

学校代码： 10286

学 号： 03014404

**软件工程基础及实践**

课程实践作业二

学生姓名**： 姚依晨**

导师姓名**： 程懋华**

2017年03月12日

**目录**

[**一、** **Python基本开发环境和程序示例：IDLE** 3](#_Toc477110751)

[**1.1** **Python语言的简介** 3](#_Toc477110752)

[**1.2** **Python 3.5.3 的安装过程** 4](#_Toc477110753)

[**1.3** **Python 3.5.3 的程序示例** 6](#_Toc477110754)

[**二、** **扩展模块** 8](#_Toc477110755)

[**2.1** **安装科学计算包** 8](#_Toc477110756)

[**2.2** **安装规范格式扩展包** 8](#_Toc477110757)

[**2.3** **安装IF97物性计算** 9](#_Toc477110758)

[**三、** **交互计算环境和程序示例：Jupyter Notebook** 10](#_Toc477110759)

[**3.1** **Jupyter Notebook 的安装过程** 10](#_Toc477110760)

[**3.2** **Jupyter notebook 的使用示例** 11](#_Toc477110761)

[四、 **集成开发环境和程序示例：Eclipse** 16](#_Toc477110762)

[**4.1** **Eclipse的安装和配置** 16](#_Toc477110763)

[**4.1.1** **安装Java SDK** 16](#_Toc477110764)

[**4.1.2** **安装Eclipse** 17](#_Toc477110765)

[**4.1.3** **配置Eclipse工作空间** 17](#_Toc477110766)

[**4.1.4** **安装和配置插件PyDev** 18](#_Toc477110767)

[**4.1.5** **基于PyDev的语言规范静态检查** 21](#_Toc477110768)

[**4.1.6** **Eclipse配置和使用** 22](#_Toc477110769)

[**4.1.7** **Markdown插件** 24](#_Toc477110770)

[**4.2** **Eclipse使用示例** 25](#_Toc477110771)

[**五、** **遇到的问题及解决方法** 27](#_Toc477110772)

[**六、** **个人小结** 30](#_Toc477110773)

[**七、** **参考文献** 31](#_Toc477110774)

1. **Python基本开发环境和程序示例：IDLE**
   1. **Python语言的简介**

Python语言中有大量的内置库以及第三方库，这为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容。[1]

Guido van Rossum给Python的定位是“优雅”、“明确”、“简单”。Python简单易读，容易上手，能够让程序员用最少的代码实现最好的功能。[2]但是同样地，为了“优雅”，Python的作者有意的设计限制性很强的语法，使得不好的编程习惯都不能通过编译。其中很重要的一项就是Python的缩进规则（例如if语句的下一行不向右缩进）。

Python的另一大特点就是开源，由于Python语言是解释型语言，它只能发布源码出去，而不能像C语言一样只发布出机械码，故而它并不能靠卖软件授权。但是随着时代的发展，靠网站和移动应用卖服务的模式越来越多了，这就并不需要程序员提供源代码。另外这种不能加密的特点同样促进了Python的交流和发展，这也和现在如火如荼的开源运动和互联网自由开放的精神是一致的。

Python的缺点也同样显而易见，它的运行速度慢，和C程序相比非常慢。因为Python是解释型语言，你的代码在执行时会一行一行地翻译成CPU能理解的机器码，这个翻译过程非常耗时，所以很慢。而C程序是运行前直接编译成CPU能执行的机器码，所以非常快。事实上，Python开发人员尽量避开不成熟或者不重要的优化。一些针对非重要部位的加快运行速度的补丁通常不会被合并到Python内。不过，根据二八定律，大多数程序对速度要求不高，因为0.1秒和0.001秒看起来似乎差别也不是那么大。而在某些对运行速度要求很高的情况，Python设计师倾向于使用JIT技术，或者用使用C/C++语言改写这部分程序。可用的JIT技术是PyPy。

* 1. **Python 3.5.3 的安装过程**

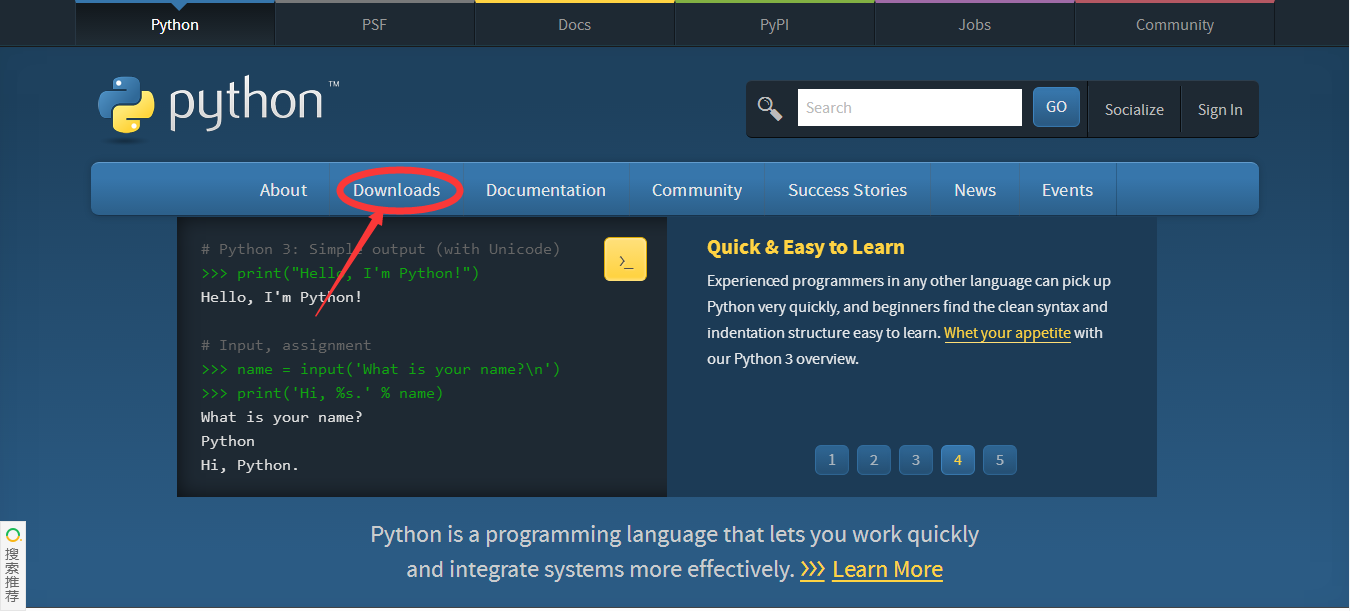
第一步是从登录官网<http://www.python.org>，点击上面目录的Download，进入版本选择的页面：

图 1 官网页面

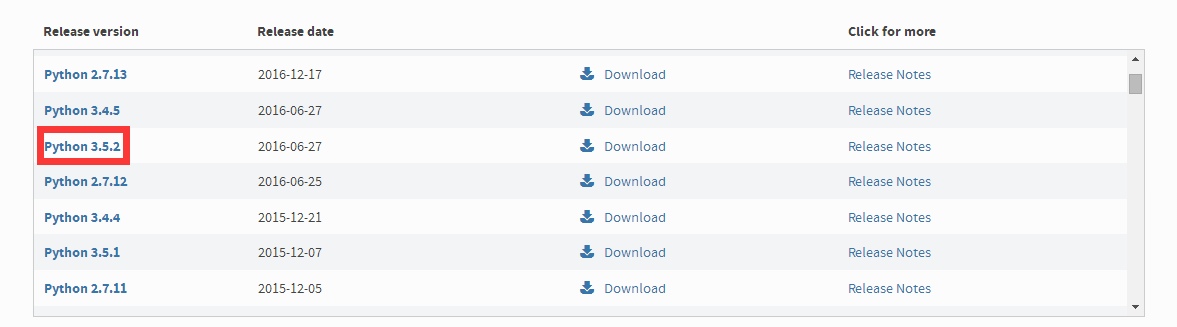
我个人选择的是Python 3.5.2版本：

图 2 版本选择

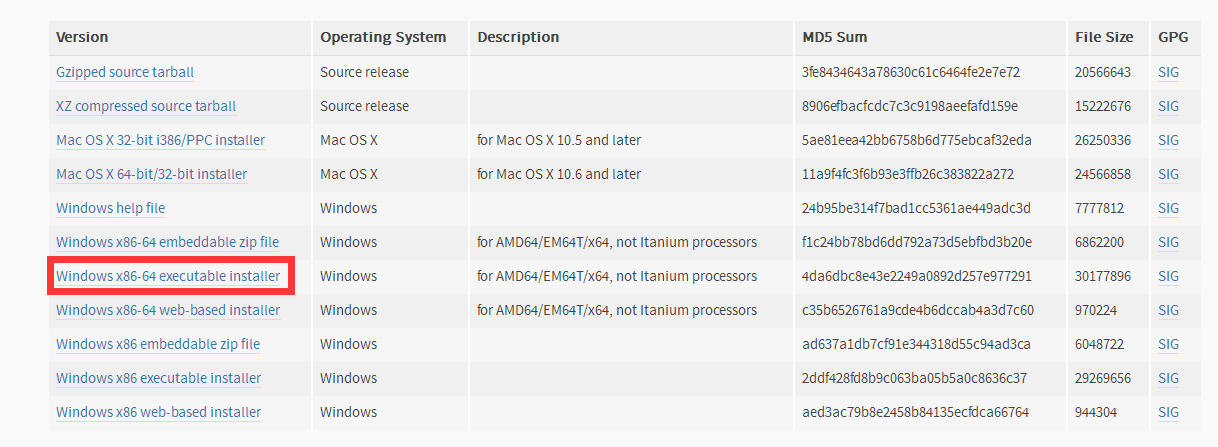
以我的Win8，64位系统为例，选择相应的系统，已在图3中标出：

图 3 Python下载区

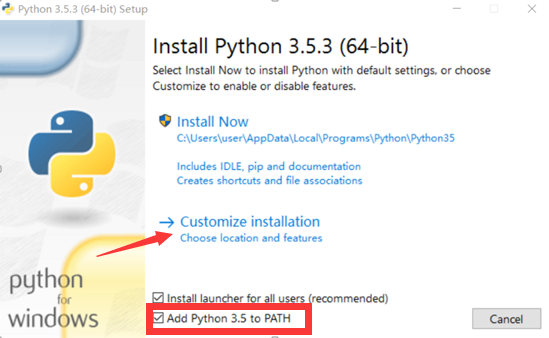
双击上图的链接，在弹出的对话框中，应当勾选Add Python 3.5 to PATH（若不选中，在命令提示符中就无法运行python，这是因为Windows会根据一个Path的环境变量设定的路径去查找python.exe，如果没找到，就会报错。如果在安装时漏掉了勾选Add Python 3.5 to PATH，那就要手动把python.exe所在的路径添加到Path中）。

