前言

Python的编辑器有很多比如`Visual Studio Code`、`Sublime Text`、`Atom`、`jupyter notebook`等等

但是功能最强使用最多的还是`PyCharm`，同时也是我使用的最顺手的一款IDE。它是由JetBrains打造的一款功能强大的Python IDE。比如代码调试、项目管理、代码跳转、智能提示、单元测试、版本控制等等。具有跨平台性，无论Windows、Mac、Linux都可以使用。

一款好的IDE能够帮我们快速入门一个新的语言，对于初次使用PyCharm的同学，可能无从下手，这也是我编写这份手册的初衷，希望能够帮助到大家更效率的使用PyCharm。

作者：阿亮。 公众号：Python极客专栏

> 本使用手册为V1.0.0版本（2021年3月17日）
>
> 操作系统Window10
>
> PyCharm版本 2020.3.3
>
> Python解释器版本：3.9.2

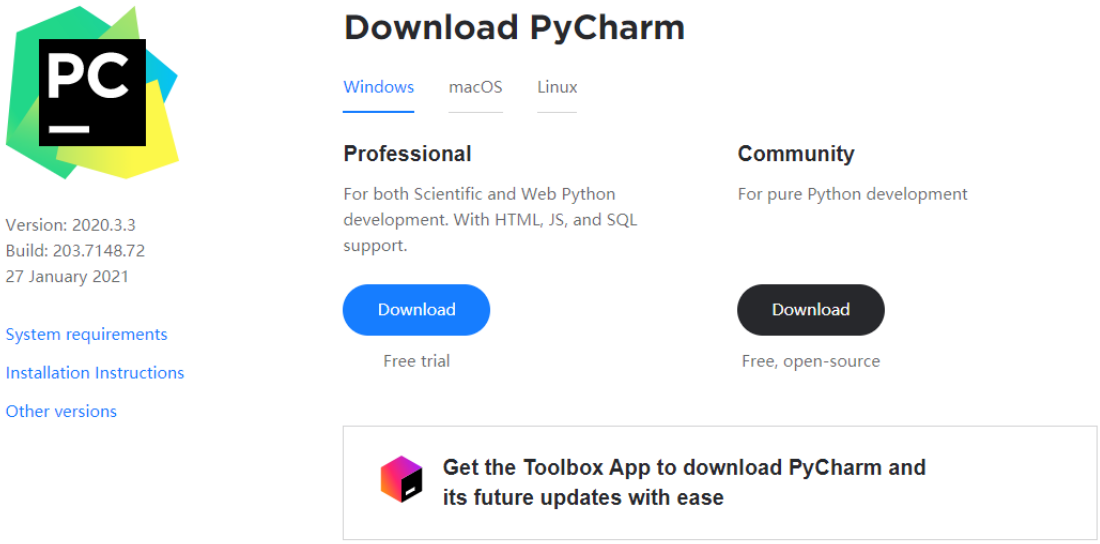
PyCharm的安装

当然时间2021年3月11日，官方最新版本为`2020.3.3`

安装PyCham系统要求：

- Microsoft Windows 10、8的64位版本
- 最低2 GB RAM，建议8 GB RAM
- 2.5 GB硬盘空间，建议使用SSD
- 最低1024x768屏幕分辨率
- Python 2.7或Python 3.5或更高版本

官方下载地址：https://www.jetbrains.com/pycharm/download



从上图可以看到，有两个版本分别是`Professional`和`Community`，他们的区别如下表所示。

	PyCharm专业版 (Professional)	PyCharm社区版本 (Community)
智能Python编辑器	支持	支持
图形化调试器	支持	支持
导航和重构	支持	支持
代码检查	支持	支持
VCS支持	支持	支持
科学工具	支持	
Web开发	支持	
Python Web框架	支持	
Python Profiler（性能分析）	支持	
远程开发功能	支持	
数据库和SQL支持	支持	

社区版本是免费的，基本满足日常开发需求

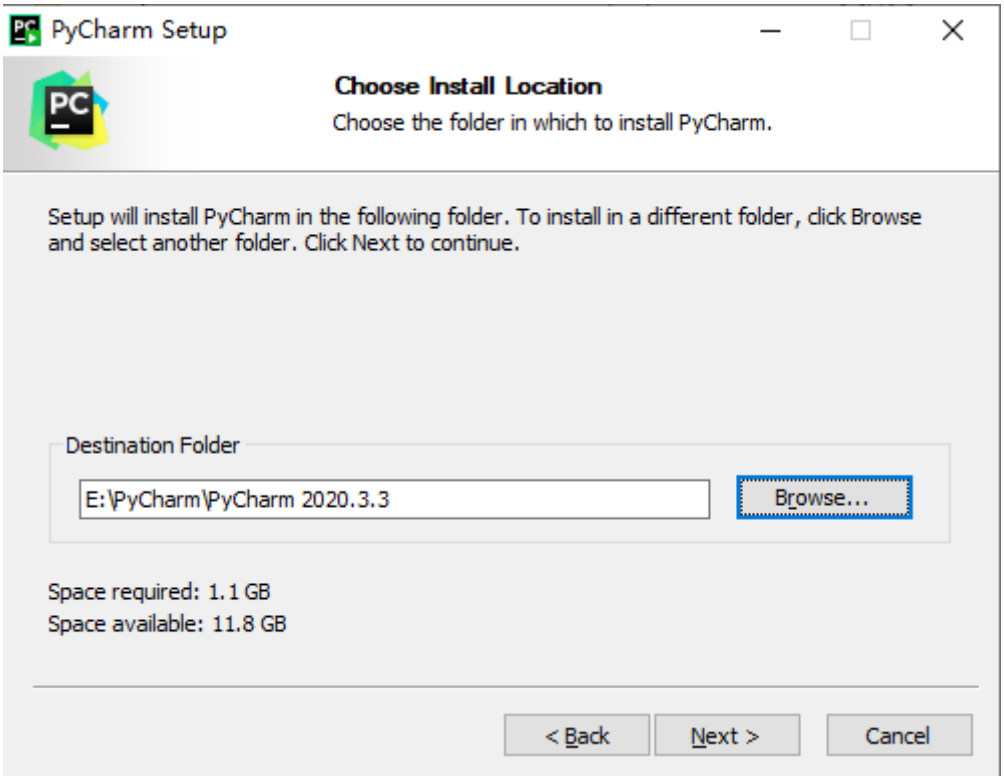
如果下载PyCharm专业版，即`Professional`。但是只能免费试用30天，可以选择以下几种激活方式

- 购买正版
- 网上也有激活方法（参考：https://shimo.im/docs/9wGqjpDWwxYcrCkH）
- 学生/老师可申请免费：申请地址：https://www.jetbrains.com/community/education/

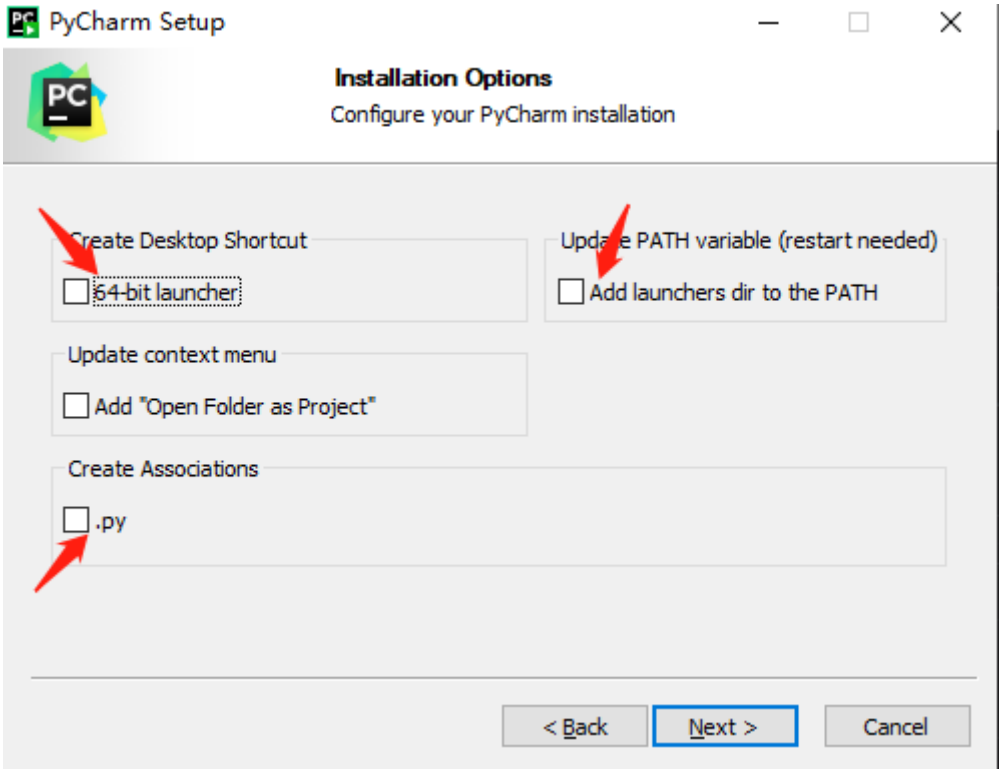
PyCharm下载

直接点击`Download`就可以进行下载了。下载完成之后双击打开`pycharm-professional-2020.3.3.exe`

然后点击`Next`，中间记得选择安装路径， 不建议安装在C盘，如果你的C盘很大很大，那完全可以。



然后点击`Next`会看到如下界面，**建议勾选箭头所指的选项**

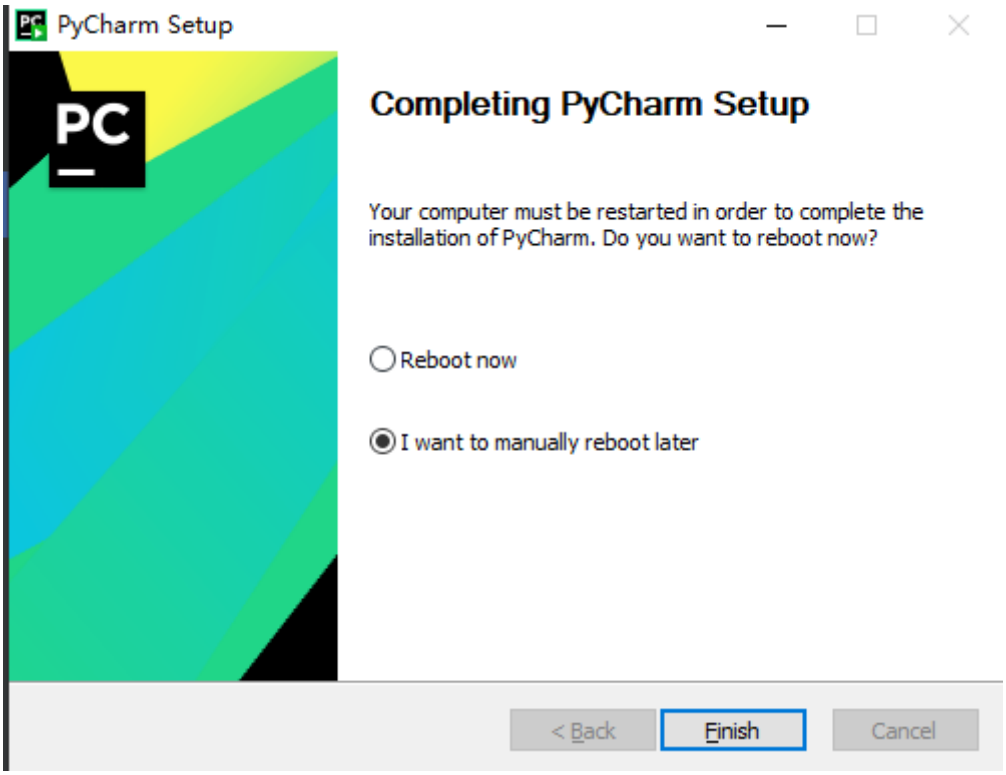


选项翻译如下：

- create desktop shortcut：创建桌面快捷方式
- update path variable(restart needed)：更新路径变量将路径添加到环境变量中
- update context menu ： 更新通知
- create associations：关联.py文件，双击都是以pycharm打开

