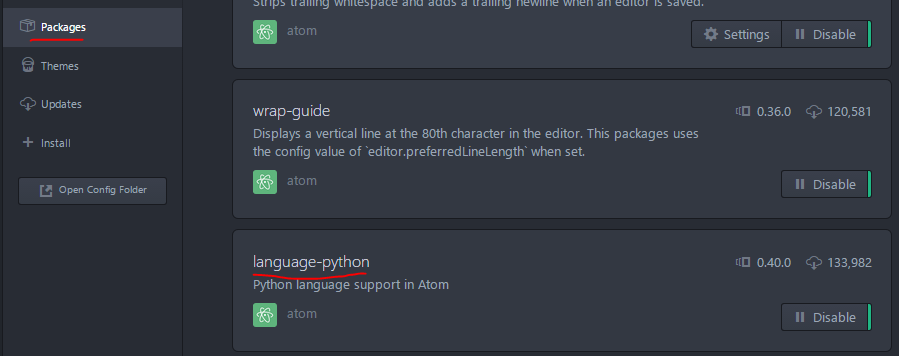
**基于ATOM的Python开发环境**

**程懋华**

1. **安装Atom**

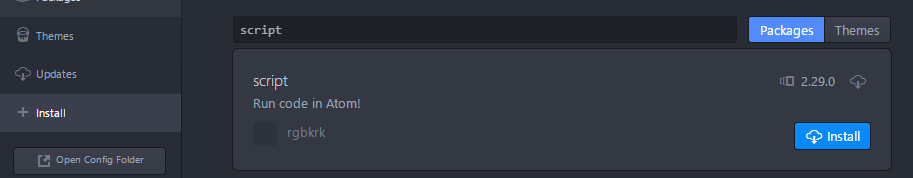
**https://atom.io/**

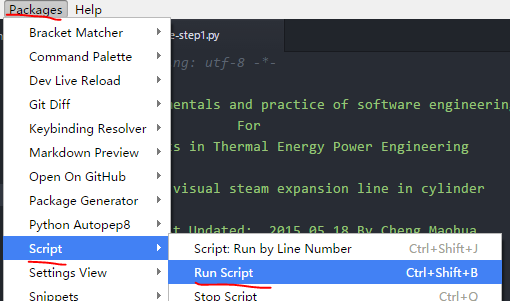
Atom已经预装了Python语言支持，是其核心包，无需安装



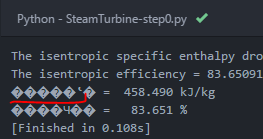
1. **运行Python**

运行Python程序需要安装atom运行脚本，具体请见：https://atom.io/packages/script



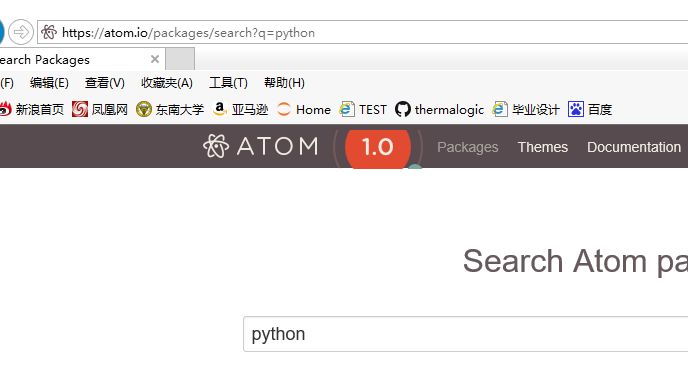


**就可以运行Python脚本程序了**



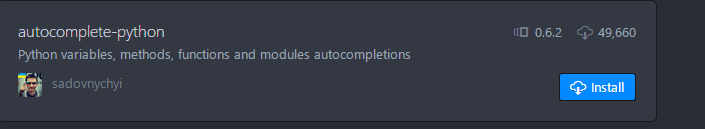
1. **安装Python插件**

可在官网，查询某类包信息，比在Atom软件中查，看起来更方便一点。有详情的插件安装和使用说明

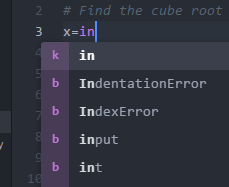


**３1 自动补全**

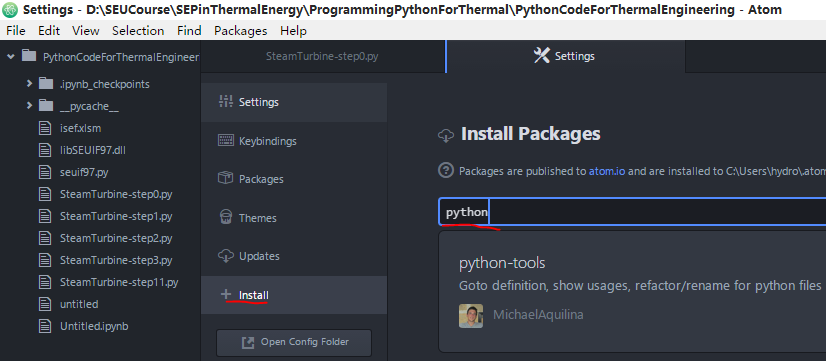
**必装**

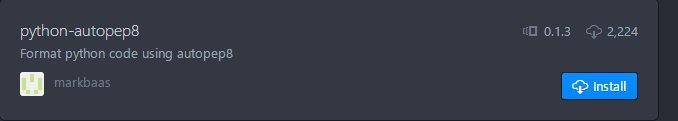


**安装后：**

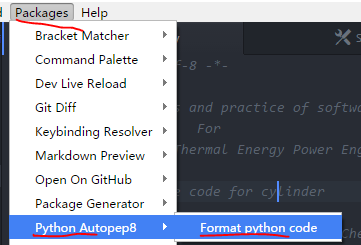


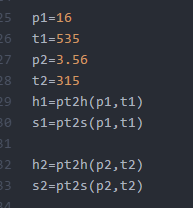
**３.２安装PEP8**



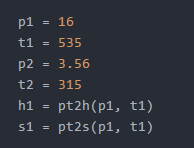


