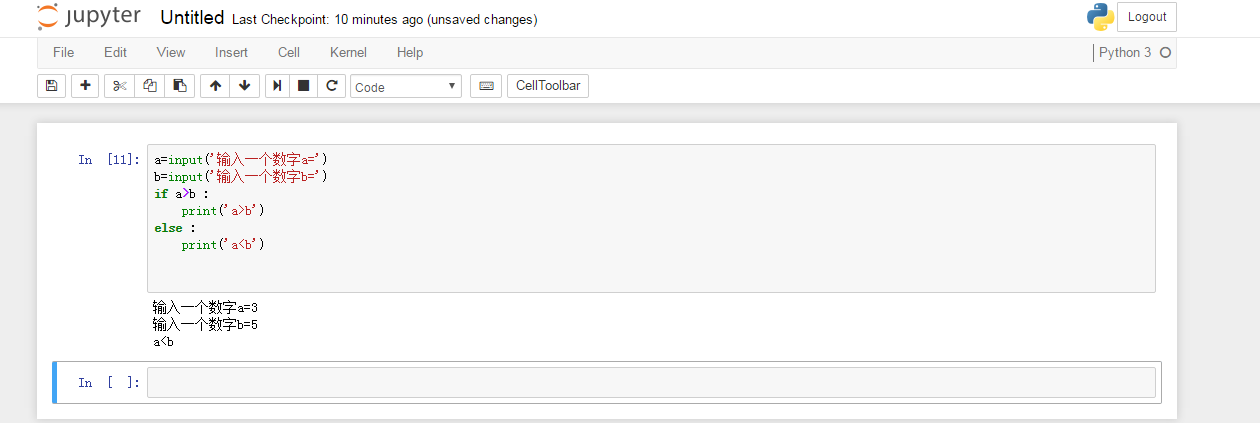
* 1. **交互计算Jupyter Notebook的程序事例**

双击Jupyter Notebook.bat 打开了Jupyter Notebook的操作网页，我们可以看出这就是对DOS文件所在文件夹的“复现”。在New中选择Python 3即可打开Python的可执行窗格，在该窗格中进行程序的书写和编译。

****

**图 15 打开程序界面**

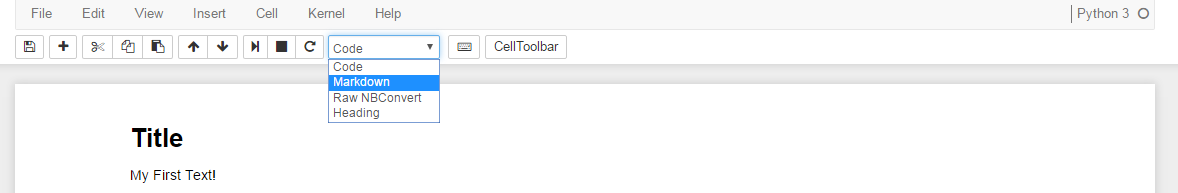
打开Python的代码文件，在其中进行代码的编写，代码完成后，点击 run cell进行代码的编译和运行。由此可知，我们所进行代码编写的一个窗口即称为一个“cell”。



Cell

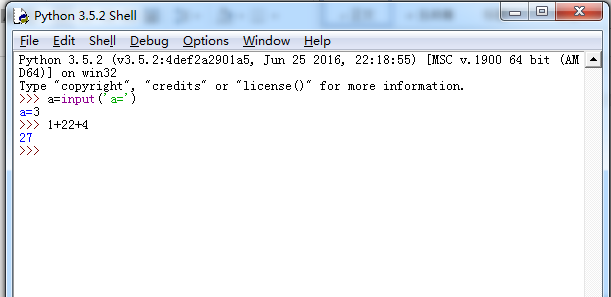
**图 16 Python程序界面**

另一方面，我们可以给我们的文章添加标题：其中，#Title 在编译完成后，显示为一级标题，##……后显示的为二级标题，###……后显示的为三级标题，通过该方法，可以把自己的文件地更有条理。

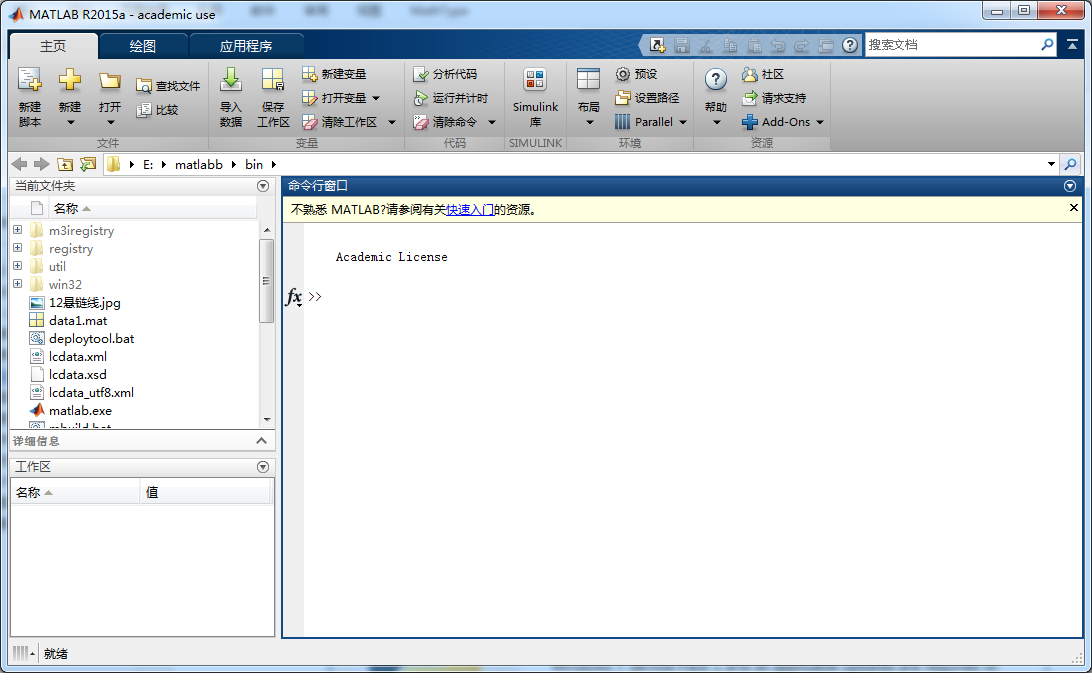


* 1. **Python3.5.2的使用事例**

IDLE是Python的集成开发环境，通过IDLE可以实现Python程序的简单开发：

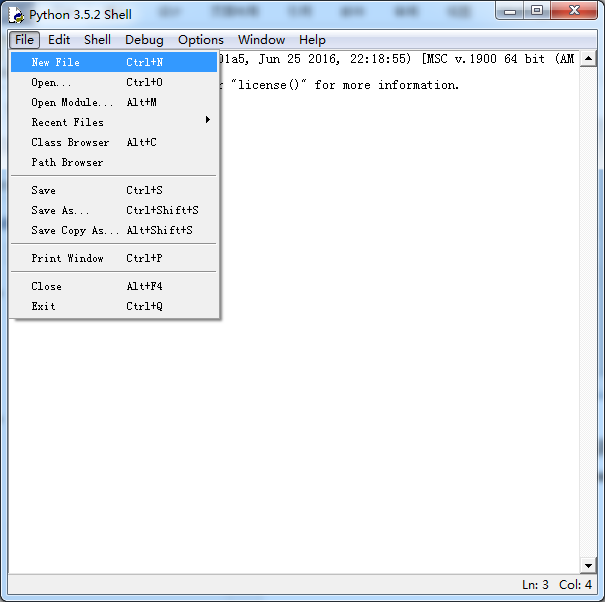
****

**图 17 简单的IDLE事例**

****

**图 18 MATLAB界面**

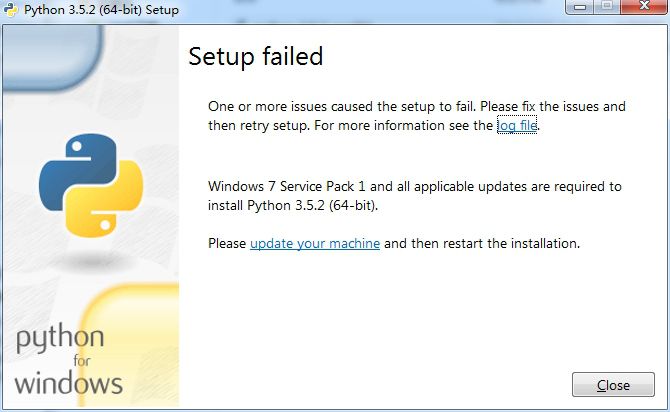
IDLE类似于MATLAB的程序页面，在主页面上，以>>提示输入，并可以进行简单的代码计算。要完成整个程序的开发，在文件新建脚本上进行程序的编写。