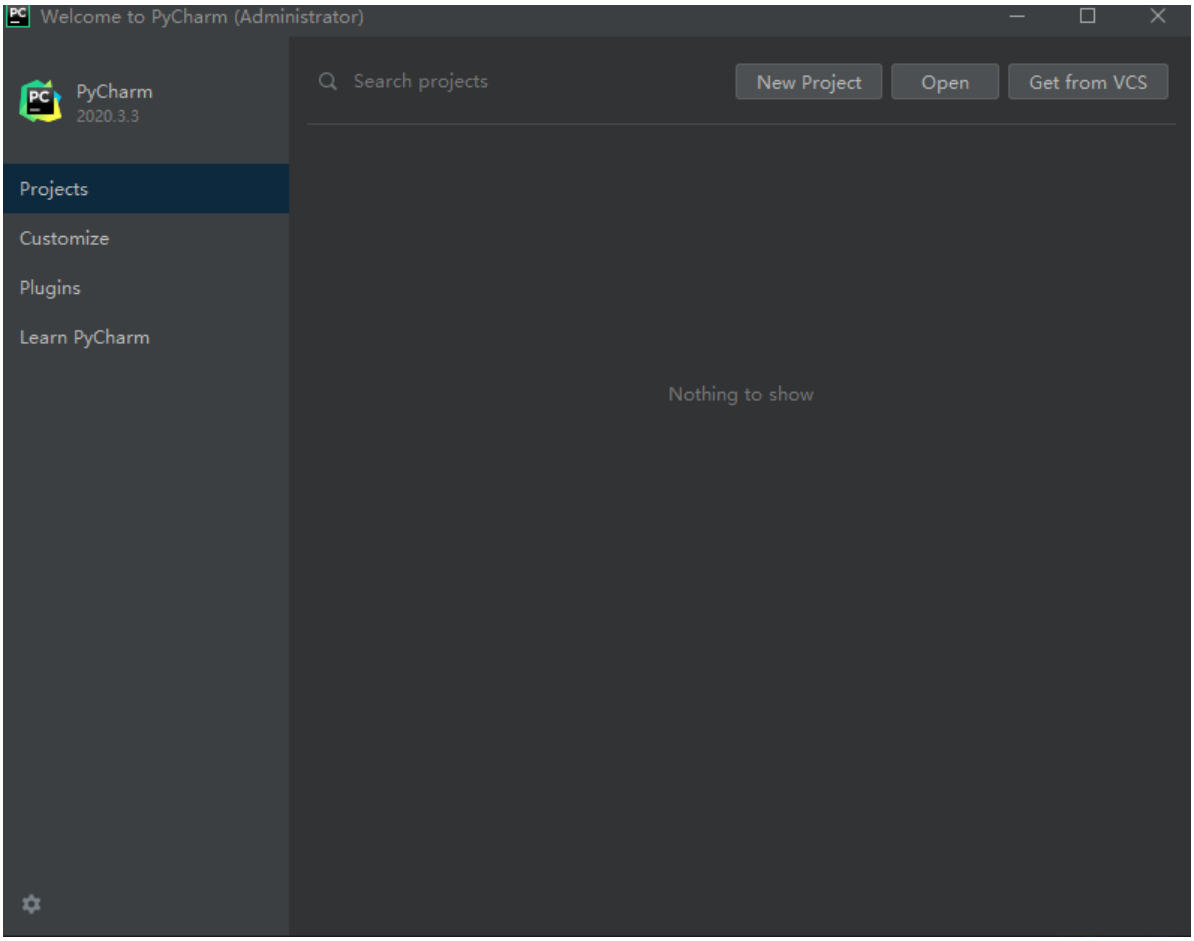
之后呢，就是一路`Next` `install`即可。这里就不一一截图了。

安装完成之后界面如下（最新版本与之前的不太一样）



提示重启，不重启也没啥影响。

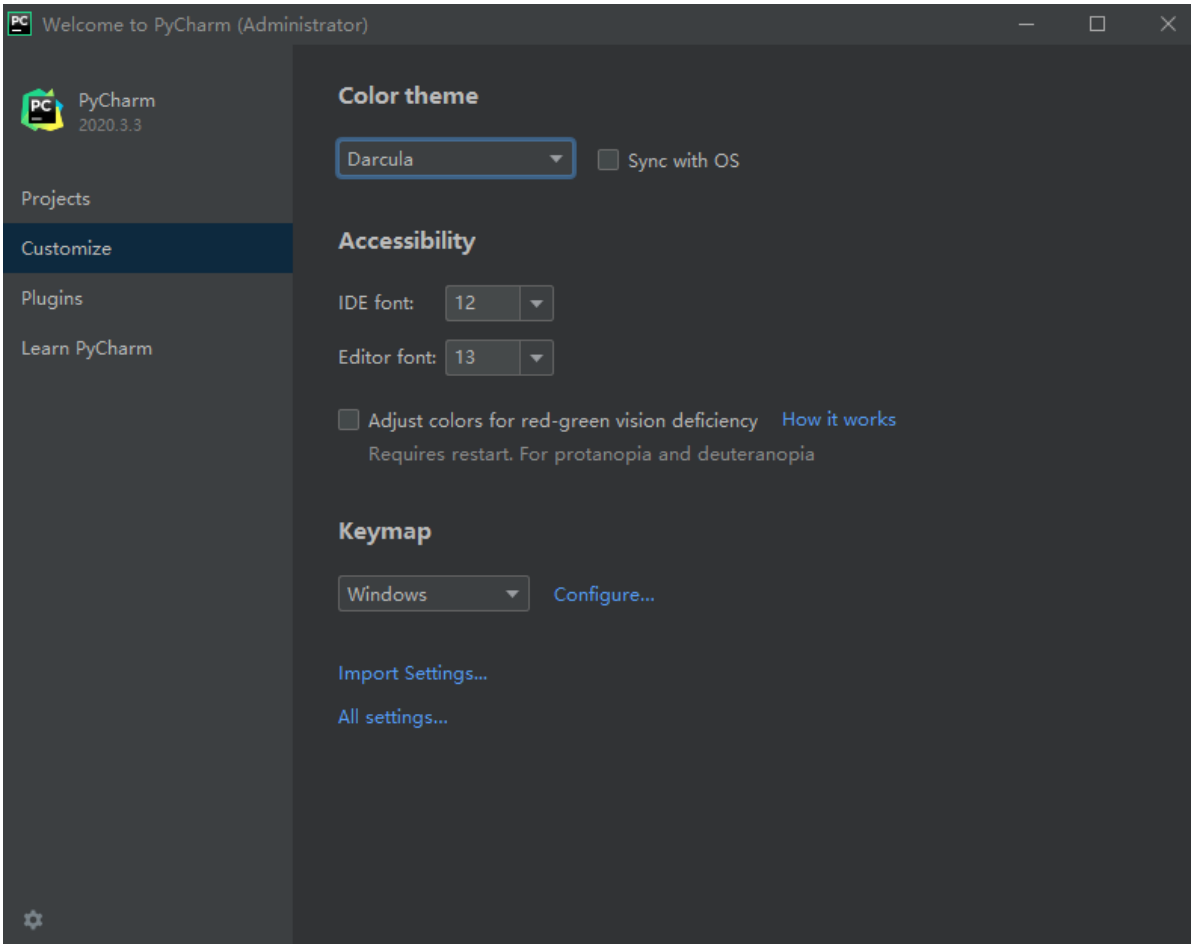
Project



- **New Project**：创建项目
- **Open**：导入并打开本地项目，
- **Get from VCS**：配置GitHub/SVN仓库地址

Customize

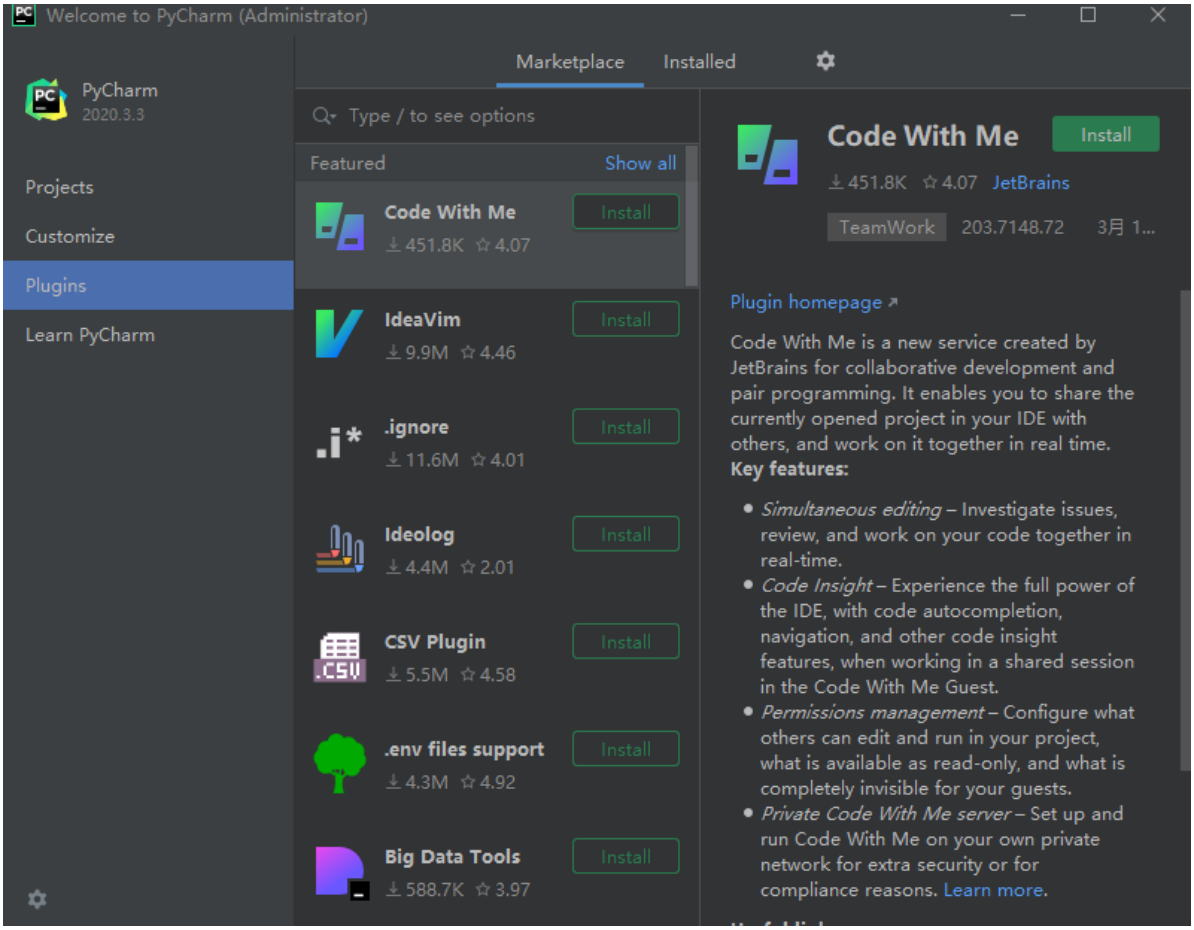
这里面的配置在后面的文档中详细介绍，这里只是做个介绍。



- **Color theme**：主题风格，以我本机为例，默认有`Darcula`、`Intellij Light`、`windows 10 Light`、`High contrast`
- **IDE font**：修改界面字体大小，建议默认12的就可以了
- **Editor font:** ：控制代码字体大小以及输出面板中的字体大小
- **Adjust ....**：针对红绿色盲用户设计的。
- **Keymap:** 快捷键风格，有Windos、EMacs、Sublime Text 风格