图 4 Python安装页面

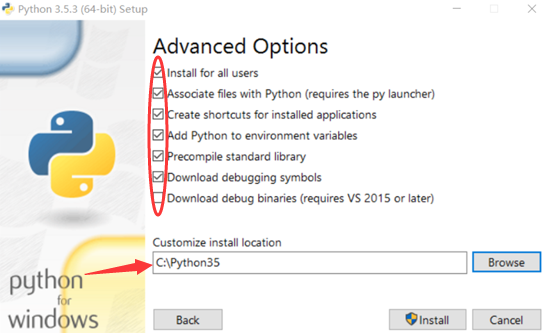
 然后勾选所弹出的下一个对话框的前六个选项，最后一个选项是基于VS2015调试的选项，由于本人的电脑并未装载该软件，故不予勾选。另外，路径最好不要使用默认路径，使用自定义路径更加便于查找。

图 5 安装选项以及路径选择

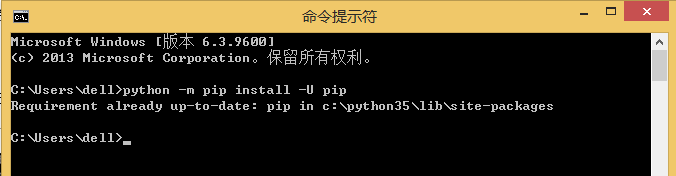
安装完成以后，打开命令提示符，输入>python –m pip install –U pip 检查安装是否成功。如果成功，则会出现如下图所示的版本号：

图 6 安装检查

* 1. **Python 3.5.3 的程序示例**

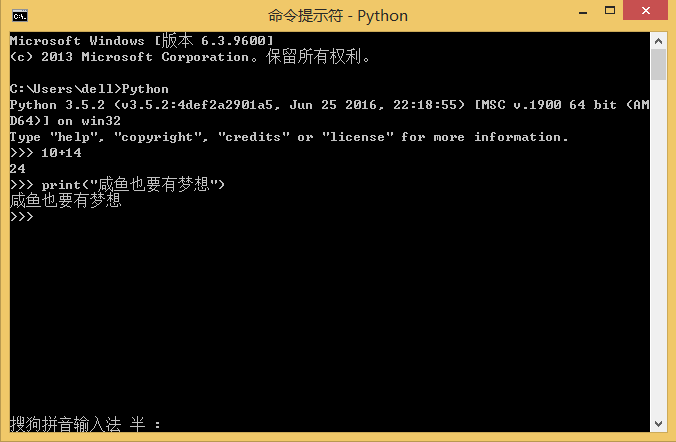
 我们可以直接在命令提示符里面输入代码，然后敲击enter键，直接运行。这种方法不利于复杂程序的运行：

图 7 命令提示符运行Python程序示例

同样的，打开IDLE窗口（可以在开始菜单里找到），我们可以直接向其中输入代码，然后点击enter键即可弹出结果，其使用方法类似mathematics的笔记本。他们都是交互式命令，使用简单，但是不便于保存。

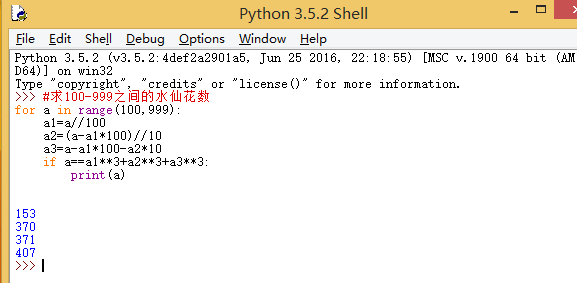


图 8 shell文件直接运行Python示例

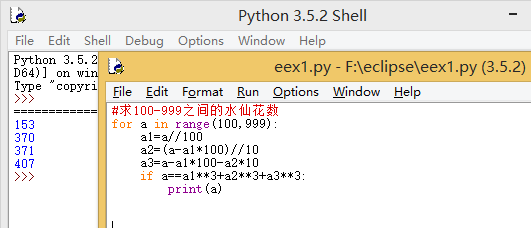
为了下次还能正常使用这个程序，我们可以将之保存在一个后缀为.py的文件中，这样就可以直接打开，点击run，它就可以自动弹出shell窗口并输出结果：

图 9 利用文件保存Python程序并运行示例

1. **扩展模块**
   1. **安装科学计算包**

这里只介绍SciPy、numpy、matplotlib三个包的安装方法。

 首先打开加州大学欧文分校的网址<http://www.lfd.uci.edu/~gohlke/pythonlibs/>，从中找到适合自己电脑版本以及适合刚刚安装的Python版本的安装包。比如下图中的NumPy安装包，第一个数表示NumPy的版本，mkl是科学计算用的，cp35对于的是python3.5的版本，最后是指系统:

图 10 科学计算包的选择示例

打开命令提示符，输入 >pip install \*.whl。\*用扩展包名字替换，如：>pip install numpy-1.12.0+mkl-cp35-cp35m-win\_amd64.whl，要注意的是，需要cd到安装包所下载的目录下才能找到并且安装。

* 1. **安装规范格式扩展包**

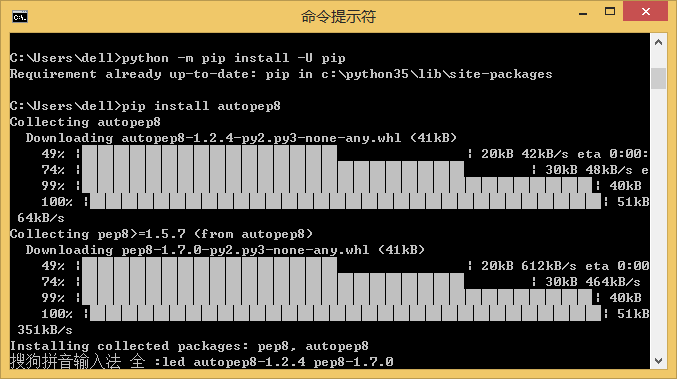
**** 安装扩展模块pep8、autopep8 和 pylint，一般不用另外下载，直接输入pip install pep8即可在线安装。这一步安装python之后就进行，这样后续环境的建立才比较顺利。

图 11 autopep8的安装

****

图 12 pylint的安装

* 1. **安装IF97物性计算**

对于IF97物性计算可使用基于C语言实现共享库、纯Python语言包。C语言共享库计算速度远快于Python语言包。C语言IF97共享库及其Python封装包仓库：

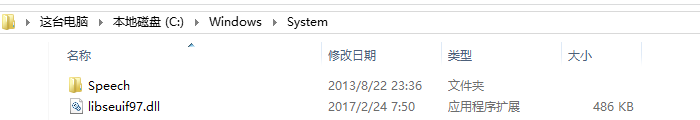
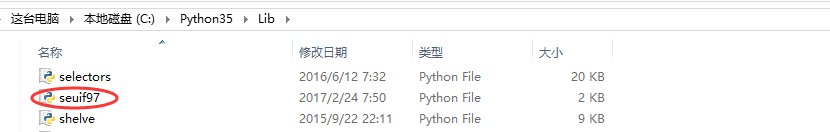
https://github.com/PySEE/SEUIF97，下载仓库的zip文件并解压，将和操作系统对应版本的libseuif97.dll和 seuif97.py，seuif97.dll拷贝到 c:\windows\system，seuif97.py拷贝到 c:\python35\Lib。

图 13 seuif97.py拷贝到 c:\python35\Lib

图 14 seuif97.dll拷贝到c:\windows\system

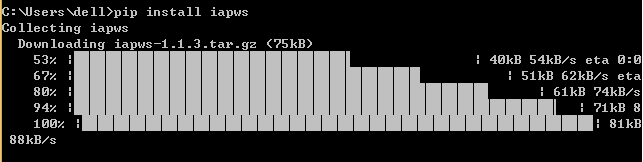
Python语言计算包，在线安装：>pip install iapws

图 15 在线安装iapws

1. **交互计算环境和程序示例：Jupyter Notebook**
   1. **Jupyter Notebook 的安装过程**

Jupyter Notebook是一个交互式开发环境，可以在安装Python的基础上直接用命令提示符进行安装。只要联网，然后输入命令>pip install jupyter即可安装。

