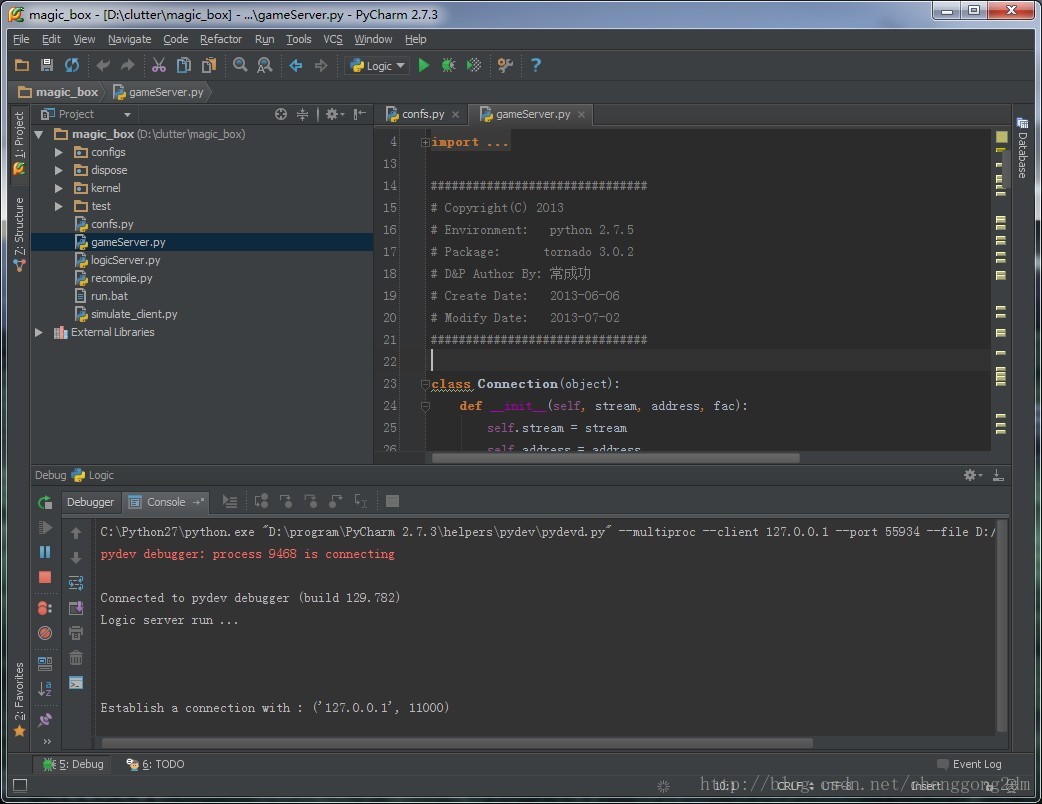
# IDE

# Pycharm

## 安装和首次使用

**PyCharm** 是我用过的python编辑器中，比较顺手的一个。而且可以跨平台，在macos和windows下面都可以用，这点比较好。

    首先预览一下 PyCharm 在实际应用中的界面：（更改了PyCharm的默认风格）



**安装**

首先去下载最新的pycharm 2.7.3，进行安装。可以直接在官网下载。

**PyCharm 的激活方式：**

1，**推荐购买正版**。

2，可以选择试用，免费试用30天。

3，网上找激活码：

（下面的激活码来自互联网，仅供学习交流之用）

**user name**: EMBRACE

**key:**

14203-12042010

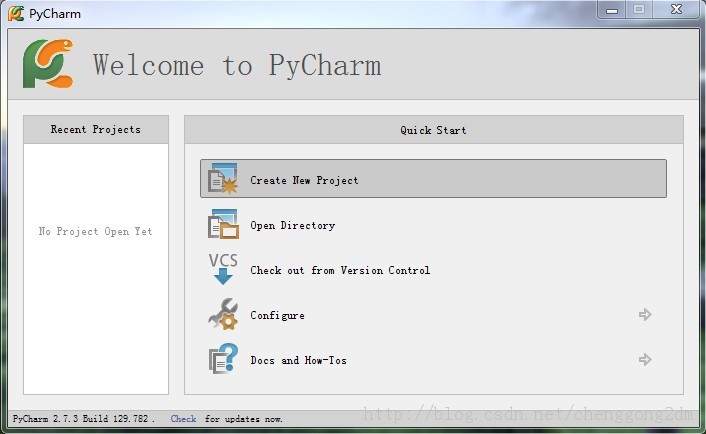
0000107Iq75C621P7X1SFnpJDivKnX

6zcwYOYaGK3euO3ehd1MiTT"2!Jny8

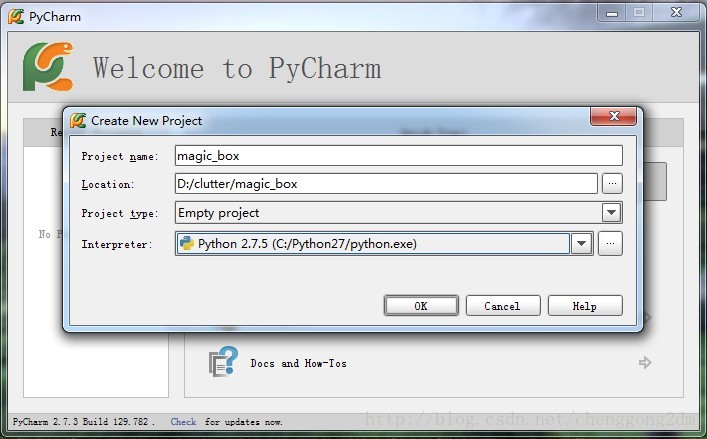
bff9VcTSJk7sRDLqKRVz1XGKbMqw3G

**首次使用**

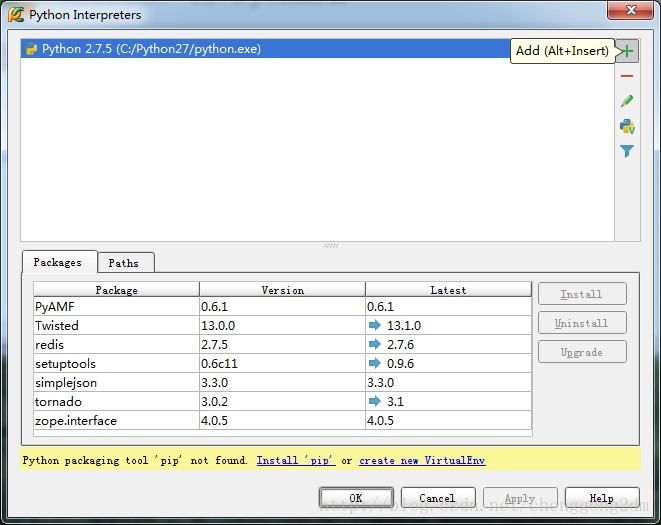
1,点击Create New Project.



2, 输入项目名、路径、选择python解释器。如果没有出现python解释器，进入步骤3.



3，选择python解释器。可以看到，一旦添加了python解释器，pycharm就会扫描出你已经安装的python扩展包，和这些扩展包的最新版本。（估计是pycharm连接了pypi）