Plugins

显示安装的插件，也可以在这个界面进行在线安装。



Learn PyCharm

官方教程，英文好的同学可以看一下

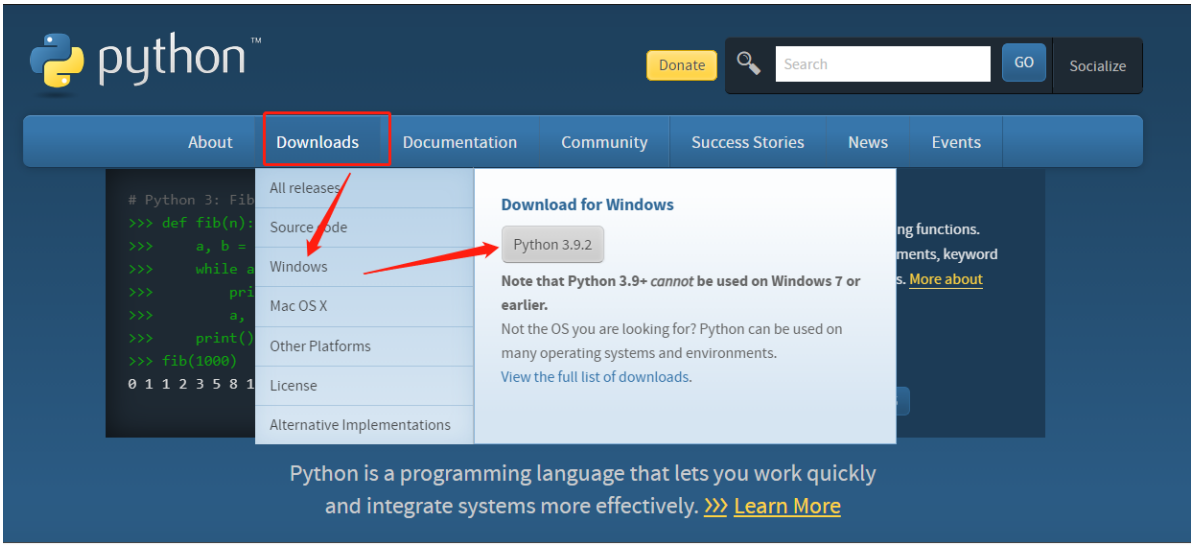
环境配置

这一篇是给新手朋友准备的，如果你本地已经安装配置，请自行跳过

Python代码运行，需要解释器，Python解释器下载地址：https://www.python.org/

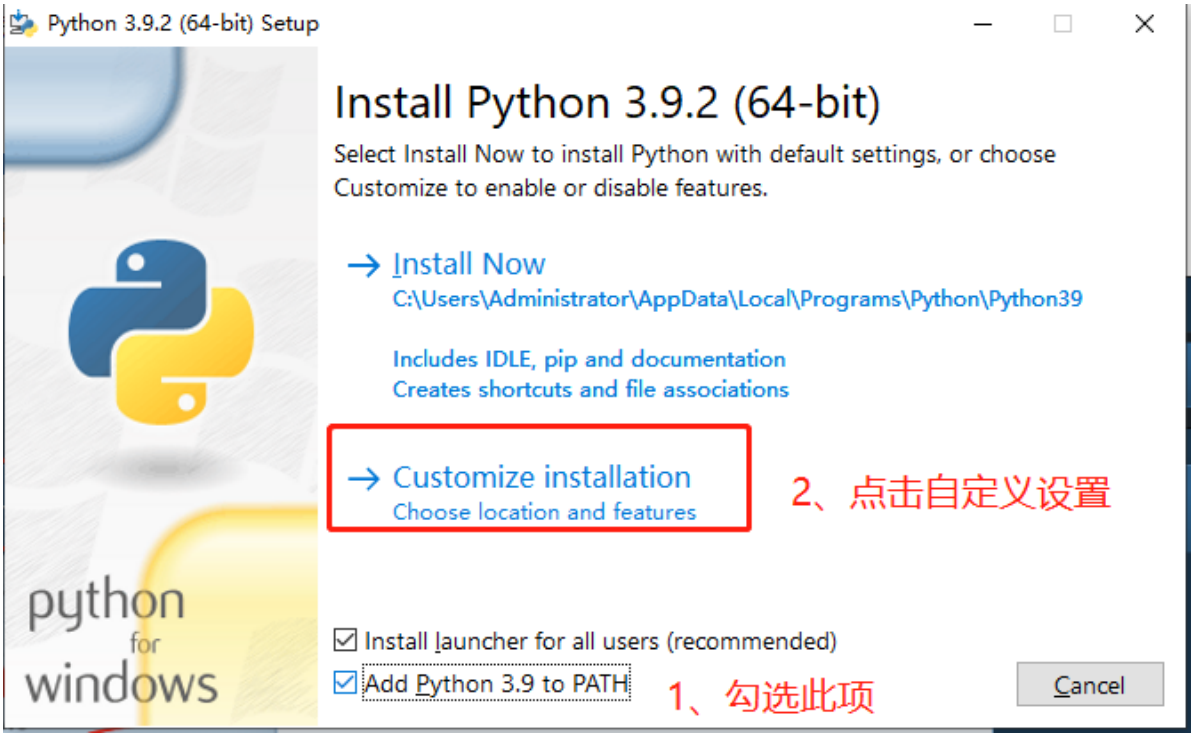
鼠标悬停在`Downloads`上，然后选择对应的操作系统，点击版本号即可。我这里以`Python 3.9.2`为例

> 官网下载较慢，可以在公众号：Python极客专栏，后台回复【python392】获取安装包。

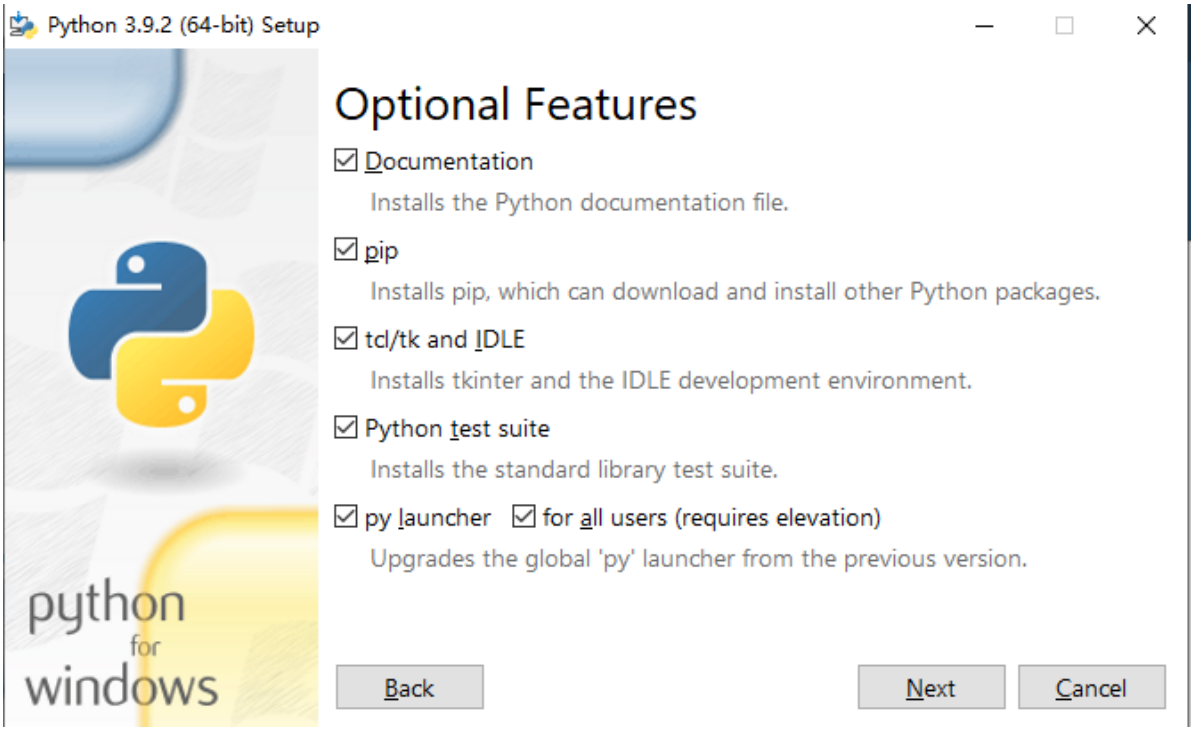


下载完毕，双击打开（**建议以管理员身份运行**）。

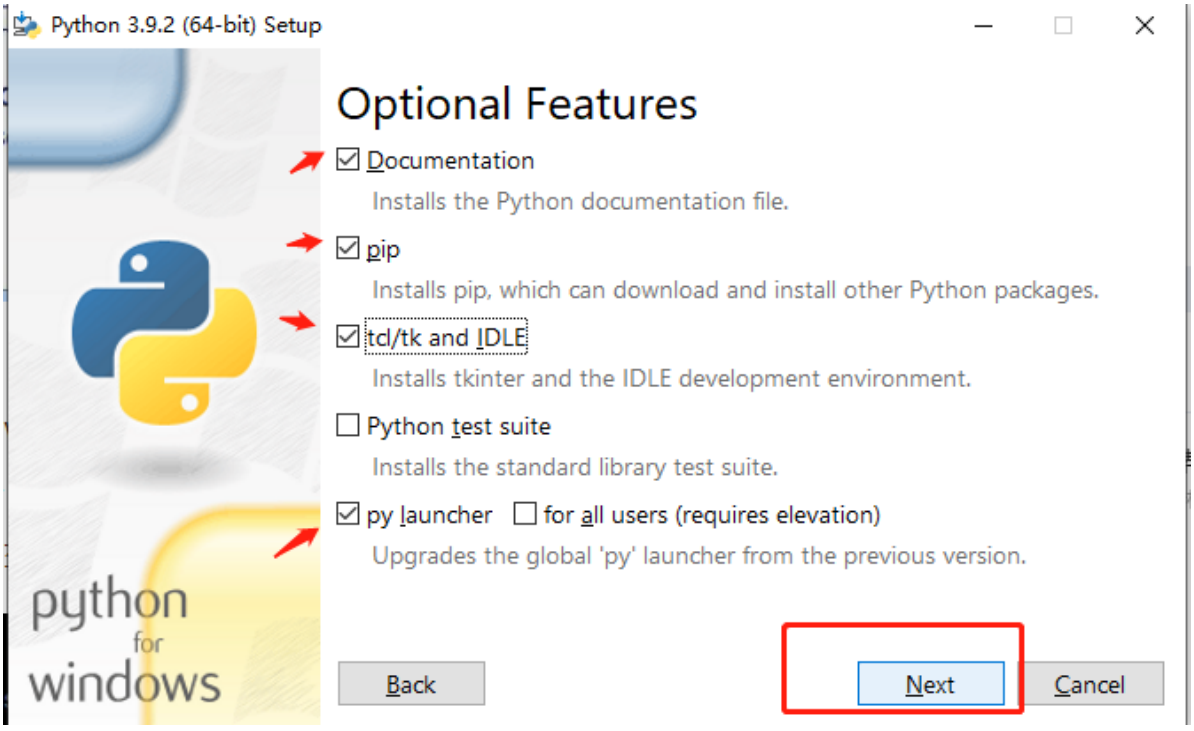
不建议按照默认的方式安装，参考下图



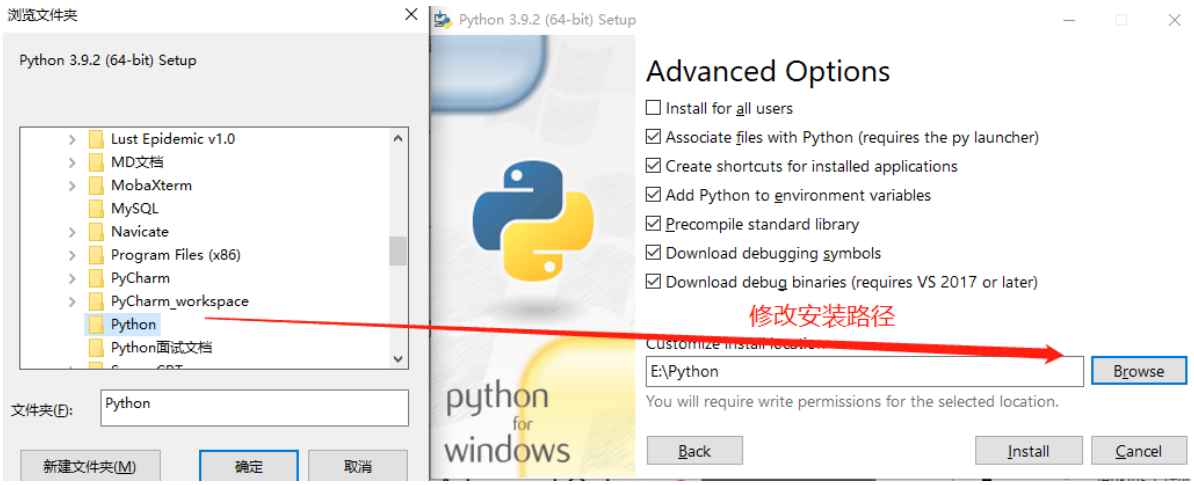
点击之后出现如下界面



- **Documentation：** 安装文献资料（选装）
- **pip：**安装pip（**必选**）
- **tcl/tk and IDLE：**安装tkinter和IDLE开发环境（选装）
- **Python test suite：**安装测试套件（测试用，选装）
- **py launcher：**Python启动器（**必选**）
- **for all users：**本机上所有用户账号登录的配置（选装）