下图是已经安装好之后的截图：

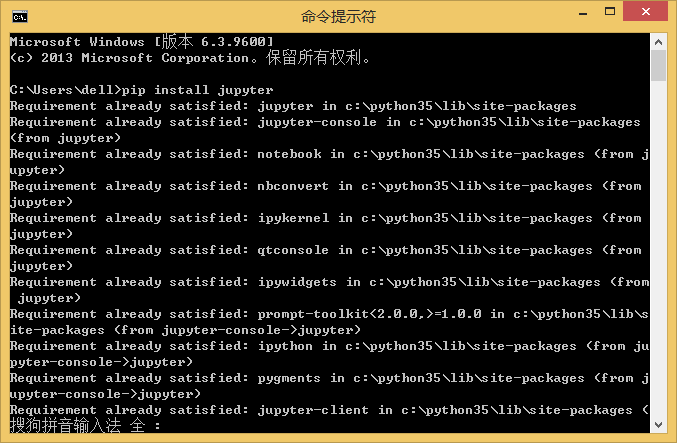
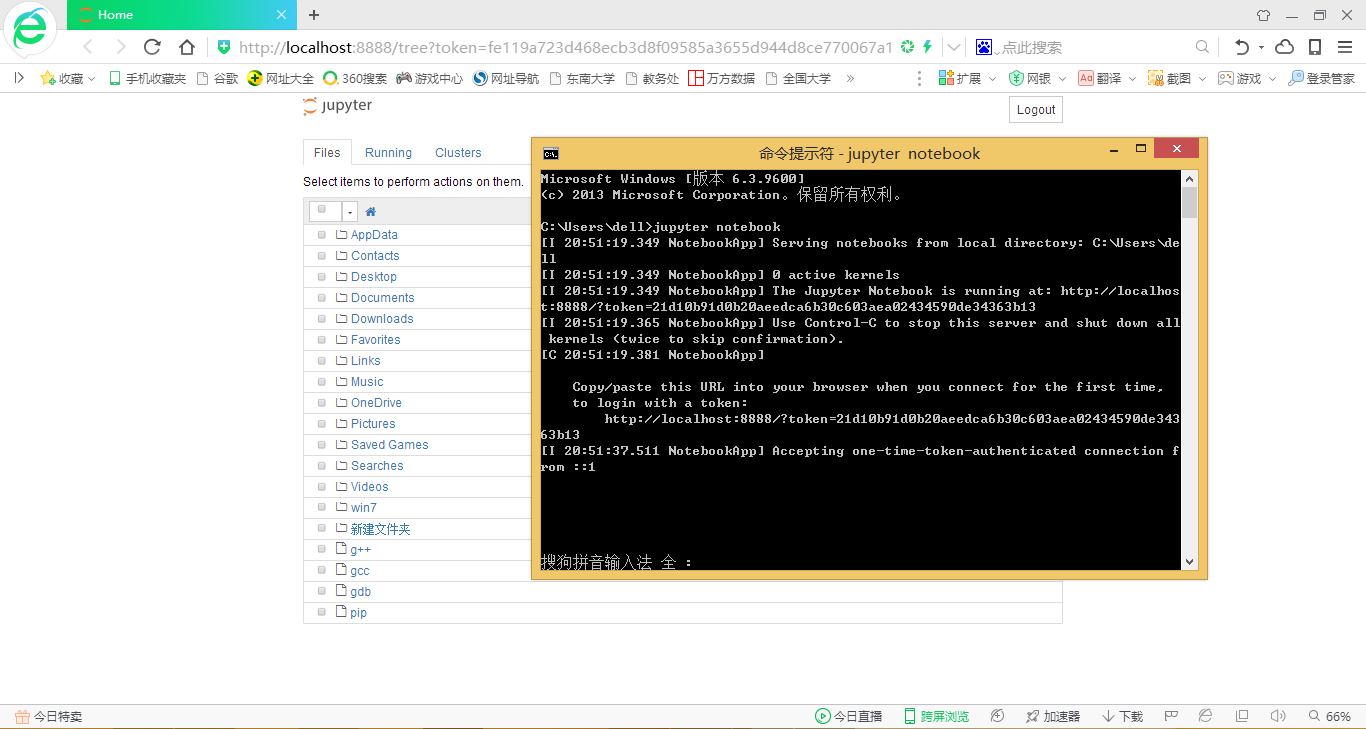
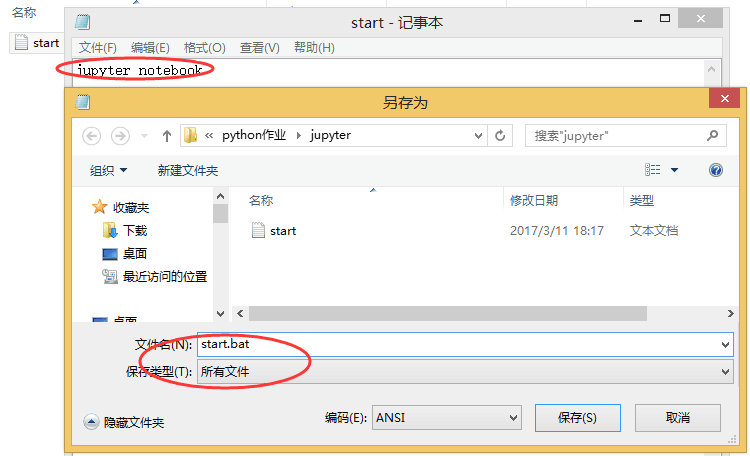
在制定目录下输入>jupyter notebook 就可以运行并用notebook模式打开这一目录。

图 16用命令提示符打开jupyter notebook

图 17 jupyter notebook的安装

* 1. **Jupyter notebook 的使用示例**

除了命令提示符之外，我们还可以用另一种方法打开jupyter notebook。先新建一个文件夹，接着在文件夹建立一个内容为jupyter notebook的txt文本，另存为所有文件，改名为.bat。之后就可以双击这个文件打开此文件夹的notebook了。同时，我们可以在jupyter notebook的这个网页上对这个文件夹进行操作，比如新建一个文件夹或是新建一个py文件，这些都会一一映照在我们本地的文件夹中：



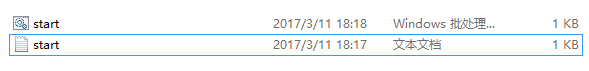
****

图 18 生成bat文件

图 19 将txt另存为bat

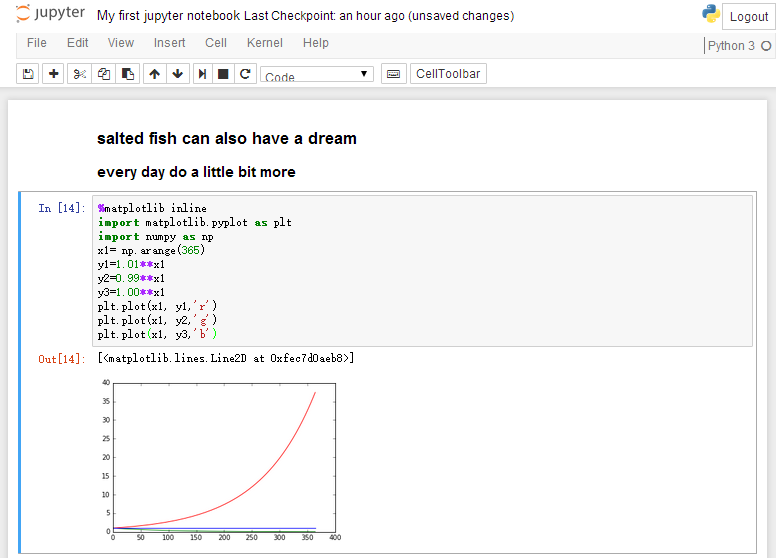
Jupyter notebook 支持多种编辑，在同一个文件中，不同的cell我们可以编辑不同的语言，比如markdown 和Python，它们可以独自编译。另外markdown还可以实现标题形式，改变颜色等，非常方便：

图 20 jupyter notebook 的使用示例[3]

上面的程序是实现了matplotlib中的画图功能在Python中的运用，这样我们就可以在jupyter notebook中画出类似MATLAB中的图。不仅如此，numpy等我们已经安装添加过的包，也可以用这种方式被调用。这就好像c++中的包含头文件#include<cmath> usingnamespace 一样，声明之后就可以调用其中已经封装的函数。

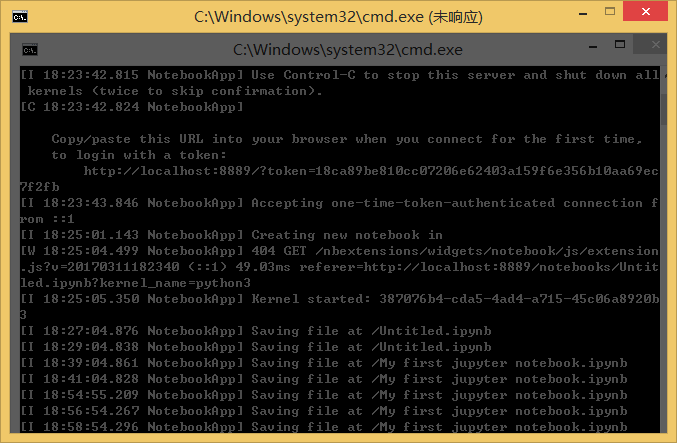
同时，命令提示符会一直记载这段时间我们对于这个文件夹的操作：

图 21 命令提示符的记录

另外我们还可以用jupyter notebook来进行有关git的操作，比如先寻找合适的目录打开jupyter notebook，创建Python3，接着利用语句来创建一个仓库，将仓库交给git管理，然后还可以自己写一个README.MD文件添加到版本库：

图 22 用jupyter notebook来创建仓库[2]