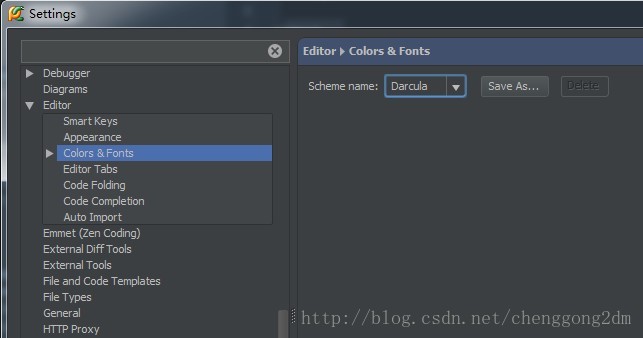
4，点击OK之后，就会创建一个空项目，里面包含一个.idea的文件夹，用于pycharm管理项目。

5，好了，写一个新的项目尝试一下pycharm吧！

■ **附注：**

pyCharm风格（配色方案）的调整，可以在：【File】-->【Settings】 的 Editor的子选项里进行设置。

比如我这个配色方案，选择的是Dracula（吸血鬼德库拉伯爵），如下图：



参考笔记:<http://blog.csdn.net/chenggong2dm/article/details/9365437>

## 设置字体大小

pycharm 是很好的一个IDE，在windows下，和macOS下，都能很好的运行。唯一缺点是启动慢。

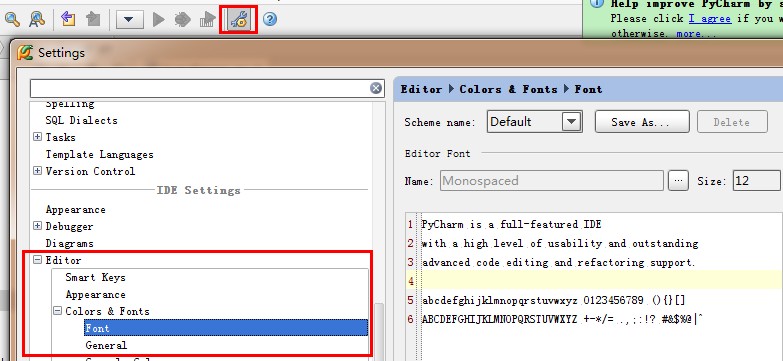
默认字体太小，在mac下，需要瞪大24K氪金狗眼才能看清。 为了保护好眼睛，我们需要把字体调整大一些：

**步骤：**

Settings-->Editor-->Colors & Fonts-->Font

然后在size那里调整。

注意，要先保存一个新的Scheme，才能调整字体大小。

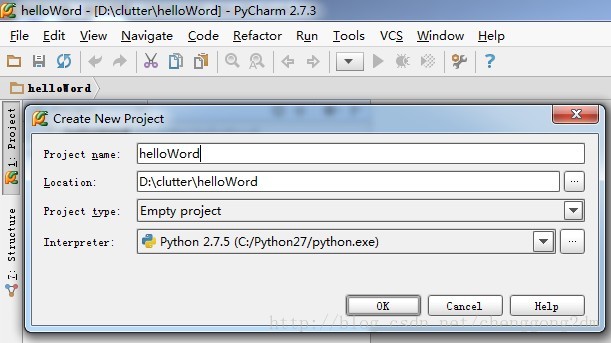


## Hello world！

作为PyCharm编辑器的起步，我们理所当然的先写一个Hello word，并运行它。（**此文献给对IDE不熟悉的初学者**）

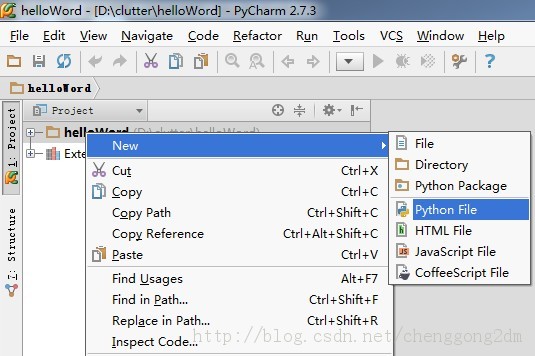
**1，新建一个项目**

File --> New Project...



