

学校代码: 10286  
学 号: 03014404



# 东南大学

## 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
1.1 Python 语言的简介 .....	3
1.2 Python 3.5.3 的安装过程.....	4
1.3 Python 3.5.3 的程序示例.....	6
二、 扩展模块 .....	8
2.1 安装科学计算包.....	8
2.2 安装规范格式扩展包 .....	8
2.3 安装 IF97 物性计算.....	9
三、 交互计算环境和程序示例：Jupyter Notebook.....	10
3.1 Jupyter Notebook 的安装过程.....	10
3.2 Jupyter notebook 的使用示例.....	11
四、 集成开发环境和程序示例：Eclipse.....	16
4.1 Eclipse 的安装和配置.....	16
4.1.1 安装 Java SDK .....	16
4.1.2 安装 Eclipse.....	17
4.1.3 配置 Eclipse 工作空间 .....	17
4.1.4 安装和配置插件 PyDev.....	18
4.1.5 基于 PyDev 的语言规范静态检查 .....	21
4.1.6 Eclipse 配置和使用 .....	22
4.1.7 Markdown 插件 .....	24
4.2 Eclipse 使用示例.....	25

<b>五、 遇到的问题及解决方法 .....</b>	<b>27</b>
<b>六、 个人小结 .....</b>	<b>30</b>
<b>七、 参考文献 .....</b>	<b>31</b>

## 一、Python 基本开发环境和程序示例：IDLE

### 1.1 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<sup>[1]</sup>

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。<sup>[2]</sup>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。

1.2 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：

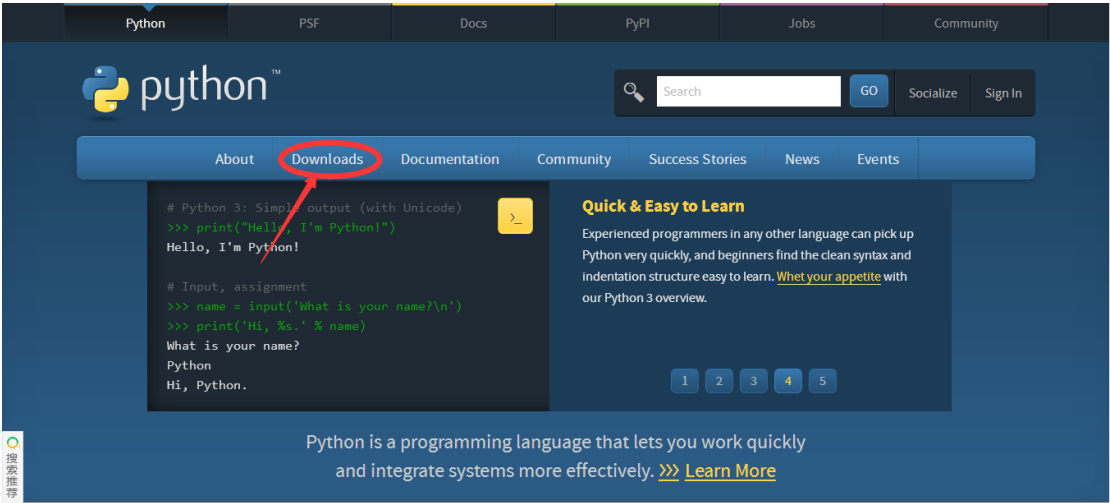


图 1 官网页面

我个人选择的是 Python 3.5.2 版本：

Release version	Release date	Click for more	
<a href="#">Python 2.7.13</a>	2016-12-17	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.4.5</a>	2016-06-27	<a href="#">Download</a>	<a href="#">Release Notes</a>
<b><a href="#">Python 3.5.2</a></b>	2016-06-27	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 2.7.12</a>	2016-06-25	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.4.4</a>	2015-12-21	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.5.1</a>	2015-12-07	<a href="#">Download</a>	<a href="#">Release Notes</a>

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：

Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		3fe8434643a78630c61c6464fe2e7e72	20566643	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		8906efbacfdc7c3c9198aeefafd159e	15222676	<a href="#">SIG</a>
<a href="#">Mac OS X 32-bit/i386/PPC installer</a>	Mac OS X	for Mac OS X 10.5 and later	5ae81eea42bb6758b6d775ebcaf32eda	26250336	<a href="#">SIG</a>
<a href="#">Mac OS X 64-bit/32-bit installer</a>	Mac OS X	for Mac OS X 10.6 and later	11a9f4fc3f6b93e3ffb26c383822a272	24566858	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		24b95be314f7bad1cc5361ae449adc3d	7777812	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64, not Itanium processors	f1c24bb78bd6dd792a73d5ebfdd3b20e	6862200	<a href="#">SIG</a>
<b><a href="#">Windows x86-64 executable installer</a></b>	Windows	for AMD64/EM64T/x64, not Itanium processors	4da6dbc8e43e2249a0892d257e977291	30177896	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64, not Itanium processors	c35b6526761a9cde4b6dccb4a3d7c60	970224	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		ad637a1db7cf91e344318d55c94ad3ca	6048722	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		2ddf428fd8b9c063ba05b5a0c8636c37	29269656	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		aed3ac79b8e2458b84135ecfca66764	944304	<a href="#">SIG</a>

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）。

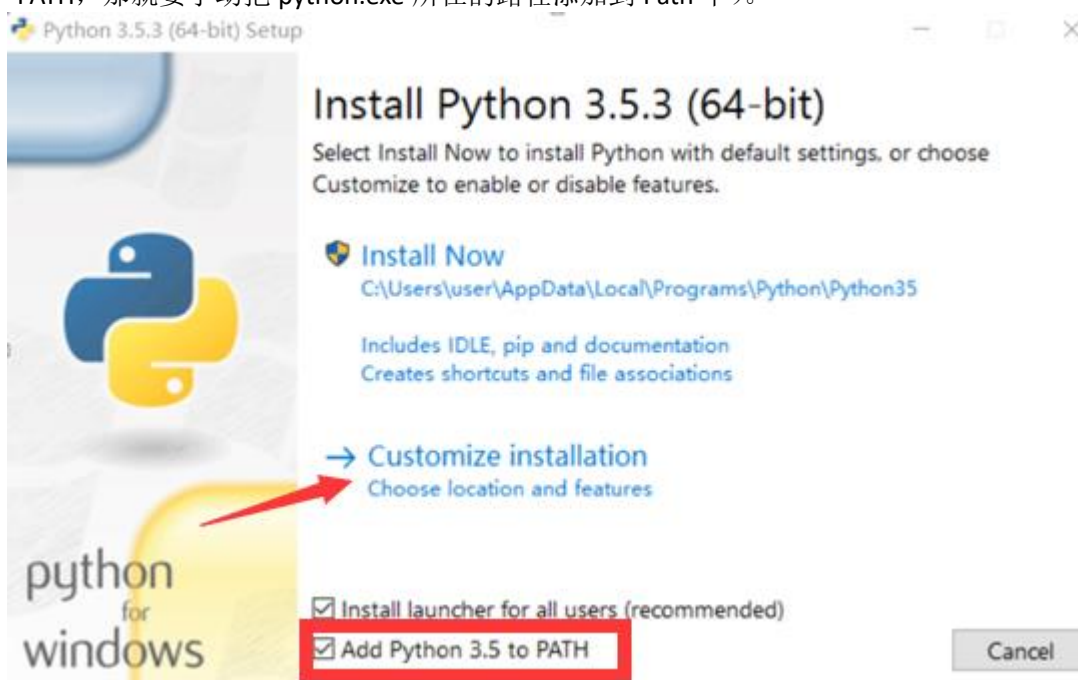


图 4 Python 安装页面

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。

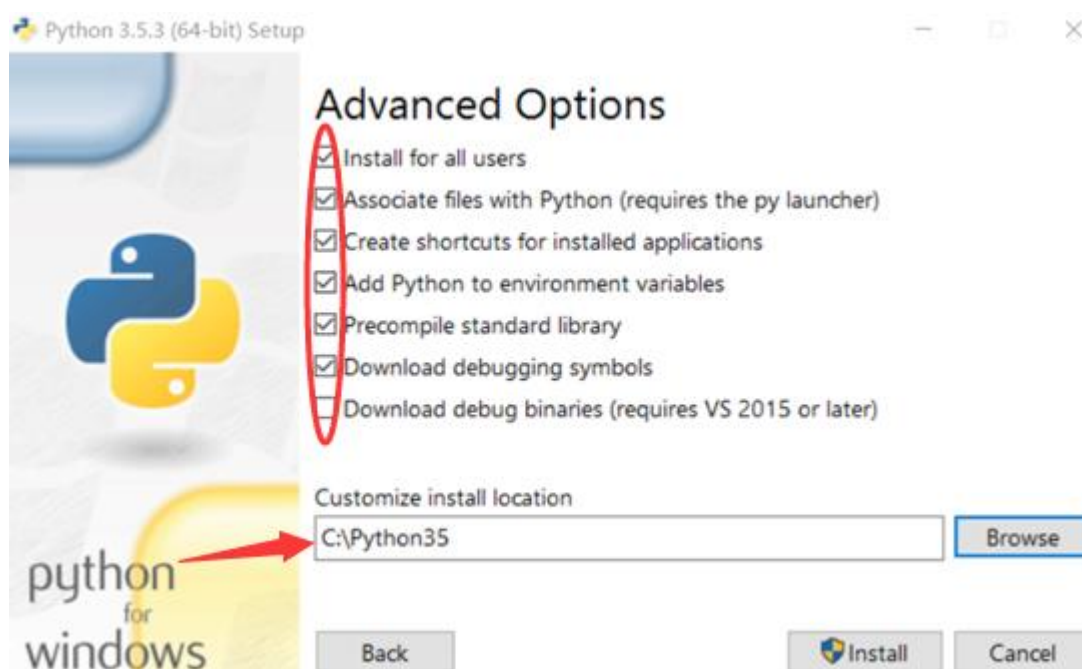


图 5 安装选项以及路径选择

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：



```
命令提示符
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

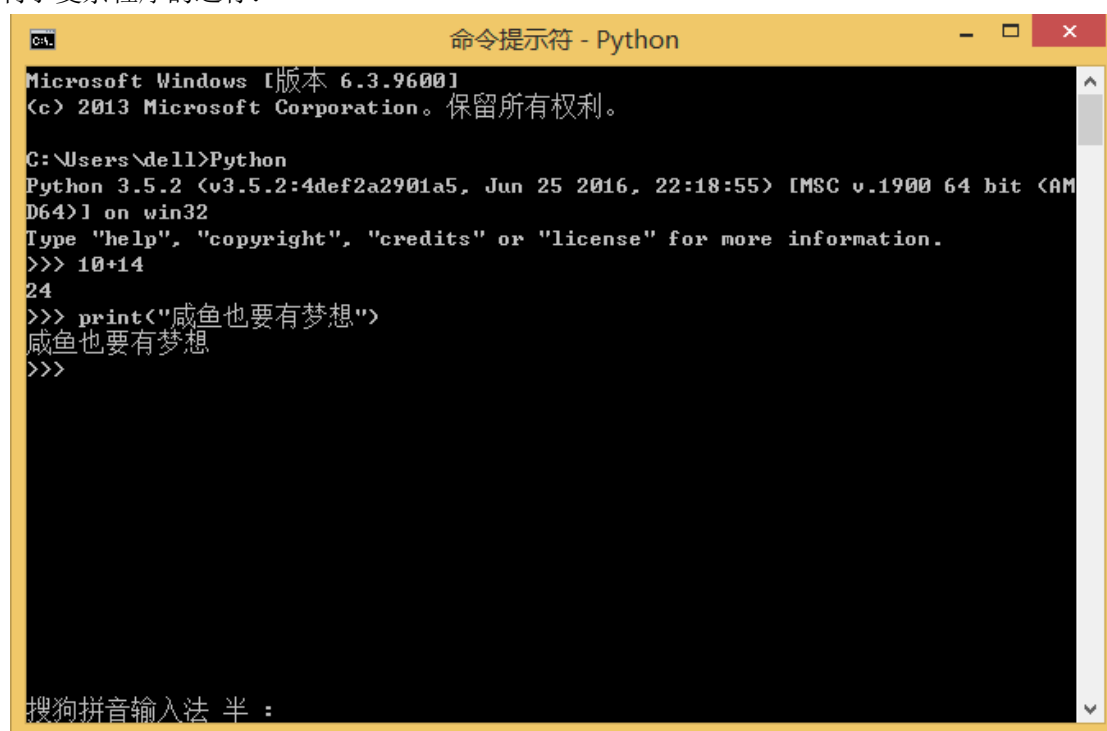
C:\Users\dell>python -m pip install -U pip
Requirement already up-to-date: pip in c:\python35\lib\site-packages

C:\Users\dell>
```