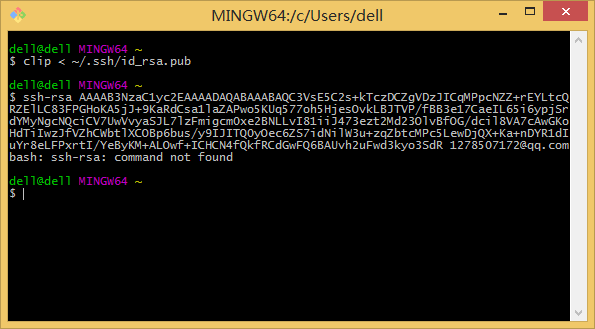
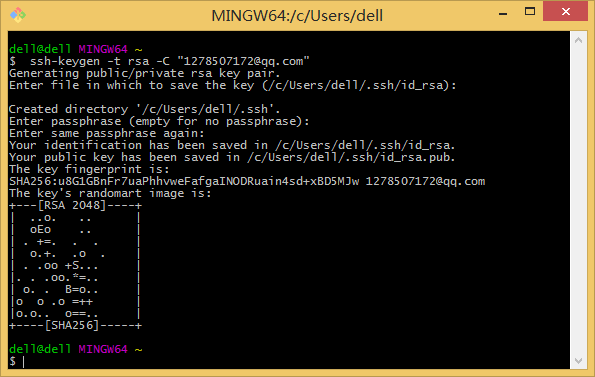
其实我们还可以把这个库与GitHub上面的远程库关联起来。先在用户目录下面寻找.ssh文件夹。由于我的电脑中没有这个文件夹，所以我需要用git bash自己创建：

图 23 git bash创建.ssh

图 24 git bash 拷贝rsa.pub

第二步是在GitHub的相应位置输入本机的ssh秘钥然后创建远程仓库:

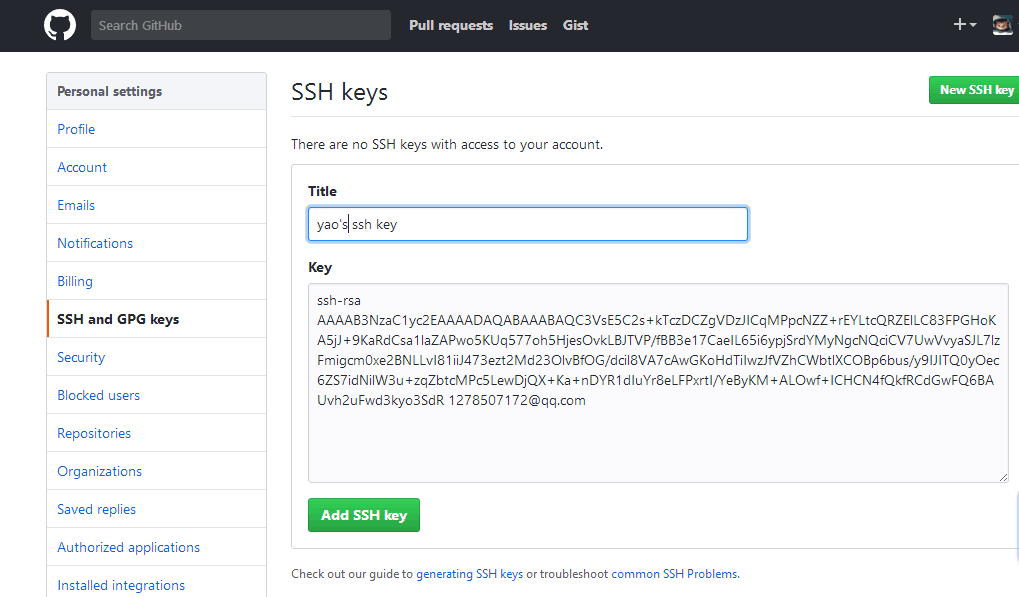


图 25 在GitHub上添加秘钥

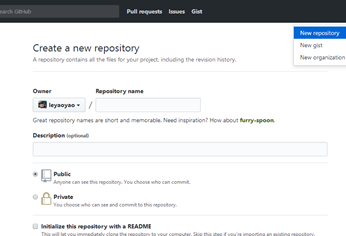


图 26 在GitHub上找到建立本地与远程仓库的联系的代码

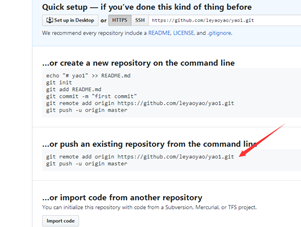


图 27 在GitHub上建立远程仓库

在建立新仓库后会弹出一个页面，找到其中指导我们建立本地与远程仓库连接的代码然后复制到jupyter notebook上，我们就完成了本地仓库与远程仓库的连接。

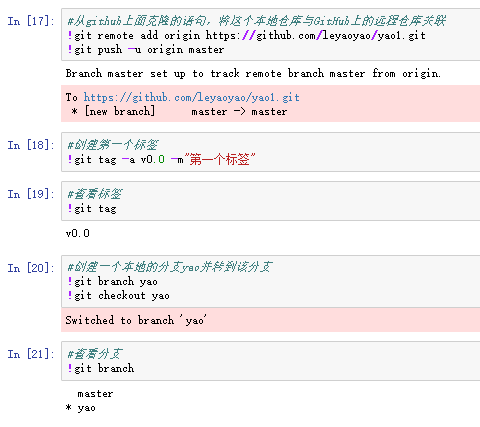
之后我们还可以用jupyter notebook来进行添加标签，添加分支，合并分支，删除分支，推送至远程仓库等一系列操作：

图 28 完成与远程仓库关联并新建标签

图 29 建立分支

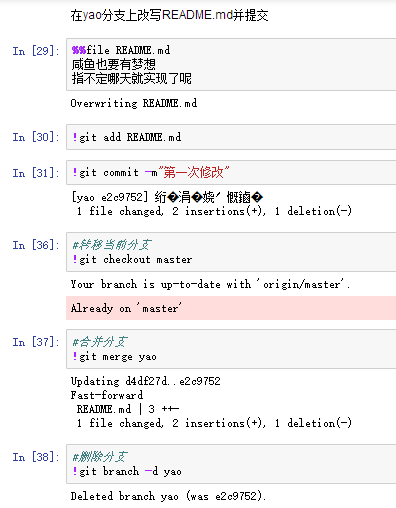
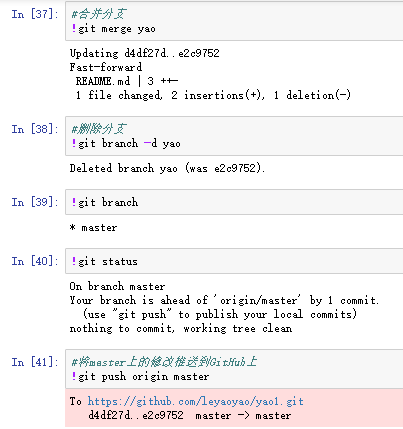


图 30 合并分支，删除分支并推送至GitHub

1. **集成开发环境和程序示例：Eclipse**
   1. **Eclipse的安装和配置**
      1. **安装Java SDK**

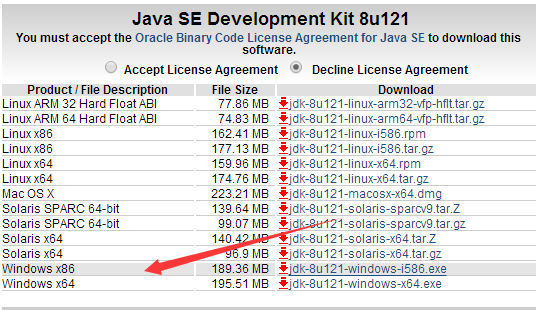
Eclipse IDE是使用Java开发的，电脑中需要预先安装好Java JRE/JDK软件包，但是我使用的电脑中并没有装过Java，故而从Oracle下载Java包。Java JDK的Oracle官方下载地址是<http://www.oracle.com/technetwork/java/javase/downloads/index.html>，下载时要注意选择与自己电脑匹配的版本，我的电脑是64位的：

图 31 JAVA的下载地址

安装好之后会有提示：

图 32 JAVA安装好的提示

* + 1. **安装Eclipse**

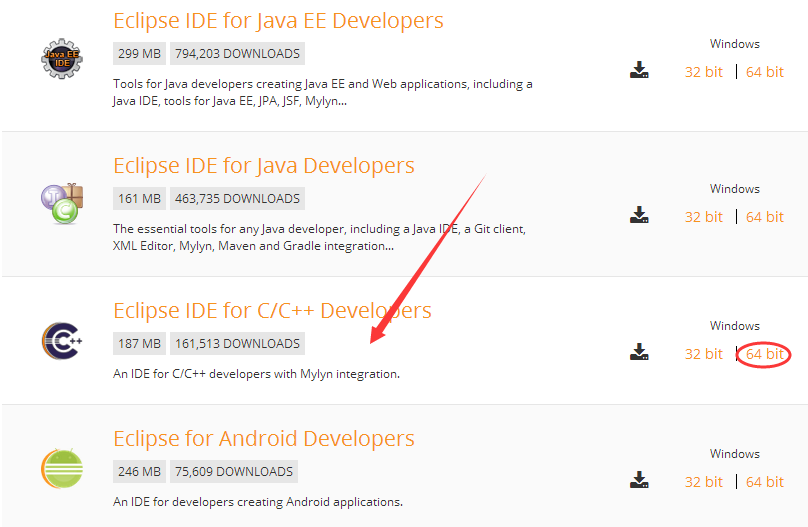
Eclipse IDE是Eclipse基金会开发的插件型集成开发环境，Eclipse IDE有很多版本。而我们这学期使用Python语言，可能会涉及C/C++开发，所以，我下载的是Eclipse CDT版.Eclipse CDT官方下载地址： <http://www.eclipse.org/downloads/eclipse-packages/>，

