**pycharm常用快捷键**

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

F1   外部文档

Shift + F1    外部文档，进入web文档主页

Ctrl + Shift + Z --> Redo 重做

Ctrl + 鼠标    简介/进入代码定义  
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/取消行注释  
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（当前位置的：函数，注释等）  
Ctrl + shift + Numpad+/-   展开/折叠所有代码块  
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  或者连续2次敲击shift   全局查找{可以在整个项目中查找某个字符串什么的，如查找某个函数名字符串看之前是怎么使用这个函数的}  
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)  
F8   跳过  
F7   进入  
Shift + F8   退出  
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，打开最近使用的文件列表  
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

**pycharm常用设置**

lz提示一下，pycharm中的设置是可以导入和导出的，file>export settings可以保存当前pycharm中的设置为jar文件，重装时可以直接import settings>jar文件，就不用重复配置了。

file -> Setting ->Editor

1. 设置[Python](http://lib.csdn.net/base/python" \o "Python知识库" \t "_blank)自动引入包，要先在 >general > autoimport -> python :show popup

     快捷键：Alt + Enter: 自动添加包

2. “代码自动完成”时间延时设置

  > Code Completion   -> Auto code completion in (ms):0  -> Autopopup in (ms):500

3. Pycharm中默认是不能用Ctrl+滚轮改变字体大小的，可以在〉Mouse中设置

4. 显示“行号”与“空白字符”

  > Appearance  -> 勾选“Show line numbers”、“Show whitespaces”、“Show method separators”

5. 设置编辑器“颜色与字体”主题

  > Colors & Fonts -> Scheme name -> 选择"monokai"“Darcula”

  说明：先选择“monokai”，再“Save As”为"monokai-pipi"，因为默认的主题是“只读的”，一些字体大小颜色什么的都不能修改，拷贝一份后方可修改！

  修改字体大小

> Colors & Fonts -> Font -> Size -> 设置为“14”

6. 设置缩进符为制表符“Tab”

  File -> Default Settings -> Code Style

  -> General -> 勾选“Use tab character”

  -> Python -> 勾选“Use tab character”

  -> 其他的语言代码同理设置

7. 去掉默认折叠

  > Code Folding -> Collapse by default -> 全部去掉勾选

8. 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：  
    > General -> Synchronization -> Save files on frame deactivation  和 Save files automatically if application is idle for .. sec 的勾去掉

    > Editor Tabs -> Mark modified tabs with asterisk 打上勾

9.>file and code template>python scripts

#!/usr/bin/env python  
# -\*- coding: utf-8 -\*-  
"""  
\_\_title\_\_ = '$Package\_name'  
\_\_author\_\_ = '$USER'  
\_\_mtime\_\_ = '$DATE'  
# code is far away from bugs with the god animal protecting  
    I love animals. They taste delicious.  
              ┏┓      ┏┓  
            ┏┛┻━━━┛┻┓  
            ┃      ☃      ┃  
            ┃  ┳┛  ┗┳  ┃  
            ┃      ┻      ┃  
            ┗━┓      ┏━┛  
                ┃      ┗━━━┓  
                ┃  神兽保佑    ┣┓  
                ┃　永无BUG！   ┏┛  
                ┗┓┓┏━┳┓┏┛  
                  ┃┫┫  ┃┫┫  
                  ┗┻┛  ┗┻┛  
"""

10 python文件默认编码

File Encodings> IDE Encoding: UTF-8;Project Encoding: UTF-8;

11. 代码自动整理设置

这里line breaks去掉√，否则bar, 和baz会分开在不同行，不好看。

File -> Settings -> appearance

1. 修改IDE快捷键方案

  > Keymap

1) execute selection in console : add keymap > ctrl + enter

  系统自带了好几种快捷键方案，下拉框中有如“defaul”,“Visual Studio”,在查找Bug时非常有用,“NetBeans 6.5”,“Default for GNOME”等等可选项，

  因为“Eclipse”方案比较大众，个人用的也比较多，最终选择了“Eclipse”。

  还是有几个常用的快捷键跟Eclipse不一样，为了能修改，还得先对Eclipse方案拷贝一份：

  (1).代码提示功能，默认是【Ctrl+空格】，现改为跟Eclipse一样，即【Alt+/】

  Main menu -> code -> Completion -> Basic -> 设置为“Alt+/”