图 6 安装检查

### 1.3 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：

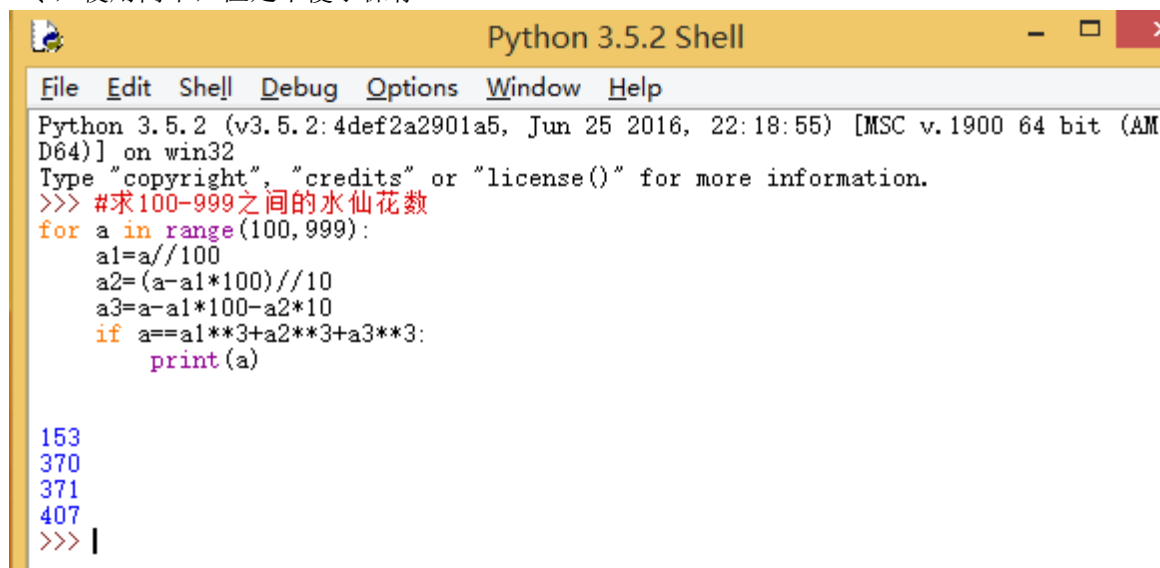


```
命令提示符 - Python
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\dell>Python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 10+14
24
>>> print("咸鱼也要有梦想")
咸鱼也要有梦想
>>>
```

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。

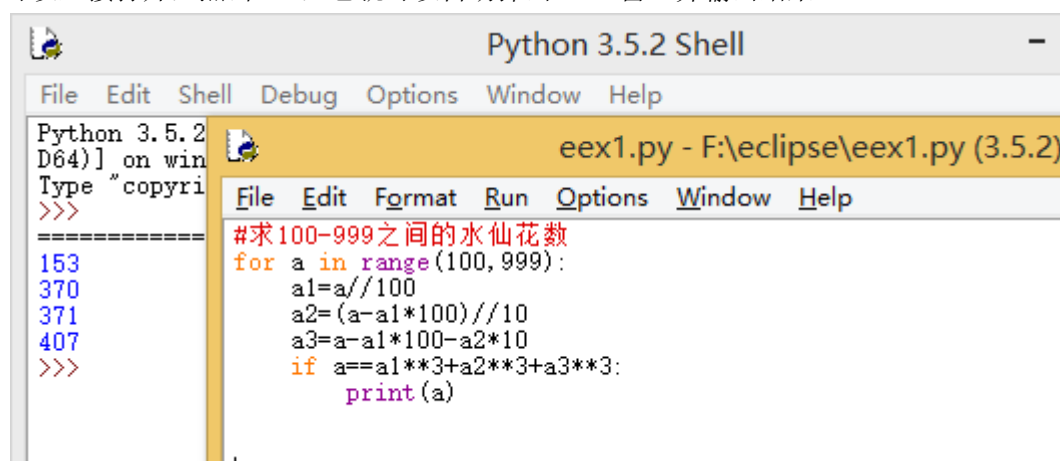


```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> #求100-999之间的水仙花数
for a in range(100,999):
    a1=a//100
    a2=(a-a1*100)//10
    a3=a-a1*100-a2*10
    if a==a1**3+a2**3+a3**3:
        print(a)

153
370
371
407
>>> |
```

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：



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=====
153
370
371
407
>>>

eex1.py - F:\eclipse\eex1.py (3.5.2)
File Edit Format Run Options Window Help
#求100-999之间的水仙花数
for a in range(100,999):
    a1=a//100
    a2=(a-a1*100)//10
    a3=a-a1*100-a2*10
    if a==a1**3+a2**3+a3**3:
        print(a)
```

图 9 利用文件保存 Python 程序并运行示例



## 二、扩展模块

### 2.1 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：

```
NumPy, a fundamental package needed for scientific computing with Python.
NumPy+MKL is linked to the Intel® Math Kernel Library and includes required DLLs in the numpy.core directory.
numpy-1.11.3+mkl-cp27-cp27m-win32.whl
numpy-1.11.3+mkl-cp27-cp27m-win_amd64.whl
numpy-1.11.3+mkl-cp34-cp34m-win32.whl
numpy-1.11.3+mkl-cp34-cp34m-win_amd64.whl
numpy-1.11.3+mkl-cp35-cp35m-win32.whl
numpy-1.11.3+mkl-cp35-cp35m-win_amd64.whl
numpy-1.11.3+mkl-cp36-cp36m-win32.whl
numpy-1.11.3+mkl-cp36-cp36m-win_amd64.whl
numpy-1.12.1rc1+mkl-cp27-cp27m-win32.whl
numpy-1.12.1rc1+mkl-cp27-cp27m-win_amd64.whl
numpy-1.12.1rc1+mkl-cp34-cp34m-win32.whl
numpy-1.12.1rc1+mkl-cp34-cp34m-win_amd64.whl
numpy-1.12.1rc1+mkl-cp35-cp35m-win32.whl
numpy-1.12.1rc1+mkl-cp35-cp35m-win_amd64.whl
numpy-1.12.1rc1+mkl-cp36-cp36m-win32.whl
numpy-1.12.1rc1+mkl-cp36-cp36m-win_amd64.whl
```

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。

### 2.2 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

2.3 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。

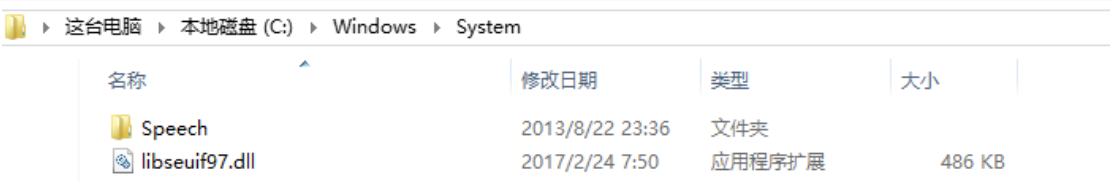


图 14 seuif97.dll 拷贝到 c:\windows\system

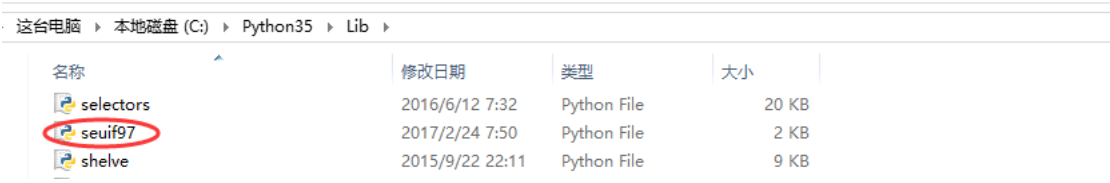


图 13 seuif97.py 拷贝到 c:\python35\Lib

Python 语言计算包，在线安装：>pip install iapws

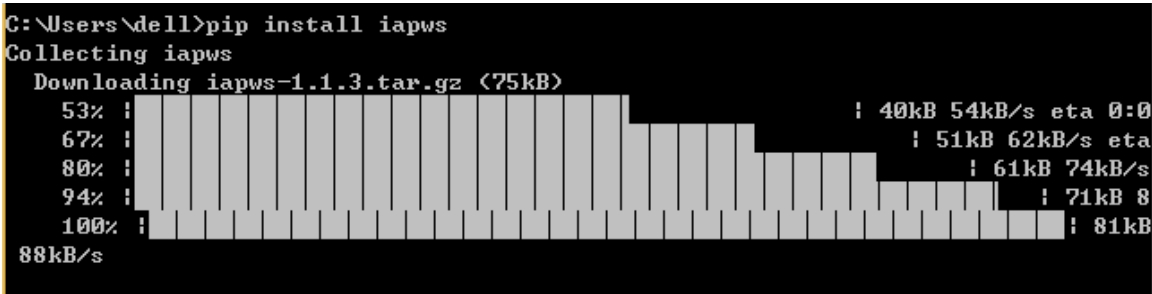


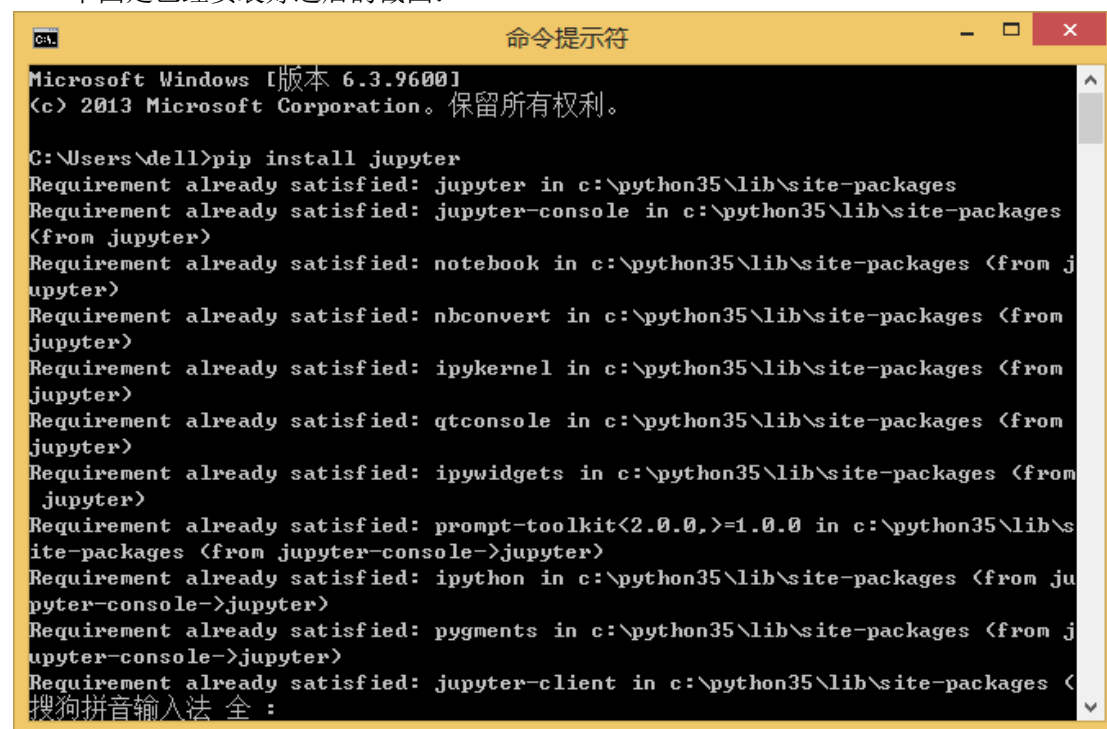
图 15 在线安装 iapws

## 三、交互计算环境和程序示例：Jupyter Notebook

### 3.1 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：