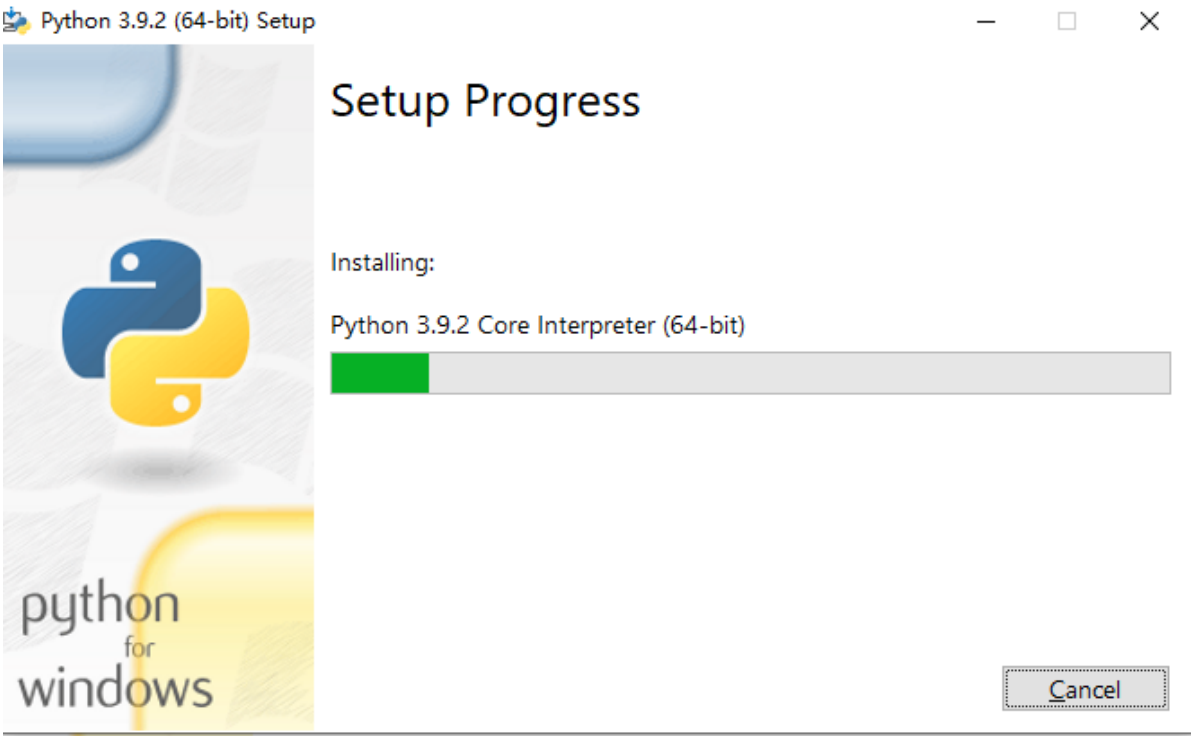


勾选需要的选项之后，点击`Next`

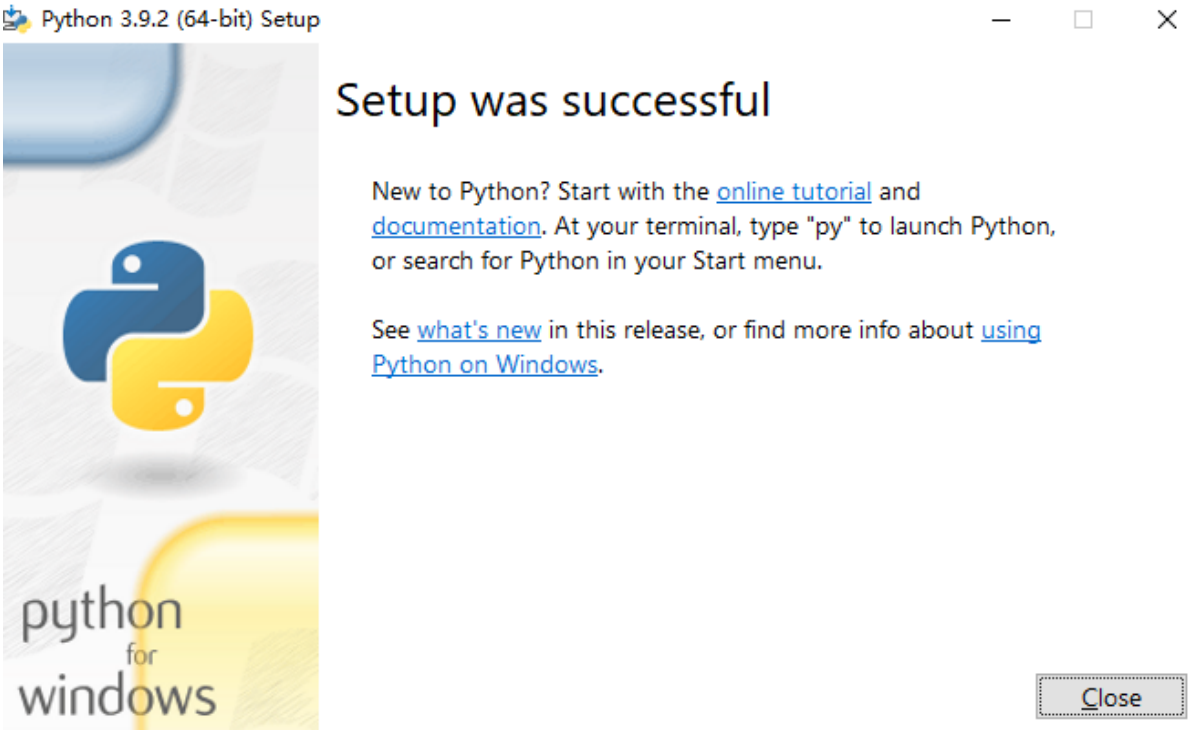


- **Install for all users**：为所有用户安装 （选装）
- **Associate files with Python**：Python文件关联 （**必选**）
- **Create shortcuts for Installed application**：为已安装的应用程序创建快捷方式（**必选**）
- **Add Python to environment variables**：添加Python环境变量（**必选**）
- **Precompile standard library**：预编译python标准文件（**推荐勾选**）
- **Download debugging symbol**：下载调试符号（**推荐勾选**）
- **Download debug binaries(require VS 2017 or later)**：下载调试二进制文件（需要VS 2017或更高版本）（**推荐勾选**）