****

**图 19 IDLE脚本创建**

* 1. **问题及解决**

1. **问题1：安装Python3.5.2过程中，安装程序显示安装失败**

****

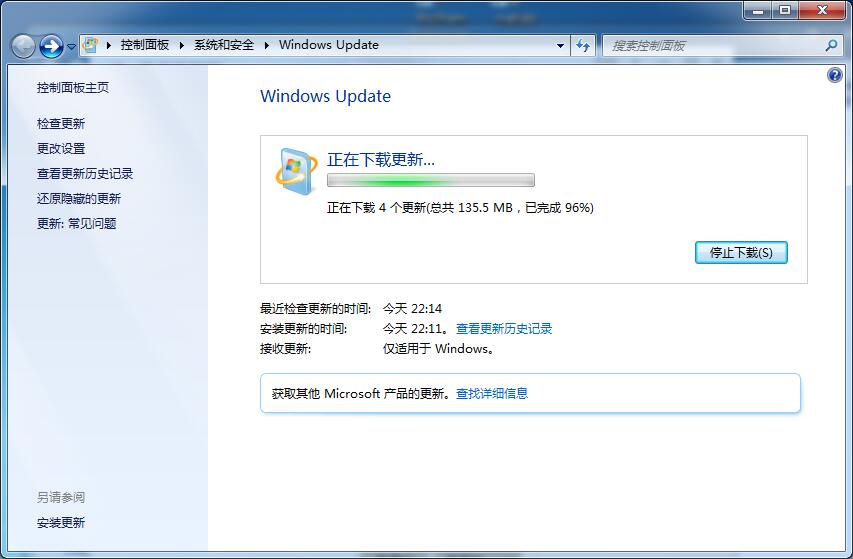
**图 20 Python安装失败**

经过检查后得知，由于系统Windows7 系统版本过低，Python的安装需要更新的系统的支持，所系需要在系统中安装Service Pack 1，来解决。

****

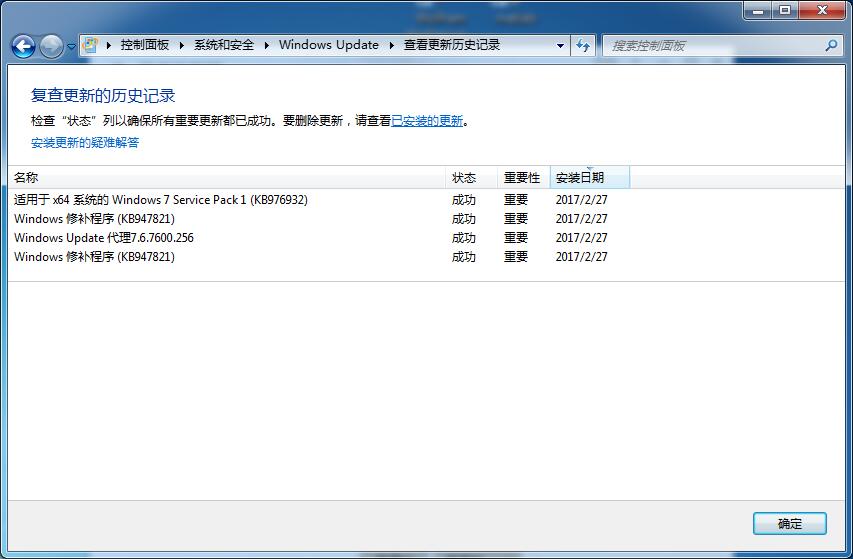
**图 21 windows版本信息**

1. **解决方案1：在Windows Update中下载更新**

****

**图 22 Windows自动更新**

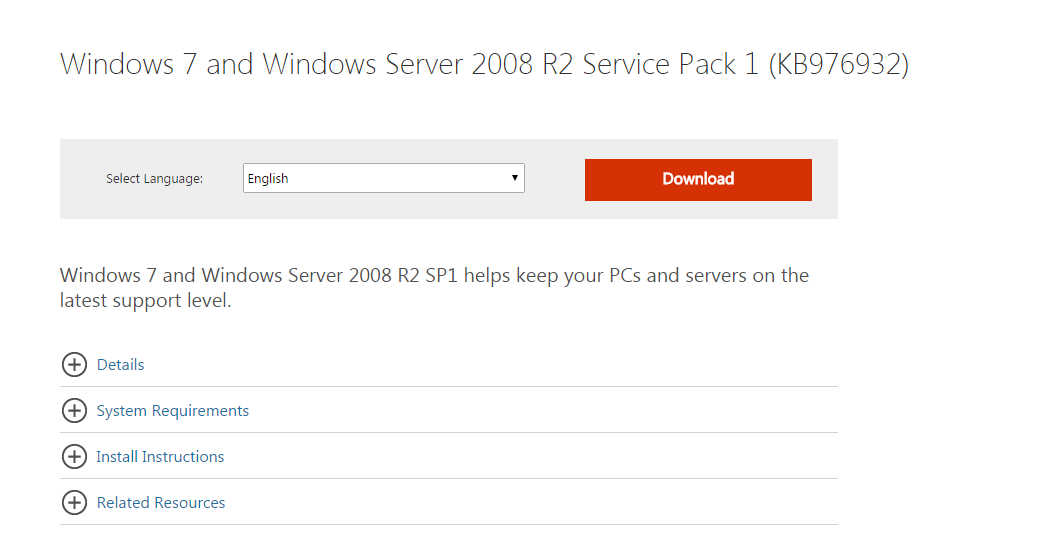
但是最终由于某些不知名原因，导致系统更新中的Service Pack 1安装失败，转而寻找其他Service Pack 1的安装方式。

****

**图 23 Windows更新界面**

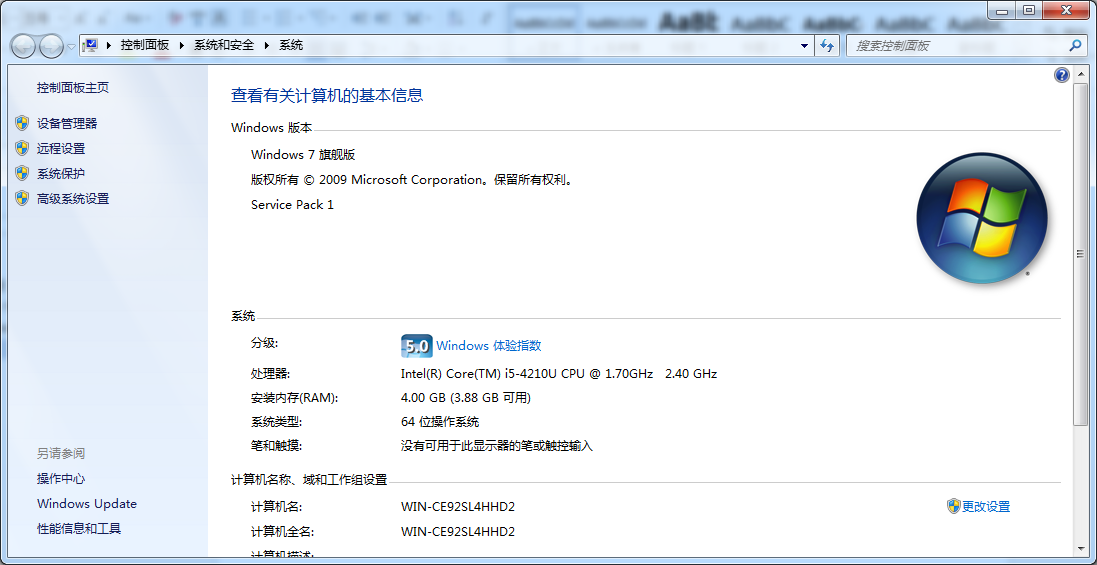
1. **解决方案2：在微软官网中，通过连接下载**

通过网页<https://www.microsoft.com/en-us/download/details.aspx?id=5842>，进入Microsoft官网进行下载：

****

**图 24 微软官网SP1下载**

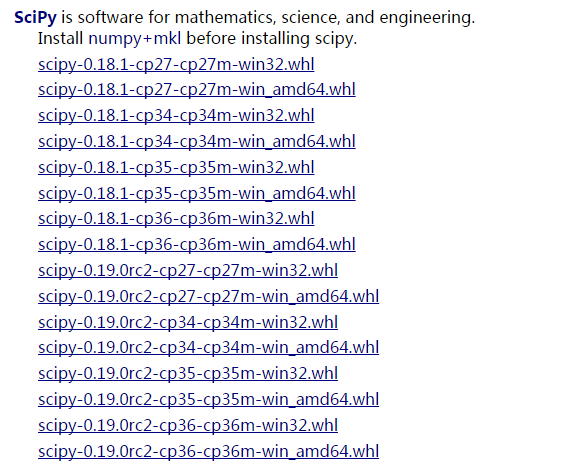
下载完成后，在控制面板中，Windows版本下面，出现Service Pack 1的项目，表示系统更新完成。