```
命令提示符
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\dell>pip install jupyter
Requirement already satisfied: jupyter in c:\python35\lib\site-packages
Requirement already satisfied: jupyter-console in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: notebook in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: nbconvert in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: ipykernel in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: qtconsole in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: ipywidgets in c:\python35\lib\site-packages (from jupyter)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.0 in c:\python35\lib\site-packages (from jupyter-console->jupyter)
Requirement already satisfied: ipython in c:\python35\lib\site-packages (from jupyter-console->jupyter)
Requirement already satisfied: pygments in c:\python35\lib\site-packages (from jupyter-console->jupyter)
Requirement already satisfied: jupyter-client in c:\python35\lib\site-packages (from jupyter-console->jupyter)
搜狗拼音输入法 全：
```

图 17 jupyter notebook 的安装

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。

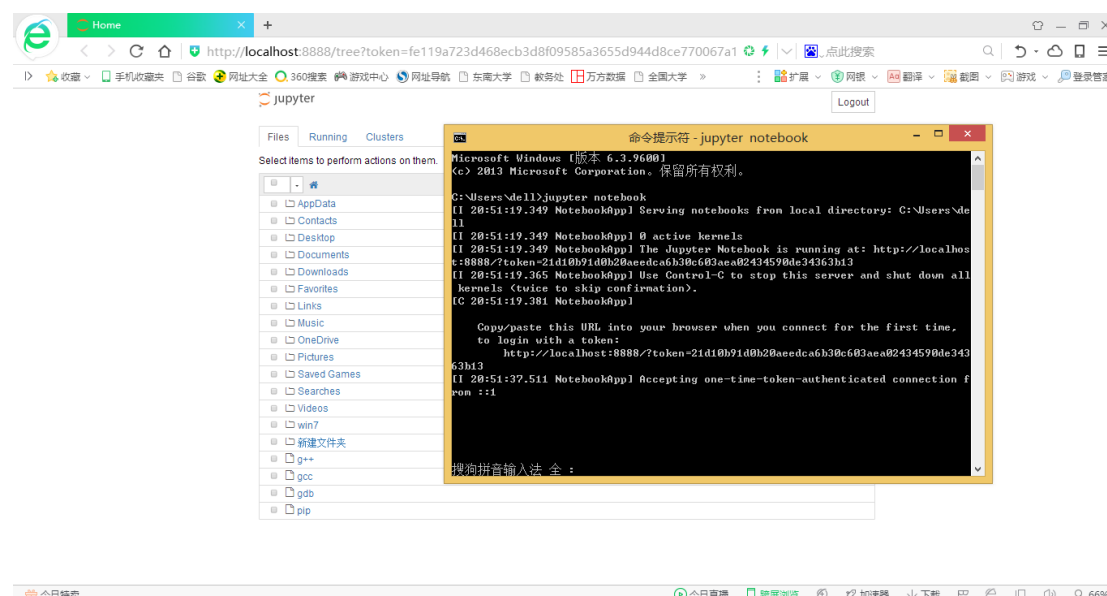


图 16 用命令提示符打开 jupyter notebook

### 3.2 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：

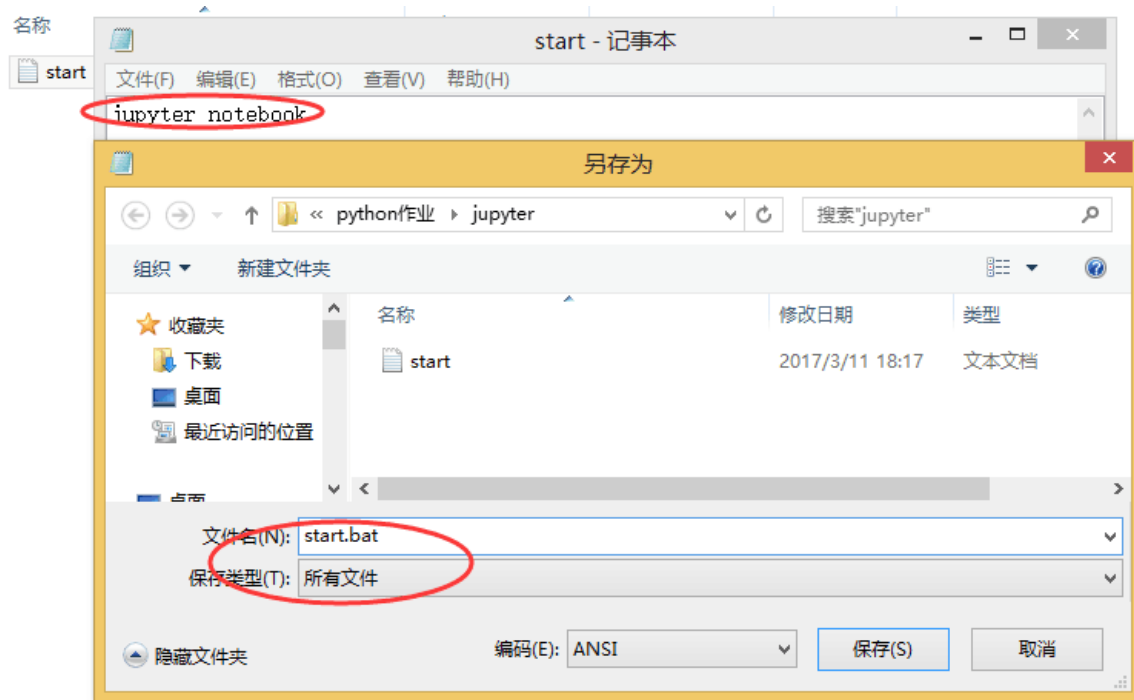


图 19 将 txt 另存为 bat

start	2017/3/11 18:18	Windows 批处理...	1 KB
start	2017/3/11 18:17	文本文档	1 KB

图 18 生成 bat 文件

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:

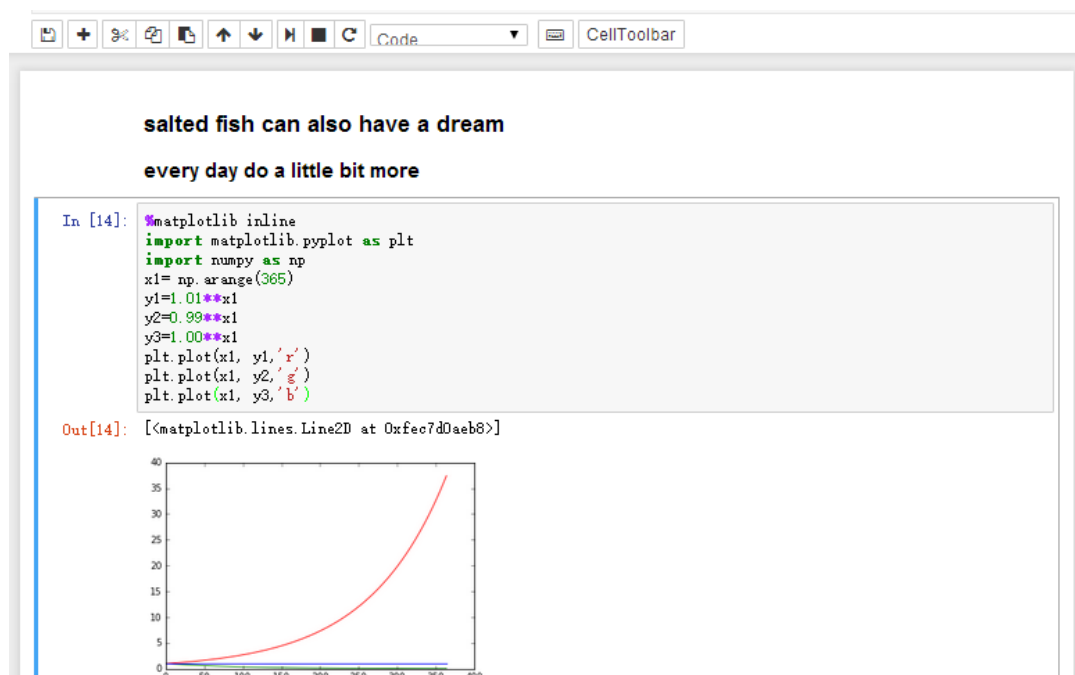


图 20 jupyter notebook 的使用示例<sup>[3]</sup>

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>` using namespace 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:

```
[I 18:23:42.815 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 18:23:42.824 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8889/?token=18ca89be810cc07206e62403a159f6e356b10aa69ec
7f2fb
[I 18:23:43.846 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
[I 18:25:01.143 NotebookApp] Creating new notebook in
[W 18:25:04.499 NotebookApp] 404 GET /nbextensions/widgets/notebook/js/extension
.js?v=20170311182340 (::1) 49.03ms referer=http://localhost:8889/notebooks/Untit
led.ipynb?kernel_name=python3
[I 18:25:05.350 NotebookApp] Kernel started: 387076b4-cda5-4ad4-a715-45c06a8920b
3
[I 18:27:04.876 NotebookApp] Saving file at /Untitled.ipynb
[I 18:29:04.838 NotebookApp] Saving file at /Untitled.ipynb
[I 18:39:04.861 NotebookApp] Saving file at /My first jupyter notebook.ipynb
[I 18:41:04.828 NotebookApp] Saving file at /My first jupyter notebook.ipynb
[I 18:54:55.209 NotebookApp] Saving file at /My first jupyter notebook.ipynb
[I 18:56:54.267 NotebookApp] Saving file at /My first jupyter notebook.ipynb
[I 18:58:54.296 NotebookApp] Saving file at /My first jupyter notebook.ipynb
```

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

```

In [1]: #查看当前路径
%pwd

Out[1]: 'F:\python作业\jupyter'

In [2]: #创建一个名字是yao1的本地仓库
%mkdir yao1

In [4]: #转到这个本地仓库目录下
%cd yao1

F:\python作业\jupyter\yao1

In [5]: #将这个仓库托付给git来管理
!git init

Initialized empty Git repository in F:/python完满版/jupyter/yao1/.git/

建立一个名叫README的markdown的文件并且书写它的内容

In [12]: %%file README.md
咸鱼也要有梦想

Writing README.md

In [13]: #将README.md从工作区提交到暂存区
!git add README.md

In [16]: #将README.md从暂存区提交到版本库
!git commit -m"README.md 咸鱼" README.md

[master (root-commit) d4df27d] README.md 猥亵README.md
1 file changed, 1 insertion(+)
create mode 100644 README.md

```

图 22 用 jupyter notebook 来创建仓库[2]

理, 然后还可以自己写一个 README.MD 文件添加到版本库:

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:

第二步是在 GitHub 的相应位置输入本机的 ssh 密钥然后创建远程仓库:



```

MINGW64:/c/Users/dell
dell@dell MINGW64 ~
$ ssh-keygen -t rsa -C "1278507172@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/dell/.ssh/id_rsa):
Created directory '/c/Users/dell/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/dell/.ssh/id_rsa.
Your public key has been saved in /c/Users/dell/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:u8G1GBnFr7uaPhhvweFafgaINODRuain4sd+x8D5MJw 1278507172@qq.com
The key's randomart image is:
+---[RSA 2048]-----+
  ..O.. ..
  oEo.. ..
  .+.. ..
  o.+.. .o.
  .oo+S...
  . .oo. +=..
  o. . B=o..
  o o.o ==+
  o.o. o=..
+---[SHA256]-----+

```

图 23 git bash 创建.ssh