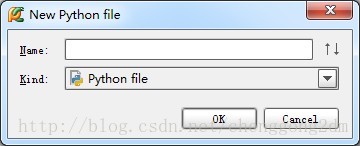
**2，新建一个文件**

右键单击刚建好的helloWord项目，选择New --> Python File



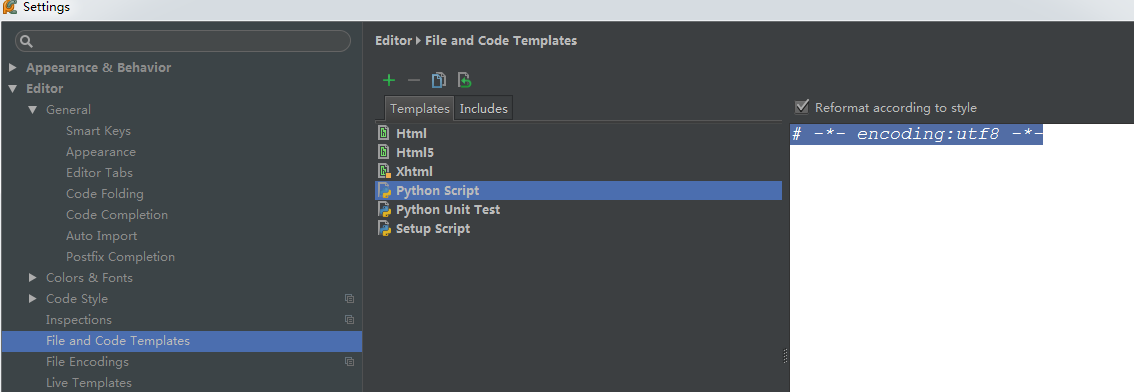
**3，输入文件名**

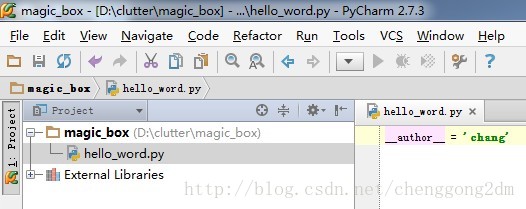
输入文件名，没什么好说的



**4，进入编写界面**

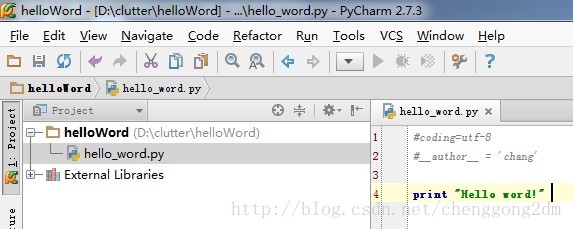
PyCharm的默认编辑界面很怪，会自动生成一行 \_\_author\_\_ = “作者” 的头。而比较常用的文件头，如：#coding=utf-8 等，反倒没有自动生成。





输入code：

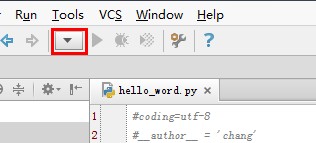
print "Hello word!"



**5，设置控制台**

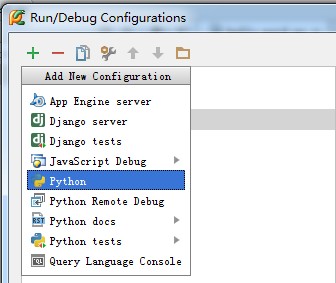
运行之前，我们发现快捷菜单上的【运行】和【调试】都是灰色的(如果不是，跳过)，不可触发状态。是因为，我们需要先配置一下控制台。

点击运行旁边的黑色倒三角，进入【Run/Debug Configurations】配置界面。（或者点击Run --> Edit Configurations 也一样）



在【Run/Debug Configurations】配置界面里，

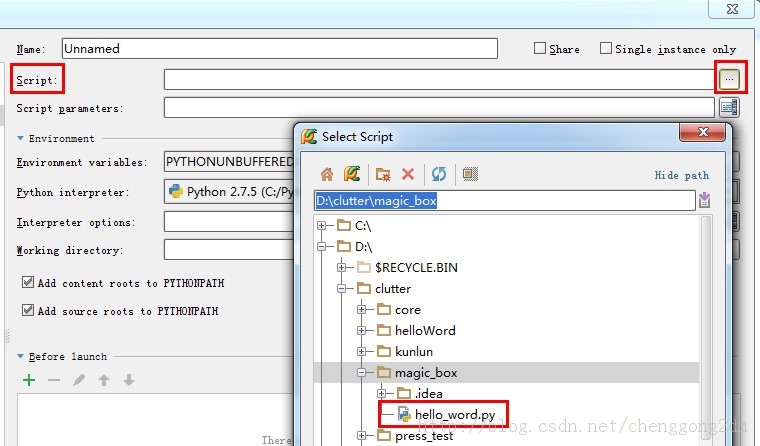
点击绿色的加号，新建一个配置项，并选择python。（因为hello word 就是个python程序）