****

**图 25 Windows更新完成**

1. **扩展模块的建立**
   1. **安装科学计算包**

在这里，下载扩展模块。基本的科学计算包有 SciPy，NumPy，Matplotlib. 在加州大学欧文分校的网站上（<http://www.lfd.uci.edu/~gohlke/pythonlibs/>）可以找到对应版本的安装包，下载后用命令提示符进行pip安装。

现以scipy安装为例：打开上面的链接网站，找到 ****

**图 26 scipy版本下载**

下载Win64位操作系统的对应于Python3.5科学计算包。注：在这里，XXX.whl表示包含了Python文件的压缩包格式，需要通过DOS命令符进行“解压”。打开DOS提示符，键入>pip install matplotlib-1.5.3-cp35-cp35m-win\_amd64.whl对科学计算包进行安装。

其他NumPy和Matplotlib计算包依照上述规范进行安装。

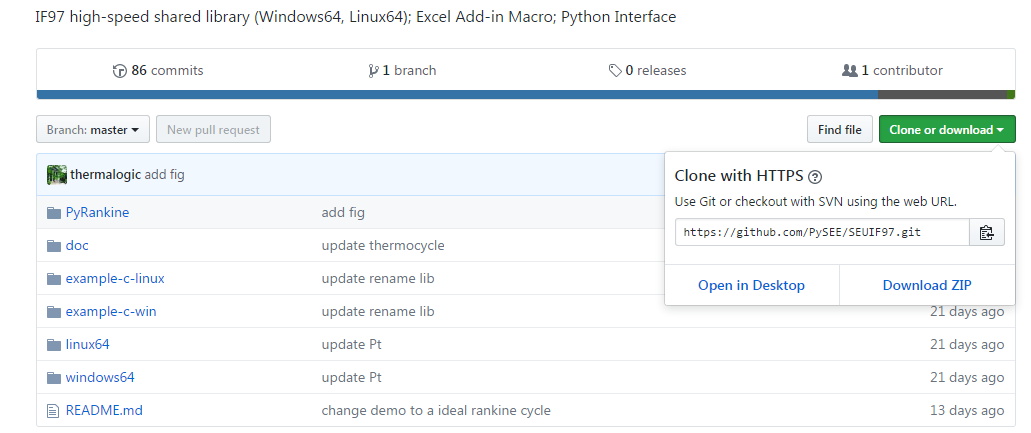
* 1. **安装规范格式扩展包**

在这里安装扩展模块pep8、autopep8 和 pylint.其中，pep8全称为：Python Enhancement Proposal，即Python增强建议书；autopep8是一个将Python代码自动排版为PEP8风格的小工具；pylint是一款代码检查工具。

其安装过程比较简单，直接在DOS窗格内，键入如>pip install pep8即可。

* 1. **安装IF97物性计算**

IF97是一款水蒸气物性计算的语言包，从github下载课程SEUIF97仓库zip文件：<https://github.com/PySEE/SEUIF97>.

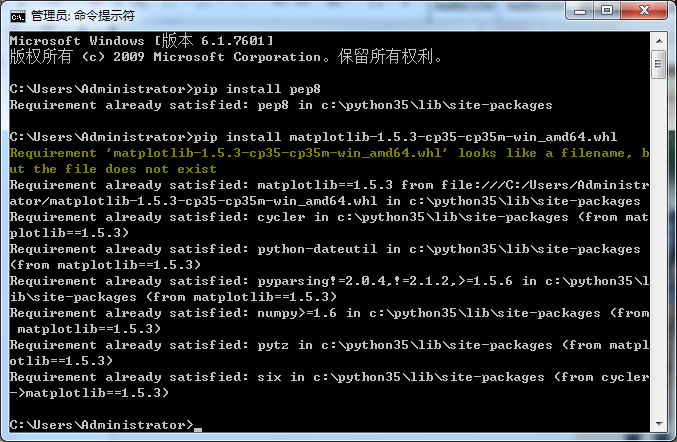
****

**图 27 IF97语言包下载**

解压下载后的，zip文件，将libseuif97.dll拷贝到 c:\windows\system.最后, 将seuif97.py拷贝到python安装目录的lib子目录下，如C:\python35\Lib.

* 1. **问题及解决**

1. **问题2：输入 >pip install \*.whl时，显示失败**

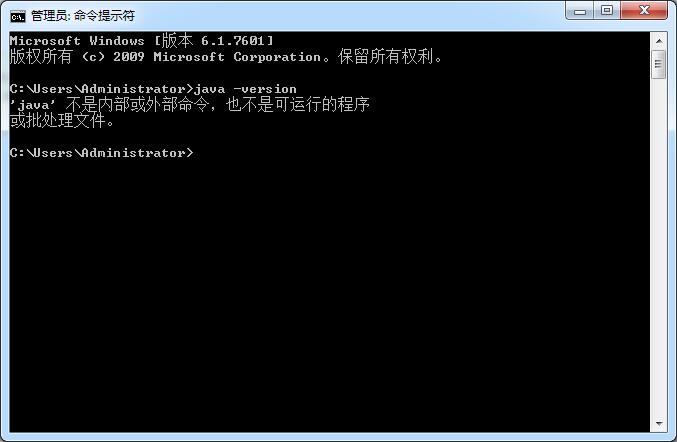
****

**图 28 扩展包安装失败**

在这里，由于文件不存在，所以导致科学计算包的安装失败。在这里，解决方法是通过键入>cd +文件路径，转到安装目录下，再进行安装。

1. **集成开发环境Eclipse**
   1. **Java SDK的安装和配置**

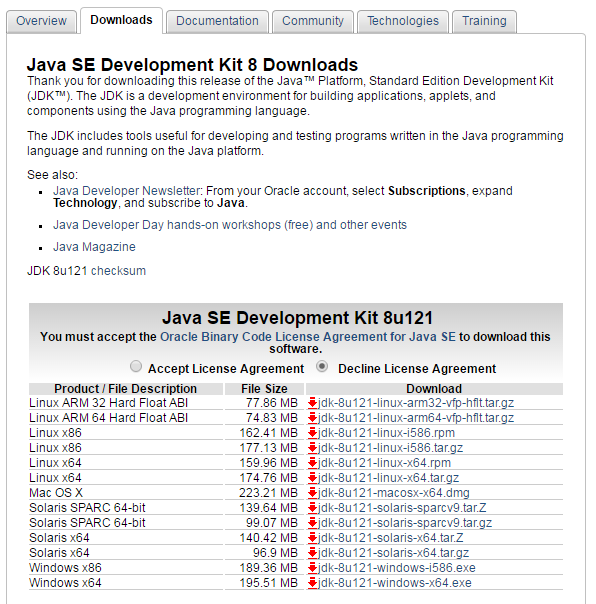
Eclipse IDE是使用Java进行开发的，所以在安装Eclipse之前，需要预先安装好Java JRE/JDK软件包。需要预先对本机的安装情况进行检验，在命令行下，输入：> java –version检查是否已经安装了Java软件包。在电脑上没有成功安装好Java的情况下，会显示如下信息：

****

**图 29 未安装Java**

因此，需要从Oracle下载Java包，以满足Eclipse IDE的最低要求：在Java SDK的官网上进行下载。

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

****

**图 30 Java版本下载**

选择对应Windows x64的Java进行下载。在Java安装完成后，注意对环境变量的配置。安装完成后，需要配置java环境变量，在我的电脑上右击-属性-高级系统设置，在弹出的窗口中点击右下角的“环境变量”按钮，接下来可以在用户环境变量或者系统环境变量中配置具体的环境变量，新建名为“classpath”的变量名，值为“.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar 。同样，新建名为”JAVA\_HOME“的变量名，变量值为之前安装jdk的目录。在已有的系统变量“path“的变量值加上 ”;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin“，自此配置完成。下面进行Eclipse的直接安装。

****

**图 31 Java版本检查**

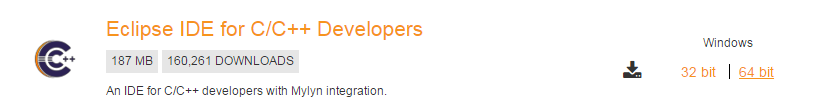
安装完成后，再次输入 >java –version，检查已安装Java版本信息如图，已安装Java版本为1.8.0\_121.

