**Python学习和开发环境的建立**

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。

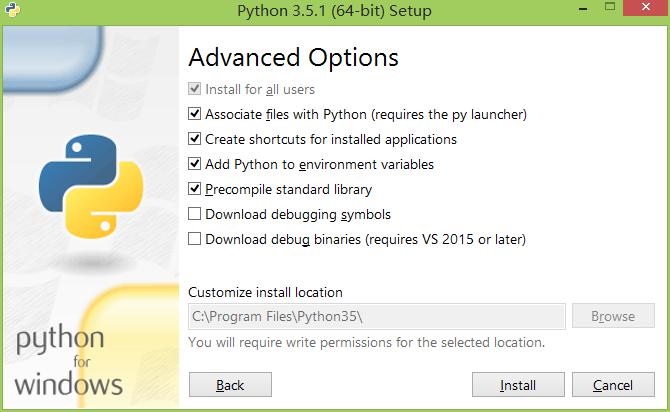
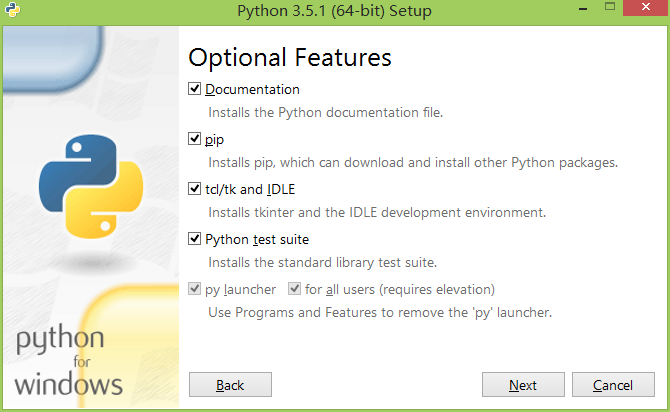
大型软件开发宜采用集成开发工具，常用的有: Eclipse，VS2015等，其中，Eclipse加PyDev插件的方式， 为较多Python程序开发人员使用。此外，还有使用编辑器(如Atom)构建开发环境的方法。

## Python基本开发环境建立

用Github中分享的安装文件安装Python3.5。

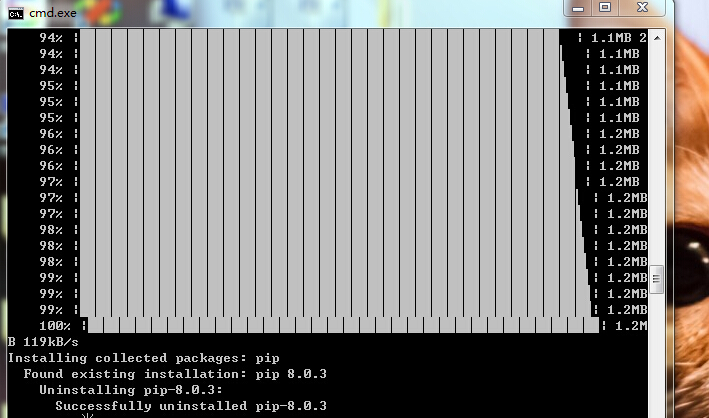
Windows系统安装： 1) 加环境变量PATH;