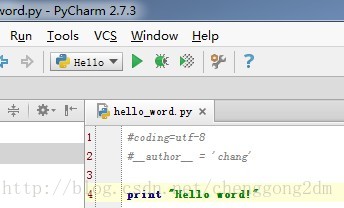
在右边的配置界面里：

Name 一栏里写个名字，比如Hello

点击Scrip选项，找到刚才我们写的hello\_word.py

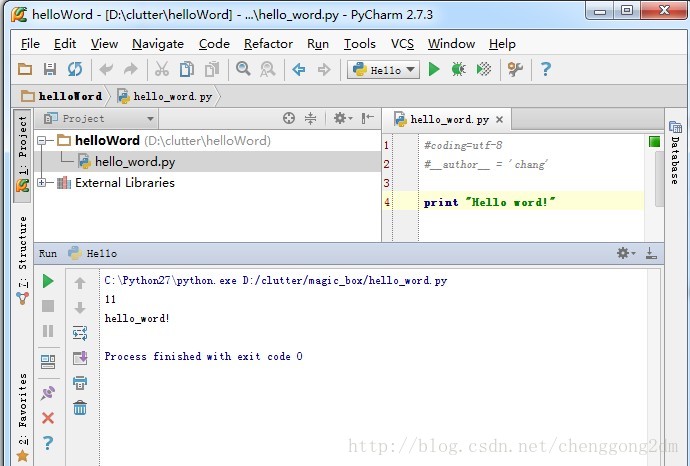


点击ok, 自动返回到编辑界面，这时候【运行】【调试】按钮全部变绿！



**6，运行**

点击绿色的运行按钮，观看输出的结果：



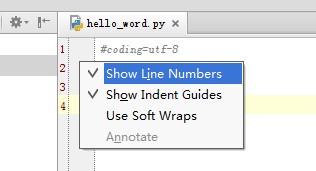
OK，PyCharm的首秀，圆满结束！

## 显示行号

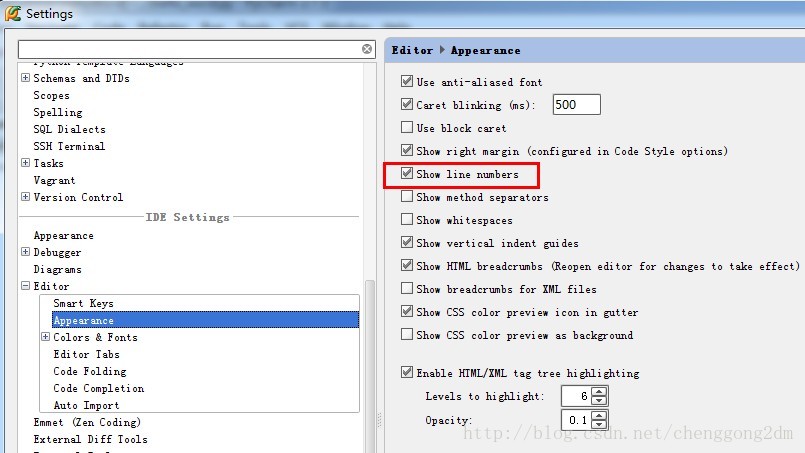
在**PyCharm**里，显示行号有两种办法：

**1，临时设置。右键单击行号处，选择 Show Line Numbers。**

但是这种方法，只对一个文件有效，并且，重启PyCharm 后消失。



**2，永久设置。File --> Settings -->Editor -->Appearance ,  之后勾选Show Line Numbers**。



## 05 断点 调试

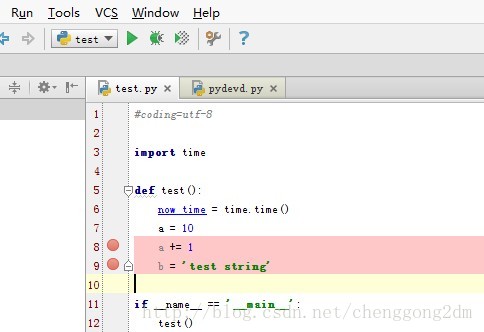
PyCharm 作为IDE，断点调试是必须有的功能。否则，我们还真不如用纯编辑器写的快。