* 1. **安装Eclipse及配置工作环境**
     1. **Eclipse的安装**

Eclipse IDE 是IDE，全称IntegratedDevelopment，是集成开发环境。Eclipse基金会在：<http://www.eclipse.org/users/> ，提供了大量资源。安装和使用中的问题，都可以在其中找到相应的帮助文档。

Eclipse IDE有很多版本。本课程使用Python语言，可能会涉及C/C++开发，所以，建议下载Eclipse CDT(Eclipse IDE　for　C/C++)版.

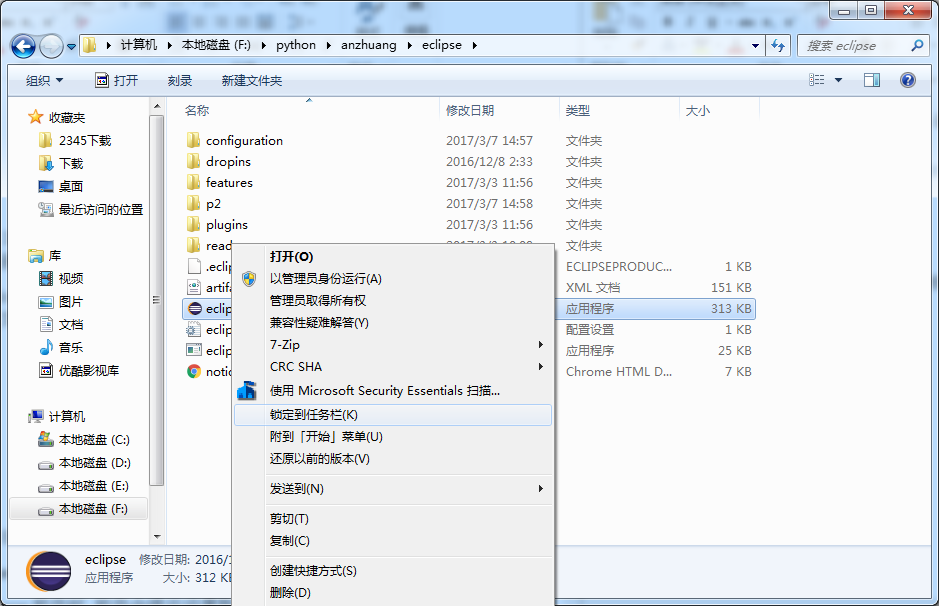
Eclipse CDT官方下载地址： [http://www.eclipse.org/downloads/eclipse-packages/](%20http:/www.eclipse.org/downloads/eclipse-packages/)选择Win64的Eclipse安装包进行下载：

****

**图 32 Eclipse安装包下载**

下载完成后，将已下载的Eclipse CDT 解压到指定目录下，运行解压目录下的：eclipse.exe进行安装。

为了方便使用，建议将eclipse.exe，固定到任务栏中，将使得Eclipse的启动和工作方便得多：

****

**图 33 锁定Eclipse到任务栏**

****

**图 34 任务栏截图**

* + 1. **Eclipse的工作空间的配置**

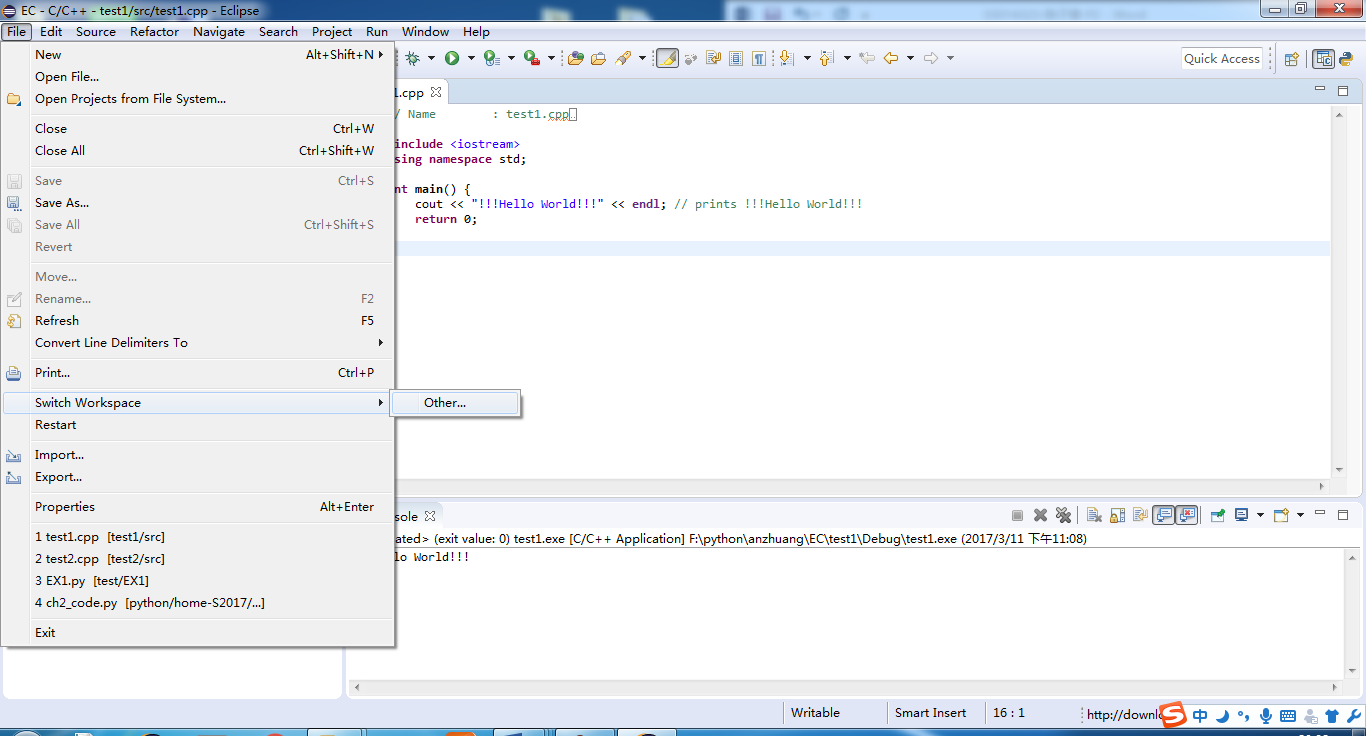
对于首次打开Eclipse软件的用户，系统会提示对默认的工作空间进行设置，以后所有开发的项目都将默认保存到该目录下。（**注：**不建议用户使用Eclipse软件初始默认的工作空间目录）

在非系统盘中，如笔者在F盘内新建一个名为“EC”的文件夹作为目录，点击Browse，浏览到F:\python\anzhuang\EC的目录之下，记得勾选“Use this as the default and do not ask again” 即在下次登录时不会对用户给出相同的提示，已经将Eclipse的目录配置为当前的目录了，即后续进行的程序的开发都将保存在这里。

****

**图 35 配置Eclipse默认路径**

另，倘若用户想要改变当前的工作目录，可以通过“File->Switch Workspace->other”进行重新配置：



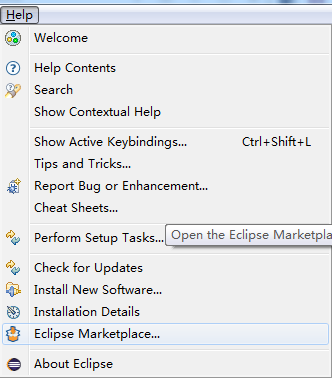
**图 36 修改默认路径**

* + 1. **Python 开发插件PyDev的配置**

Eclipse是一款集成的软件开发平台，为了实现Python程序在该平台下的使用，需要进行以下步骤：① 安装必须的Python插件；② 配置使用的Python解释器版本。

**① PyDev插件的安装**

该过程较为简单，通过Help->Eclipse Marketplaces进入市场，输入pydev，找到PyDev项目，点“install”在线安装即可：

****

**图 37 打开市场**

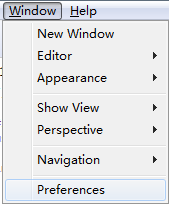
倘若安装速度较慢，可以从PyDev官网上(<http://www.pydev.org/>)所提供的下载地址进行下载：