2) 自定义安装目录： C:\python3.5。



然后，命令行下更新pip到新版本：

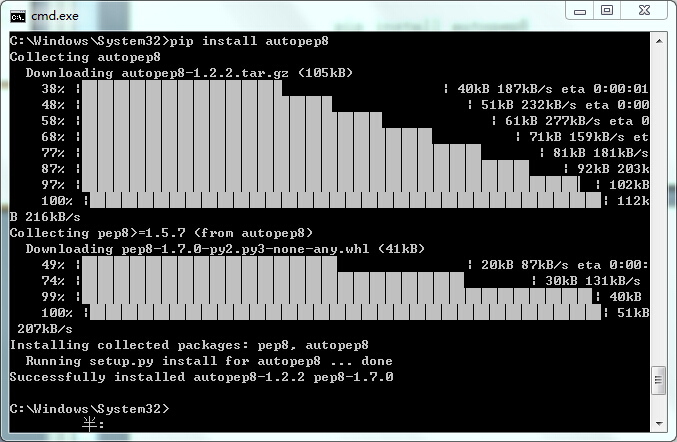
>pip install --upgrade pip



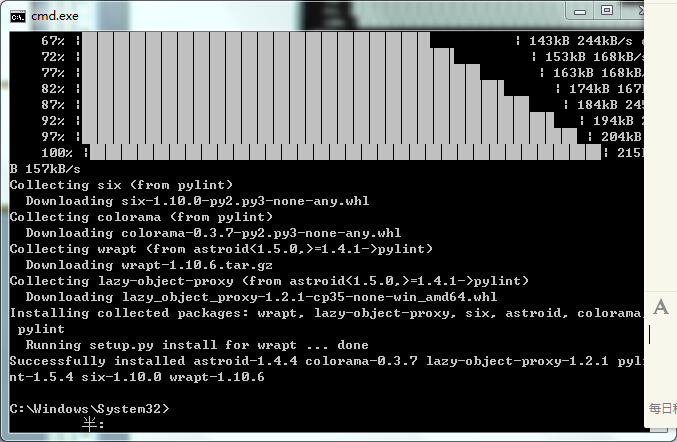
## 二、 Python拓展包安装

**2.1 编码规范**

pip install autopep8



pip install pylint



**2.2 科学计算软件包**

2.2.1 SciPy

（因暂时用不上，故没有安装）

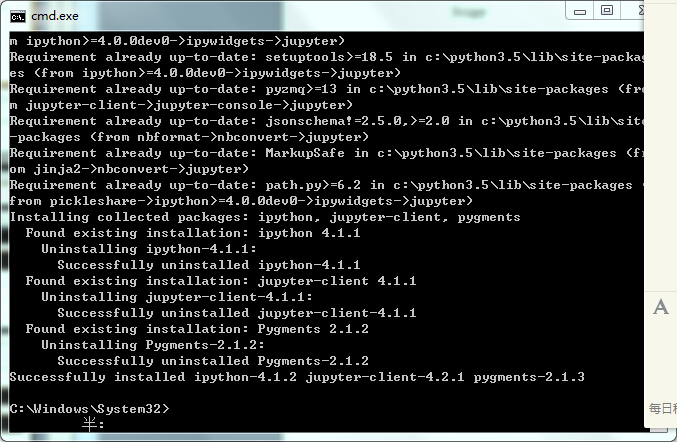
Window环境下Scipy 软件包的 安装，建议从加州大学欧文分校下载编译好的\*.whl，然后，用pip逐个本地安装：