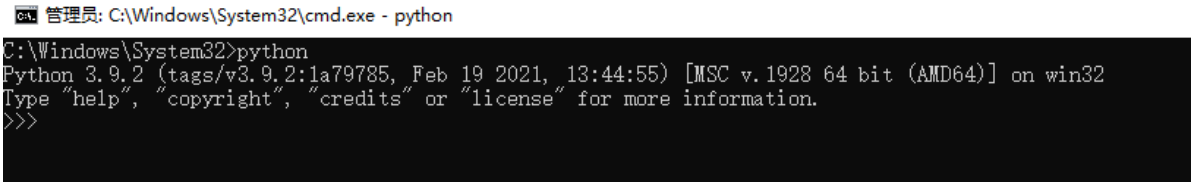
勾选完毕之后点击`Install`进行安装，安装速度可能有点慢，耐心等耐即可。



安装完毕之后如下图所示。

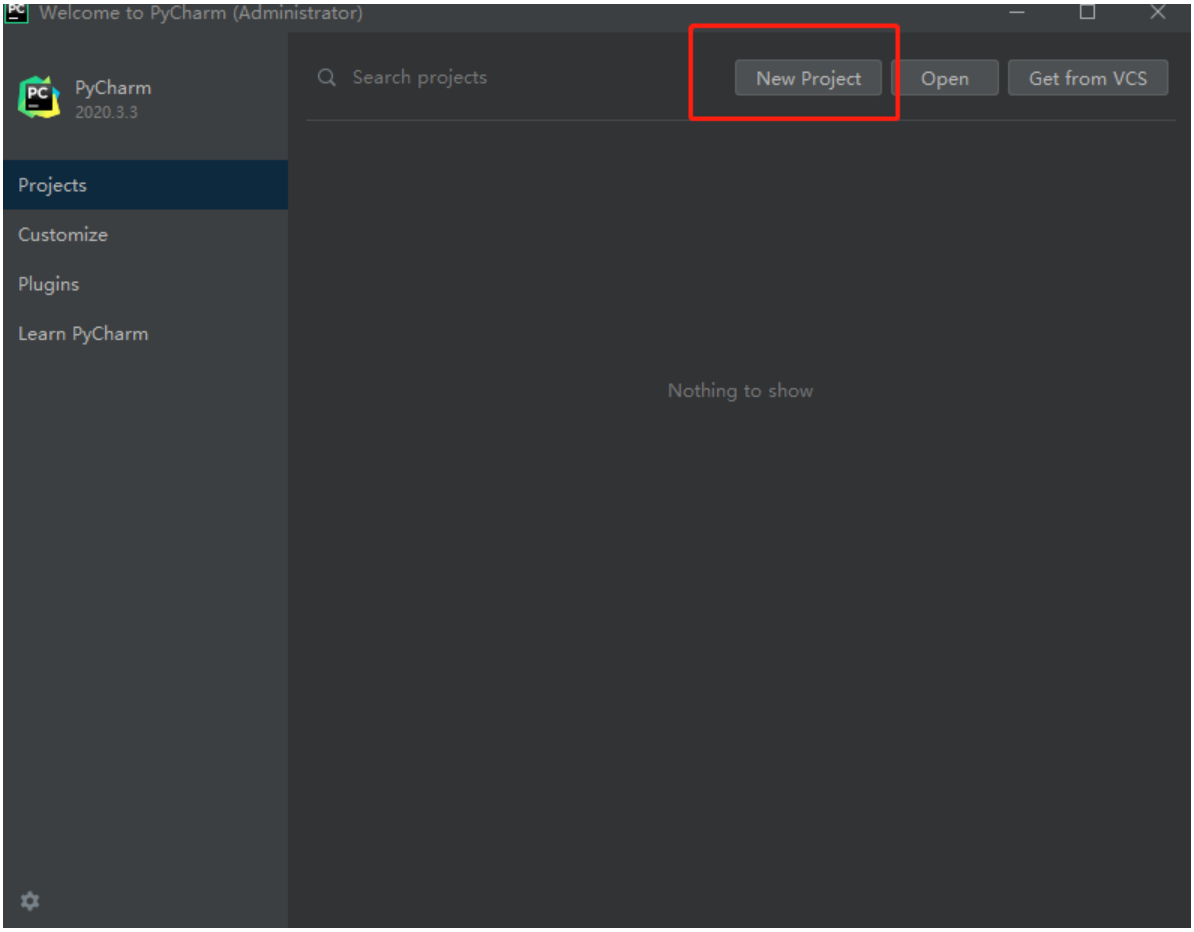


查看是否安装成功

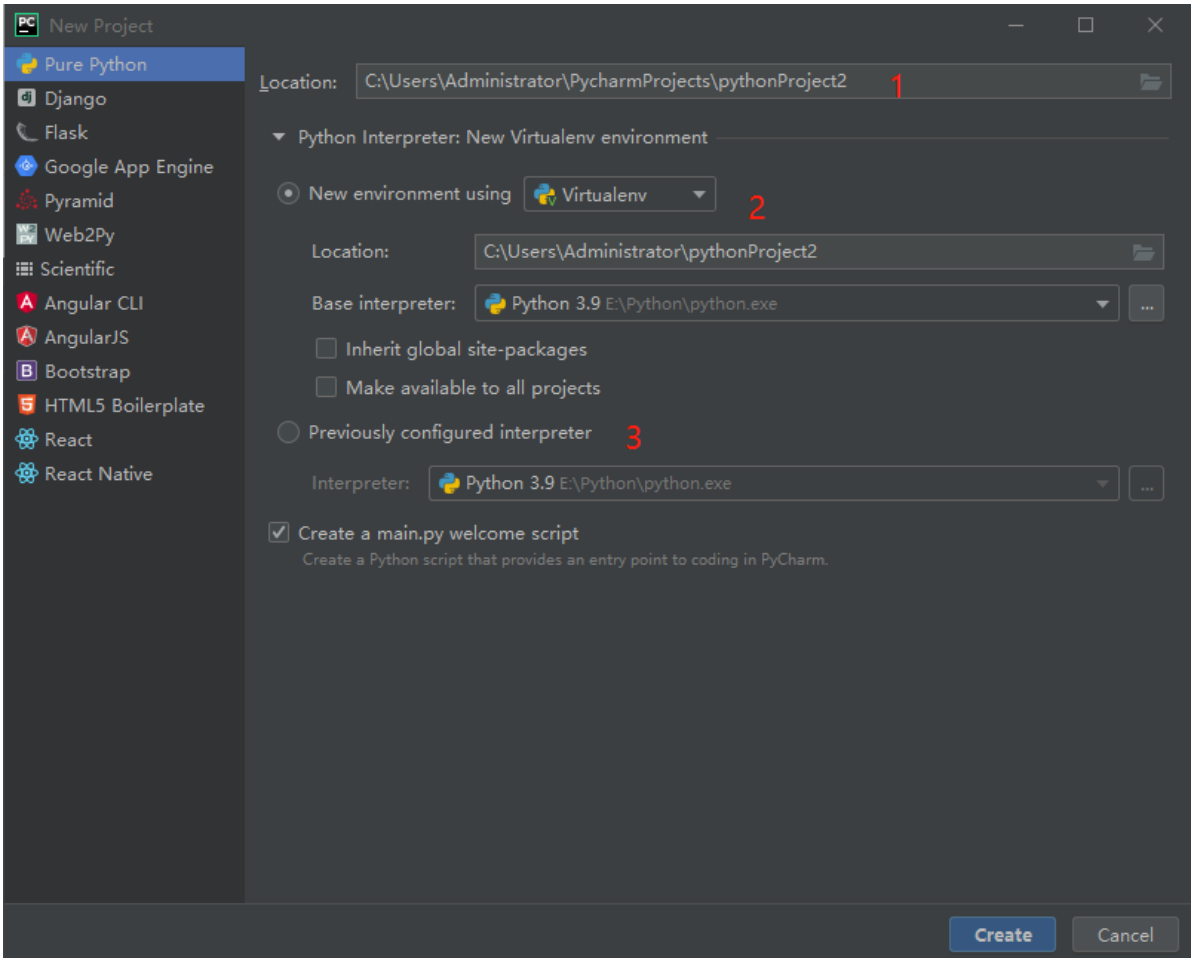


创建项目

点击`New Project`创建一个新的项目



界面如下

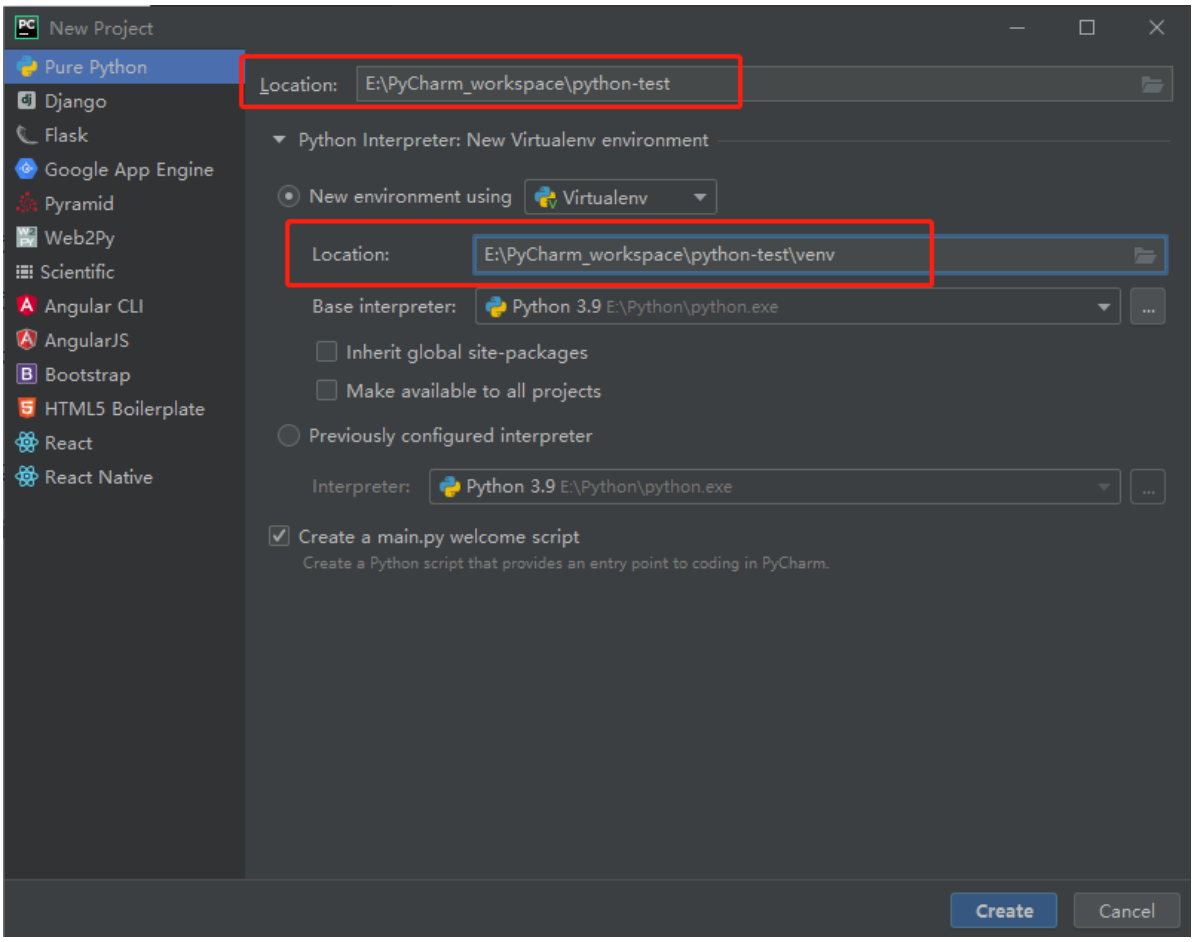


1、选择项目的存储路径

2、选择项目依赖的Python库，会在项目中创建一个`venv`的虚拟环境

3、关联本地的Python解释器，如果不想使用venv可以选择本地解释器的可执行文件（也就是我们上面安装的内容）

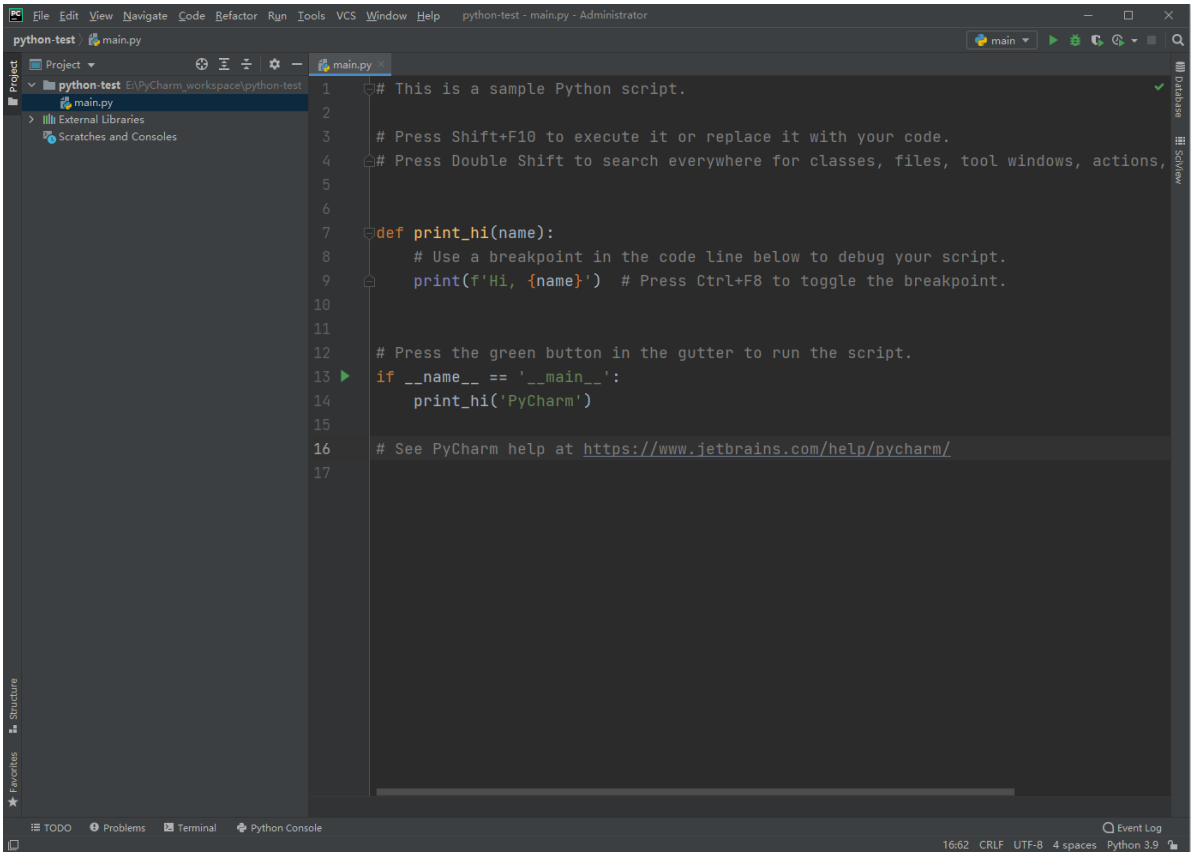
建议选择`New environment using`， 然后在`Base interpreter`中选择前面安装的Python解释器的路径。如下图所示