图 33 eclipse的安装选择

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。

* + 1. **配置Eclipse工作空间**

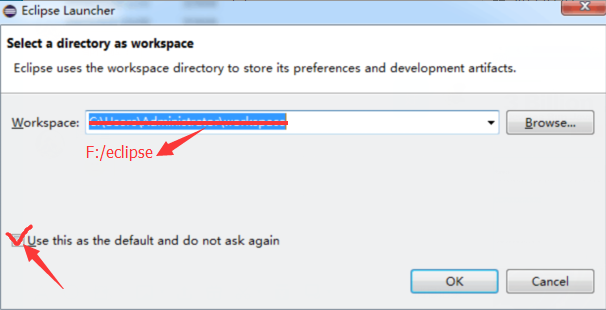
首次打开Eclipse软件时,软件会提示设置默认的工作空间。一般不建议使用Eclipse软件默认的工作空间目录。建议配置自己的工作空间：第一步，在非系统盘中，如F盘中建立一个目录，如:F:\eclipse，然后，打开Eclipse，在其提示设置默认的工作空间时，“Browser”到F:\eclipse，勾选“Use this as the default and do not ask again",就将该目录配置为Eclipse默认当前工作空间目录了，则以后开发的程序代码等都在这个目录下。

图 34 eclipse的工作空间配置

* + 1. **安装和配置插件PyDev**

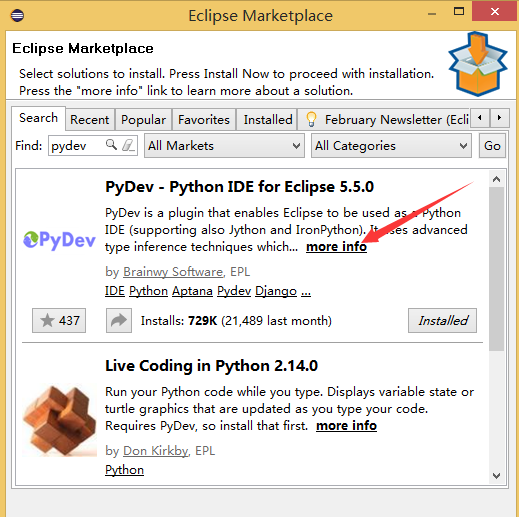
使用Eclipse IDE作为Python开发环境，需要：1）安装PyDev插件；2）配置使用的Python解释器版本。首先来安装PyDev插件，通过Help->Eclipse Marketplaces进入市场，输入pydev，找到Pydev项目，点“install”在线安装。

图 35 安装PyDe插件

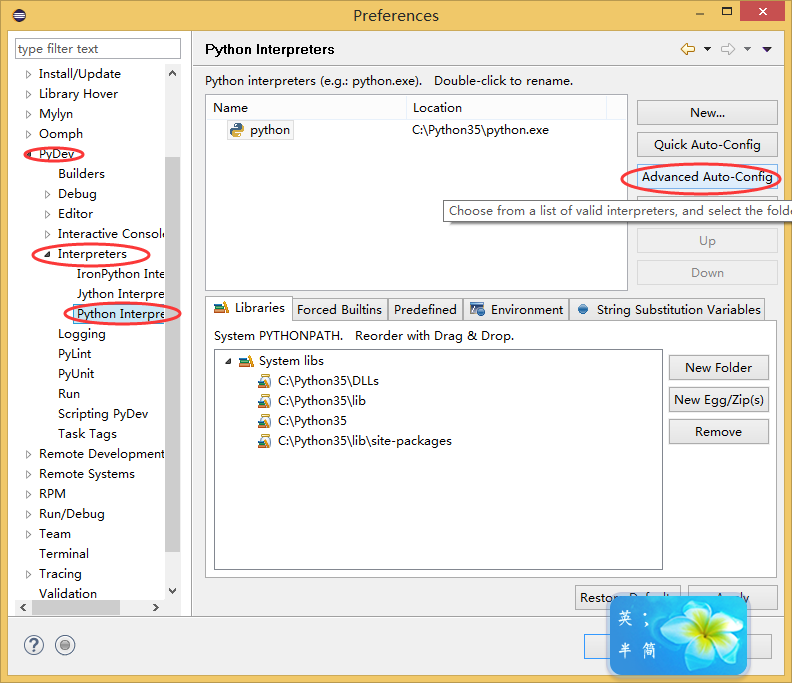
安装好后，重新启动。通过：Windows->Preference->Pydev->Interperters->Python Interperter点其中的：Advanced Auto-config配置开发使用的 Python解释器版本:

图 36 配置Python解释器版本

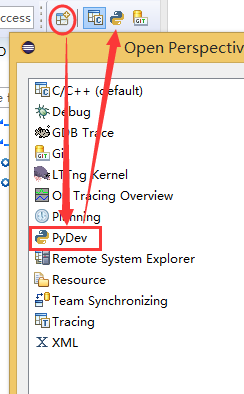
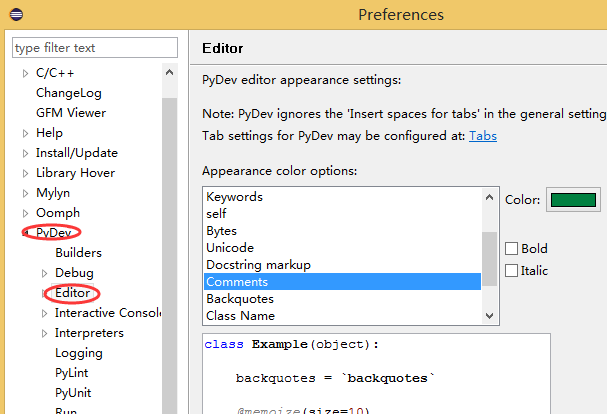
配置好后，添加Python场景到工作页面,如图38：

图 37 修改编辑器的配色

图 38 添加Python场景到工作页面

除此以外，为了使我们的编程更加方便，我们还可以修改编辑器的配色和进行任务标签的配置。

其中修改编辑器的配色示例见上图37. 通过从Window → Preferences→PyDev→Editor,进入配置界面。

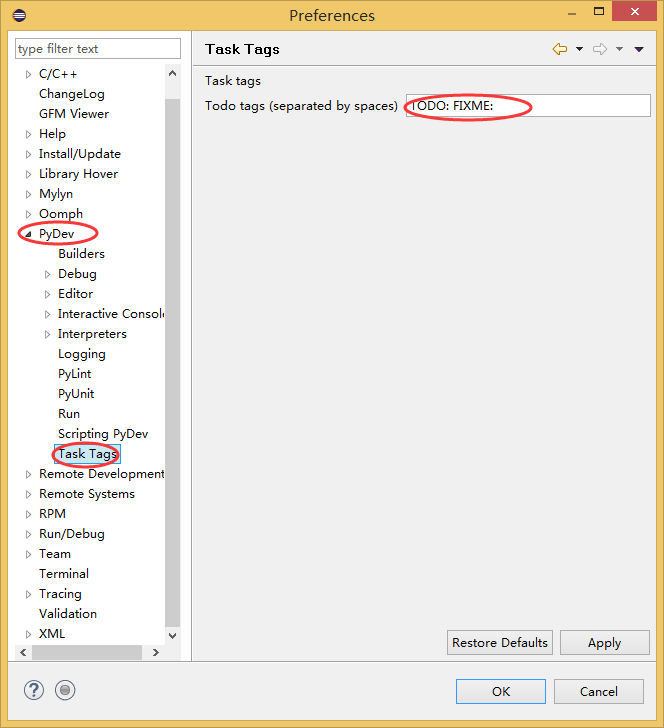
程序开发中，可在代码中标识当前任务状态，计划开发工作。使用“任务标签”在代码中标识任务，然后，让开发环境将其识别出来，加入工作空间的任务列表中。首先，要通过PyDev->Task Tags中配置任务标签：

图 39 配置任务标签

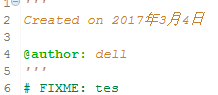
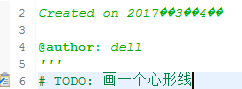
 然后我们就可以添加任务标签了，其实任务标签本身可以理解为是一种注释，只是它用了任务标签规定的关键词（比如上面设定的“TODO”和“FIXME”），故而他可以被检索到：

图 40 添加任务标签

图 41 添加任务标签

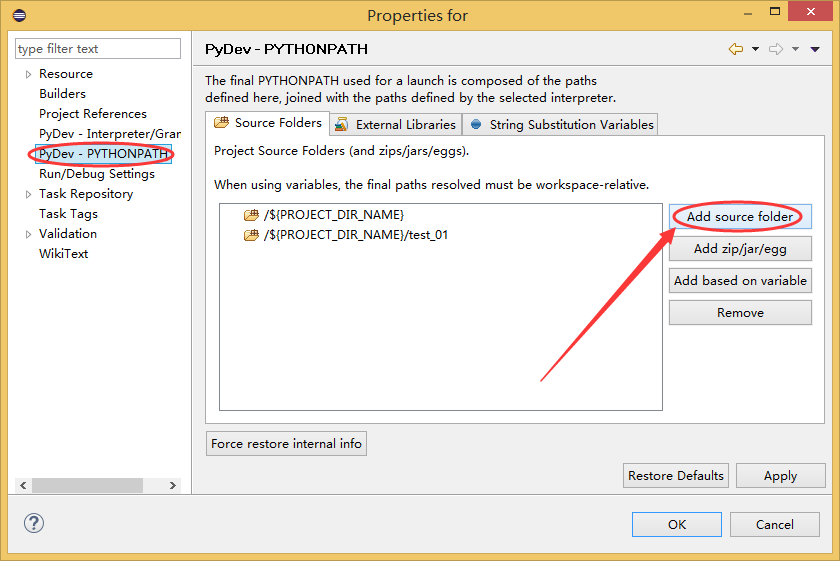
接着，为了能够让他们被自动识别，我们需要配置当前工程的PyDev-PYTHONPATH的source folder。在源码工程的Properties->PyDev-PYTHONPATH配置项中，点“Add source folder”将源码目录加入。