>pip install \*.whl

2.2.2 交互计算 Jupiter

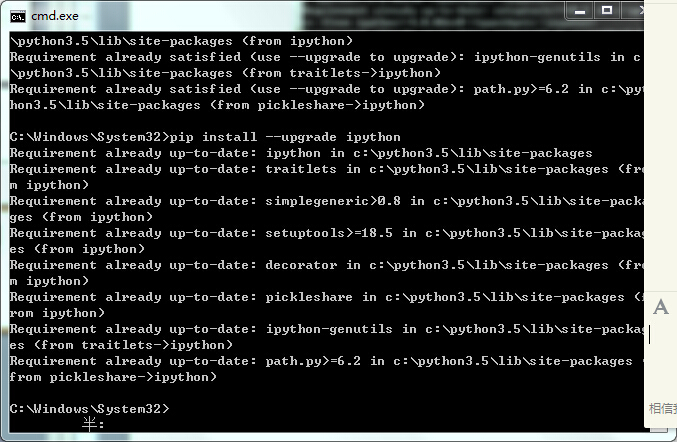
1） 安装Jupyter ：

>pip install jupyter 在线安装



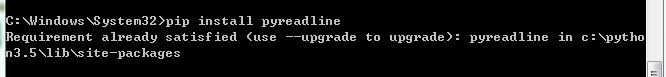
2） 安装Python语言内核

>pip install ipython ，支持Python语言

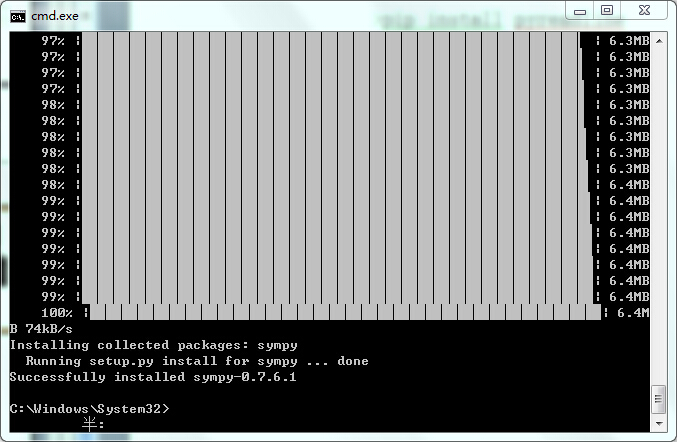


3） 安装依赖包

>pip install pyreadline



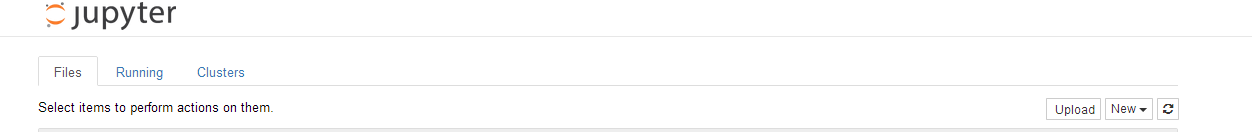
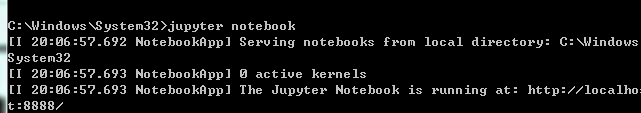
>pip install sympy



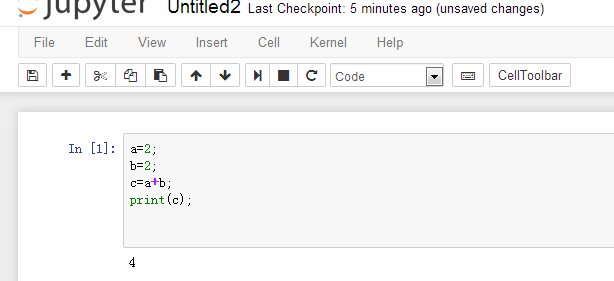
4） 运行notebook:

在iPython notebook文件所在目录下，打开命令行窗口， 运行jupyter notebook.exe 。

也可在记事本写入 jupyter notebook,保存为jupyter.bat,进行快捷呼出。



可利用其code 模式进行一些简单计算演示。如下图。



2.2.3 IF97 for Python

Windows 32/64位版: 从

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

下载：SEUIF97.dll和 seuif97.py，然后：

1) SEUIF97.dll拷贝到 c:\windows\system

2) seuif97.py拷贝到 c:\python35\Lib

## 三、 基于Eclipse的Python开发环境

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台，专注于为高度集成的工具开发提供一个全功能的、具有商业品质的工业平台。Eclipse只是一个框架和一组服务，用于通过插件组件构建开发环境。Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。Eclipse 开源，而且一般Java开发更倾向于Eclipse，使用安装免费，操作简单，且包含很多插件，使用方便，但程序启动较慢。

**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，因此，安装前，在命名行下，输入：>java –version，检查是否已经安装了Java软件包。如果电脑中已经安装了Java ,会显示有关版本。