  Main menu -> code -> Completion -> SmartType -> 设置为“Alt+Shift+/”

  不过“Alt+/”默认又被

  Main menu -> code -> Completion -> Basic -> Cyclic Expand Word 占用，先把它删除再说吧（单击右键删除）！

  (2).关闭当前文档，默认是【Ctrl+F4】，现改为跟Eclipse一样，即【Ctrl+W】

  Main menu -> Window -> Active Tool Window -> Close Active Tab -> 设置为 “Ctrl+F4”;

  Main menu -> Window -> Editor -> Close -> 设置为 “Ctrl+W”;

2.设置IDE皮肤主题

 > Theme -> 选择“Alloy.IDEA Theme”

  或者在setting中搜索theme可以改变主题，所有配色统一改变

File > settings > build.excution

每次打开python控制台时自动执行代码

> console > pyconsole

import sys

# print('Python %s on %s' % (sys.version, sys.platform))

sys.path.extend([WORKING\_DIR\_AND\_PYTHON\_PATHS])

import os

print('current workdirectory : ', os.getcwd() )

import numpy as np

import scipy as sp

import matplotlib as mpl

如果安装了ipython，则在pyconsole中使用更强大的ipython

> console

选中use ipython if available

这样每次打开pyconsole就会打开ipython

Note: 在virtualenv中安装ipython: (ubuntu\_env) pika:/media/pika/files/mine/python\_workspace/ubuntu\_env$pip install ipython

File > settings > Languages & Frameworks

如果在项目设置中开启了django支持，打开python console时会自动变成打开django console，当然如果不想这样就关闭项目对django的支持：

如果打开支持就会在 settings > build.excution > console下多显示一个django console:

Django console设置如下

import sys

print('Python %s on %s' % (sys.version, sys.platform))

import django

print('Django %s' % django.get\_version())

sys.path.extend([WORKING\_DIR\_AND\_PYTHON\_PATHS])

if 'setup' in dir(django): django.setup()

import django\_manage\_shell; django\_manage\_shell.run(PROJECT\_ROOT)

File > settings > Project : initial project

project dependencies > LDA > project depends on these projects > 选择sim\_cluster就可以在LDA中调用sim\_cluster中的包

**pycharm环境和路径配置**

**python解释器路径**

**python项目解释器路径**

用于配置python项目执行的python路径

比如，有的项目是运行的是系统python2.7下的环境；有的是3.4；有的项目使用的是virtualenv的python环境[[python虚拟环境配置](http://blog.csdn.net/pipisorry/article/details/47008981) - pycharm中的项目配置]

在pycharm > file > settings > project:pythonworkspace > project interpreter > 选择对应项目 > project interpreter中指定python解释器

pycharm中运行configuration有一个选项add content roots to pythonpath

选中后sys.path中会多一整个项目project的路径/media/pika/files/mine/python\_workspace，里面的目录就被当成包使用，这样就可以通过from SocialNetworks.SocialNetworks引入不是python包的目录中的文件了。

不过最好使用sys.path.append(os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0],"../.."))来添加，这样在pycharm外也可以运行不出错 。

**pycharm中进行python包管理**

pycharm中的项目中可以包含package、目录（目录名可以有空格）、等等

目录的某个包中的某个py文件要调用另一个py文件中的函数，首先要将目录设置为source root，这样才能从包中至上至上正确引入函数，否则怎么引入都出错：

SystemError: Parent module '' not loaded, cannot perform relative import

Note:目录 > 右键 > make directory as > source root

**python脚本解释路径**

ctrl + shift + f10 / f10 执行python脚本时

当前工作目录cwd为run/debug configurations 中的working directory

可在edit configurations > project or defaults中配置

**console执行路径和当前工作目录**

python console中执行时

cwd为File > settings > build.excution > console > pyconsole中的working directory

并可在其中配置

**pycharm配置os.environ环境**

pycharm中os.environ不能读取到terminal中的系统环境变量

pycharm中os.environ不能读取.bashrc参数

使用pycharm，无论在python console还是在module中使用os.environ返回的dict中都没有~/.bashrc中的设置的变量，但是有/etc/profile中的变量配置。然而在terminal中使用python，os.environ却可以获取~/.bashrc的内容。

解决方法1：

在~/.bashrc中设置的系统环境只能在terminal shell下运行[Spark](http://lib.csdn.net/base/spark" \o "Apache Spark知识库" \t "_blank)程序才有效，因为.bashrc is only read for interactive shells.