这样做的好处：每个项目都是独立的空间，不会存在版本依赖冲突的问题，冲分发挥了虚拟环境的灵活性。

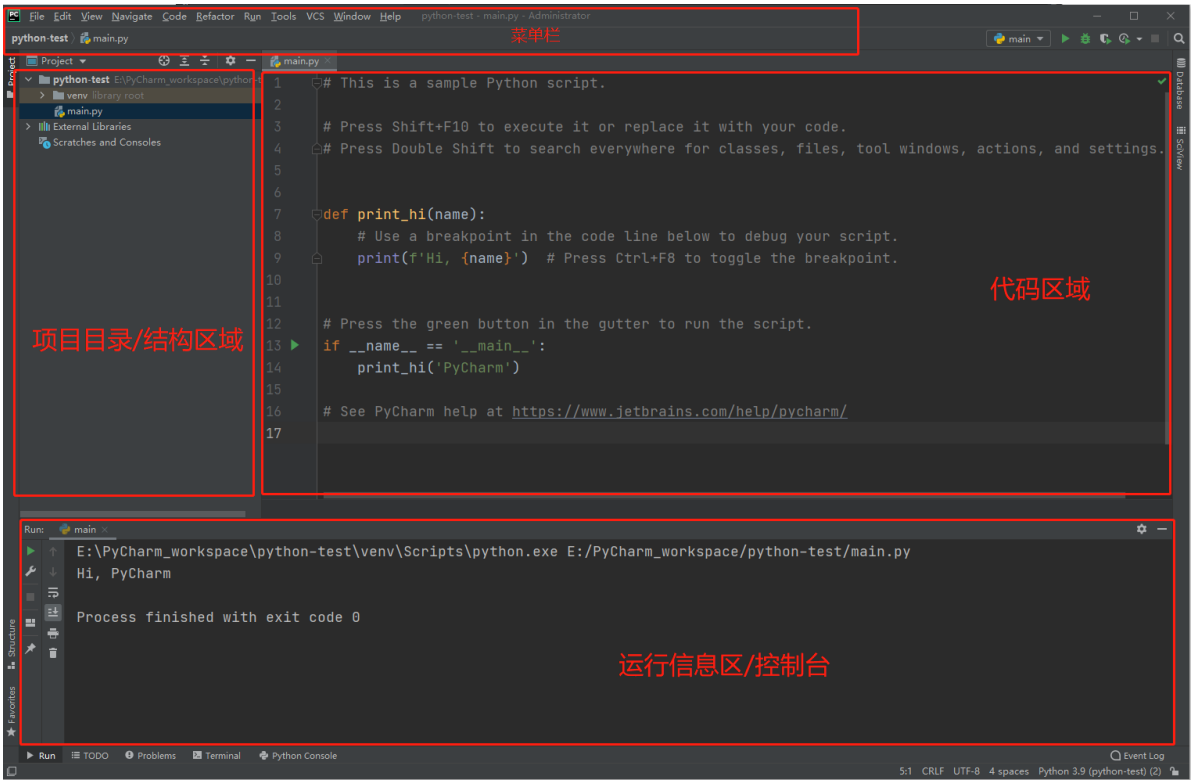
修改完毕之后，点击`Creat`，创建项目。

这样就创建了一个最初始的空项目



界面介绍

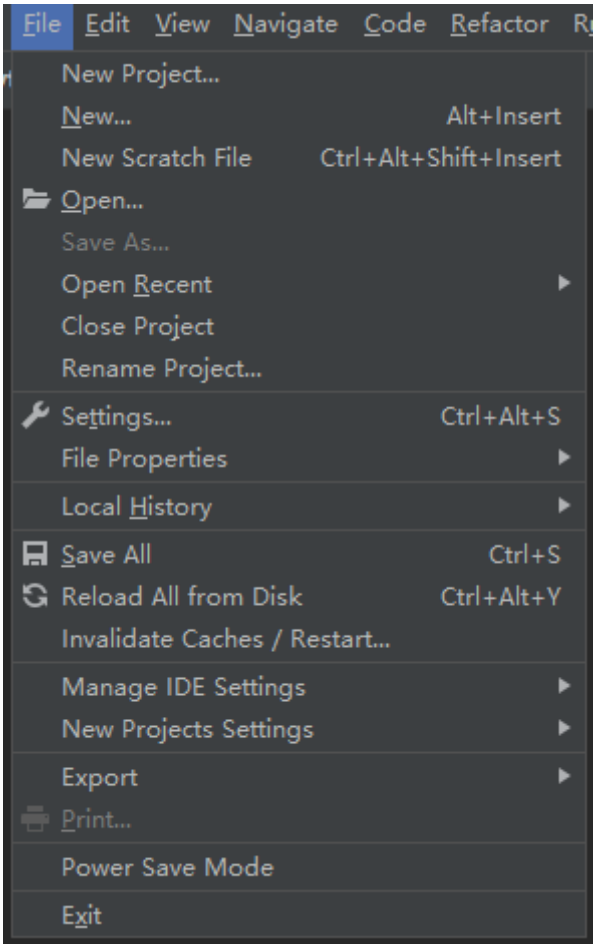
从大的方向来看PyCharm分为 **菜单栏区域 / 项目结构区域 / 代码区域 / 运行信息区**



菜单栏

> 提示：菜单栏 快捷键为`Alt + 首字母`，比如`File`的快捷键`Alt + F`，`Edit`的快捷键`Alt +E`

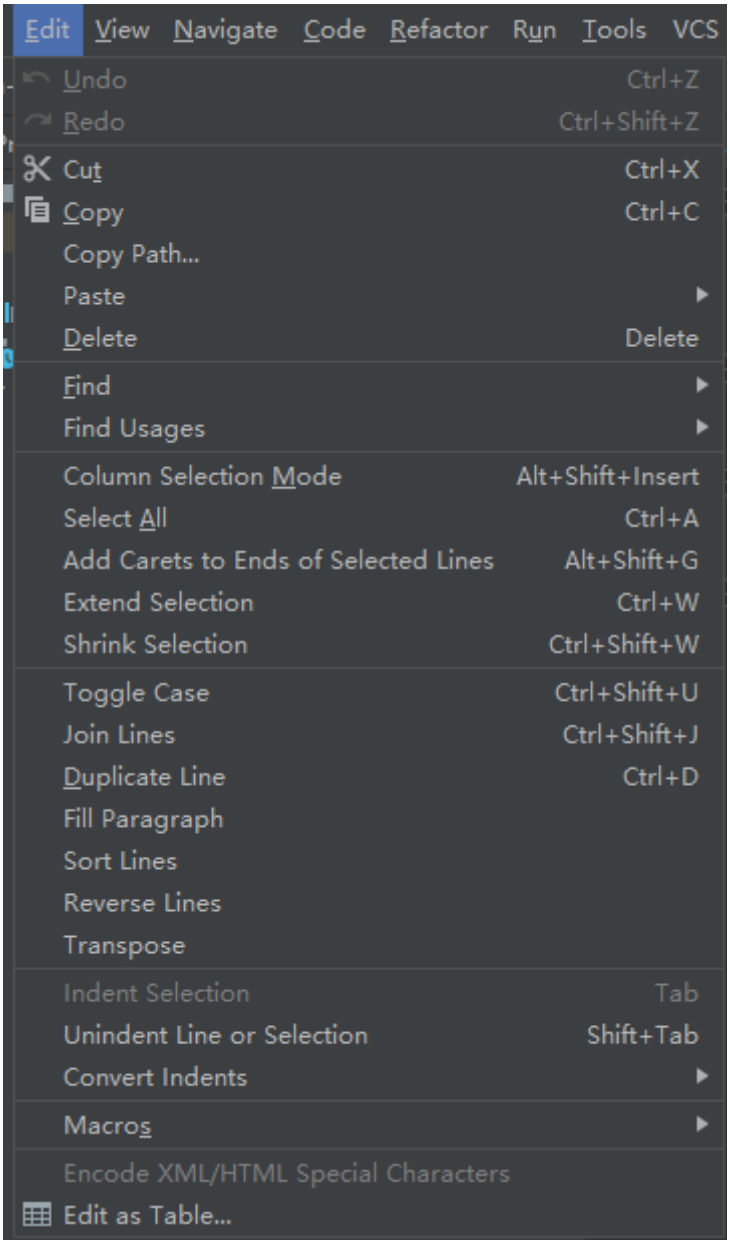
1、File（文件）



- `New Project`：创建新的项目

- `New ...`：新建一些中间件配置，如MySQL、MongoDB、DDL等以及相关驱动
- `New Scratch File`：划痕文档，也称为临时文件，可以**创建各种类型**的文件进行临时处理，在里面“打草稿”，可运行并且可调试（非常棒的一个功能，在最近的版本才有的）
- `Open`：打开项目目录
- `Save as`： 另存为
- `Close Project`：关闭项目并回答创建项目页面
- `Rename Project`：给项目重命名
- `Settings`：设置选项，重点☆☆☆☆☆
- `File Properties`：文件的相关属性，包括编码等
- `Invalidate Caches /Restart..`： 是缓存失效，并重启

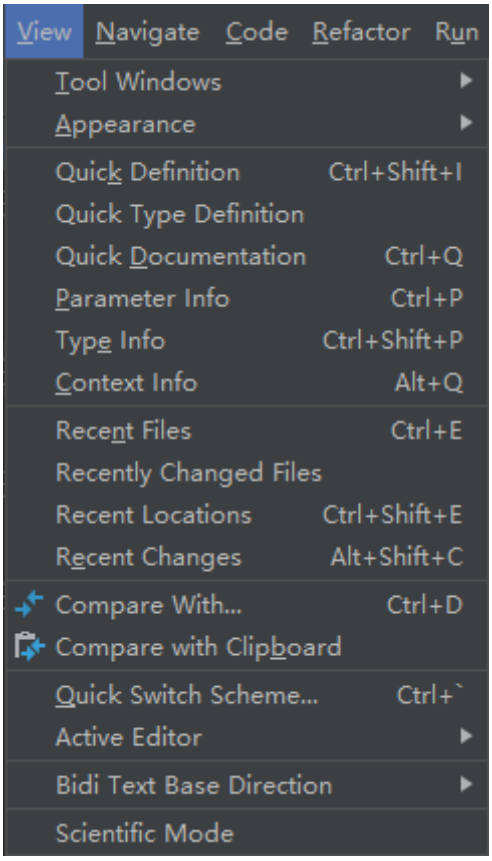
2、Edit（编辑）



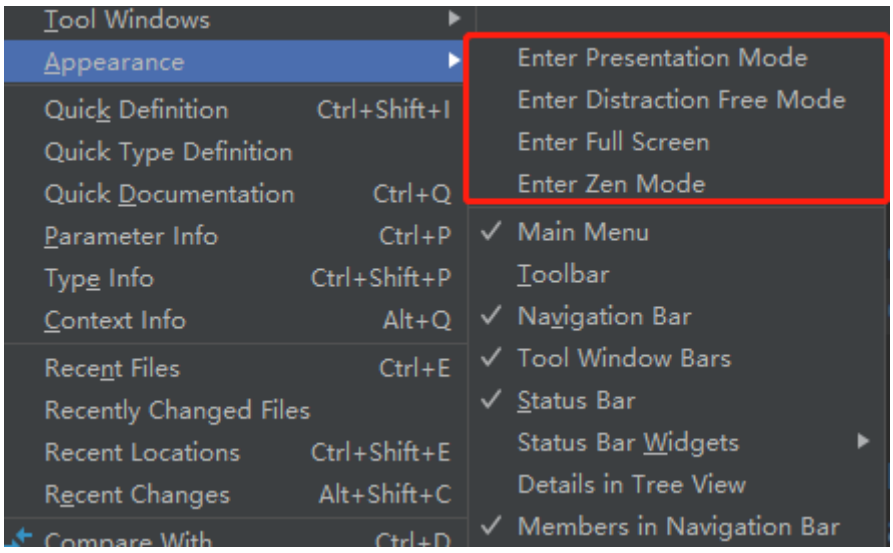
- `Find`： 编辑窗口中用的最多的就是Find选项中的，例如`Ctrl + F`文件内查找，`Ctrl +Shift + F`项目中搜索，以及`Ctrl + R`文件内替换，`Ctrl+Shift+R`全文替换（慎用！）

> windows下ctrl+shift+F快捷键如果无效，大概率是因为装了搜狗输入法，快捷键冲突导致的。只需要修改输入法中对应的快捷键即可，或者修改PyCharm的快捷键。

3、View（视图）

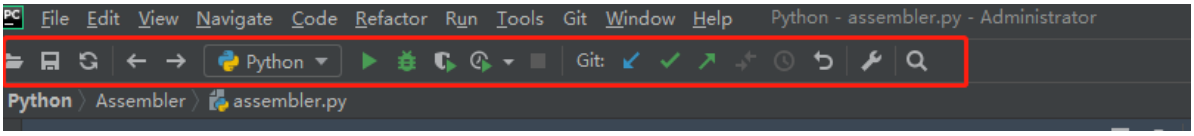


- `Tool Windows`： 工具窗口，如果主页面中某些窗口不小心关了，可以在这里面重新找到。
- `Appearance:`：外观设置，除了基本的布局调整，最强大的莫过于这四种模式（在阅读代码的时候真的很爽！）

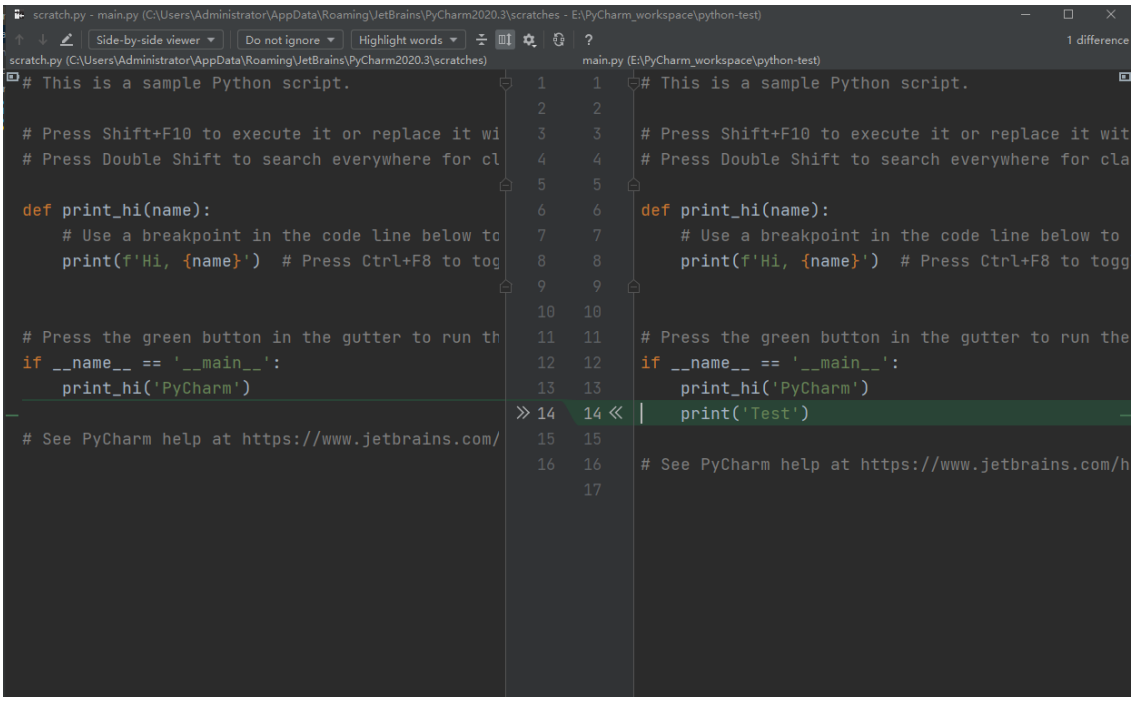


> Enter/Exit Presentation Mode：进入/退出 展示模式
>
> Enter/Exit Distraction Free Mode：进入/退出 免打扰模式
>
> Enter/Exit Full Screen：进入/退出 全屏模式
>
> Enter/Exit Zen Mode：进入/退出 禅模式（一个终极模式，包含以上3种模式）