如果没有安装java 或者版本不能满足Eclipse IDE最低要求，需要从Oracle下载Java包。

Java JDK的Oracle官方下载地址如下：

<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 32位：\*-windows-i586.exe

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/>

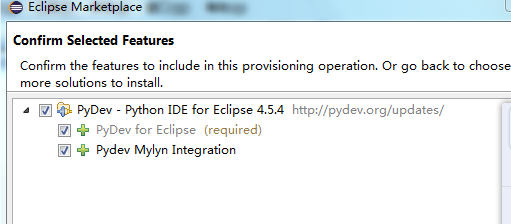
注：eclipse为解压即用文件，不需要进行安装。

**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

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。



也可从pydev官网（<http://www.pydev.org/>）提供的下载地址：

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

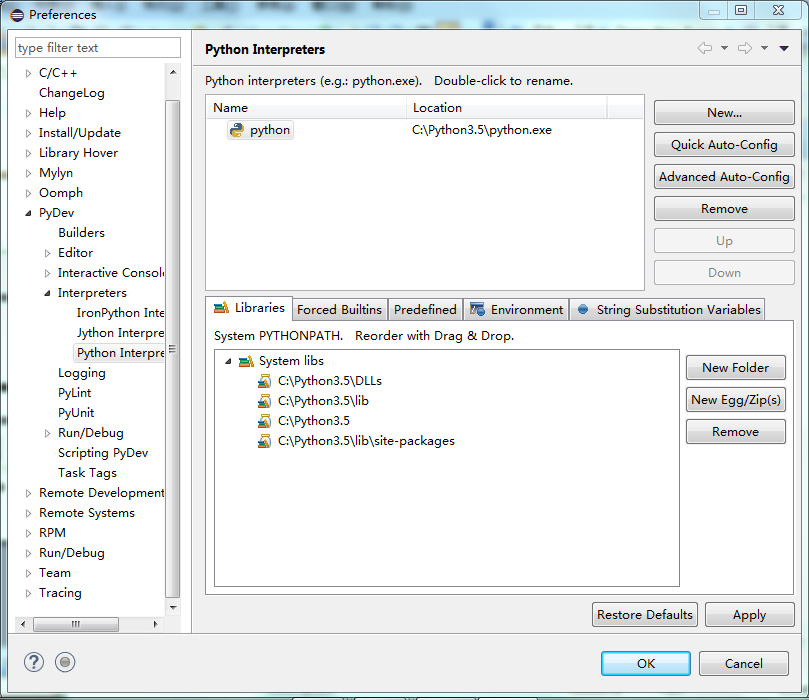
下载插件包，然后，通过：

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:

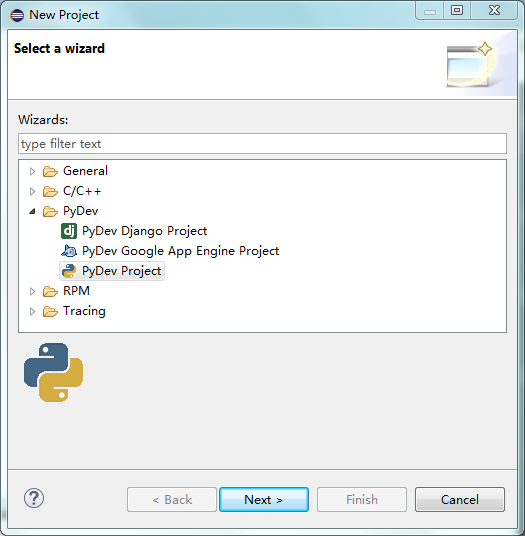


配置好后切换到Python场景，就可以进行Python开发了。

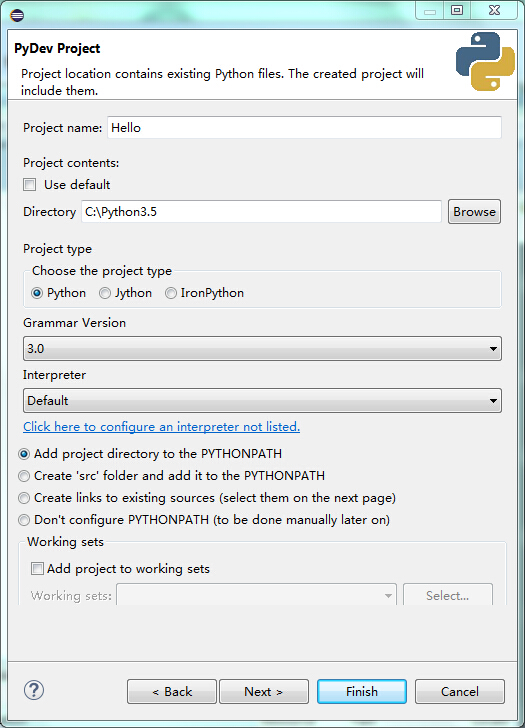
**3.3 创建python工程的方法**

3.3.1创建python工程

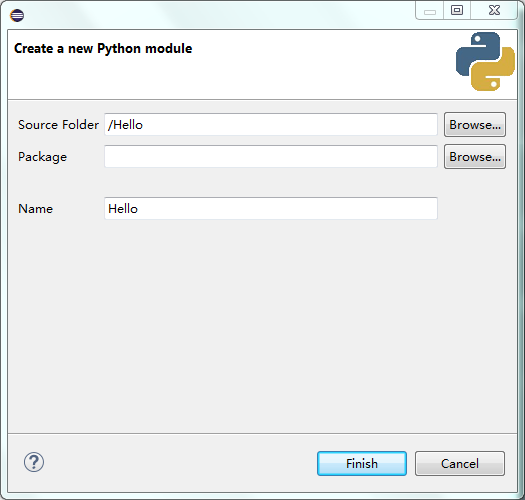
通过File->New->Project，在New Project中选择Pydev->Pydev Project，建立新的工程