如果要在当前用户整个系统中都有效（包括pycharm等等IDE），就应该将系统环境变量设置在~/.profile文件中。如果是设置所有用户整个系统，修改/etc/profile或者/etc/environment吧。

如SPARK\_HOME的设置[[Spark：相关错误总结](http://blog.csdn.net/pipisorry/article/details/52916307)]

解决方法2：在代码中设置，这样不管环境有没有问题了

# spark environment settings

import sys, os

os.environ['SPARK\_HOME'] = conf.get(SECTION, 'SPARK\_HOME')

sys.path.append(os.path.join(conf.get(SECTION, 'SPARK\_HOME'), 'python'))

os.environ["PYSPARK\_PYTHON"] = conf.get(SECTION, 'PYSPARK\_PYTHON')

os.environ['SPARK\_LOCAL\_IP'] = conf.get(SECTION, 'SPARK\_LOCAL\_IP')

os.environ['JAVA\_HOME'] = conf.get(SECTION, 'JAVA\_HOME')

os.environ['PYTHONPATH'] = '$SPARK\_HOME/python/lib/py4j-0.10.3-src.zip:$PYTHONPATH'

**pycharm配置第三方库代码自动提示**

**Pycharm实用拓展功能**

**pycharm中清除已编译.pyc中间文件**

选中你的workspace > 右键 > clean python compiled files

还可以自己写一个清除代码

**pycharm设置外部工具**

[[python小工具](http://blog.csdn.net/pipisorry/article/details/46754515)]针对当前pycharm中打开的py文件对应的目录删除其中所有的pyc文件。如果是直接运行（而不是在下面的tools中运行），则删除E:\mine\python\_workspace\WebSite目录下的pyc文件。

**将上面的删除代码改成外部工具**

PyCharm > settings > tools > external tools > +添加

Name: DelPyc

program: $PyInterpreterDirectory$/python Python安装路径

Parameters: $ProjectFileDir$/Oth/Utility/DelPyc.py $FileDir$

Work directory: $FileDir$

Note:Parameters后面的 $FileDir$参数是说，DelPyc是针对当前pycharm中打开的py文件对应的目录删除其中所有的pyc文件。

**之后可以通过下面的方式直接执行**

Note:再添加一个Tools名为DelPycIn

program: Python安装路径，e.g.     D:\python3.4.2\python.exe

Parameters: E:\mine\python\_workspace\Utility\DelPyc.py

Work directory 使用变量 $FileDir$

参数中没有$FileDir$，这样就可以直接删除常用目录r'E:\mine\python\_workspace\WebSite'了，两个一起用更方便

**代码质量**

当你在打字的时候，PyCharm会检查你的代码是否符合PEP8。它会让你知道，你是否有太多的空格或空行等等。如果你愿意，你可以配置PyCharm运行pylint作为外部工具。

**python2转python3最快方式**

/usr/bin/2to3 -wn $FileDir$

这样在pycharm中打开某个文件，右键external tools > py2topy3就可以瞬间将当前文件所在目录下的所有py2转换成py3，是不是很机智！

1. 7

代码快速格式化，让代码简洁规整：**Ctrl + Alt + L**

1. 8

自动缩进：**Ctrl + Alt + I**

1. 9

 缩进/取消缩进：**Tab / Shift + Tab**

1. Ctrl+鼠标点击函数名，查看内置函数介绍。
2. 6

非常实用的一个命令，编码过程中快速复制当前行，大大提升编码效率：**Crtl+D**

4

快速查看文档：**Ctrl + q**

3

行注释/取消行注释：**Ctrl+/**

块注释：**Ctrl+Shift+/**

1. 1

代码快速运行：**Ctrl+Shift+F10**

1. 2

编辑代码的时候经常的要换下一行，但是光标没有在行末，可以用这个命令直接换行：**Shift+Enter**