```

MINGW64:/c/Users/dell
dell@dell MINGW64 ~
$ clip < ~/.ssh/id_rsa.pub

dell@dell MINGW64 ~
$ ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC3VsE5C2s+kTczDCZgVDzJICqMPpcNZZ+rEYLtcQ
RZE1LC83FPGHoKAsjj+9KaRdCsa1aZAPwo5KUq577oh5HjesOvkL8JTVP/FB83e17CaeIL65i6ypjSr
dYMyNgcNQciCV7UwVvyaS3L71zFmigm0xe2BNLLvI81iJ473ezt2Md2301vBfOG/dcl18VA7cAwGKo
HdTiIwzJfVZhCwbt1XC08p6bus/y9IJIITQ0yOec6ZS7idNi1w3u+qzbtCMpc5LewDjQX+Ka+nDYR1dI
uYr8eLFPxrtI/YeByKM+AL0wF+ICHcN4fQkfRCdGwFQ6BAUvh2uFwd3kyo3SdR 1278507172@qq.com
bash: ssh-rsa: command not found

dell@dell MINGW64 ~
$ |

```

图 24 git bash 拷贝 rsa.pub

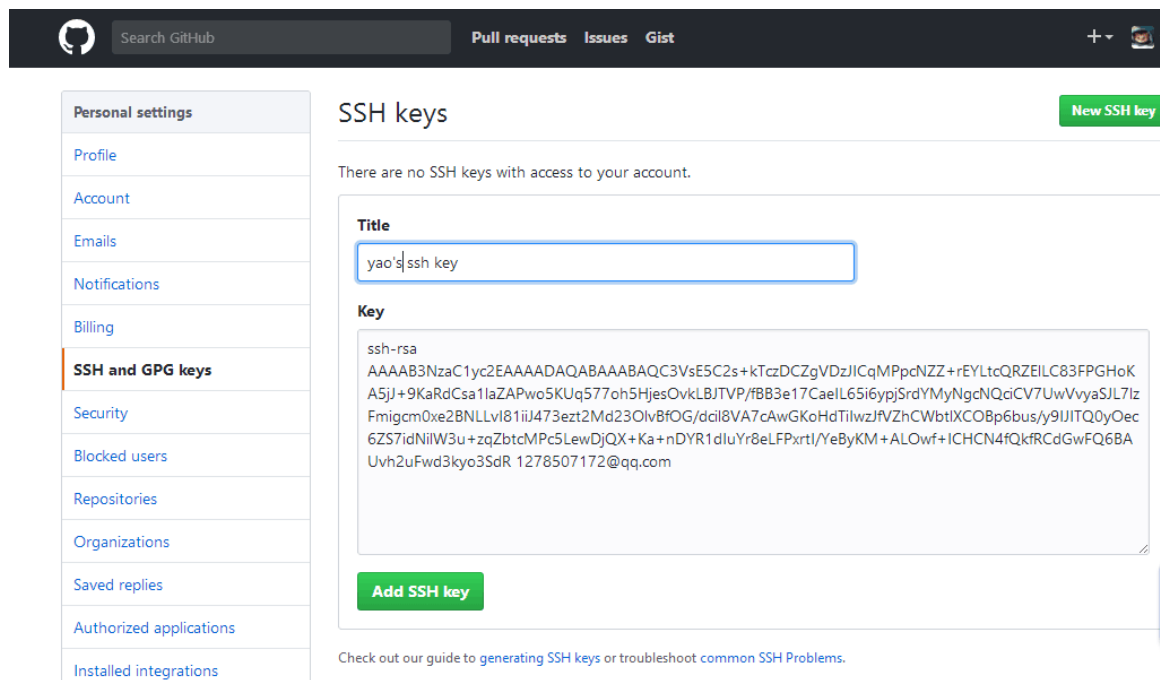


图 25 在 GitHub 上添加秘钥

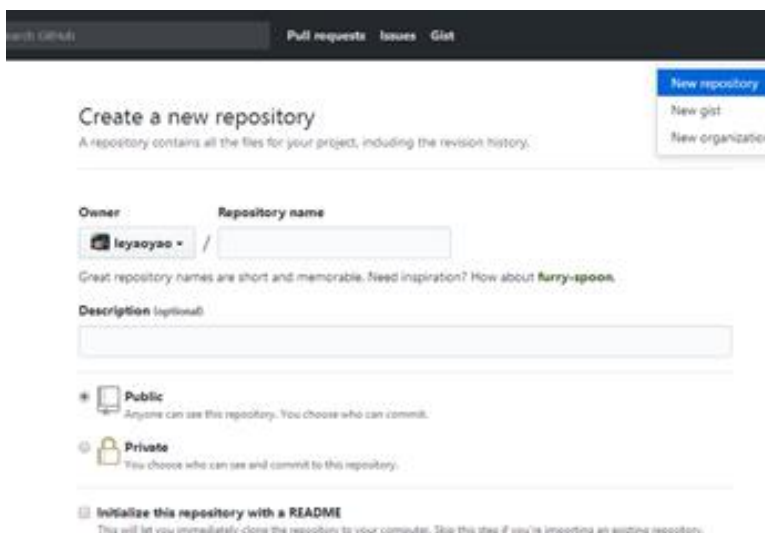


图 27 在 GitHub 上建立远程仓库

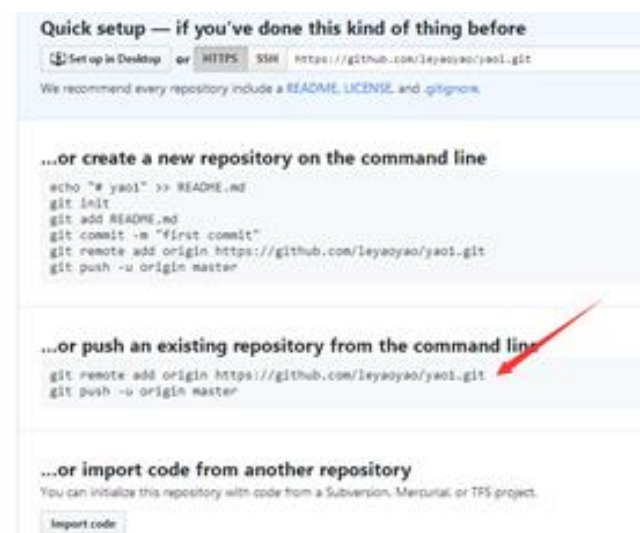


图 26 在 GitHub 上找到建立本地与远程仓库的联系代码

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：

```
In [17]: #从github上面克隆的语句，将这个本地仓库与GitHub上的远程仓库关联
!git remote add origin https://github.com/leyaoyao/yao1.git
!git push -u origin master

Branch master set up to track remote branch master from origin.

To https://github.com/leyaoyao/yao1.git
* [new branch]      master -> master

In [18]: #创建第一个标签
!git tag -a v0.0 -m "第一个标签"

In [19]: #查看标签
!git tag

v0.0

In [20]: #创建一个本地的分支yao并转到该分支
!git branch yao
!git checkout yao

Switched to branch 'yao'

In [21]: #查看分支
!git branch

master
* yao
```

图 28 完成与远程仓库关联并新建标签

在yao分支上改写README.md并提交

```
In [29]: %%file README.md
咸鱼也要有梦想
说不定哪天就实现了呢

Overwriting README.md

In [30]: !git add README.md

In [31]: !git commit -m "第一次修改"

[yao e2c9752] 绉◆涓◆娆◆ 慨◆鎭◆
1 file changed, 2 insertions(+), 1 deletion(-)

In [36]: #转移当前分支
!git checkout master

Your branch is up-to-date with 'origin/master'.
Already on 'master'

In [37]: #合并分支
!git merge yao

Updating d4df27d..e2c9752
Fast-forward
 README.md | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

In [38]: #删除分支
!git branch -d yao
```

图 29 建立分支

```
In [37]: #合并分支
!git merge yao

Updating d4df27d..e2c9752
Fast-forward
 README.md | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

In [38]: #删除分支
!git branch -d yao

Deleted branch yao (was e2c9752).