【运行】和【调试】前的设置，详见前面的文章，helloword。0，

**0.调试的快捷键**

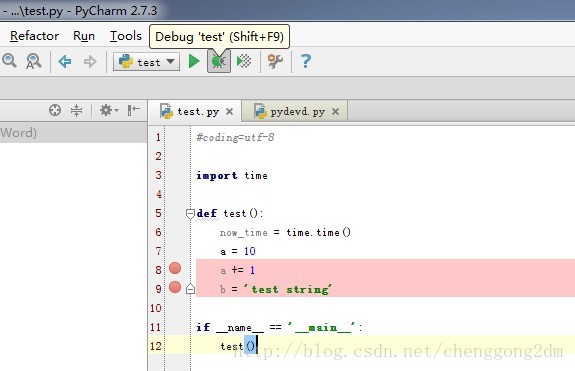
**1，设置断点**

在代码前面，行号的后面，鼠标单击，就可以设置断点。如下：

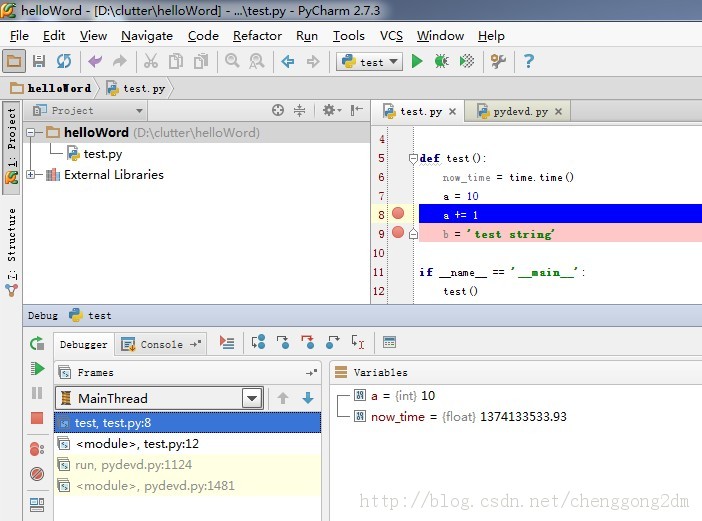


**2，调试 断点**

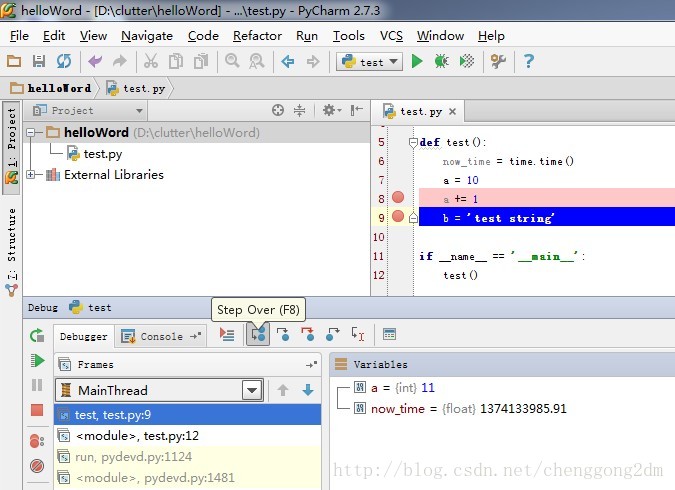
点击那个绿色的甲虫图标（似乎甲虫已经成为debug专用图标了），进行断点调试。



点击后，会运行到第一个断点。会显示该断点之前的变量信息。

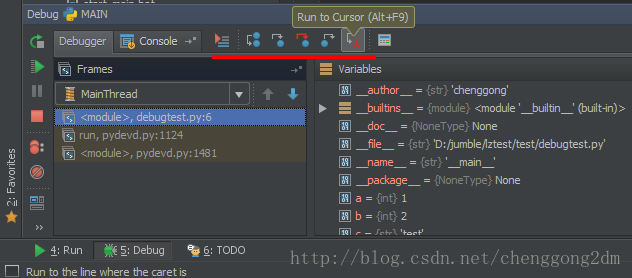


点击Step Over 或者按F8，我们继续往下运行，到下一个断点：



补充：

断点的其他操作，和功能，大家可以把鼠标移到相应的按钮上，看一下，试试便知。如下：

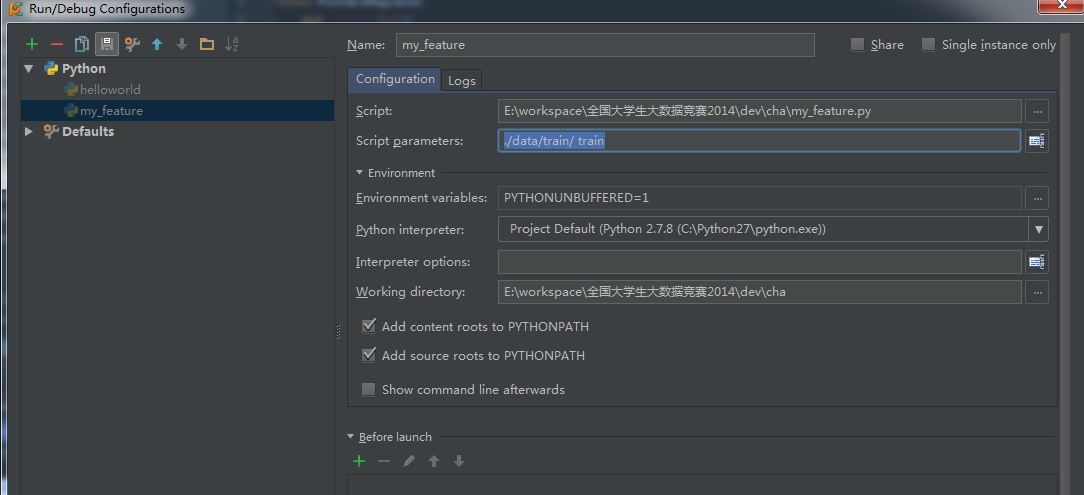


**总结：**

这就是PyCharm的断点功能，还是很简单易用的。

## 设置命令行参数

Script parameter:就是参数



## PEP8 Python 编码规范整理&配置autopep8到菜单

**目录(?)[-]**

1. **规范**
2. **配置**
3. **参考**

**1 规范**

**第一部分**

决定开始Python之路了，利用业余时间，争取更深入学习Python。编程语言不是艺术，而是工作或者说是工具，所以整理并遵循一套编码规范是十分必要的。所以今天下午我根据PEP 8整理了一份，以后都照此编码了，还会持续更新。