其次`Toolbar`也是一个不错的功能，开启之后，会在菜单栏有一个导航

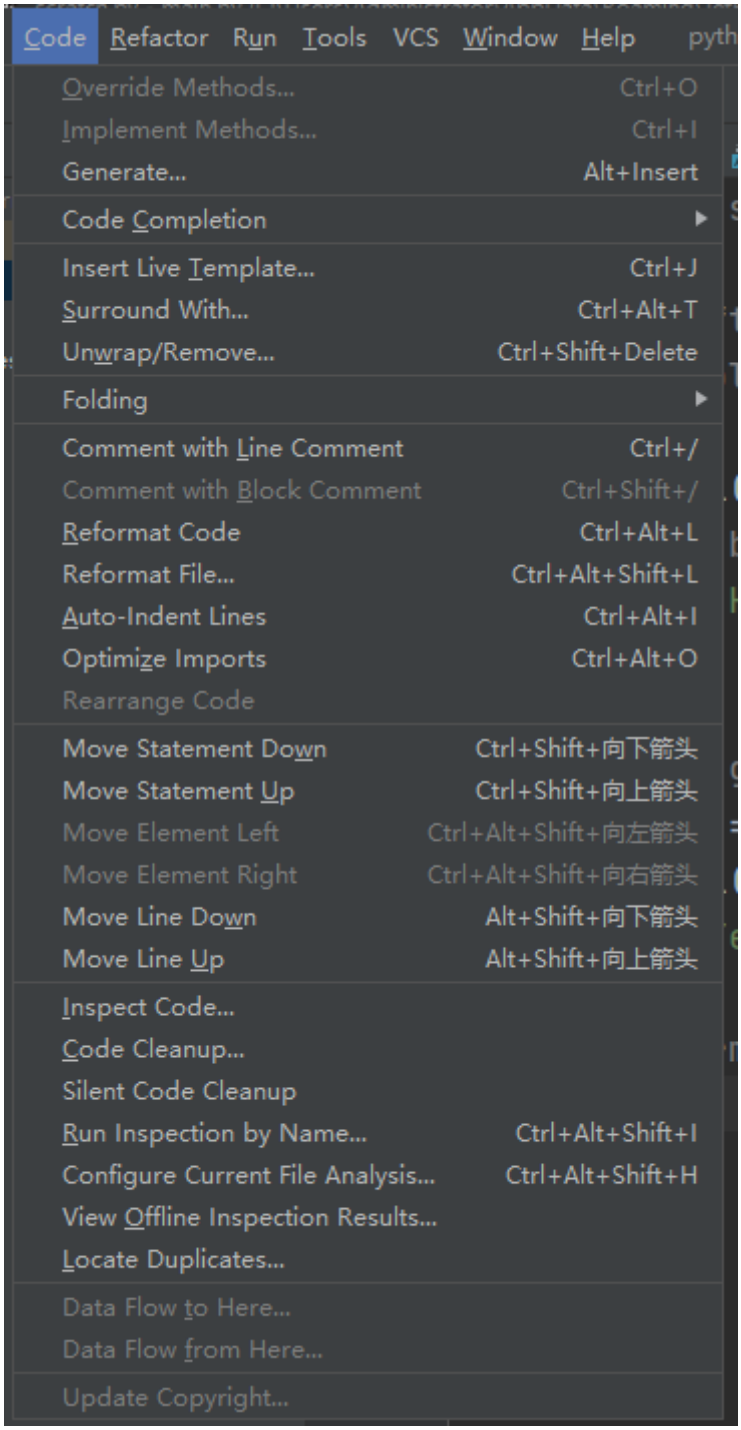


- `Recent Files:`： 最近打开的文件，快捷键`Ctrl + E`
- `Recent Locations`：最近修改的内容
- `Compare With:`：比较文件之间的差异