In [39]: !git branch

* master

In [40]: !git status

On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working tree clean

In [41]: #将master上的修改推送到GitHub上
!git push origin master

To https://github.com/leyaoyao/yao1.git
d4df27d..e2c9752 master -> master
```

图 30 合并分支，删除分支并推送至 GitHub



## 四、集成开发环境和程序示例：Eclipse

### 4.1 Eclipse 的安装和配置

#### 4.1.1 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：

Java SE Development Kit 8u121		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.86 MB	<a href="#">jdk-8u121-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.83 MB	<a href="#">jdk-8u121-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	162.41 MB	<a href="#">jdk-8u121-linux-i586.rpm</a>
Linux x86	177.13 MB	<a href="#">jdk-8u121-linux-i586.tar.gz</a>
Linux x64	159.96 MB	<a href="#">jdk-8u121-linux-x64.rpm</a>
Linux x64	174.76 MB	<a href="#">jdk-8u121-linux-x64.tar.gz</a>
Mac OS X	223.21 MB	<a href="#">jdk-8u121-macosx-x64.dmg</a>
Solaris SPARC 64-bit	139.64 MB	<a href="#">jdk-8u121-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	99.07 MB	<a href="#">jdk-8u121-solaris-sparcv9.tar.gz</a>
Solaris x64	140.42 MB	<a href="#">jdk-8u121-solaris-x64.tar.Z</a>
Solaris x64	96.9 MB	<a href="#">jdk-8u121-solaris-x64.tar.gz</a>
Windows x86	189.36 MB	<a href="#">jdk-8u121-windows-i586.exe</a>