**使用PEP8：**

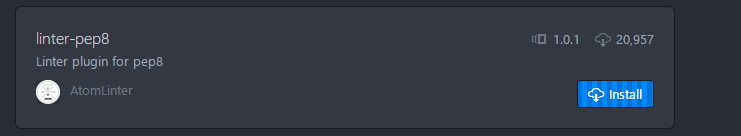




**使用后：**



**更多人用：**



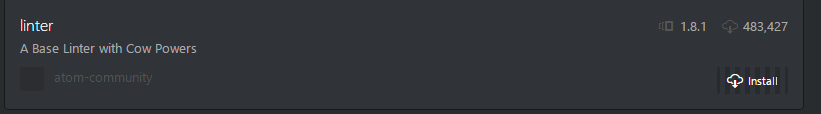
**这个安装后看不见，在哪里操作？？**

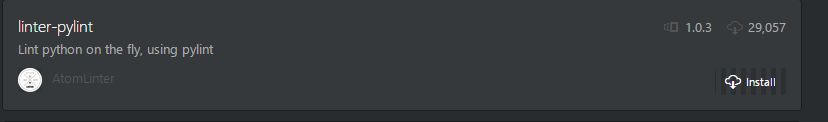
**３3 安装PEP257**

**首先需要安装：**

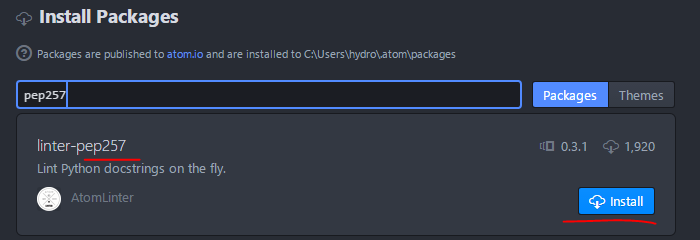
**Pip install pep257**

**然后，安装：**[linter](https://github.com/atom-community/linter)

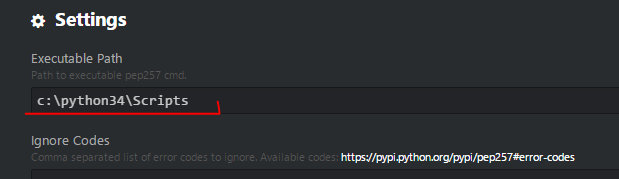




**再安装linter-pep257**



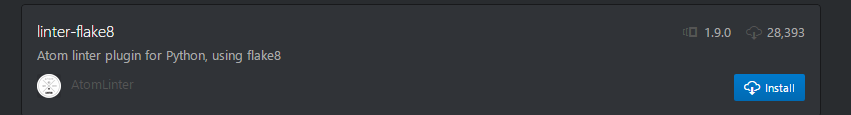
**配置：**



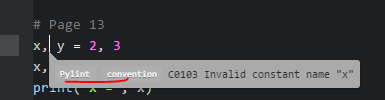
**使用PEP257?　没有看到效果？**

**安装问题?**

**装ｆｌａｋｅ８的很多**



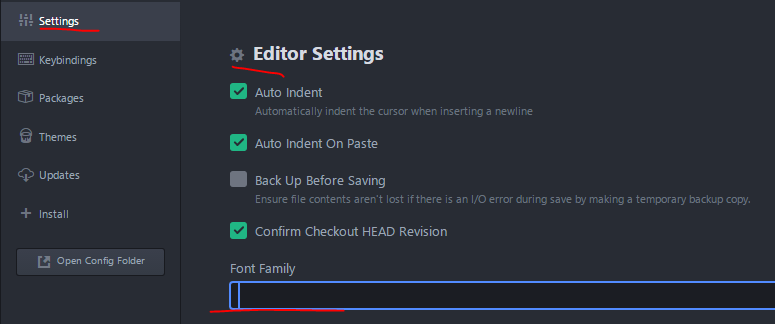
**３４pylint**



**但是停止提示后，再启动就无效了？**

1. **Jupyter集成**
2. **汉字乱码**

字体家族怎么空？ 加了也无效，输出还是乱码，



源码有：

# encoding: utf-8

# -\*- coding: utf-8 -\*-

源码部分正常，但是输出异常

Python3内部都是unicode表示，print(str)的结果重定向以后，**依赖于平台的编码设置**，比如