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

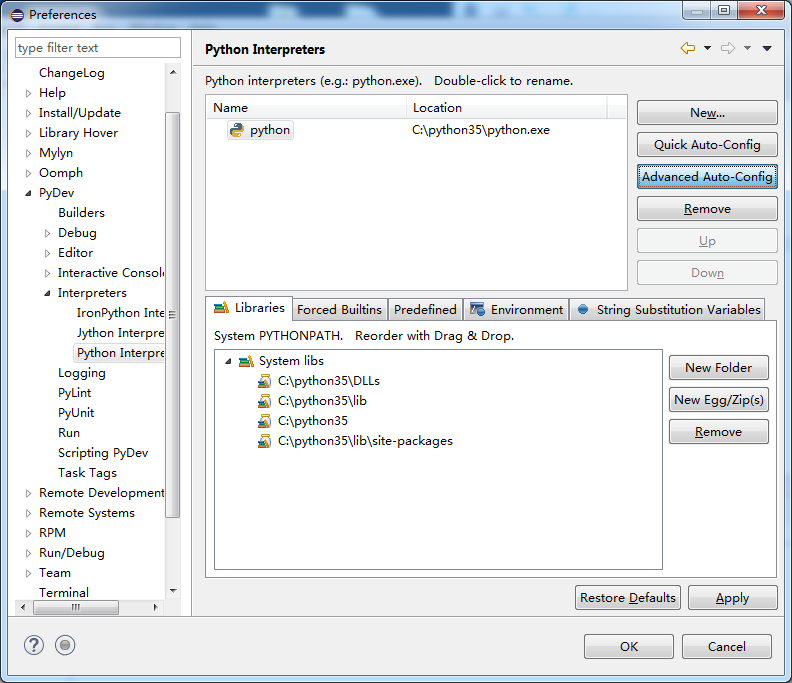
下载插件包，然后，通过Eclipse中：Help->Install->New Software->Add->Archive进行离线安装。正常情况，Eclipse会自动解压，安装好插件。如解压中异常，可手工解压插件包，将解压后的所有文件，拷贝到Eclipse的插件目录plugins下。

**② 配置使用的Python解释器版本**

安装好PyDev插件后，重新启动Eclipse，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:

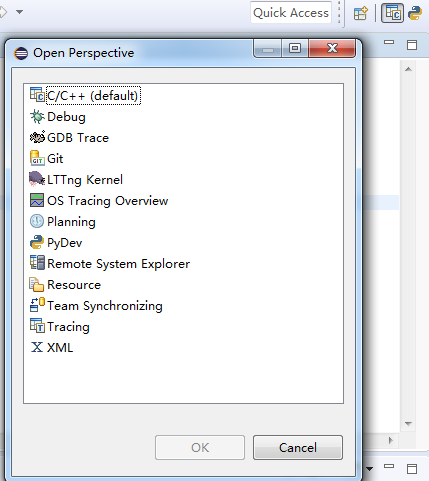
****

**图 38 Preferences 打开**

****

**图 39 Python解释器版本**

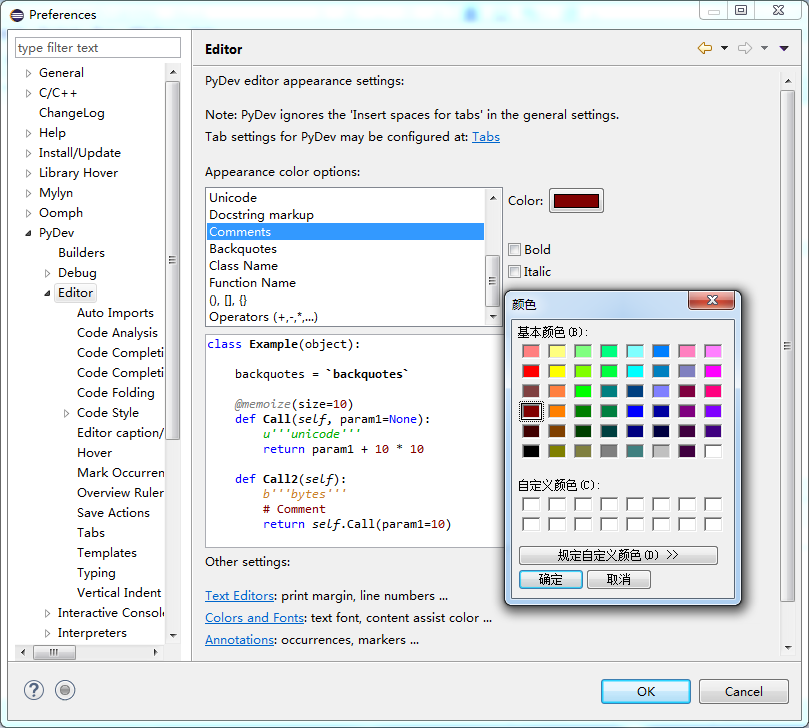
配置好后，添加Python场景到工作页面上，方便以后在Eclipse中对Python的操作和使用：

****

**图 40 添加Python工作场景**

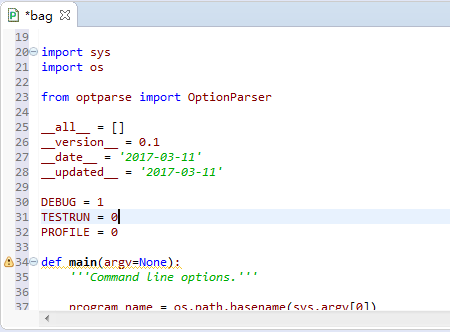
* 1. **配置PyDev以提高效率**
     1. **修改编辑器配色提高可读性**

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

****

**图 41 修改编辑器配色**

配置完成后，在.py文件内进行程序编写：

****

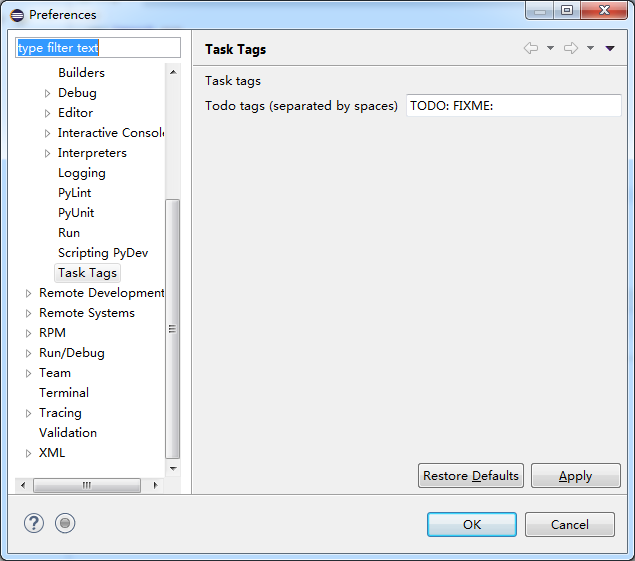
**图 42 Python程序界面**

* + 1. **配置任务标签**

在一个完整的程序的开发中，在代码中对当前任务进行标识，计划开发工作。可以通过“任务标签”在代码中对任务进行标识，在开发环境中进行识别，加入工作空间的任务列表中：

**① 配置任务标签**

通过Eclipse中Window → Preferences→PyDev→Task Tags进行配置：

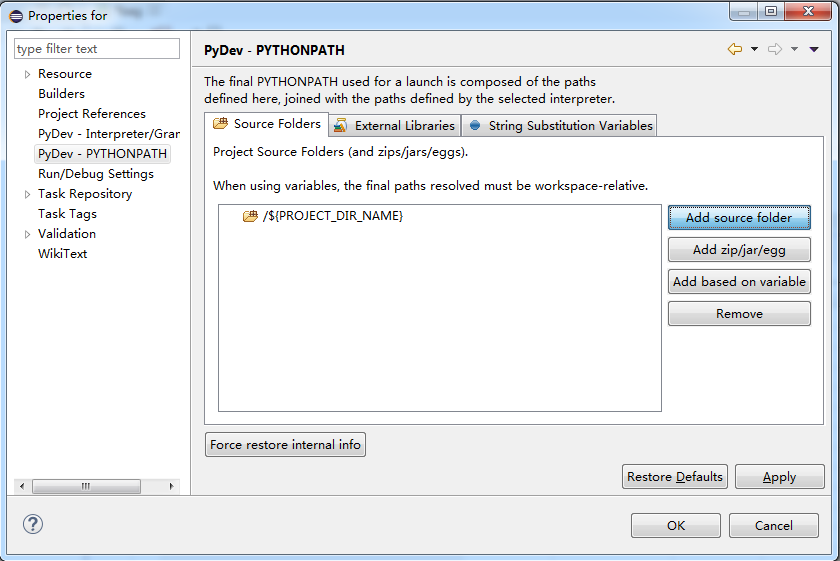
****

**图 43 任务标签设置**

其中，TODO和FIXME是系统默认的两个标签符号，对于用户可以根据个人喜好，对任务标签进行配置。

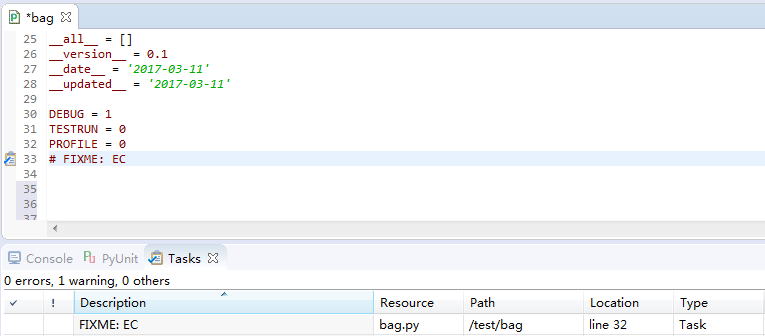
**② 添加任务标签注释并识别任务**

任务标签是一种“注释”，以任务标签为关键词开头。需要配置当前工程的PyDev-PYTHONPATH的source folder。在源码工程的Properties->PyDev-PYTHONPATH配置项中，点“Add source folder”将源码目录加入。