Windows x64	195.51 MB	<a href="#">jdk-8u121-windows-x64.exe</a>

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

### 4.1.2 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/>,

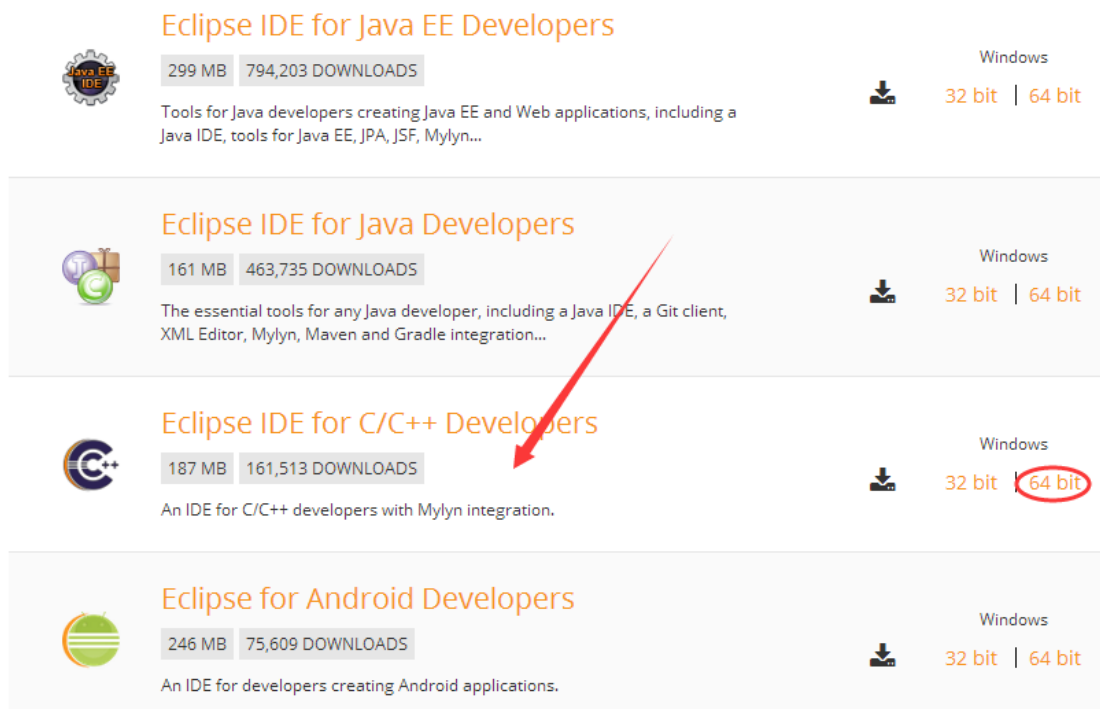


图 33 eclipse 的安装选择

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。

### 4.1.3 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。

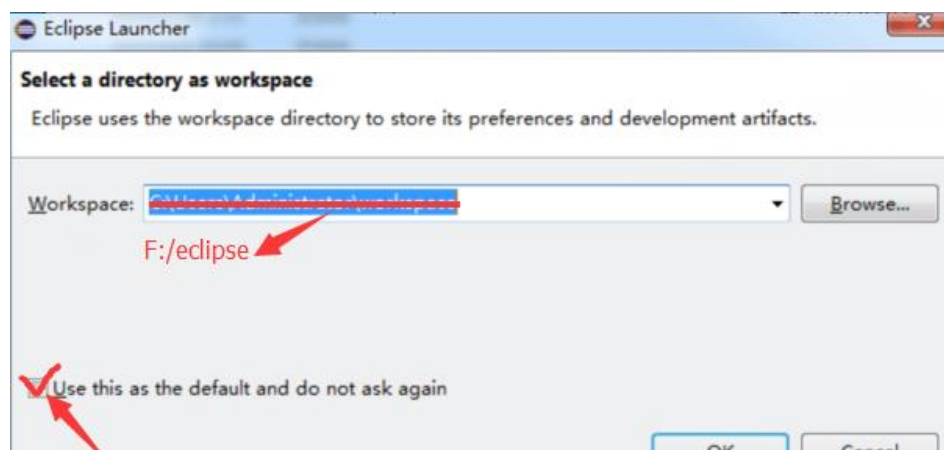


图 34 eclipse 的工作空间配置

#### 4.1.4 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。

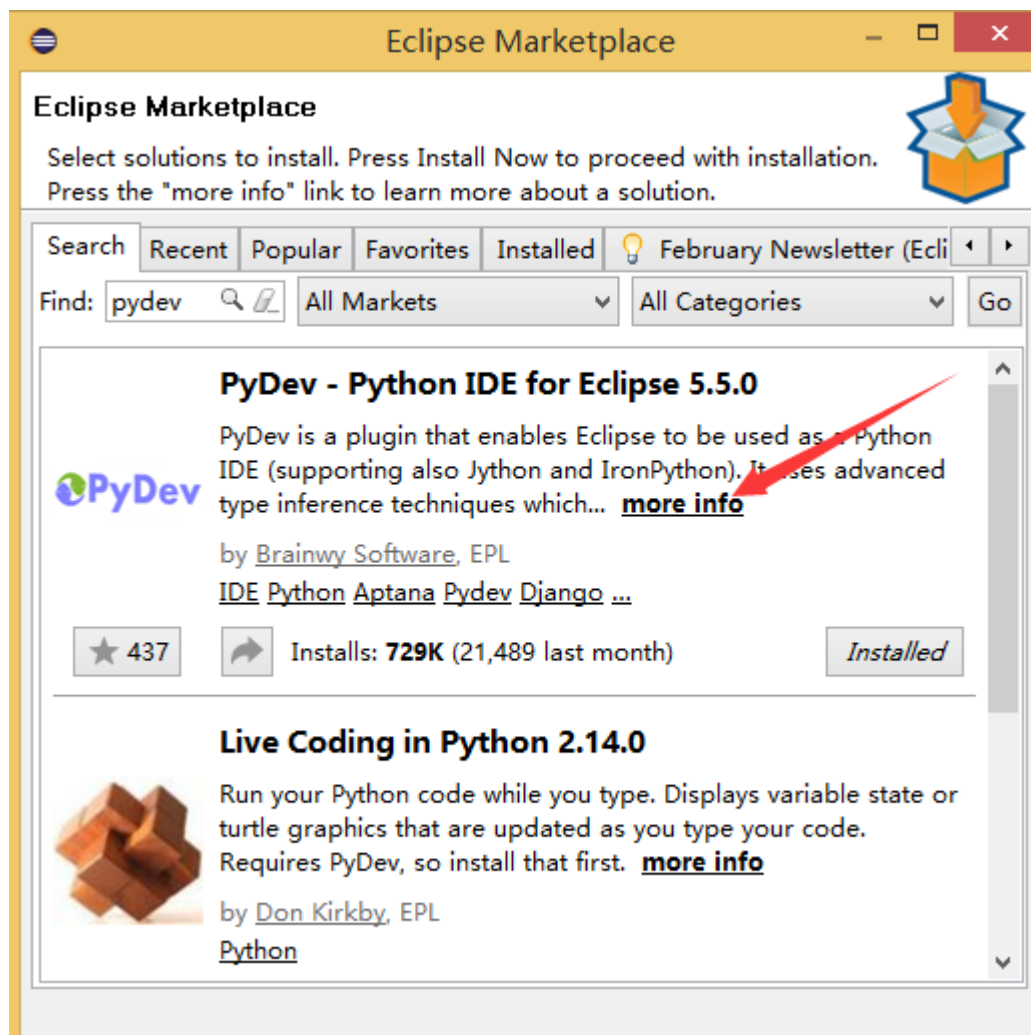


图 35 安装 PyDe 插件

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

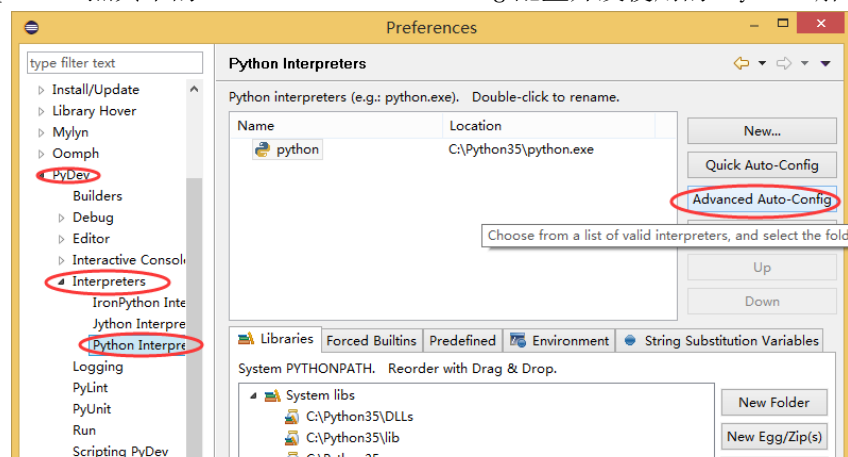


图 36 配置 Python 解释器版本

配置好后，添加 Python 场景到工作页面，如图 38：

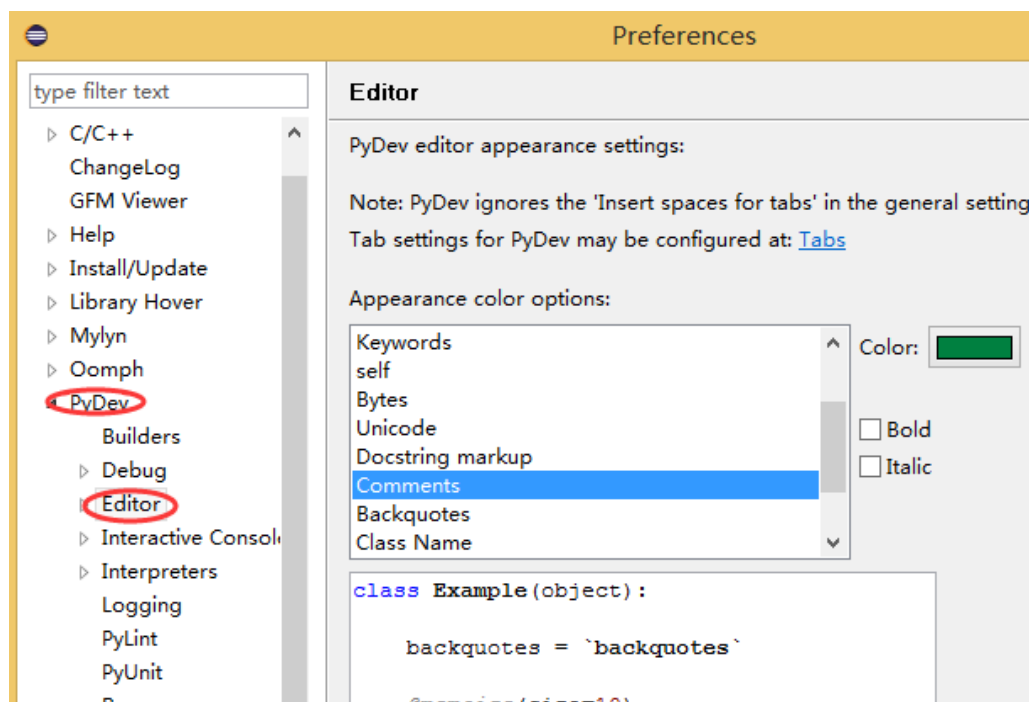
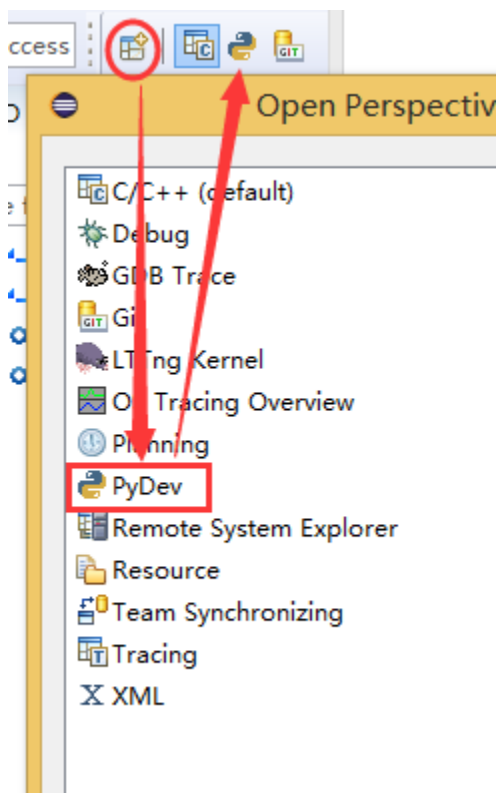


图 38 添加 Python 场景到工作页面

图 37 修改编辑器的配色

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：

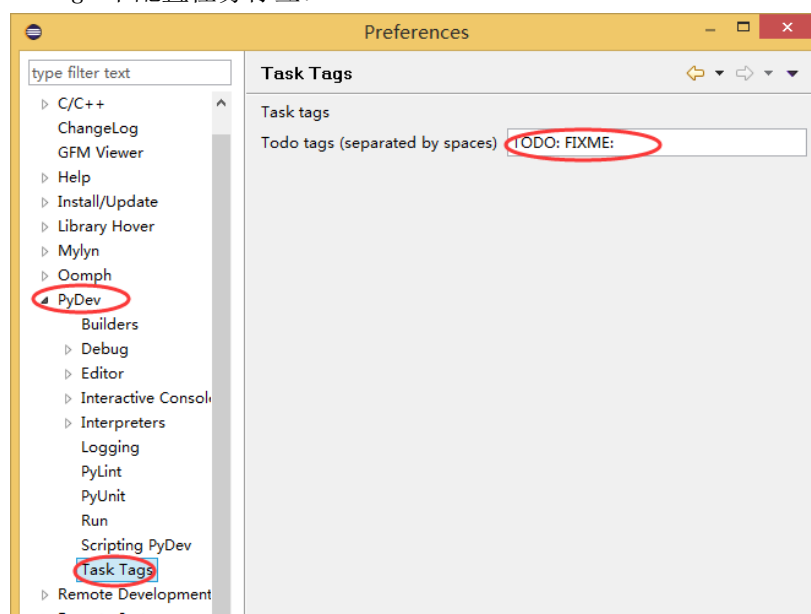


图 39 配置任务标签

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”），故而它可以被检索到：

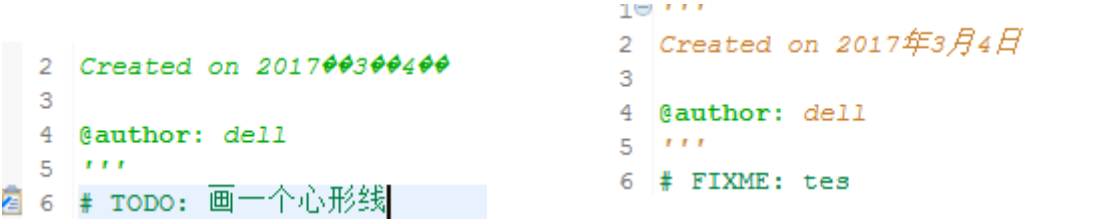


图 41 添加任务标签

图 40 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。

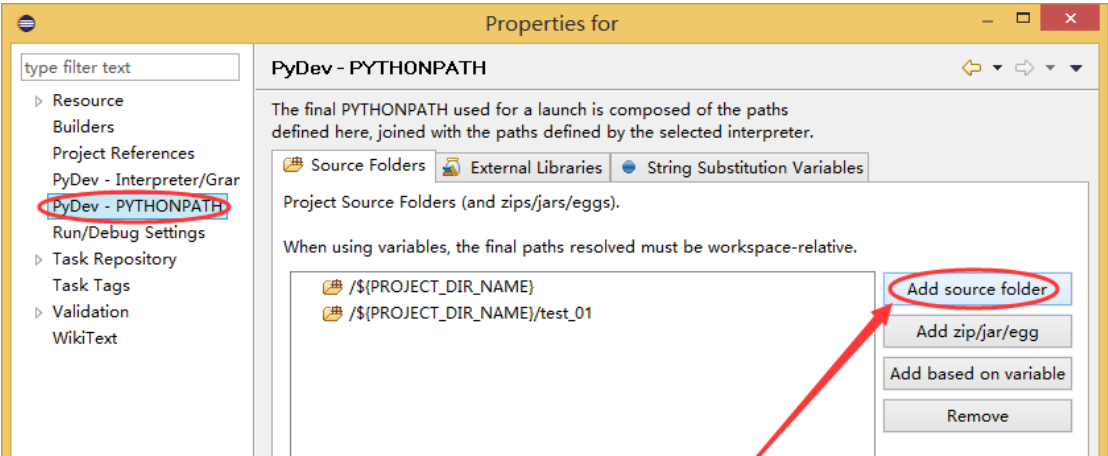


图 42 自动识别

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。