PEP8 Python 编码规范

一 代码编排

1 缩进。4个空格的缩进（编辑器都可以完成此功能），不使用Tap，更不能混合使用Tap和空格。

2 每行最大长度79，换行可以使用反斜杠，最好使用圆括号。换行点要在操作符的后边敲回车。

3 类和top-level函数定义之间空两行；类中的方法定义之间空一行；函数内逻辑无关段落之间空一行；其他地方尽量不要再空行。

二 文档编排

1 模块内容的顺序：模块说明和docstring—import—globals&constants—其他定义。其中import部分，又按标准、三方和自己编写顺序依次排放，之间空一行。

2 不要在一句import中多个库，比如import os, sys不推荐。

3 如果采用from XX import XX引用库，可以省略‘module.’，都是可能出现命名冲突，这时就要采用import XX。

三 空格的使用

        总体原则，避免不必要的空格。

1 各种右括号前不要加空格。

2 逗号、冒号、分号前不要加空格。

3 函数的左括号前不要加空格。如Func(1)。

4 序列的左括号前不要加空格。如list[2]。

5 操作符左右各加一个空格，不要为了对齐增加空格。

6 函数默认参数使用的赋值符左右省略空格。

7 不要将多句语句写在同一行，尽管使用‘；’允许。

8 if/for/while语句中，即使执行语句只有一句，也必须另起一行。

四 注释

        总体原则，错误的注释不如没有注释。所以当一段代码发生变化时，第一件事就是要修改注释！

        注释必须使用英文，最好是完整的句子，首字母大写，句后要有结束符，结束符后跟两个空格，开始下一句。如果是短语，可以省略结束符。

1 块注释，在一段代码前增加的注释。在‘#’后加一空格。段落之间以只有‘#’的行间隔。比如：

# Description : Module config.

#

# Input : None

#

# Output : None

2 行注释，在一句代码后加注释。比如：x = x + 1 # Increment x

但是这种方式尽量少使用。

3 避免无谓的注释。

五 文档描述

1 为所有的共有模块、函数、类、方法写docstrings；非共有的没有必要，但是可以写注释（在def的下一行）。

2 如果docstring要换行，参考如下例子,详见PEP 257

"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""

六 命名规范

        总体原则，新编代码必须按下面命名风格进行，现有库的编码尽量保持风格。

1 尽量单独使用小写字母‘l’，大写字母‘O’等容易混淆的字母。

2 模块命名尽量短小，使用全部小写的方式，可以使用下划线。

3 包命名尽量短小，使用全部小写的方式，不可以使用下划线。

4 类的命名使用CapWords的方式，模块内部使用的类采用\_CapWords的方式。

5 异常命名使用CapWords+Error后缀的方式。

6 全局变量尽量只在模块内有效，类似C语言中的static。实现方法有两种，一是\_\_all\_\_机制;二是前缀一个下划线。

7 函数命名使用全部小写的方式，可以使用下划线。

8 常量命名使用全部大写的方式，可以使用下划线。

9 类的属性（方法和变量）命名使用全部小写的方式，可以使用下划线。

9 类的属性有3种作用域public、non-public和subclass API，可以理解成C++中的public、private、protected，non-public属性前，前缀一条下划线。

11 类的属性若与关键字名字冲突，后缀一下划线，尽量不要使用缩略等其他方式。

12 为避免与子类属性命名冲突，在类的一些属性前，前缀两条下划线。比如：类Foo中声明\_\_a,访问时，只能通过Foo.\_Foo\_\_a，避免歧义。如果子类也叫Foo，那就无能为力了。

13 类的方法第一个参数必须是self，而静态方法第一个参数必须是cls。

七 编码建议

1 编码中考虑到其他python实现的效率等问题，比如运算符‘+’在CPython（Python）中效率很高，都是Jython中却非常低，所以应该采用.join()的方式。

2 尽可能使用‘is’‘is not’取代‘==’，比如if x is not None 要优于if x。

3 使用基于类的异常，每个模块或包都有自己的异常类，此异常类继承自Exception。

4 异常中不要使用裸露的except，except后跟具体的exceptions。

5 异常中try的代码尽可能少。比如：

try:

value = collection[key]

except KeyError:

return key\_not\_found(key)

else:

return handle\_value(value)

要优于

try:

# Too broad!

return handle\_value(collection[key])

except KeyError:

# Will also catch KeyError raised by handle\_value()

return key\_not\_found(key)

6 使用startswith() and endswith()代替切片进行序列前缀或后缀的检查。比如：

Yes: if foo.startswith('bar'):优于

No: if foo[:3] == 'bar':

7 使用isinstance()比较对象的类型。比如

Yes: if isinstance(obj, int): 优于

No: if type(obj) is type(1):

8 判断序列空或不空，有如下规则

Yes: if not seq:

if seq:

优于

No: if len(seq)

if not len(seq)