选择语法版本为3.0，取消使用默认保存位置，改为如图所示位置。



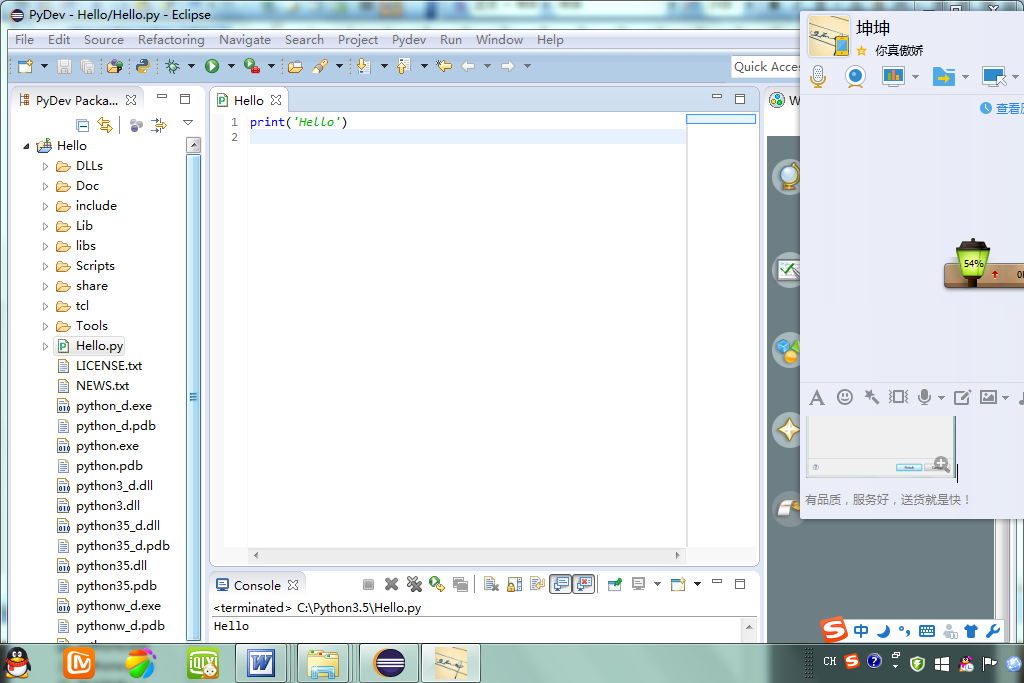
在左侧workplace中选择新建的工程，File->New->Pydev Module以建立可编写窗口。



编写最简单的工程示例Hello !作为示范

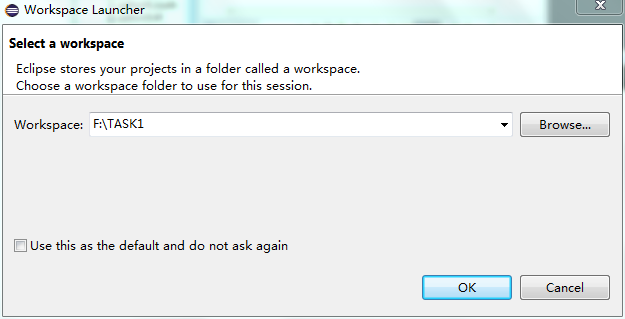
代码内容：print('Hello ')

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。

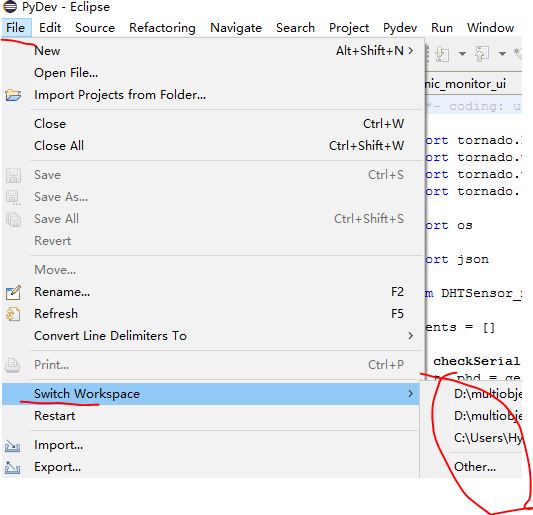


3.3.2配置Eclipse工作空间

当打开Eclipse时，会弹出以下对话框：



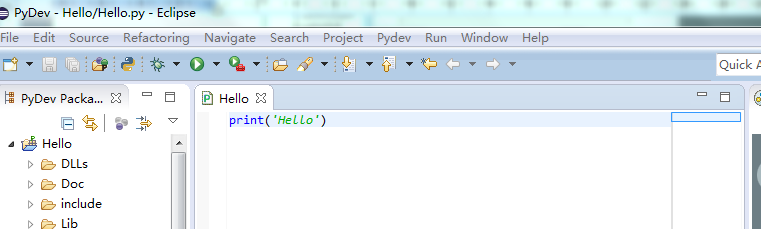
点击Broswse,选择合适的工作空间存放位置，需注意：工作空间不能存放在C盘系统盘，因为一旦系统崩溃，所编辑的文件将全部丢失，且工作空间需要建立独立的阶段程序开发任务相关的一级目录。也可以通过点击File-Switch workspace-other来配置合适的工作空间：