****

**图 44 加入源目录**

保存修改，即build当前项目（Ctrl+b）或者Project->Clean:

****

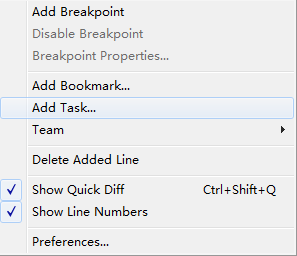
**图 45 添加任务标签**

在下面的任务列表中，我们可以看到名为“EC”的任务标签，而在程序中，我们看见被标注行左侧出现了任务标识。

**③ 手动添加任务**

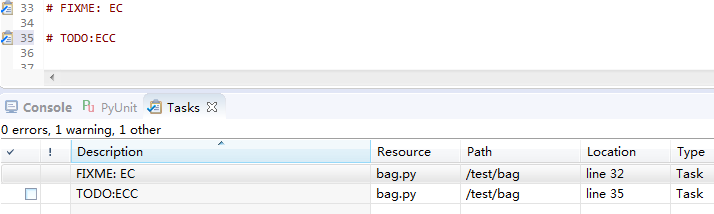
除了在上文中通过#+任务标签来添加以外，在PyDev环境下，系统还为用户提供了通过手动添加任务标识的方法：

在所需设置为任务行的最左侧，右键->Add Task，即可将其加入任务列表

****

**图 46 手动添加任务标签**

则刚刚手动添加的任务标签TODO:ECC也成功地加进了任务列表中：

****

**图 47 任务标签**

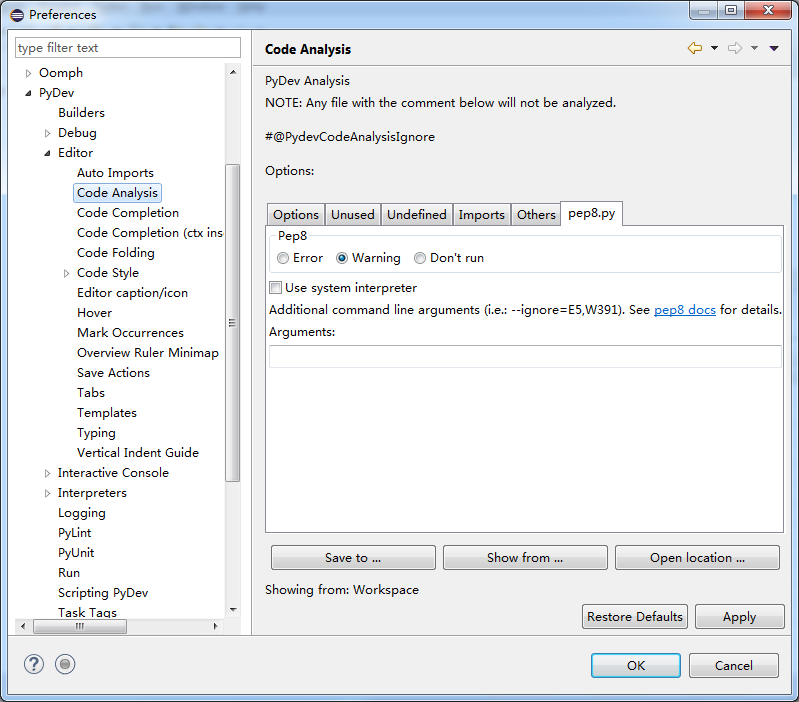
* 1. **基于PyDev的语言规范静态检查**

PyDev中集成了PEP8、和Pylint代码检查功能，这些功能默认状态都是关闭的，现在需要通过手动开启这些功能。Pep8是Eclipse里的一个标准，配合pylint对程序代码进行检查。其中，前者只能检查程序代码格式，而后者不仅检测格式，还检测语义。

程序开发过程中，要有代码规范意识，但过分注意规范会影响开发进程。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，开发进程中默认关闭，在程序开发一个阶段结束时，开启规范性检查检查更好，现在我们手动对代码进行检查。

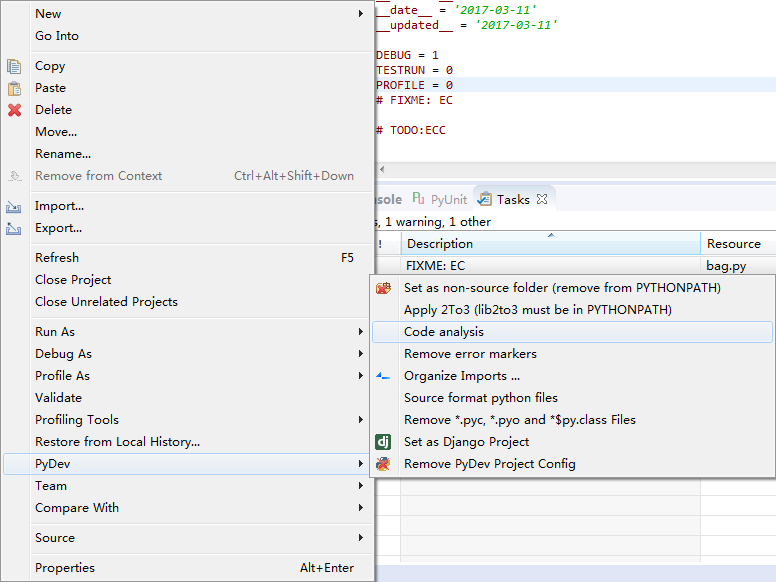
* + 1. **Pep8的检查和修改**

在Eclipse界面，启动Window->Preferences打开配置窗口后，通过PyDev->Editor->Code Analysis->pep8.py进行配置。

****

**图 48 PEP8检查设置**

配置完成后，手动右键当前工程，选择PyDev->code analysis即可完成对所有Python源程序的代码进行PEP8检查：

****

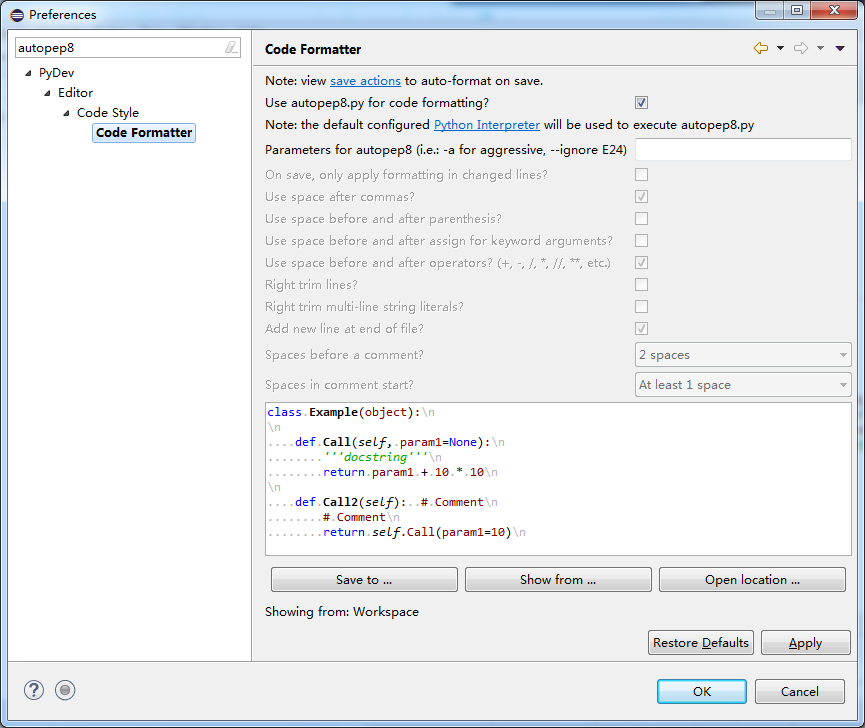
**图 49 PEP8代码检查**

即系统完成了对代码的检查：

****

**图 50 代码检查视图**

为了方便用户操作，Eclipse提供了对代码自动修改的功能即autopep8，通过Windows->Preferences，在项目栏中输入“autopep8”,选择Code Formatt中“Use autopep8 for code formatting?”进行勾选：