9 字符串不要以空格收尾。

10 二进制数据判断使用 if boolvalue的方式。

**2 配置**

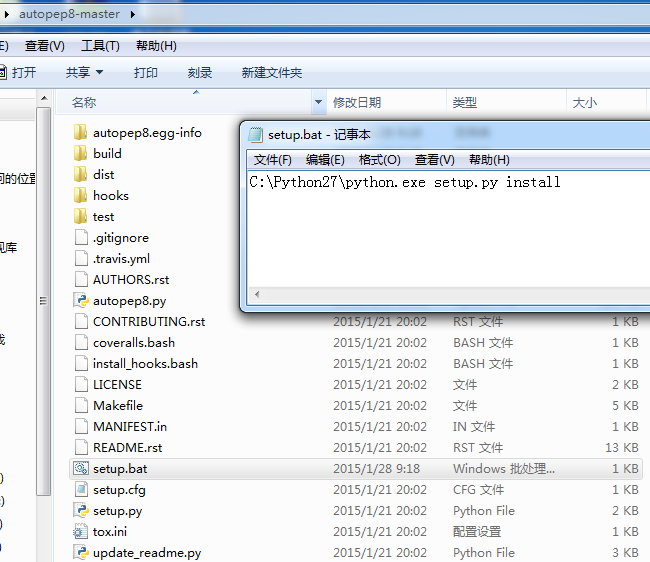
Pycharm 可以自动检测PEP8规范。

我们可以安装autopep8来自动修改文件实现PEP8规范。

0.下载&安装

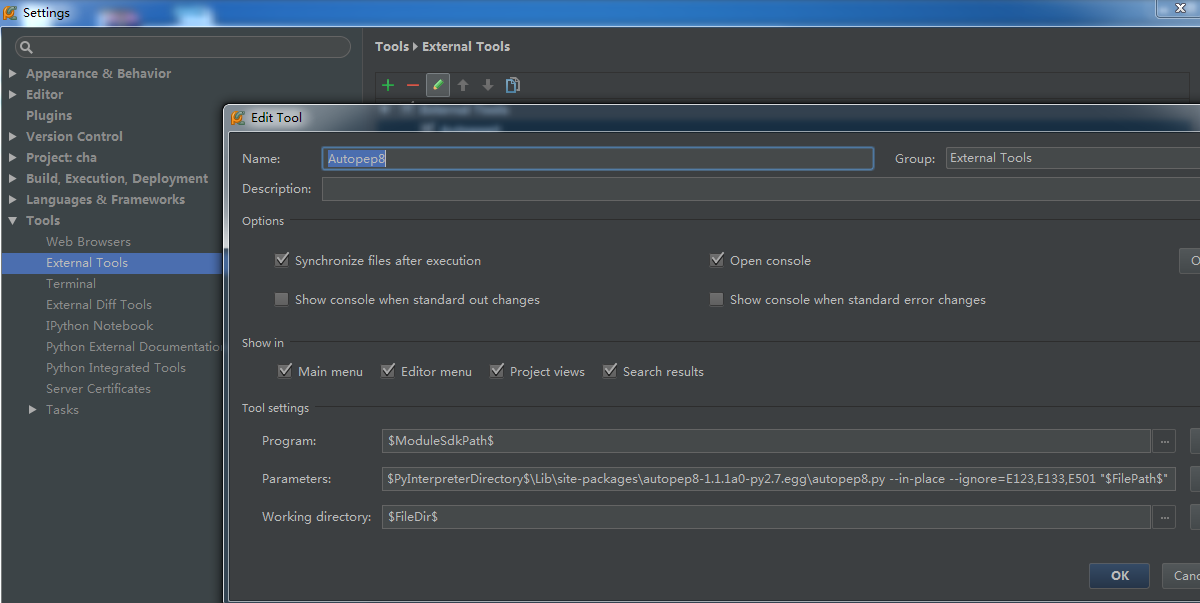
       PEP8下载地址:<https://github.com/hhatto/autopep8>

       新建一个bat文件安装即可(win7我经常这么安装，快吧)



1.通过Pycharm安装autopep8

2.File->Setting->Externel Tools-> 添加工具



配置External Tool

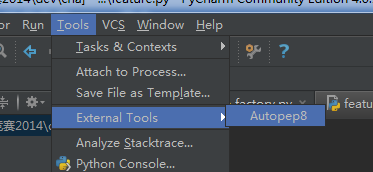
在最后书写命令的时候应该这么写：

$ModuleSdkPath$

$PyInterpreterDirectory$\Lib\site-packages\autopep8-1.1.1a0-py2.7.egg\autopep8.py --in-place --ignore=E123,E133,E501 "$FilePath$"

$FileDir$

3.使用



**3 参考**

PEP8规范:参考:[**http://www.douban.com/note/134971609/**](http://www.douban.com/note/134971609/)