**四、配置 PyDev提高效率**

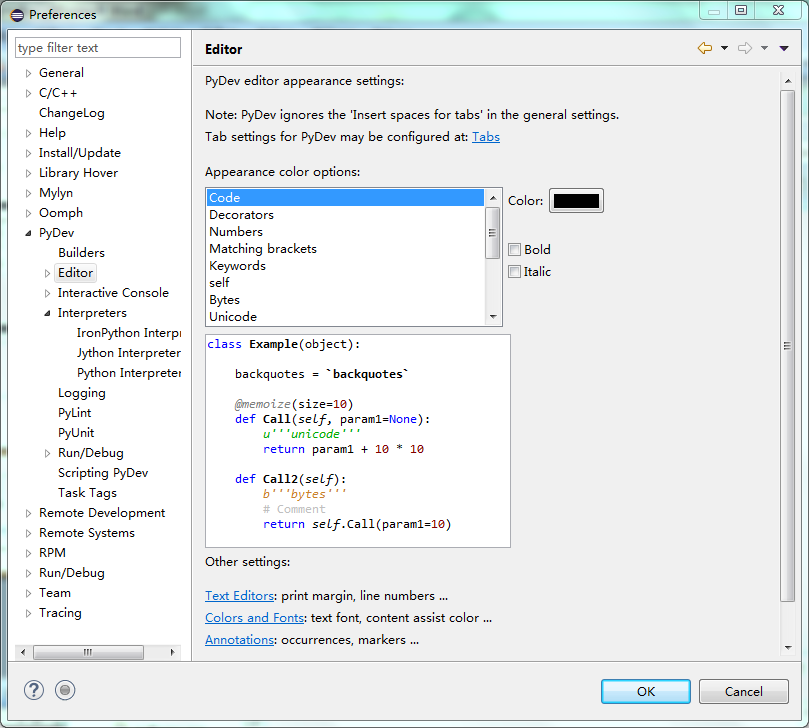
**4.1 显示源码行号**

右键源码的左边缘，选中“Show Line Number”,则代码显示为



**4.2 修改注释颜色提高可读行**

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

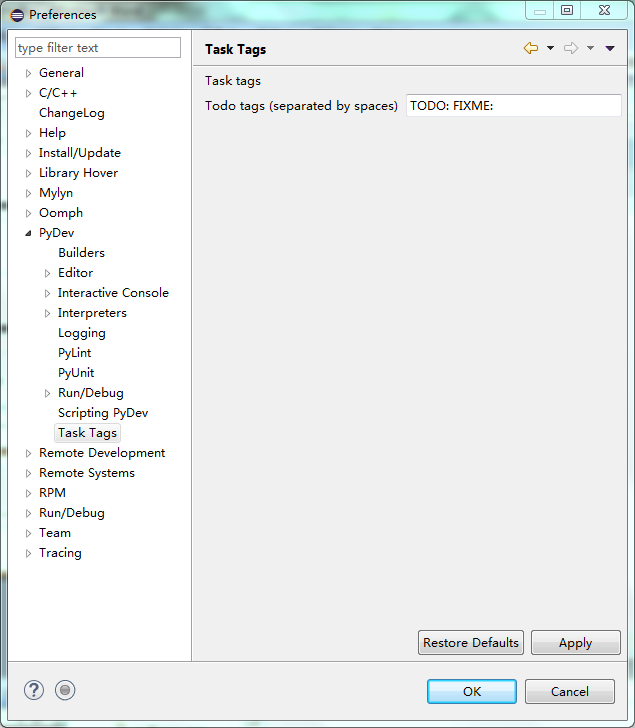


**4.3 配置任务标签**

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Window -> Preferences -> PyDev->Task Tags中配置任务标签:



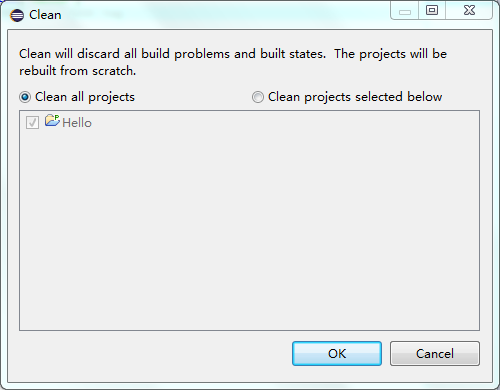
可以根据不同的任务类型，自己增加自定义标签

2）添加任务标签注释

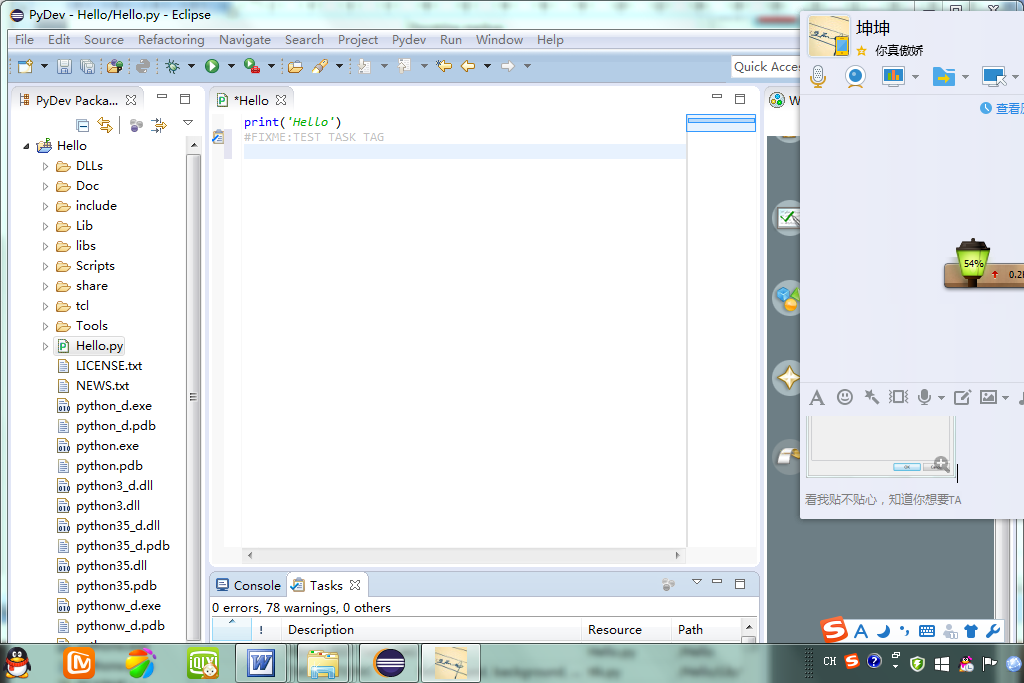
在程序源码中加入使用任务标签的注释：

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。

**五、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。

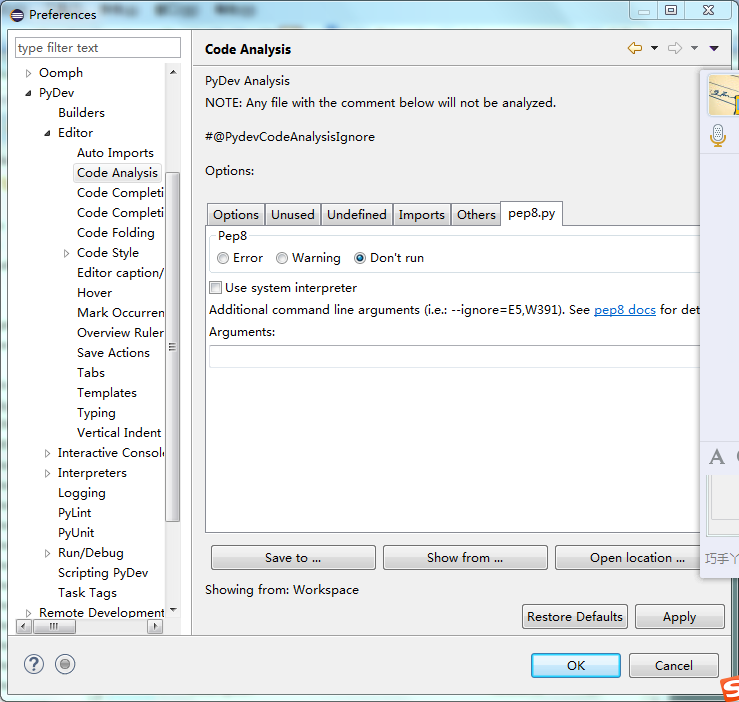
**5.1 PEP8检查和修改**

1）启动 pep8 检查：

Window > Preferences

PyDev > Editor > Code Analysis > pep8.py

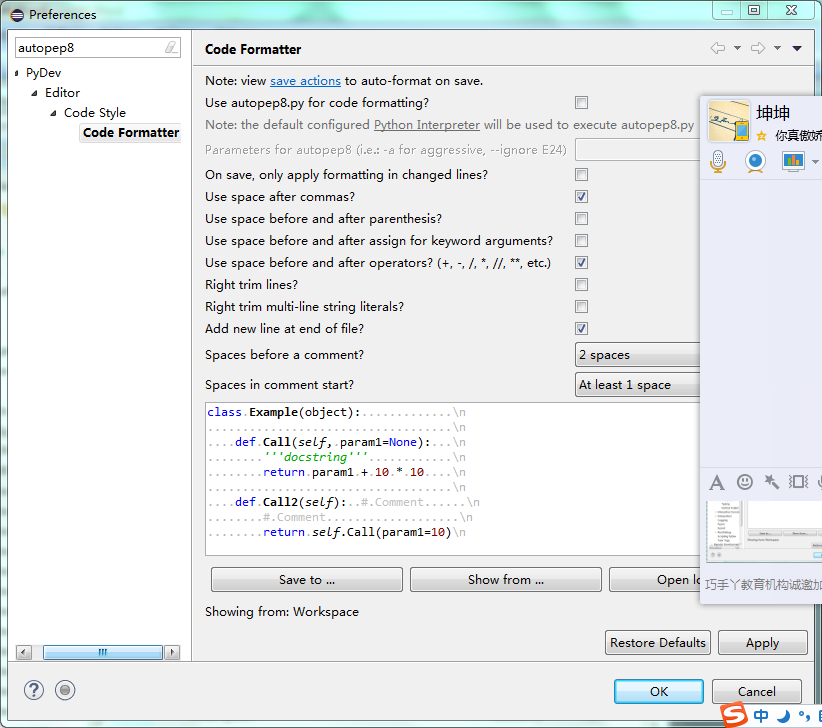