✓	!	Description	Resource	Path	Location	Type
		FIXME: tes	test_01.py	/EX2/test_01	line 5	Task
		TODO: 画一个心形线	test_01.py	/EX1/test_01	line 5	Task

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。

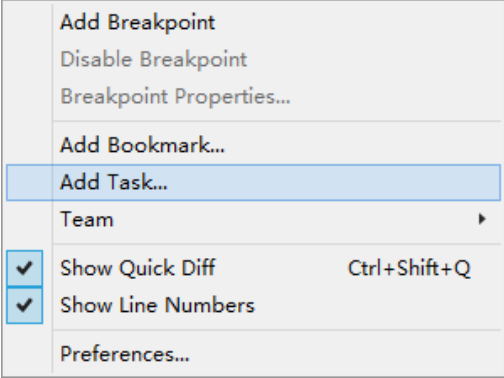


图 44 手动加入任务标签

### 4.1.5 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：

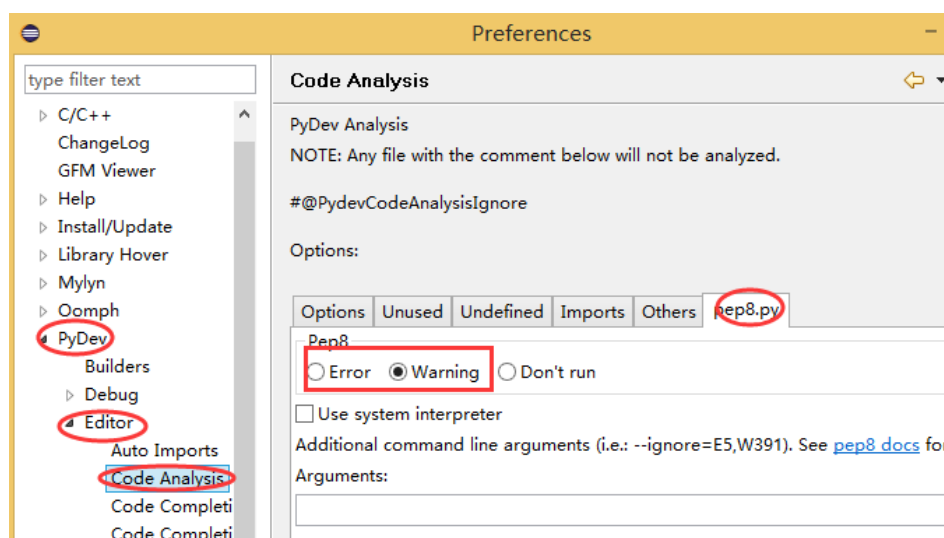


图 45 启动 pep8 检查

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?

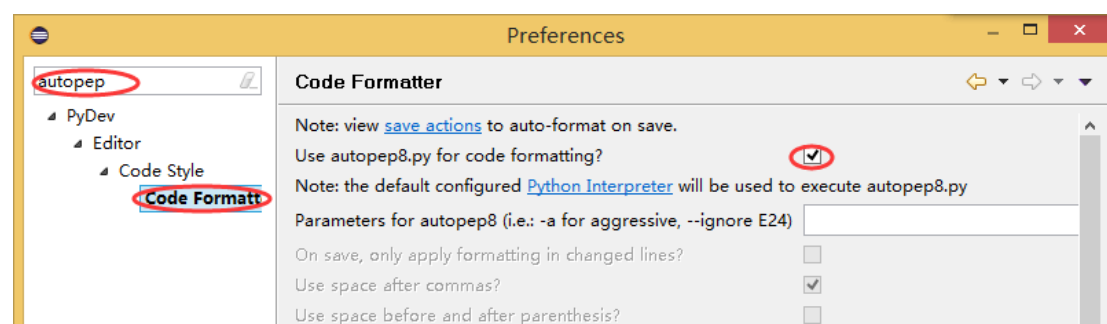


图 46 启动 autopep8 自动修改

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”

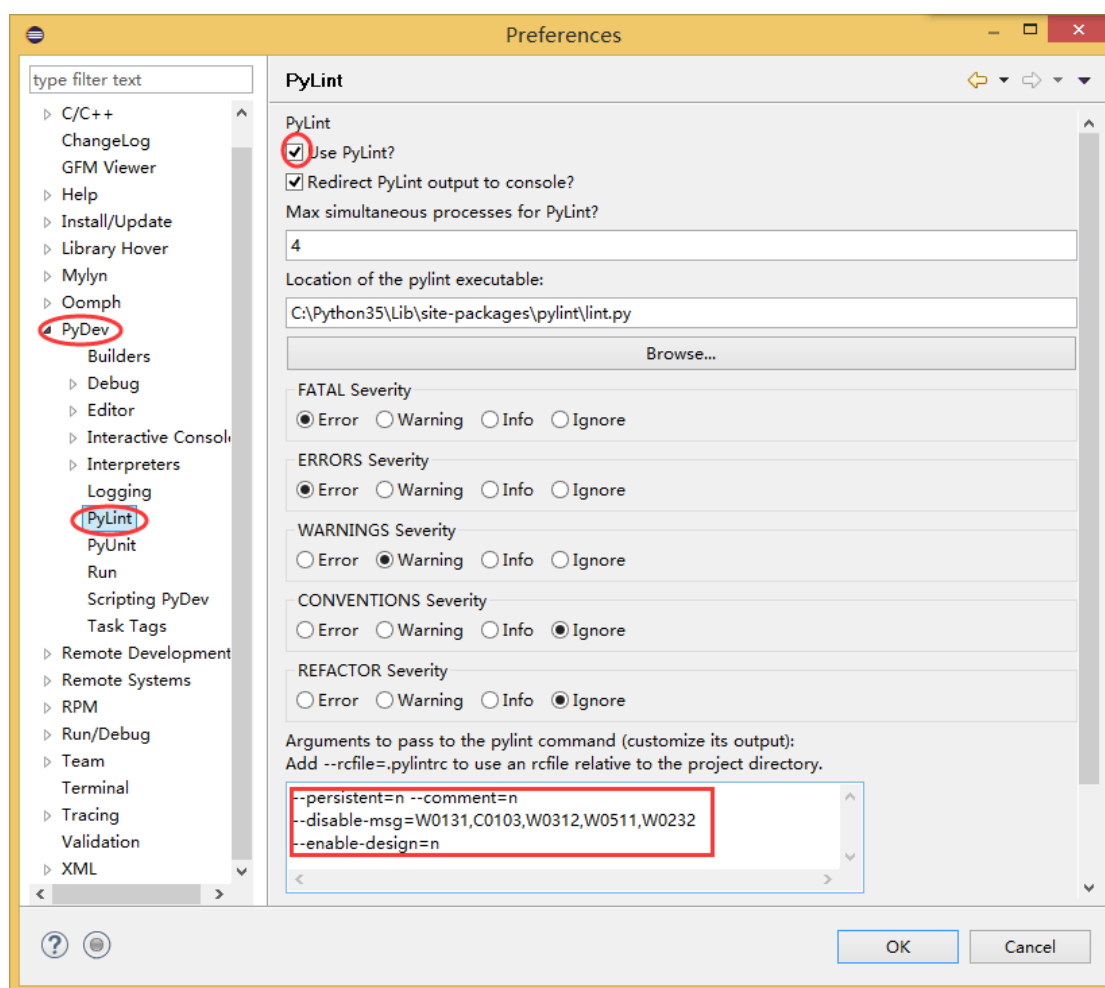


图 47 开启 Pylint

#### 4.1.6 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：

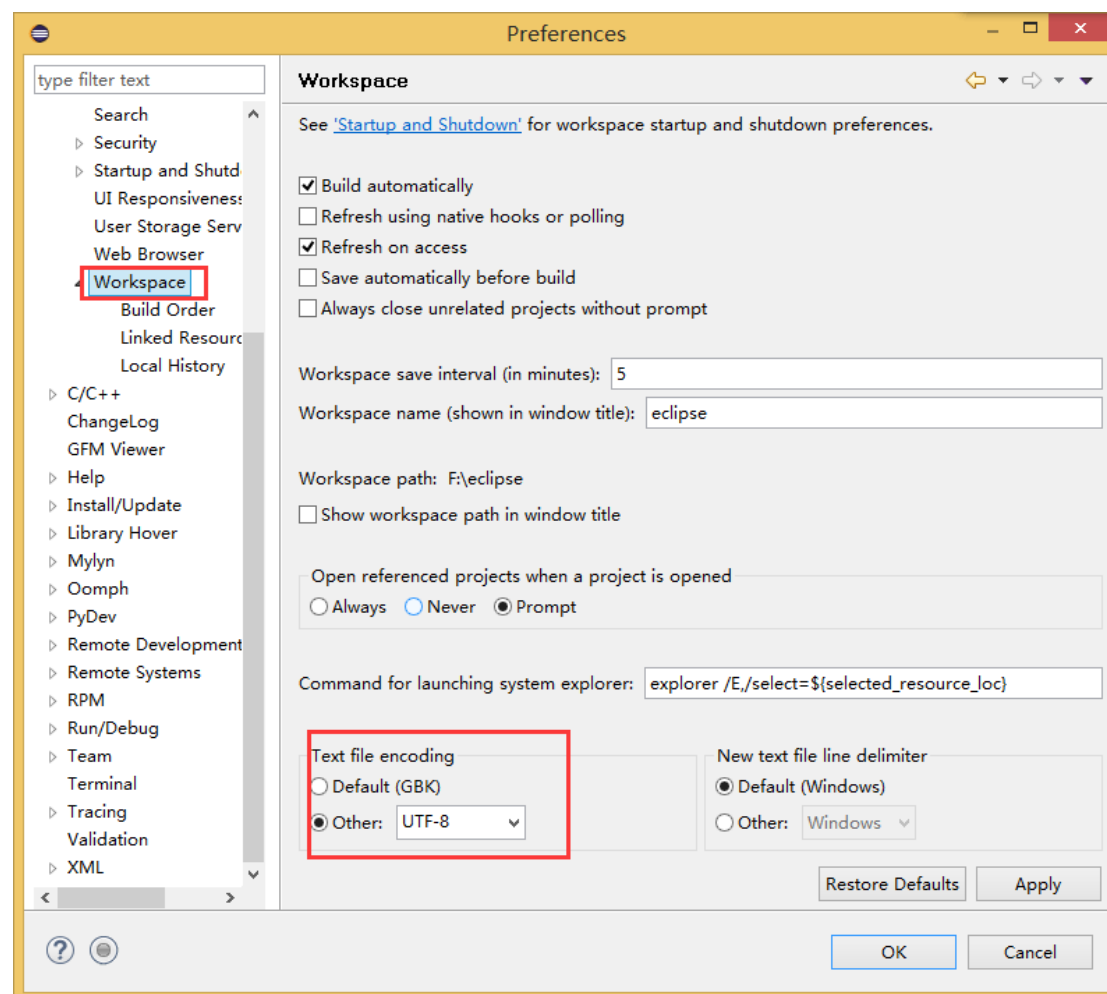


图 49 工作空间编码方式

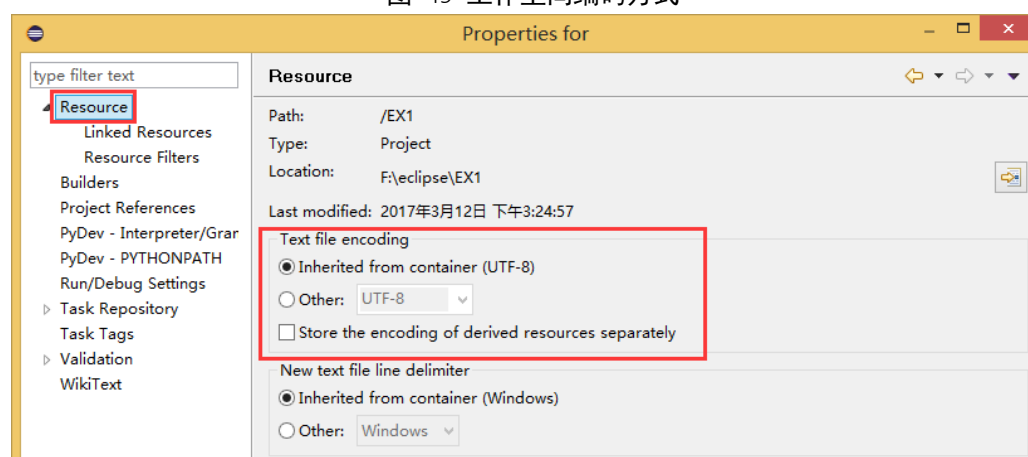


图 48 工程编码方式

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：

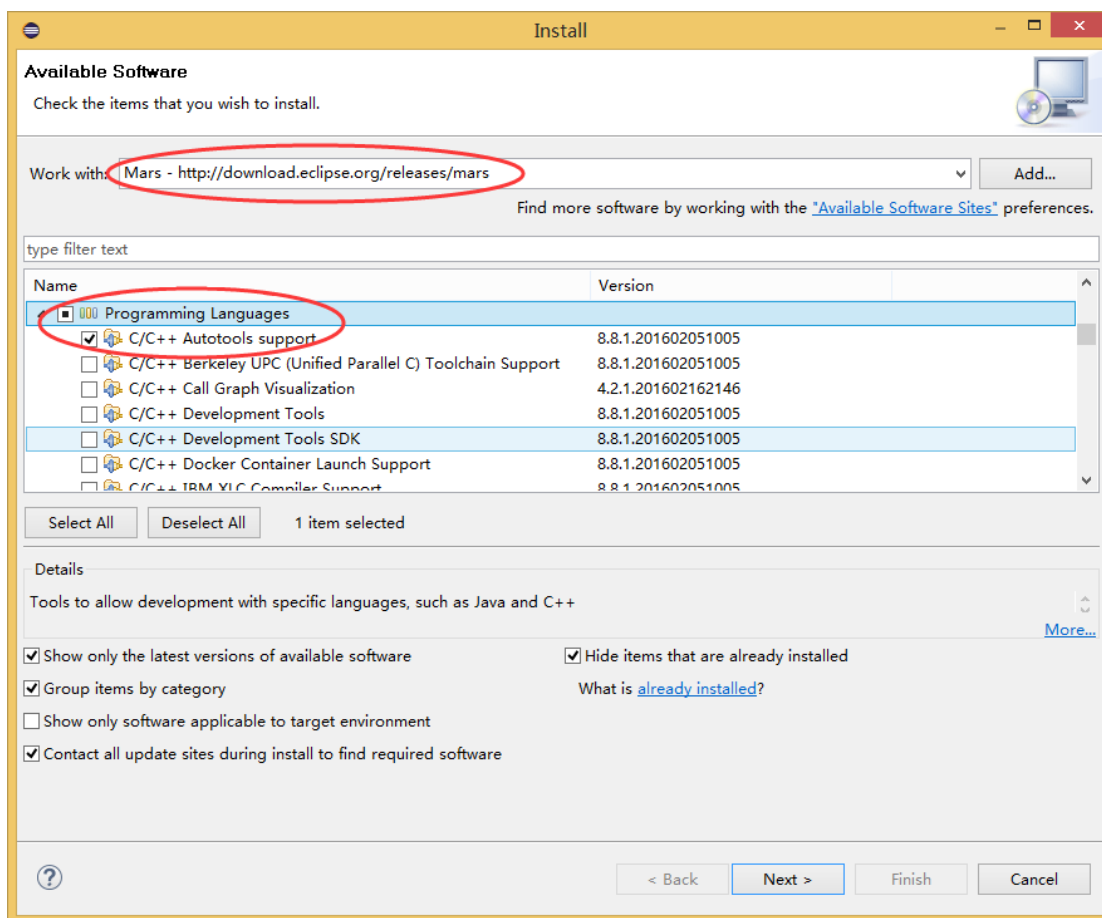


图 50 插件的补充和更新

#### 4.1.7 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。：

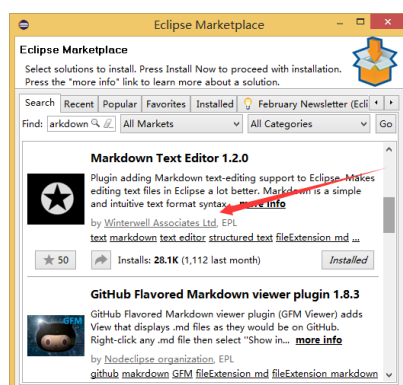


图 52 Markdown 插件

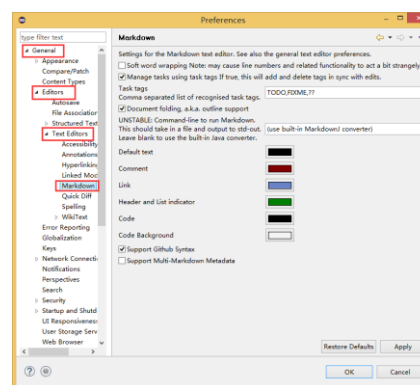


图 51 Markdown 插件配置

## 4.2 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 file->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：

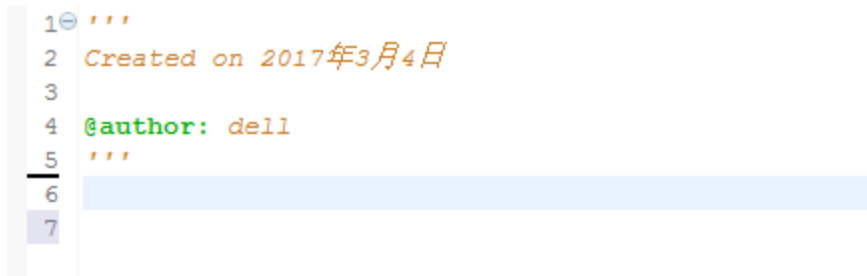


图 53 pydev module

编辑代码如图所示：

