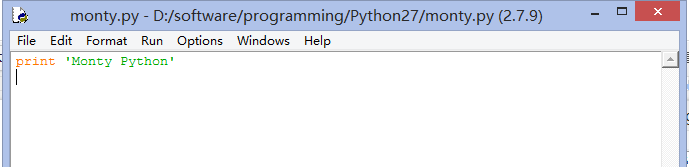
使用文本编辑器创建程序

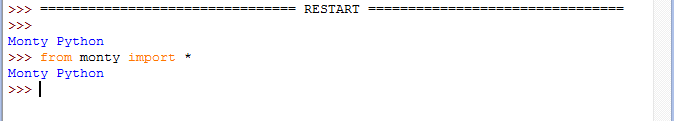
除了在命令行中输入命令，我们还可以在文本编辑器中运行。有时候要输入的代码段比较长，在IDLE中修改会比较方便。

打开IDLE

点FIle，选New File，在新窗口print ‘Monty Python’保存文件为monty.py，点run，选 Run Module运行。或者在解释器中输入from monty import \*运行。

注意：IDLE中使用中文输入法例如搜狗会出现退格变成方框的问题，所以必须使用英文输入法，很麻烦





函数

函数是指带命名的代码块，可以执行一些明确的任务。函数通常使用称为参数的变量进行输入，并且可能会生成一些结果，也称为返回值。我们使用关键字def加函数名及所有输入参数来定义一个函数，接下来是函数的主体。使用return来表示函数输出所产生的值。

**>>> from \_\_future\_\_ import division  
>>> def lexical\_diversity(text):  
... return len(text) / len(set(text))**

**>>> def lexical\_diversity(my\_text\_data):  
... word\_count = len(my\_text\_data)  
... vocab\_size = len(set(my\_text\_data))  
... diversity\_score = vocab\_size / word\_count  
... return diversity\_score**

|  |  |
| --- | --- |
| |  | | --- | | **def plural(word):  if word.endswith('y'):  return word[:-1] + 'ies'  elif word[-1] in 'sx' or word[-2:] in ['sh', 'ch']:  return word + 'es'  elif word.endswith('an'):  return word[:-2] + 'en'  else:  return word + 's'** | |
| |  |  | | --- | --- | |  | **>>> plural('fairy') 'fairies' >>> plural('woman') 'women'** | |

## 3.3 Modules

模块

一个文件中的变量和函数定义的集合被称为模块。相关模块的集合称为包。NLTK本身是包的集合，又是称为库。

我们把 plural(word)函数存在monty中

def plural(word):

if word.endswith('y'):

return word[:-1] + 'ies'

elif word[-1] in 'sx' or word[-2:] in ['sh', 'ch']:

return word + 'es'

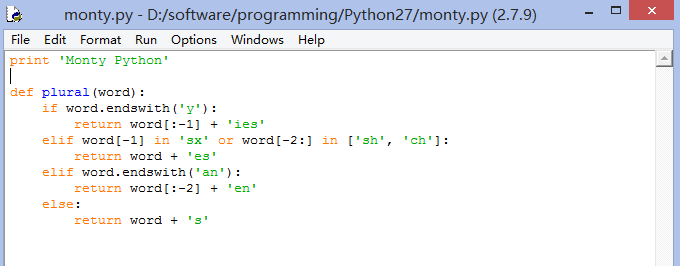
elif word.endswith('an'):

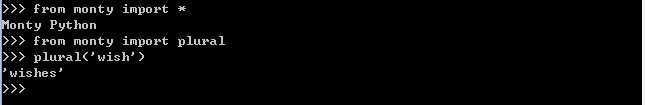
return word[:-2] + 'en'

else:

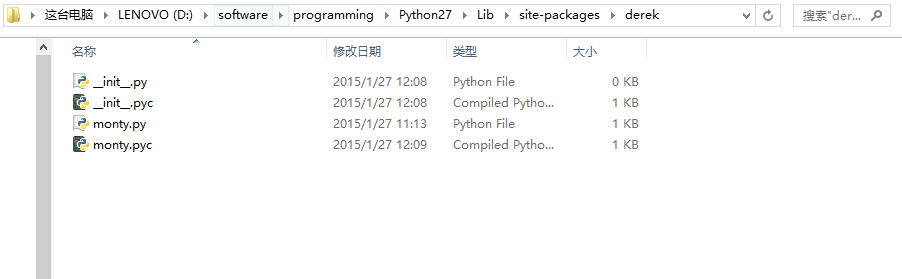
return word + 's'

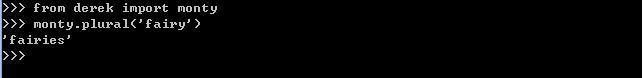
之后，我们就可以导入monty，引用plural函数了。





如果不想把文件放在根目录，而是想放在一个单独的文件夹中，比如nltk放在D:\software\programming\Python27\Lib\site-packages\nltk中，那么我们在D:\software\programming\Python27\Lib\site-packages\中新建一个derek文件夹，吧mont.py放进去，却发现不能导入。原因是我们还需要一个初始化文件\_init\_.py 这个文件可以是空文件，放入之后就可以引用了





Over time you will find that you create a variety of useful little text processing functions, and you end up copying them from old programs to new ones. Which file contains the latest version of the function you want to use? It makes life a lot easier if you can collect your work into a single place, and access previously defined functions without making copies.

To do this, save your function(s) in a file called (say) text\_proc.py. Now, you can access your work simply by importing it from the file:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> from text\_proc import plural >>> plural('wish') wishes >>> plural('fan') fen** | |

Our plural function obviously has an error, since the plural of *fan* is *fans*. Instead of typing in a new version of the function, we can simply edit the existing one. Thus, at every stage, there is only one version of our plural function, and no confusion about which one is being used.

A collection of variable and function definitions in a file is called a Python **module**. A collection of related modules is called a **package**. NLTK's code for processing the Brown Corpus is an example of a module, and its collection of code for processing all the different corpora is an example of a package. NLTK itself is a set of packages, sometimes called a **library**.

Caution!

If you are creating a file to contain some of your Python code, do *not* name your file nltk.py: it may get imported in place of the "real" NLTK package. When it imports modules, Python first looks in the current directory (folder).