str="中文"

print(str)

>python  test.py  > out.txt

打开out.txt　是gb2312编码的,print(str)有没有办法象写文件那样可以指定编码, 比如:

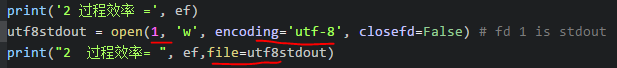
f=open('out.txt','w',**encoding='utf-8'**)

解决方法：

**utf8stdout = open(1, 'w', encoding='utf-8', closefd=False)**

**print(str, file=utf8stdout)**

其中，# fd 1 is stdout，1表示系统的stdout，输出就是正常中文了：





实际是将输出到stdout 当成一个文件，指定文件字符是采用encoding='utf-8’。

**问题：还是不能在print中直接指定字符是采用encoding='utf-8’**

注意：

**print**('2 等熵焓降 ='.encode('utf-8'), hdis)不行,输出是bytes

b'2 \xe7\xad\x89\xe7\x86\xb5\xe7\x84\x93\xe9\x99\x8d =' 425.003134565186

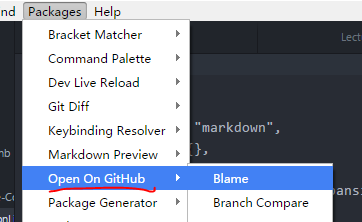
如果你用python2.7，那么在源代码开头加上codeing:gb18030，这样会让你在windows下更自在，基本上不出错。  
  
如果你用的是python3，那么**要输出到“控制台”**，或者是输出到文件时均要编码。编码成"gb18030"  
比如这样：  
  
s="中文"  
print（s.encode("gb18030")）

输出为bytes?????