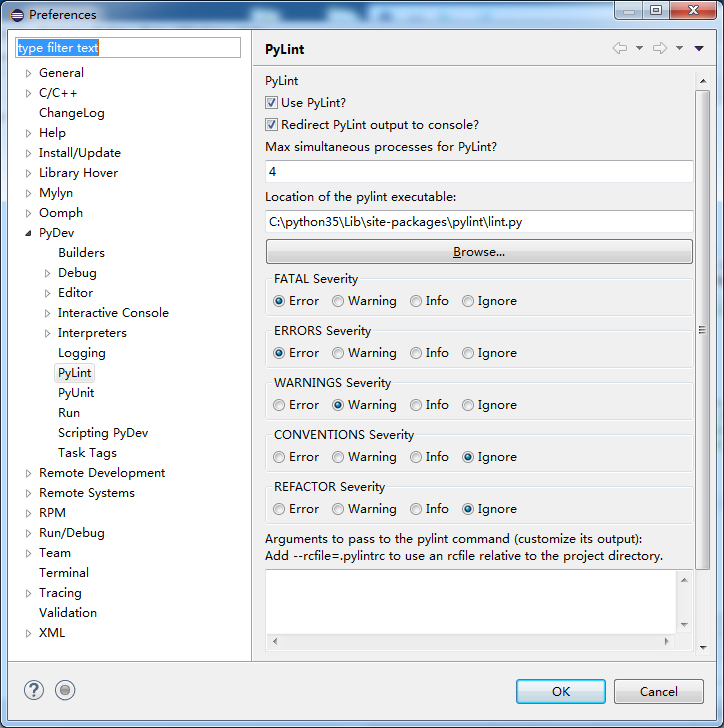
****

**图 51 代码自动检查**

完成上述设置后，通过Crtl+Shift+F就可以自动完成对代码的修改任务。

* + 1. **Pylint的使用**

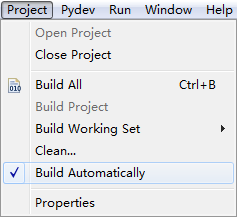
同pep8的检查一样，程序默认不自动开启Pylint的检验，用户可以通过Window -> preferences -> Pydev -> Pylint,选中"Use pylint?"对代码进行Pylint检查：

****

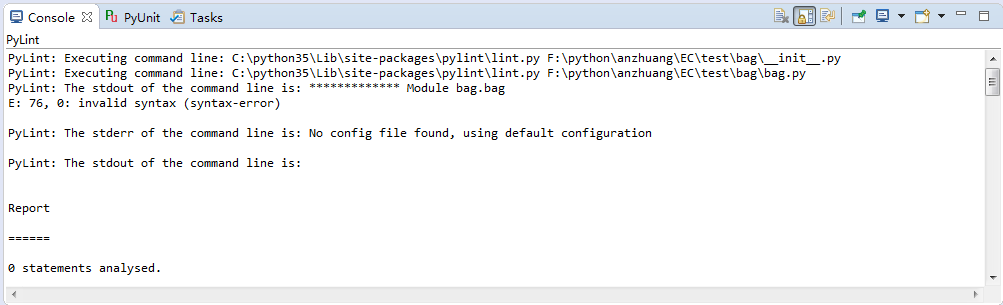
**图 52 Pylint设置**

通过“Browse...”找到“lint.py”的文件地址，如笔者安装在“C:\Python36\Lib\site-packages\pylint\lint.py”完成Pylint的设置。

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.

****

**图 53 自动检查**

****

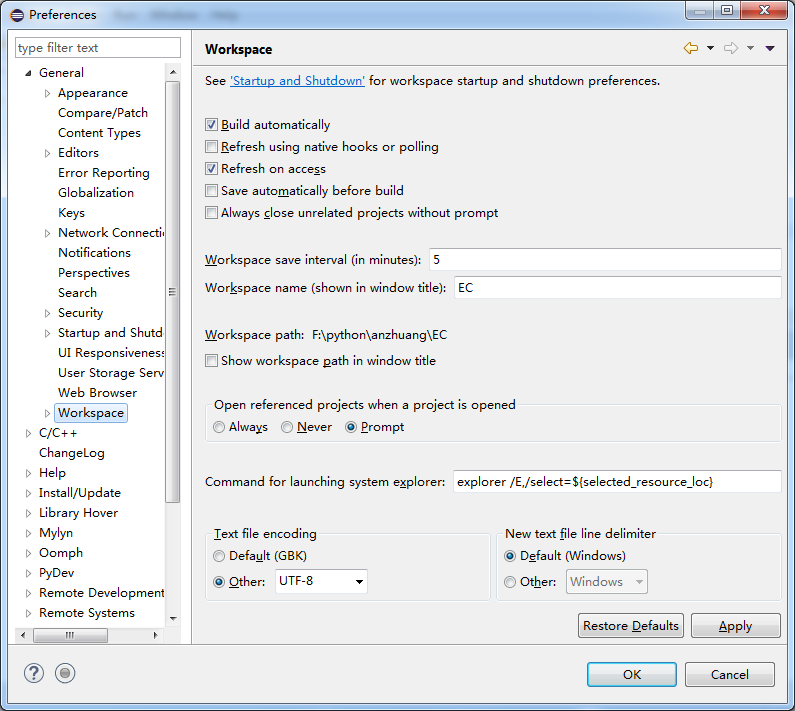
**图 54 Pylint检查**

* 1. **Eclipse的配置**
     1. **文件字符编码方式**

中文操作系统下，Eclipse默认工作空间编码方式为GBK, 这样的编码方式下，含中文字符的文件，在其他文本编辑器打开可能会乱码，因此，建议配置编码为UTF-8。

* + 1. **空间编码方式**

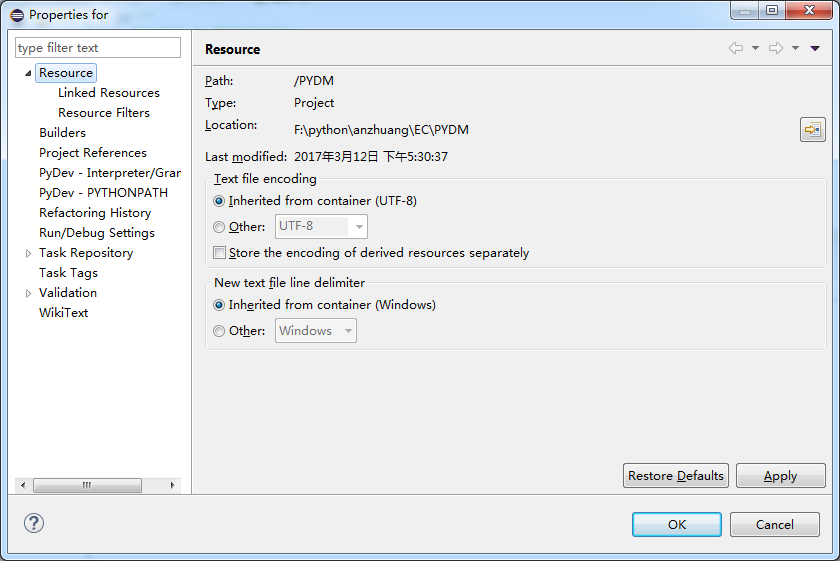
操作Window->Preference，在弹出窗口中，点击general->workspace，修改Text file encoding为UTF-8：

****

**图 55 空间编码设置**

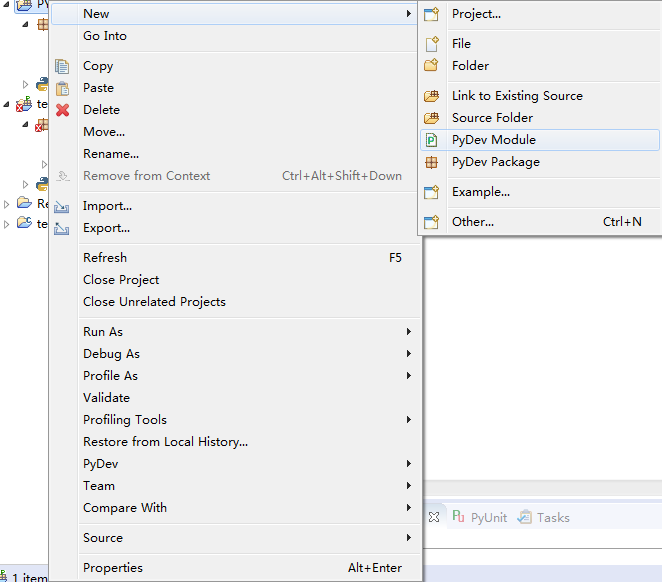
* + 1. **工程编码方式**

将鼠标移动到项目名上，点击右键，选择“properties”， 在弹出的对话框中 ，选中“resources”， 修改“Text file encoding”为UTF-8(如果工作空间配置为UTF-8会继承过来)：