```

1 '''
2 Created on 201700300400
3
4 @author: dell
5 '''
6 # FIXME: 画一个心形线
7 import numpy as np
8 import matplotlib.pyplot as plt
9 plt.rcParams['font.sans-serif']=['SimHei'] #用来显示中文
10 a=float(input("输入你想要的心的尺寸'o^"))
11 alpha=np.arange(0,2*np.pi,0.02)
12 ax1=plt.subplot(121,polar=True)#121表示把图分成1*2的各式，且现在操作第一个图
13 l=a*(1-np.sin(alpha))
14 plt.plot(alpha,l,'r')
15 plt.text(1.4*np.pi,a,"love")
16
17 ax2=plt.subplot(122,polar=True)
18 plt.fill(alpha,l,'r')
19 ax2.set_title("来自咸鱼妹妹的告白~~",va='bottom')
20
21 plt.show()

```

图 54 心形线的代码<sup>[2][4][5][6][7]</sup>

由于我设置了标签，故而 task 栏会出现：

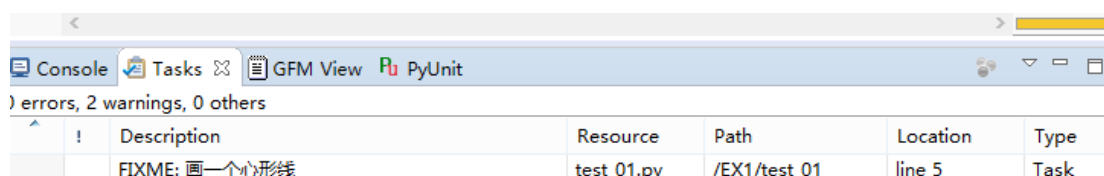


图 55 task 栏

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

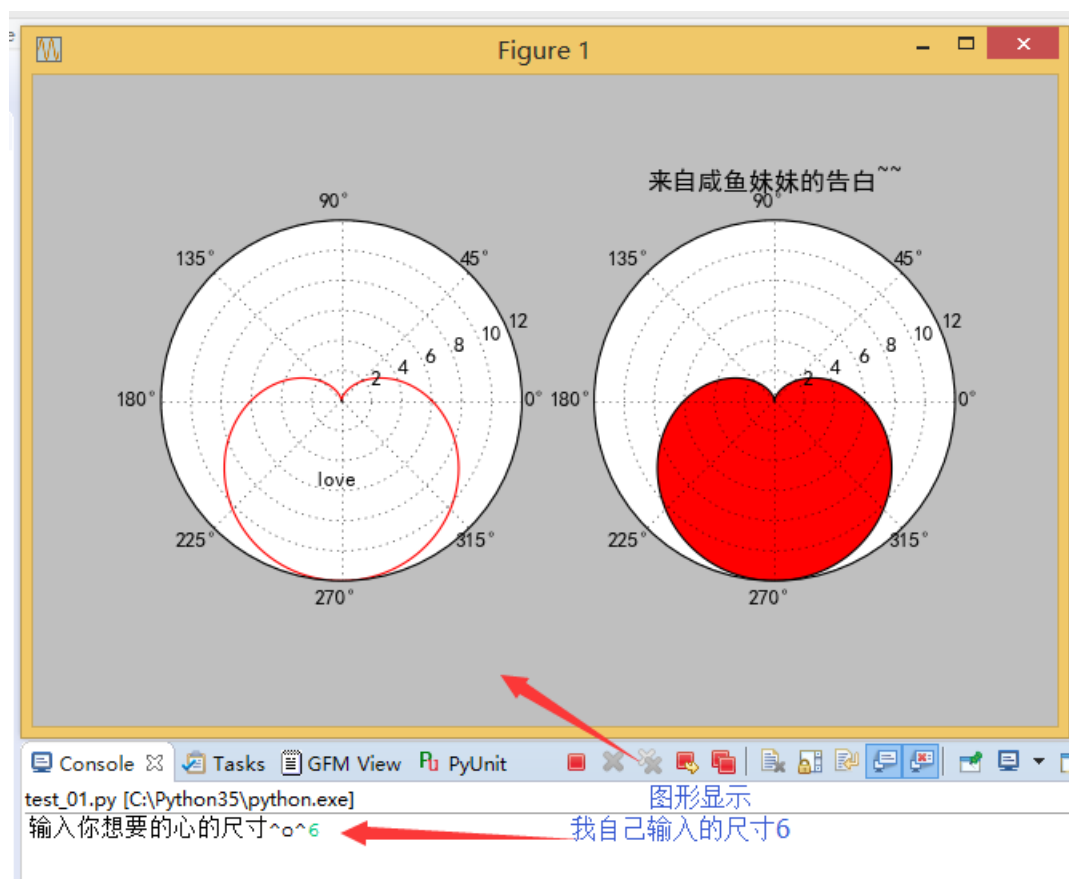


图 56 eclipse 示例输出

## 五、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

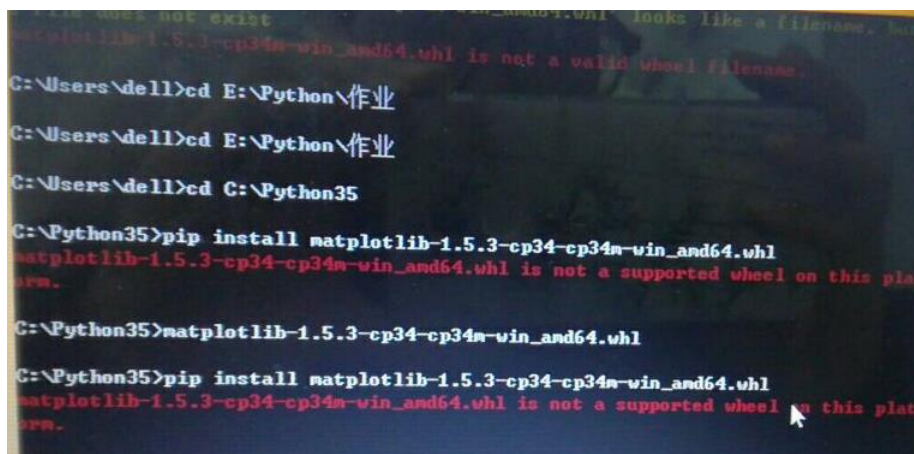


图 57 问题一

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

### Creating and cloning a repository

```
In [4]: # create a new git repository called gitdemo:
!git init learngit #把learngit目录变成git管理的仓库
usage: git init [-q | --quiet] [--bare] [--template=<template-directory>] [--shared[=<permissions>]] [<directory>]
```

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

```
In [5]: # create a new git repository called gitdemo:# 感觉像是在当前目录下创建一个新的仓库并交给git管理
!git init gitdemo
Initialized empty Git repository in E:\Python\浣漓瓶\home-S2017\home-S2017\myfirstrepo\gitdemo\.git\
```

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

```
In [6]: #建立一个名叫README的markdown的文件并且书写它的内容
%%file README.md
咸鱼也要有梦想

File "<ipython-input-6-401e50aaa505>", line 2
%%file README.md
^
SyntaxError: invalid syntax
```

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

```
In [12]: %%file README.md
          咸鱼也要有梦想
          #建立一个名叫README的markdown的文件并且书写它的内容
Writing README.md
```

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

建立一个名叫README的markdown的文件并且书写它的内容

```
In [12]: %%file README.md
         咸鱼也要有梦想

Writing README.md
```

图 62 问题三的解决

问题四：jupyter notebook 不能够删除分支：

```
In [32]: #合并分支
         !git merge yao
Already up-to-date.

In [33]: #删除分支
         !git branch -d yao
error: Cannot delete branch 'yao' checked out at 'F:/python/姚潘宸/jupyter/yaol'
```

图 63 问题四

解决方法：并没有将 head 指针切换回来

```
In [36]: #转移当前分支
         !git checkout master
Your branch is up-to-date with 'origin/master'.
Already on 'master'

In [37]: #合并分支
         !git merge yao
Updating d4df27d..e2c9752
Fast-forward
 README.md | 3 ++
 1 file changed, 2 insertions(+), 1 deletion(-)

In [38]: #删除分支
         !git branch -d yao
Deleted branch yao (was e2c9752).
```

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

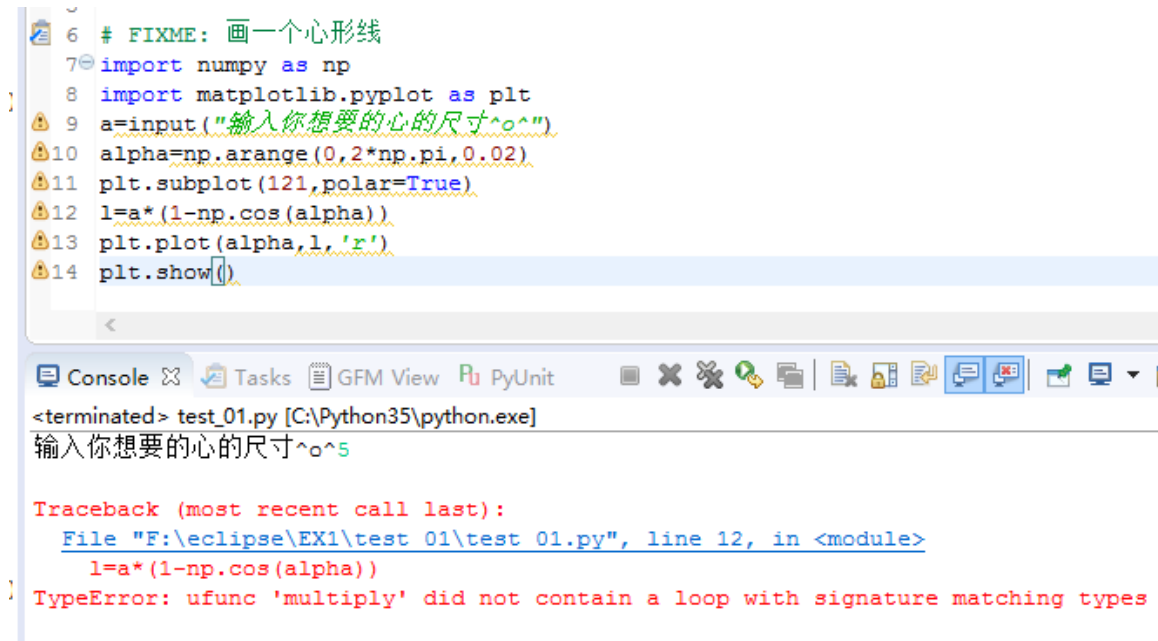
```
In [15]: #将README.md从暂存区提交到版本库
         !git commit-m"README.md 咸鱼"README.md
git: 'commit-mREADME.md 搁搁奔README.md' is not a git command. See 'git --help'.
```

图 66 问题五

```
In [16]: #将README.md从暂存区提交到版本库
         !git commit -m"README.md 咸鱼"README.md
[master (root-commit) d4df27d] README.md 搁搁奔README.md
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

图 65 问题五的解决

问题六: eclipse 编写的程序提示语法错误



```

6 # FIXME: 画一个心形线
7 import numpy as np
8 import matplotlib.pyplot as plt
9 a=input("输入你想要的心的尺寸^o^")
10 alpha=np.arange(0,2*np.pi,0.02)
11 plt.subplot(121,polar=True)
12 l=a*(1-np.cos(alpha))
13 plt.plot(alpha,l,'r')
14 plt.show()

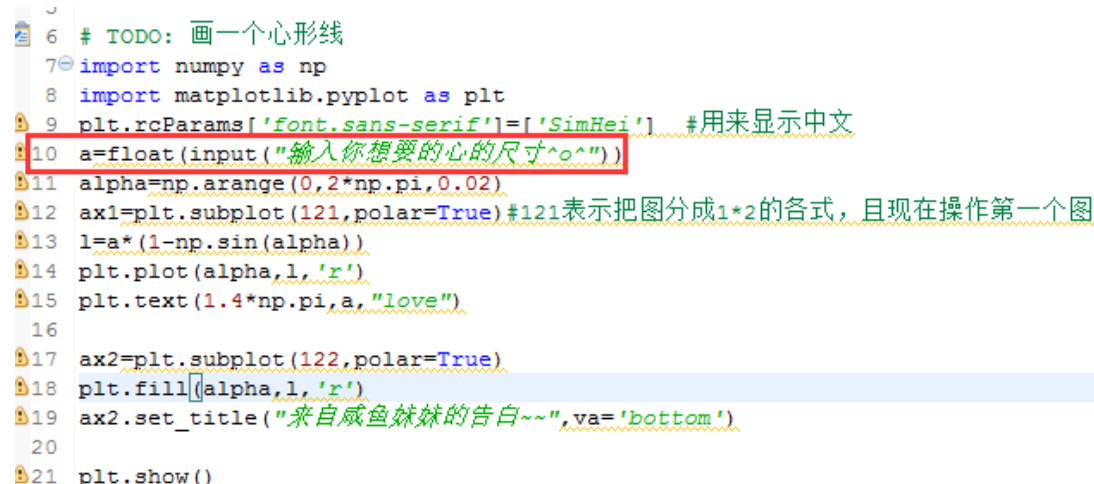
```

Console: <terminated> test\_01.py [C:\Python35\python.exe]  
 输入你想要的心的尺寸^o^5

Traceback (most recent call last):  
 File "F:\eclipse\EX1\test 01.py", line 12, in <module>  
 l=a\*(1-np.cos(alpha))  
 TypeError: ufunc 'multiply' did not contain a loop with signature matching types

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



```

6 # TODO: 画一个心形线
7 import numpy as np
8 import matplotlib.pyplot as plt
9 plt.rcParams['font.sans-serif']=['SimHei'] #用来显示中文
10 a=float(input("输入你想要的心的尺寸^o^"))
11 alpha=np.arange(0,2*np.pi,0.02)
12 ax1=plt.subplot(121,polar=True)#121表示把图分成1*2的各式,且现在操作第一个图
13 l=a*(1-np.sin(alpha))
14 plt.plot(alpha,l,'r')
15 plt.text(1.4*np.pi,a,"love")
16
17 ax2=plt.subplot(122,polar=True)
18 plt.fill(alpha,l,'r')
19 ax2.set_title("来自咸鱼妹妹的告白~~",va='bottom')
20
21 plt.show()

```

图 68 问题六的解决

## 六、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，大道理程序语言的选择，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。

## 七、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 plotting — 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