图 42 自动识别

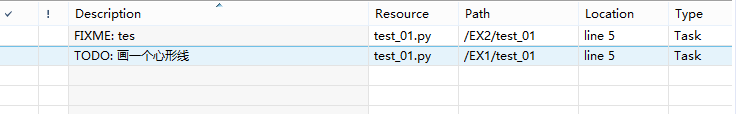
保存新修改或运行程序一次（Ctrl+b）或选择 Project → Clean，就可将使用任务标签注释加入任务列表,相应的任务注释行左边会有 标识#加任务标签关键字所在行为任务标识行。如果当前任务窗口包含“Tasks”，刚加的任务，就会立刻显示在“Tasks”窗口中。如“Tasks”窗口没有打开，可Window->Show View->Tasks显示任务窗口。

图 43 task窗口

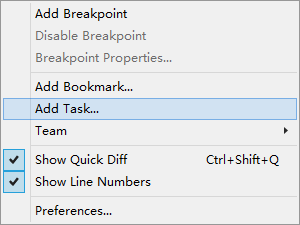
 不止如此，在PyDev 环境下，也可手动添加任务。将鼠标移动到需要添加为任务行的最左侧，点右键，选择“Add Task”,即将该行加入任务列表。

图 44 手动加入任务标签

* + 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。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。

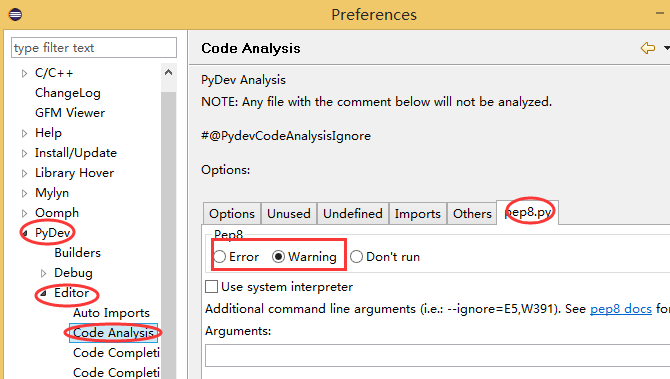
首先，启动 pep8 检查：Window > Preferences PyDev > Editor > Code Analysis > pep8.py选择Errors/Warnings其中之一：

图 45 启动 pep8 检查

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?

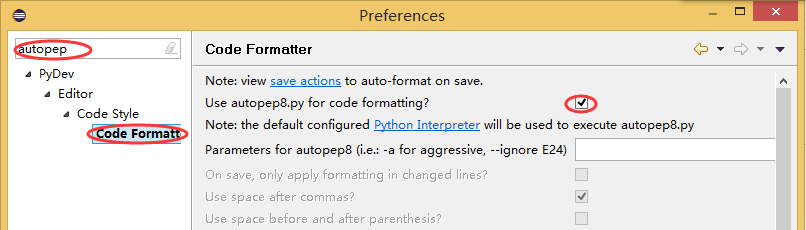


图 46 启动autopep8自动修改

而PyDev 默认不开启Pylint。我们要通过Window -> preferences -> Pydev -> Pylint,选中"Use pylint?",“Browse…”找到安装好的lint.py的地址,例如"C:\Python36\Lib\site-packages\pylint\lint.py"

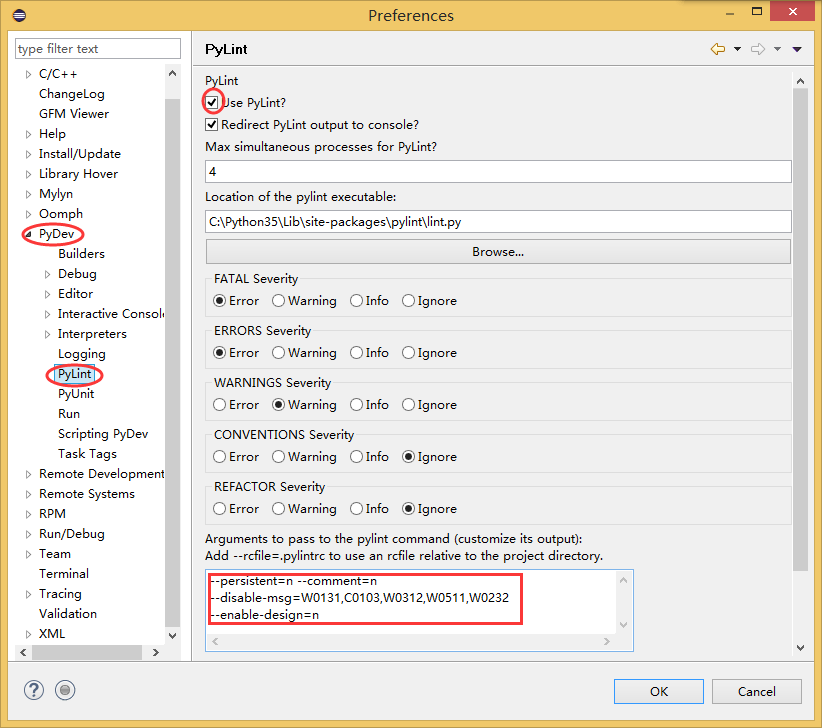


图 47 开启Pylint

* + 1. **Eclipse配置和使用**

开始中文操作系统下，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。

编码方式可以设定到：工作空间、工程和文件类型的不同等级的文件范围上。建议配置编码方式UTF-8到工作空间，工程继承工作空间配置的方式。

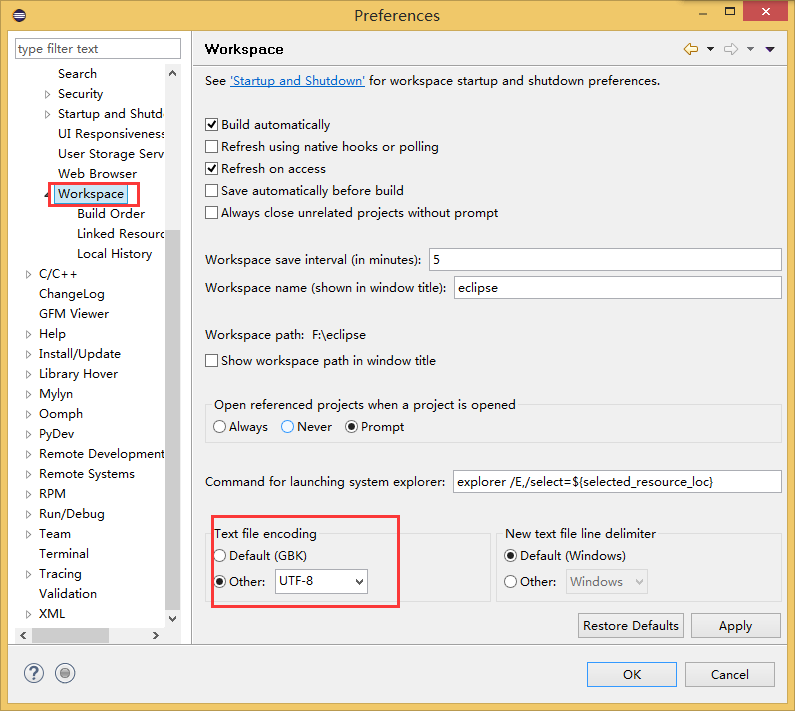
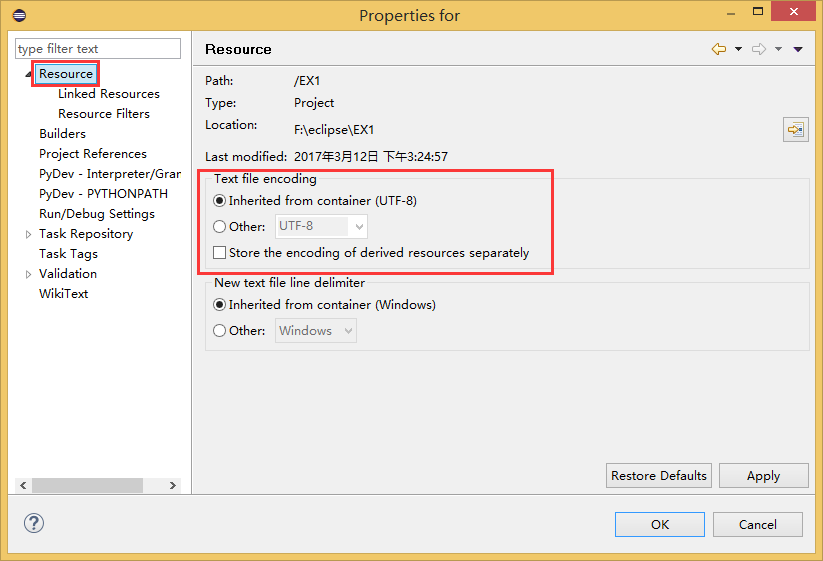
首先是工作空间的编码方式：点“Window”->“Preference”，在弹出窗口中，点击“general”-“workspace”，修改“Text file encoding”为UTF-8：

图 48 工程编码方式

图 49 工作空间编码方式

接着是工程空间的编码方式：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)，如上图48.