****

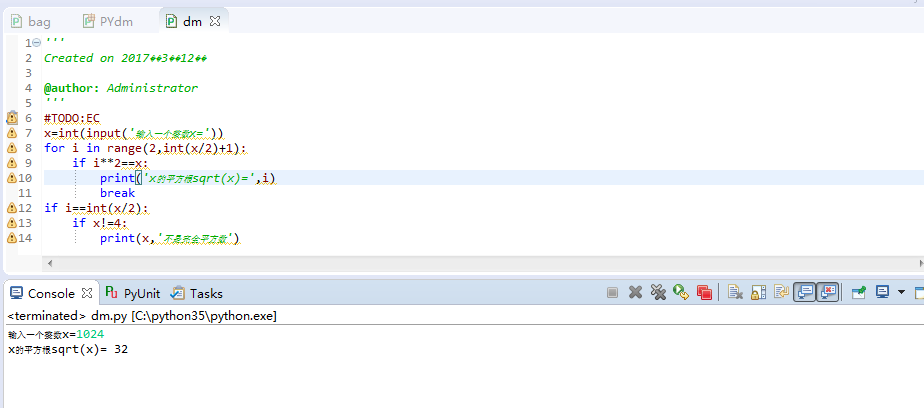
**图 56 工程编码设置**

* 1. **Eclipse使用事例**

****

**图 57 创建Python可执行程序**

在自己创建的PyDev的项目中，顺序创建PyDev Packages->PyDev Module.创建完成后，即可在.py文件内进行编程。

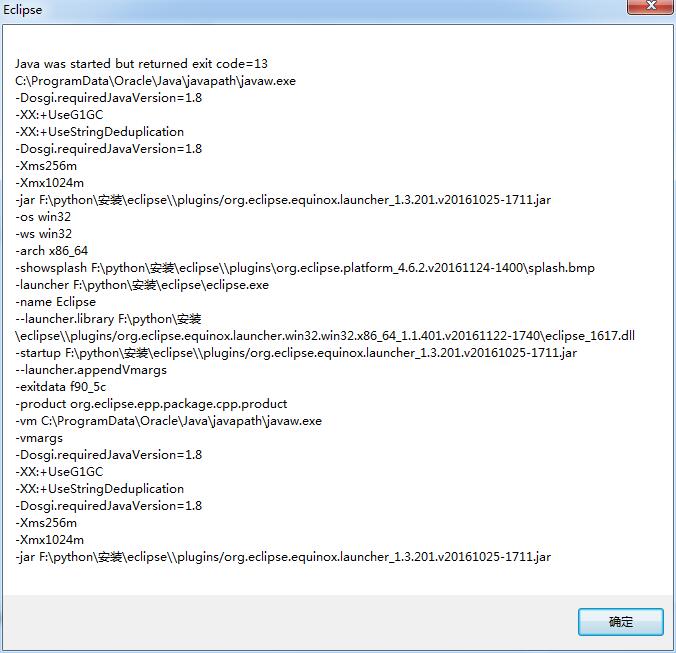
****

**图 58 求平方根程序事例**

图示为一个求解一个整数的平方根的简单程序，通过for循环，对从2到x/2所有数字进行遍历，寻找满足条件的数并对外输出。

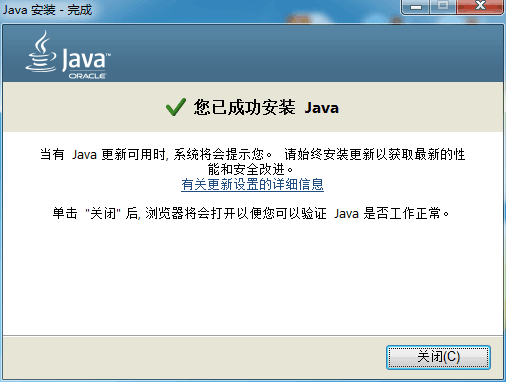
* 1. **问题及解决**

1. **问题3：Eclipse安装失败**

****

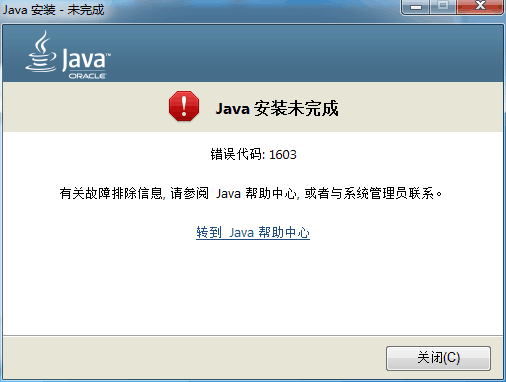
**图 59 Eclipse安装失败**

由图示可知，是由于安装Java过程中，Java版本过低或者说Java安装失败导致。回顾Java 的安装过程，在Java安装结束后：

****

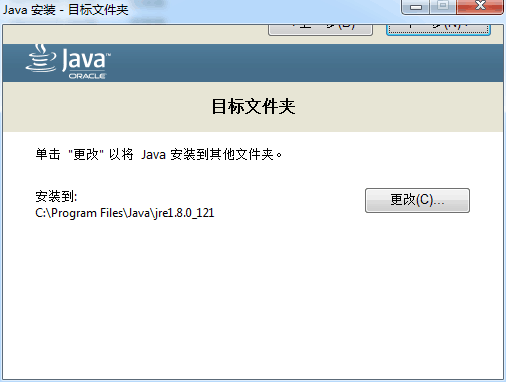
**图 60 Java安装结束**

有如上提示，笔者认为Java已经成功安装，而忽视了后面的提示：

****

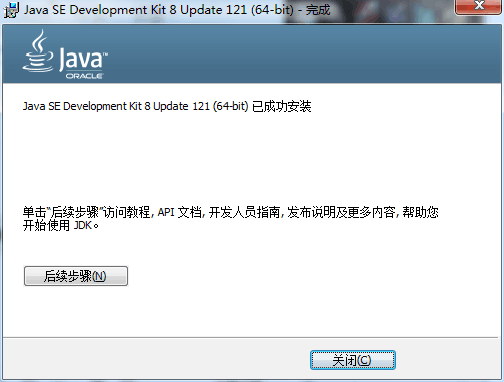
**图 61 Java安装未完成**

笔者在系统程序中找到了Java软件，所以就没有及时检查，最终导致了Eclipse软件的安装失败。回顾整个Java安装过程，发现导致Java安装失败的根源是笔者更改了Java安装的默认文件夹：

****

**图 62 Java目标文件夹更改**

在程序->卸载程序中卸载Java并重新安装推荐文件夹进行安装后，笔者成功的完成了Eclipse的安装。正确的Java安装完成界面为：

****

**图 63 Java安装完成**

1. **个人小结**

在笔者的安装过程中，出现了许多其他人所不曾遇到的问题，究其主要原因，是笔者在对Python软件安装前不久，重装了Win7的系统，当时的考虑是出于Win7系统的稳定，由于也曾使用过Win10系统，在使用过程中不停地出现新的问题，对于一个非计算机专业的学生而言，笔者选择使用更为保守的Win7系统。对于重装后的系统，有许多需要进行更新的地方，所以导致某些软件在旧版本的系统中无法正常安装，白白浪费了许多时间。

另外，在安装软件的过程中，笔者对于整个计算机系统不是十分熟悉，对于整个指导文件中的说明步骤知其然而不知其所以然，这就导致了当按部就班走下去之后，倘若发现问题不能及时地找到根源，对于一些计算机提示性语言也没有很好的反应能力，因此导致了诸多问题。

最后，经过了本次软件的安装过程，笔者深深地认识到了这样一个问题：在大一的C++学习过程中，我们更多地接触到的是VS6.0或者是VS的稍高版本的软件，通过这样的一个平台进行程序的编写，而对计算机底层的运行了解不多。这就导致了离开了一个熟悉的平台，学生很难得心应手地将自己所学知识进行举一反三，比如在笔者学习Matlab语言的时候，相似的算法和表达，却花费了许多时间去适应。对于以后想要朝“程序猿”方向发展同学，所需要掌握的绝不仅仅是基于某一个平台的软件开发能力，更需要掌握计算机开发底层的基础知识，能够做到触类旁通。

1. **参考文献**
2. Python学习和开发环境的建立(教学版).docx

<https://github.com/PySEE/RecommendedPracticestree/S2016/P1>

1. RecommendedPractices

<https://github.com/PySEE/RecommendedPractices/tree/S2016/P1>

1. 郑伟芳.PyDev for Eclipse简介.

<http://www.ibm.com/developerworks/cn/opensource/os-cn-ecl-pydev/.2008.11>

1. 廖雪峰的官方网站-Python教程

<http://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000>