## pycharm快捷键及一些常用设置

**PyCharm3.0默认快捷键(翻译的)**

**1、编辑（Editing）**

Ctrl + Space 基本的代码完成（类、方法、属性）

Ctrl + Alt + Space 快速导入任意类

Ctrl + Shift + Enter 语句完成

Ctrl + P 参数信息（在方法中调用参数）

Ctrl + Q 快速查看文档

Shift + F1 外部文档

Ctrl + 鼠标 简介

Ctrl + F1 显示错误描述或警告信息

Alt + Insert 自动生成代码

Ctrl + O 重新方法

Ctrl + Alt + T 选中

Ctrl + / 行注释

Ctrl + Shift + / 块注释

Ctrl + W 选中增加的代码块

Ctrl + Shift + W 回到之前状态

Ctrl + Shift + ]/[ 选定代码块结束、开始

Alt + Enter 快速修正

Ctrl + Alt + L 代码格式化

Ctrl + Alt + O 优化导入

Ctrl + Alt + I 自动缩进

Tab / Shift + Tab 缩进、不缩进当前行

Ctrl+X/Shift+Delete 剪切当前行或选定的代码块到剪贴板

Ctrl+C/Ctrl+Insert 复制当前行或选定的代码块到剪贴板

Ctrl+V/Shift+Insert 从剪贴板粘贴

Ctrl + Shift + V 从最近的缓冲区粘贴

Ctrl + D 复制选定的区域或行

Ctrl + Y 删除选定的行

Ctrl + Shift + J 添加智能线

Ctrl + Enter 智能线切割

Shift + Enter 另起一行

Ctrl + Shift + U 在选定的区域或代码块间切换

Ctrl + Delete 删除到字符结束

Ctrl + Backspace 删除到字符开始

Ctrl + Numpad+/- 展开折叠代码块

Ctrl + Numpad+ 全部展开

Ctrl + Numpad- 全部折叠

Ctrl + F4 关闭运行的选项卡

**2、查找/替换(Search/Replace)**

F3 下一个

Shift + F3 前一个

Ctrl + R 替换

Ctrl + Shift + F 全局查找

Ctrl + Shift + R 全局替换

**3、运行(Running)**

Alt + Shift + F10 运行模式配置

Alt + Shift + F9 调试模式配置

Shift + F10 运行

Shift + F9 调试

Ctrl + Shift + F10 运行编辑器配置

Ctrl + Alt + R 运行manage.py任务

**4、调试(Debugging)**

shift+ F9 开始调试

7上8下的

F8 跳过这条语句

F7 进入这条语句

Shift + F8 退出，到下一个断点

ctrl + F2 强制退出调试

Alt + F9 运行游标

Alt + F8 验证表达式

Ctrl + Alt + F8 快速验证表达式

F9 恢复程序

Ctrl + F8 断点开关

Ctrl + Shift + F8 查看断点

**5、导航(Navigation)**

Ctrl + N 跳转到类

Ctrl + Shift + N 跳转到符号

Alt + Right/Left 跳转到下一个、前一个编辑的选项卡

F12 回到先前的工具窗口

Esc 从工具窗口回到编辑窗口

Shift + Esc 隐藏运行的、最近运行的窗口

Ctrl + Shift + F4 关闭主动运行的选项卡

Ctrl + G 查看当前行号、字符号

Ctrl + E 当前文件弹出

Ctrl+Alt+Left/Right 后退、前进

Ctrl+Shift+Backspace 导航到最近编辑区域

Alt + F1 查找当前文件或标识

Ctrl+B / Ctrl+Click 跳转到声明

Ctrl + Alt + B 跳转到实现

Ctrl + Shift + I查看快速定义

Ctrl + Shift + B跳转到类型声明

Ctrl + U跳转到父方法、父类

Alt + Up/Down跳转到上一个、下一个方法

Ctrl + ]/[跳转到代码块结束、开始

Ctrl + F12弹出文件结构

Ctrl + H类型层次结构

Ctrl + Shift + H方法层次结构

Ctrl + Alt + H调用层次结构

F2 / Shift + F2下一条、前一条高亮的错误

F4 / Ctrl + Enter编辑资源、查看资源

Alt + Home显示导航条F11书签开关

Ctrl + Shift + F11书签助记开关

Ctrl + #[0-9]跳转到标识的书签

Shift + F11显示书签

**6、搜索相关(Usage Search)**

Alt + F7/Ctrl + F7文件中查询用法

Ctrl + Shift + F7文件中用法高亮显示

Ctrl + Alt + F7显示用法

**7、重构(Refactoring)**

F5复制F6剪切

Alt + Delete安全删除

Shift + F6重命名

Ctrl + F6更改签名

Ctrl + Alt + N内联

Ctrl + Alt + M提取方法

Ctrl + Alt + V提取属性

Ctrl + Alt + F提取字段

Ctrl + Alt + C提取常量

Ctrl + Alt + P提取参数

**8、控制VCS/Local History**

Ctrl + K提交项目

Ctrl + T更新项目

Alt + Shift + C查看最近的变化

Alt + BackQuote(’)VCS快速弹出

**9、模版(Live Templates)**

Ctrl + Alt + J当前行使用模版

Ctrl +Ｊ插入模版

**10、基本(General)**

Alt + #[0-9]打开相应的工具窗口

Ctrl + Alt + Y同步

Ctrl + Shift + F12最大化编辑开关

Alt + Shift + F添加到最喜欢

Alt + Shift + I根据配置检查当前文件

Ctrl + BackQuote(’)快速切换当前计划

Ctrl + Alt + S　打开设置页

Ctrl + Shift + A查找编辑器里所有的动作

Ctrl + Tab在窗口间进行切换

**一些常用设置：**

1. pycharm默认是自动保存的，习惯自己按ctrl + s 的可以进行如下设置：

1. file -> Setting -> General -> Synchronization -> Save files on frame deactivation 和 Save files automatically if application is idle for .. sec 的勾去掉

2. file ->Setting -> Editor -> Editor Tabs -> Mark modified tabs with asterisk 打上勾

2. Alt + Enter: 自动添加包

3. 对于常用的快捷键，可以设置为visual studio(eclipse...)一样的：

file -> Setting -> Keymap -> Keymaps -> vuisual studio -> Apply

4. Pycharm中默认是不能用Ctrl+滚轮改变字体大小的，可以在file -> Setting ->Editor-〉Mouse中设置

5. 要设置Pycharm的字体，要先在file -> Setting ->Editor-〉Editor中选择一种风格并保存，然后才可以改变

6. 在setting中搜索theme可以改变主题，所有配色统一改变

7.新建.py文件 文件内容会自动添加作者信息，如何把自动添加去掉？ file->setting->Editor->file and code templatea ->python script

pycharm快捷键及一些常用设置:<http://www.2cto.com/os/201410/341542.html>