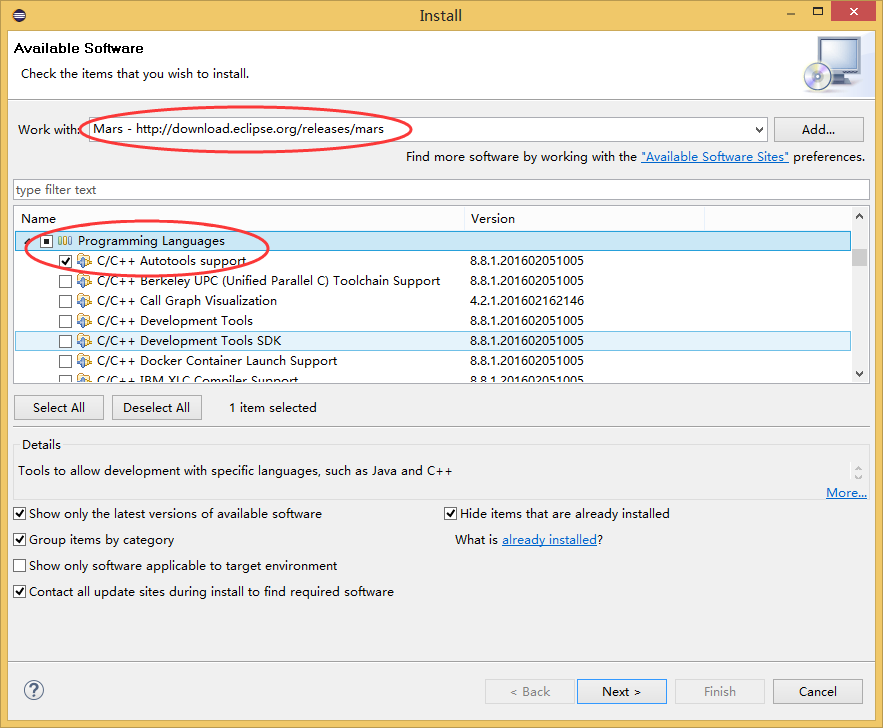
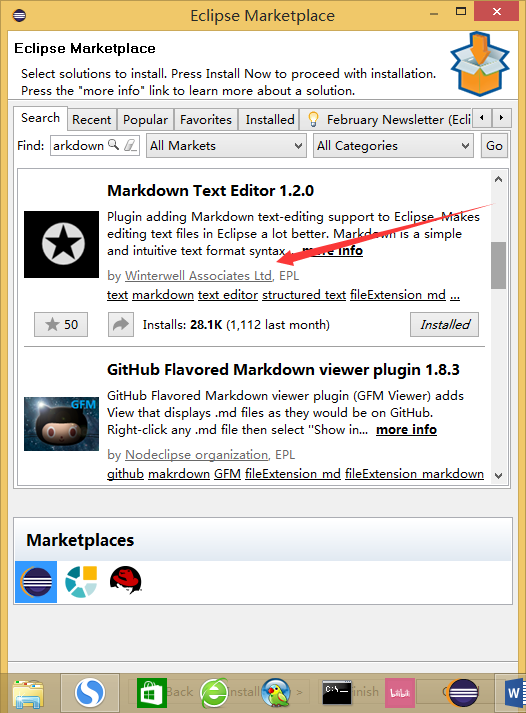
最后是有关插件的补充和更新部分：电脑联网，然后，Help->Install New Software，进入安装软件页面，选择插件的更新网址，然后可看到插件列表，勾选需要安装的插件，按照要求继续下面步骤即可安装好指定的插件：

图 50 插件的补充和更新

* + 1. **Markdown插件**

Eclipse默认安装对Markdown支持较弱，需要安装Markdown插件。Help->Eclipse Marketplace使用Markdown关键字可以找到2个插件，然后选择其中一个安装我选择的是Markdown Editor插件，可Windows->Preferences->General->Editors->Text Editors->Markdown配置有关参数。：

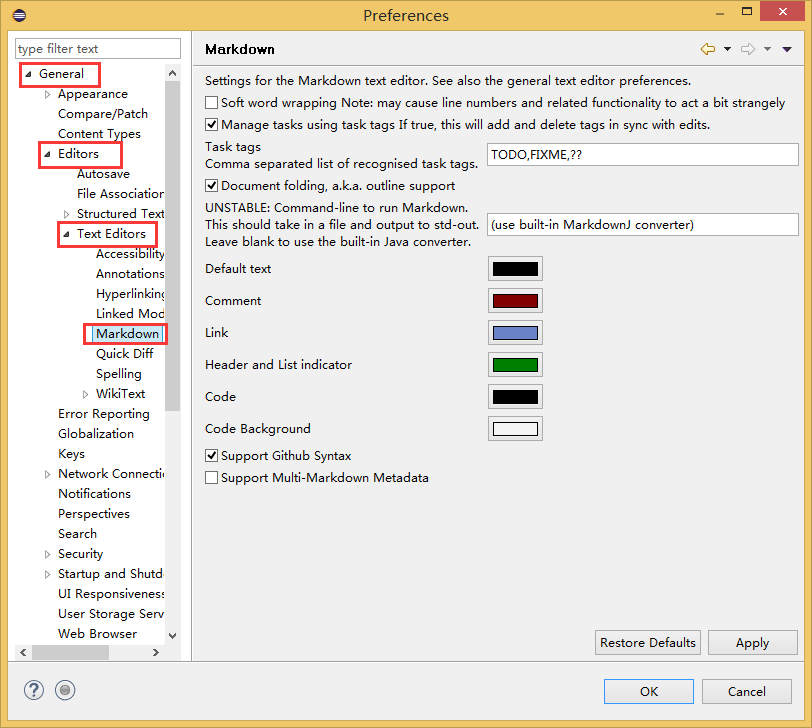


图 51 Markdown插件配置

图 52 Markdown插件

* 1. **Eclipse使用示例**

又是一年女生节，又是一年告白季，还有什么比笛卡尔的心形线更浪漫的情书？我打算在eclipse上用Python语言编一个能够画心形线的小程序自娱自乐。

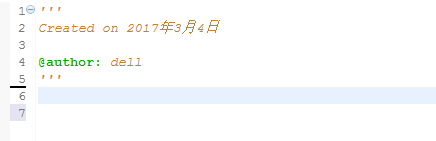
首先在eclipse上的右角选择语言Python，然后在菜单flie->new->pydev project，新建一个pydev project，输入名字为EX1，然后右击workspace里出现的EX1，new->pydev module,在弹出的对话框中输入该module的名字为test\_01,点击OK。再弹出的对话框中选择类型为empty。代码区就会出现如图42所示，就可以欢快地编程了：

图 53 pydev module

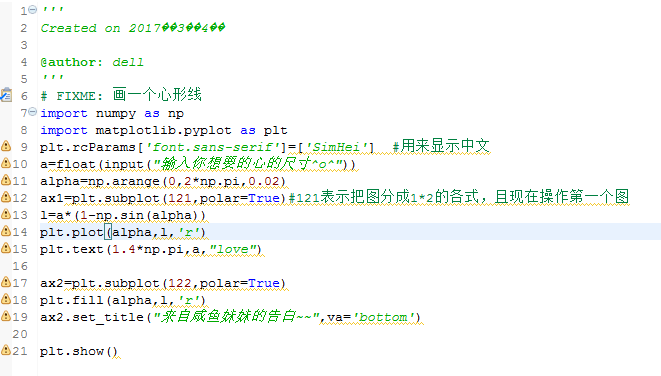
编辑代码如图所示：

图 54 心形线的代码[2,4-7]

由于我设置了标签，故而task栏会出现：

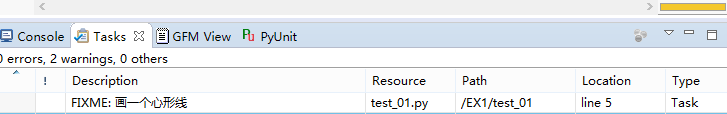


图 55 task栏

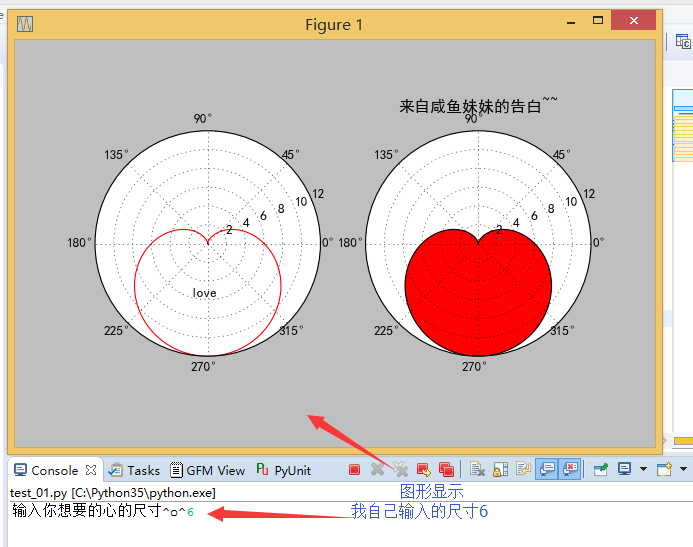
点击菜单里的run->run as->1python run，输出如图56所示：

图 56 eclipse示例输出

1. **遇到的问题及解决方法**

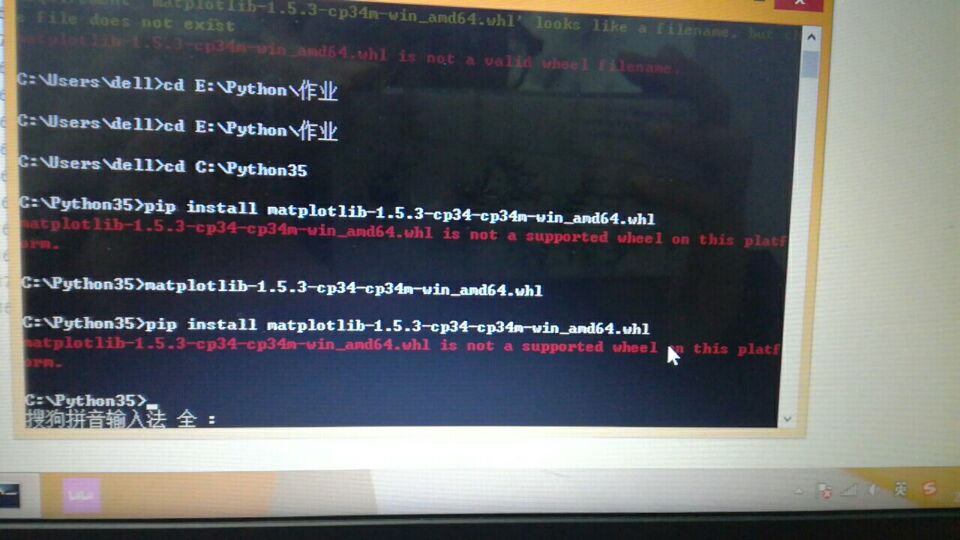
问题一：安装科学计算包的时候提示安装失败，如下图

图 57 问题一

解决方法：下载的时候只注意了与电脑操作系统的匹配，没有注意与Python的匹配。cp34中的34指的是Python3.4,而我的Python安装的是3.5的版本。