如果字符串是这样定义**：s=u'中文' ,则该字符串的编码就被指定为unicode了，即python的内部编码，而与代码文件本身的编码无关**

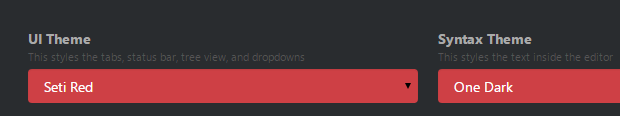
1. **GITHUB**

使用ATOM的优点是 ，连接GITHUB和本地一样方便

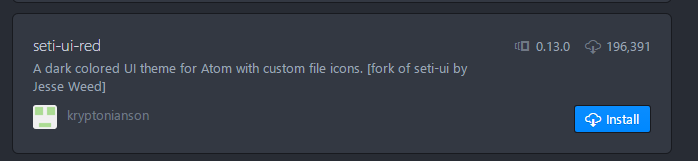


1. **增加主题**

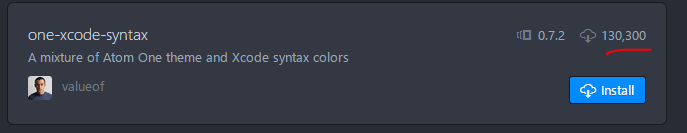
**主题有2类： UI和Syntax**



**UI**



**Syntax**



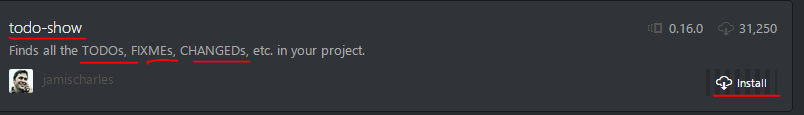
**用的人不少，但是，效果不如缺省的，可以不装**

1. **提高效率质量**

**1）关闭不常有的核心支持包，如Ruby语言等。**

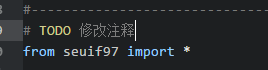
**2）安装插件**

**7.1 任务注释管理插件**

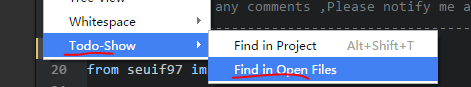


**安装后：**

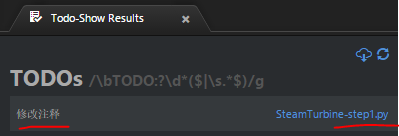
**加任务注释**



**提取任务注释**



**显示结果：**



**7.2 调试插件**



**atom-python-debugger package**

Quick and Easy way to add Python Debugger statements

**Usage**

Use "F7" / "shift+F7" to insert/remove ipdb.set\_trace() statements on current line. Takes into consideration indentation and tab preference. Enter as many ipdb.set\_trace() statemements as you like. Only one import statement is ever used. "shift+f7" will remove all debug statements.

Mac users need to use the function key: Fn+F7 and Fn+Shift+F7

**Example**

Before:

import datetime

class MyClass(object):

    def \_\_init\_\_(self, \*args, \*\*kwargs):

        pass

    def my\_function(self, \*args, \*\*kwargs):

        print date][time.datetime.now()  # ][ denotes cursor location

After pressing F7:

import ipdb

import datetime

class MyClass(object):

    def \_\_init\_\_(self, \*args, \*\*kwargs):

        pass

    def my\_function(self, \*args, \*\*kwargs):

        ipdb.set\_trace() ################## Break Point ######################

        print datetime.datetime.now()

Using Shift+F7 will return the code to the "Before" state.

1. **小结：**

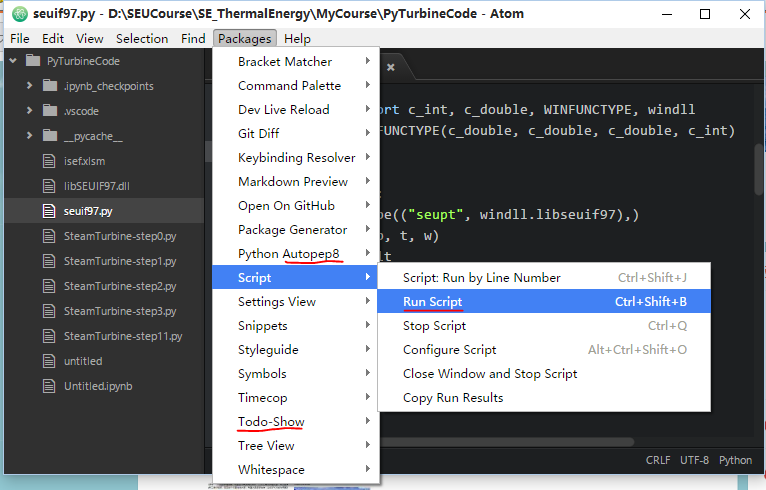
安装下面4个包。

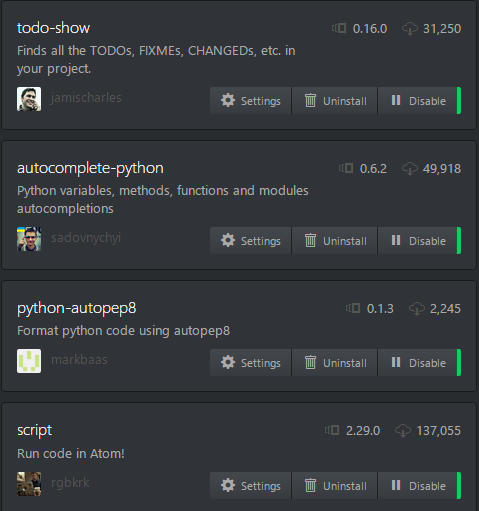
1. 脚本:script

2）自动补全:autocomplete-python

3）自动PEP8规范修订：python-autopep8

4) 提取任务注释：todo-show





一个有效轻便的Python 开发就建立起来了

**其他，插件异常，需要太多安装配置了，以后再说**