选择Errors/Warnings其中之一..



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?

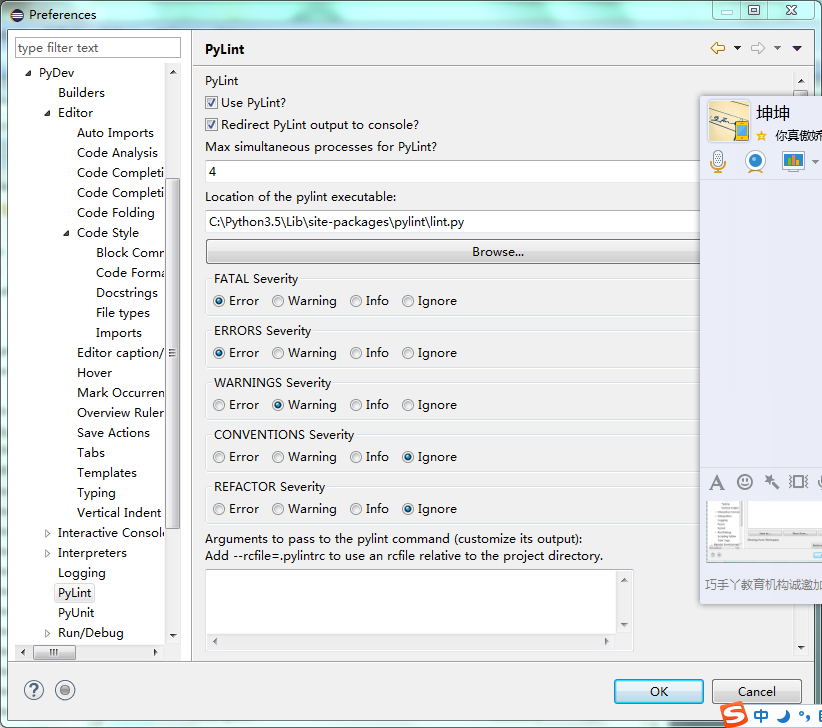


在Python源码窗口，按 CTRL-SHIFT-F 就可以自动修改代码。

**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。

**六、小结**

完成“课程实践作业一”即“Python学习和开发环境的建立”的过程对我来说可能更多的是照猫画虎，根据教学版一步一步的安装学习，或许还不能熟记理解每一步骤及功能，但总算对软件工程有了大概的轮廓。老师一直在强调这门课程的学习就是一个发现问题解决问题的过程，我也感觉到了这门课程与其他一些基础课程学习方法的不同，找到了自己的又一个薄弱环节。相信通过逐渐深入了解学习软件工程这门课程，我一定可以强化自己这方面的能力。