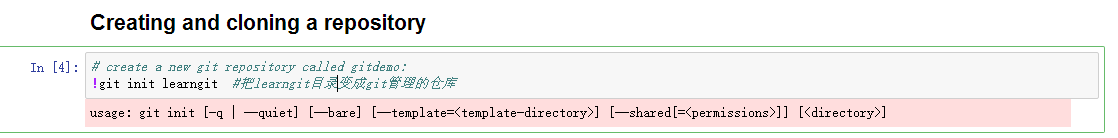
问题二：在用jupyter notebook操作git的时候，发现仓库创建不了

图 58 问题二

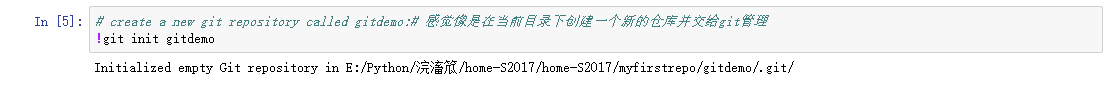
解决方法：！git inti 语句后面不能加注释，他会把注释当成一个参数

图 59 问题二的解决

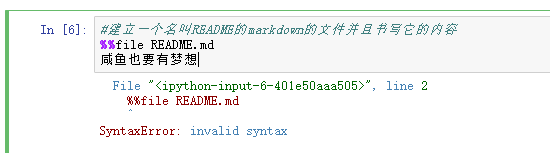
问题三：jupyter notebook中在git管理的仓库下添加文件失败

图 60 问题三

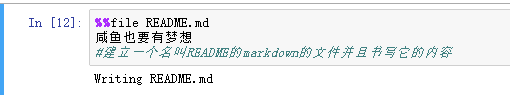
且改变注释的地方后，连同注释一并被写入文件

图 61 问题三

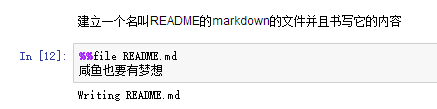
解决方法：注释不能写在%%前面，而且注释也不能写在%%file README.md 的后面，jupyter notebook 会以为这也是写入文件的内容。此时注释可以用markdown的形式另起一个cell

图 62 问题三的解决

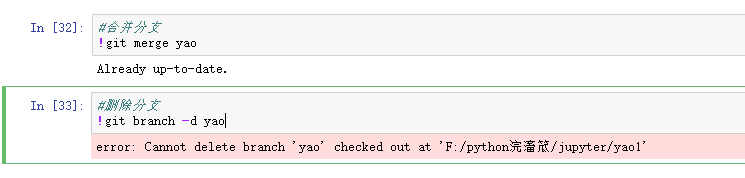
问题四：jupyter notebook不能够删除分支：

图 63 问题四

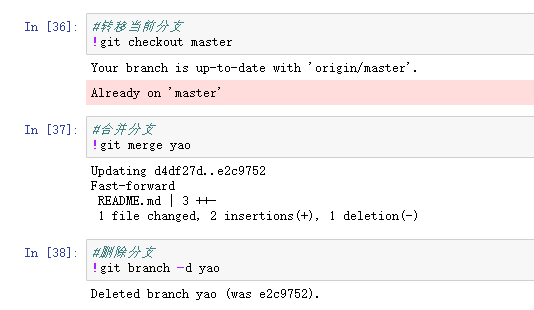
解决方法：并没有将head指针切换回来

图 64 问题四的解决

问题五：jupyter notebook中修改后的文件不能够提交

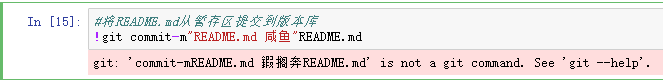
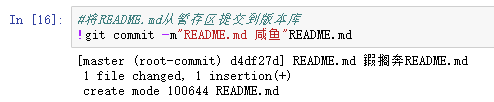
解决方法：commit和-m之间没有加空格

图 65 问题五的解决

图 66 问题五

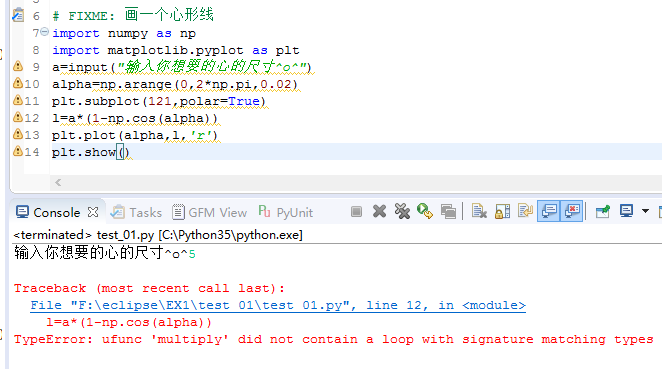
问题六：eclipse编写的程序提示语法错误

图 67 问题六

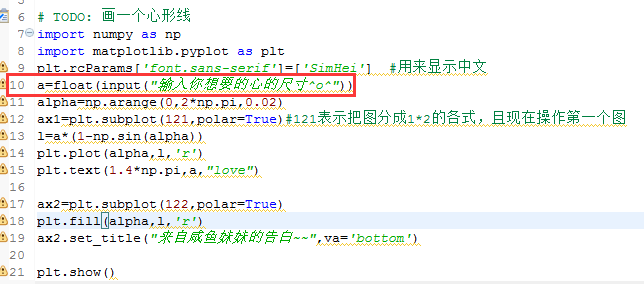
解决方法：第九行的input默认是输入的一个字符串，即a是一个字符串

图 68 问题六的解决

1. **个人小结**

这篇报告几乎花了我一个星期的时间。因为在安装eclipse软件和使用jupyter notebook操作git版本库的时候，我并没有跟上老师的节奏，导致课堂上有很大的遗失。课后专门到网上找到了一些操作步骤而且结合老师的讲义，一点一点理解和安装，但是在写报告的时候还是有很多图忘记截下来，所以可能显得有些地方不够完整。

但是在尝试使用jupyter notebook和eclipse的过程中，我还是感到很愉快。虽然现在还是不能非常熟练地使用和编写相对复杂的程序，但是我对他们感觉到非常亲切。兴许是所有计算机语言都有他们相似的地方。Jupyter notebook 的交互式页面让我想起了mathematics的笔记本形式，同样是In[]和Out[]的形式。而eclipse的环境也和VS2013很想，菜单的形式，workspace的默认位置，import的用法和#include的声明，报错的形式等等。同时调用matplotlib生成的图像的页面形式和numpy的arange的数据的生成方法又很像是MATLAB的图像和语句形式。而调用matplotlib生成的图像的语句也和假期MFC课程设计中对弹出对话框的设计很是神似（这两个可能是因为都是用句柄操作？）。总而言之，这一切的一切都莫名的熟悉，或许这就是Python语言作为高级语言的特点，可读性强且语句简单优雅明确，上手容易。

另一方面，在MFC课程设计之后，Python语言的学习再次让我体会到了互联网上材料之丰富。在自己编写程序之前，我参阅了许多git和matplotlib的教程和代码。但是事实上，看得懂和用的对还是有很大的差距。在自己手敲代码的时候，总是会出现类似空格、大小写、函数误用、字符类型和格式等大大小小的问题。

当然，环境的搭建至关重要。这应该是我第一次自己动手搭建一个合适的开发环境，大道程序语言的选择，编译器的安装，小到字符形式的设置，标签的建立，其实无论大小，在真正的编程应用中就会发现，他们都至关重要。

1. **参考文献**

[1] 刘振平 贺怀建 李强 朱发华 基于Python的三维建模可视化系统的研究 岩土力学[J] 2009.30(10):50-52

[2] 廖雪峰 廖雪峰的官方网站 [EB/OL] [2017.03.10] <http://www.liaoxuefeng.com/>

[3] xiaoiker python科学计算学习二：matplotlib绘图（极坐标 3D绘图等）（3）[EB/OL] 2017.03.10<http://blog.csdn.net/ikerpeng/article/details/20523679>

[4]Michael翔 Python--matplotlib绘图可视化知识点整理 [EB/OL] 2017.03.10

<http://python.jobbole.com/85106/>

[5]matplotlib: python ploting ——matplotlib 2.0.0 documentation [CP] 2017.03.10 <http://matplotlib.org/index.html>

[6] Joshua Reed 在Linux下使用Python的matplotlib绘制数据图的教程 [CP]

2017.03.10 <http://www.jb51.net/article/67626.htm>

[7]笛卡尔的第十三封情书 [CP] 2017.03.11 <http://www.docin.com/p-593395681.html>

[8]程懋华 Python学习和开发环境的建立[Z] 2017.03

[9]吉珣碧 软件工程基础及实践课程实践作业二[Z] 2017.03