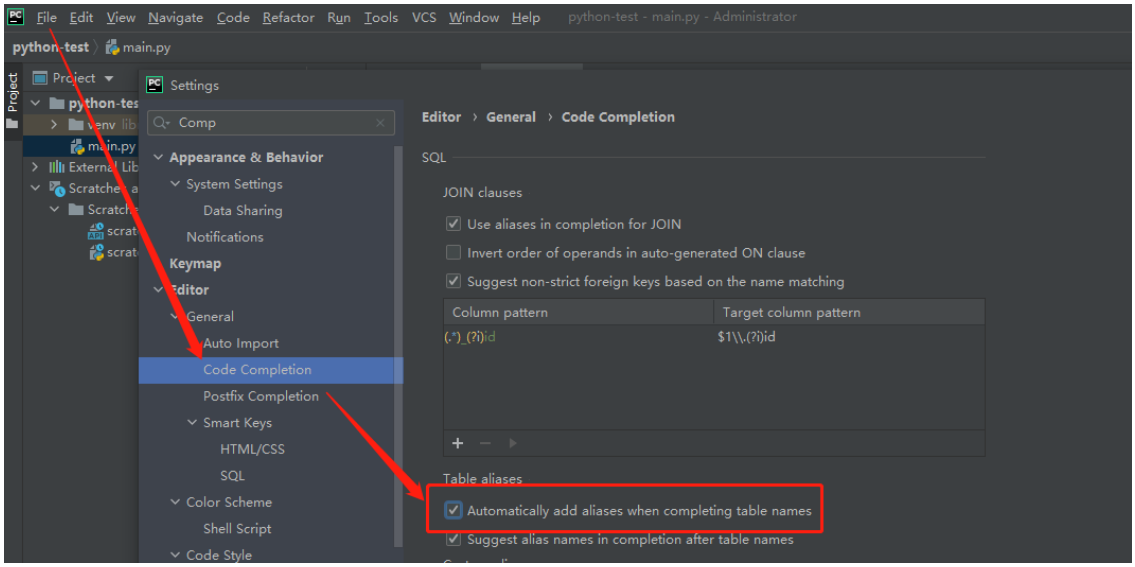
- `Compare with Cliboard:` 与剪切板上的内容做比较

4、Code（编码）

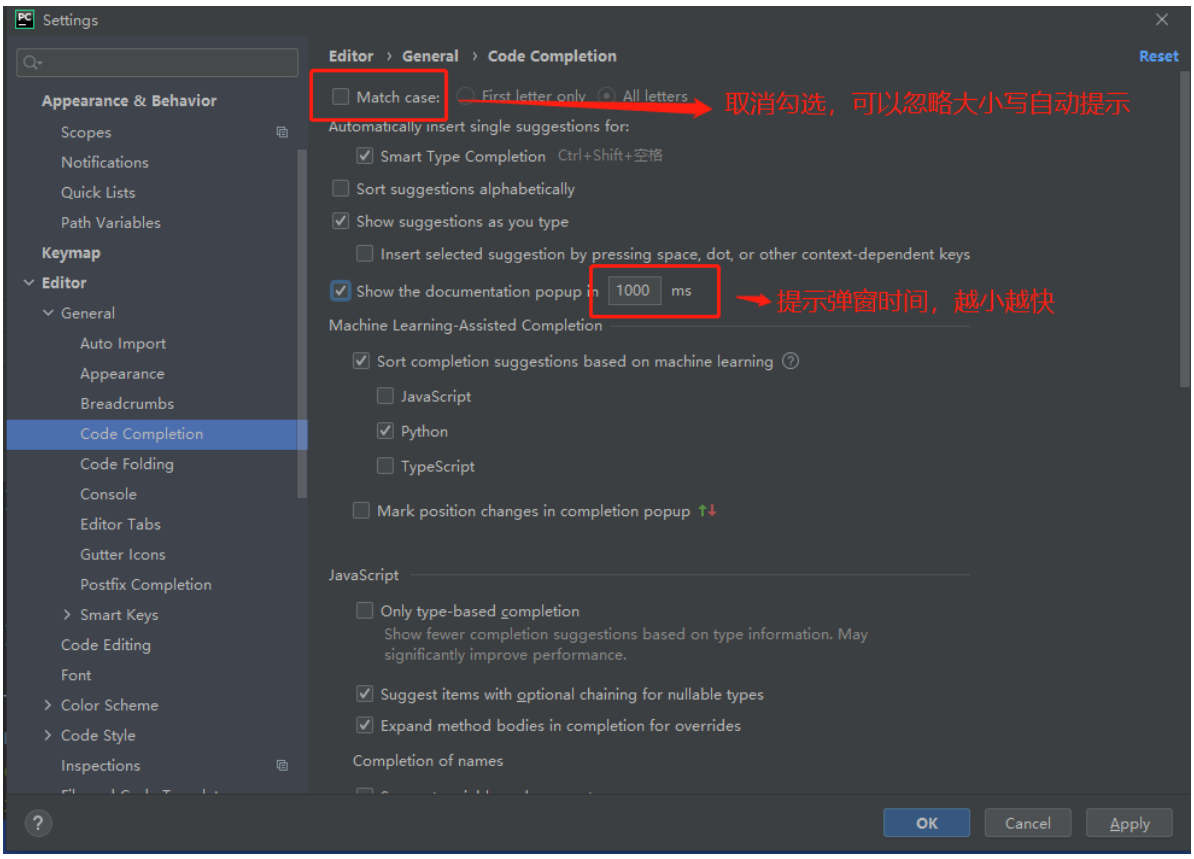


- `Code Completion:` 代码补全，不过可以进行全局设置，每次敲入字母时会自动提示进行补全

  设置步骤如下：`File -> settings`

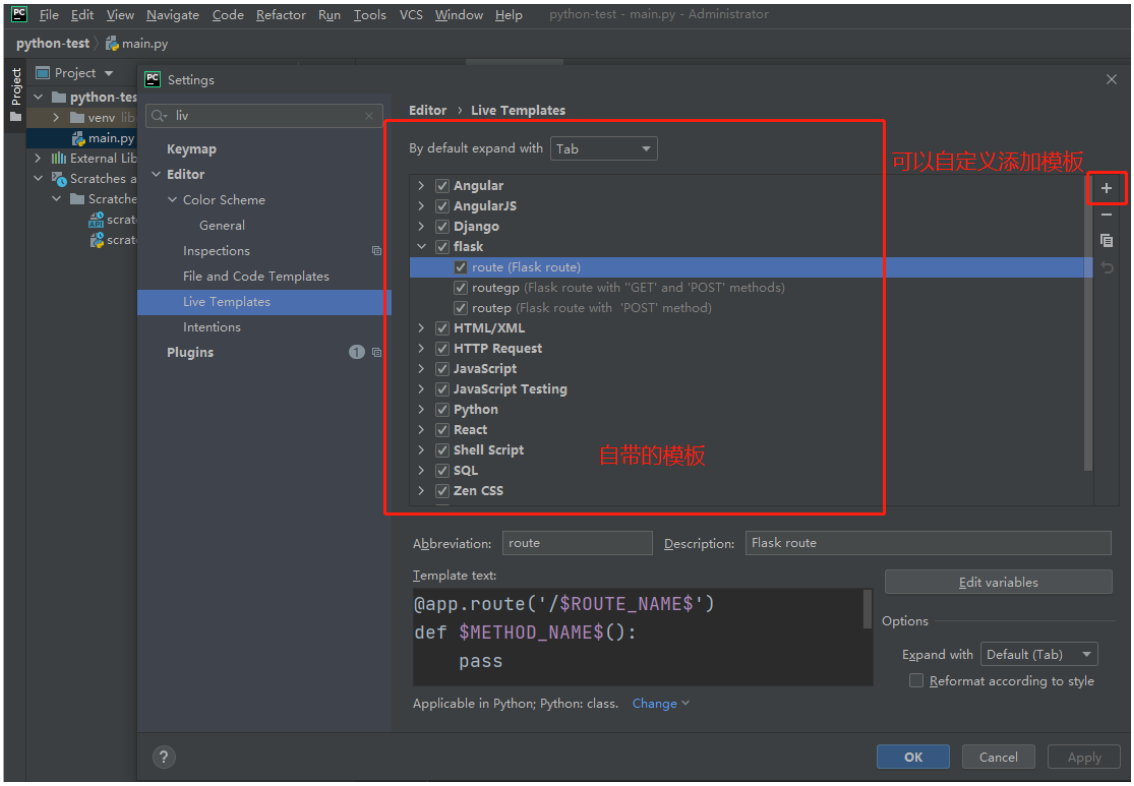


在同样的窗口，可以设置忽略大小写补全

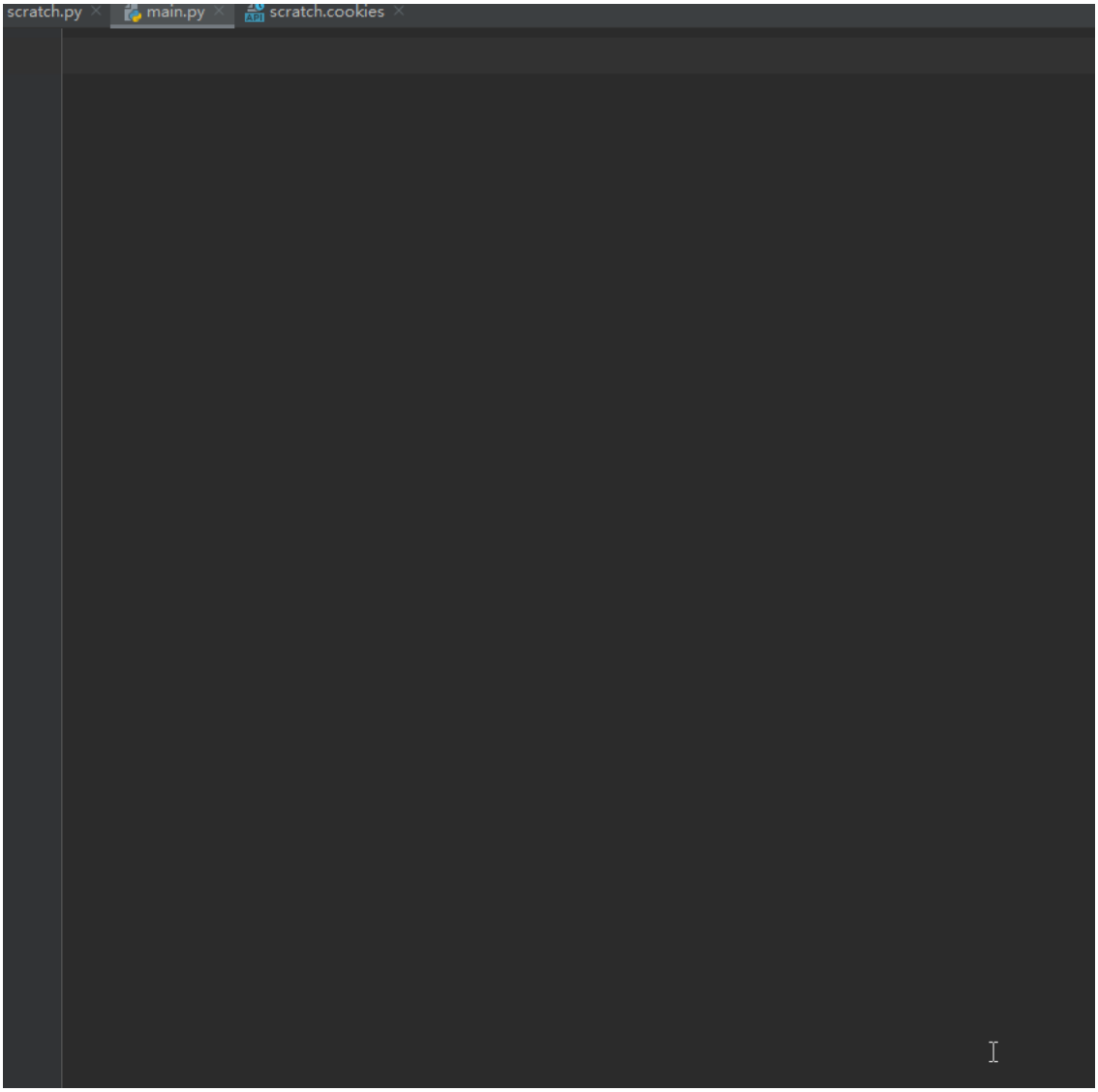


- `Insert Live Template:` 快速插入模板。

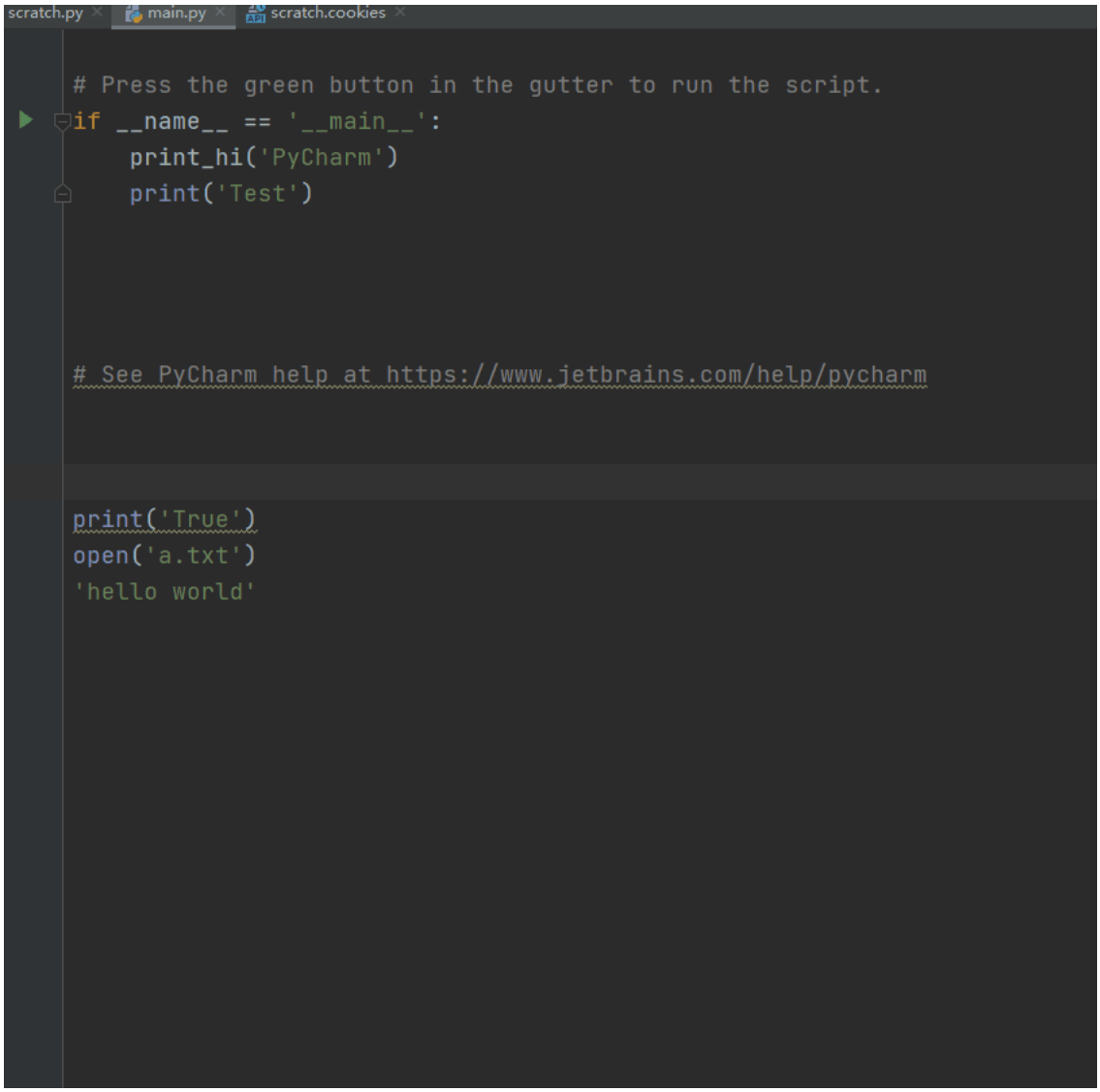
  默认模板，路径`File -> Settings`，也可以通过点击`+`号 自己添加



以上面的`flask`下的`route`为例，写代码时，直接输入`route`就可以完成预先设置的模板内容了

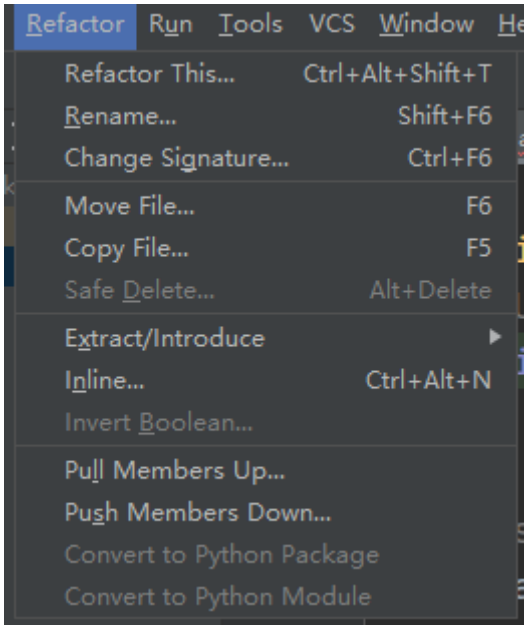


- `surround With:` 将选择的代码进行包裹，如`if/while/for/try..catch`包裹住。快捷键`Ctrl +Alt + T`



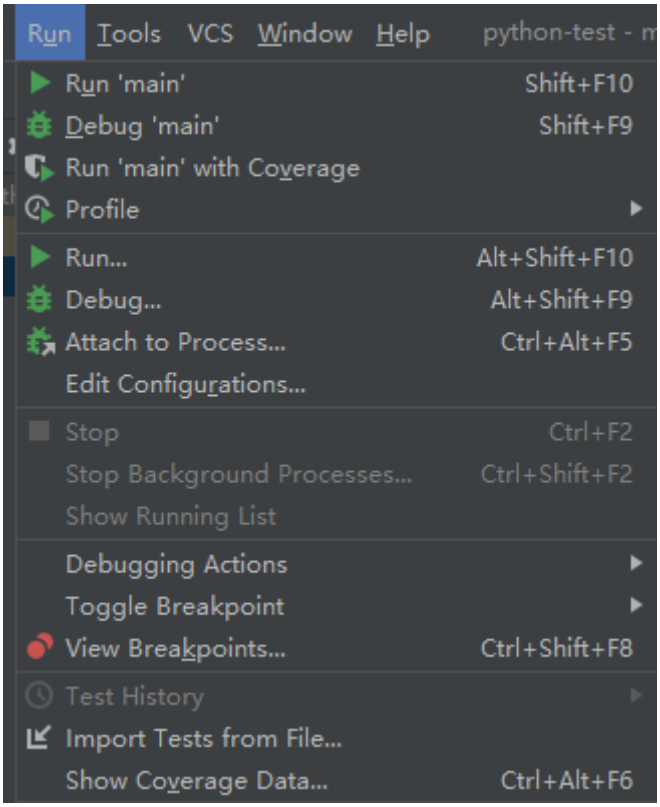
- `Reformat Code:`格式化代码，快捷键`Ctrl +Alt +L`
- `Auto-Indent Lines`: 自动缩进，快捷键`Ctrl + Alt + I`
- `Move Statement/Line Down/Up:` 向上向下移动，快捷键`Ctrl +Shift +` 向上箭头/向下箭头

5、Refactor（重构）

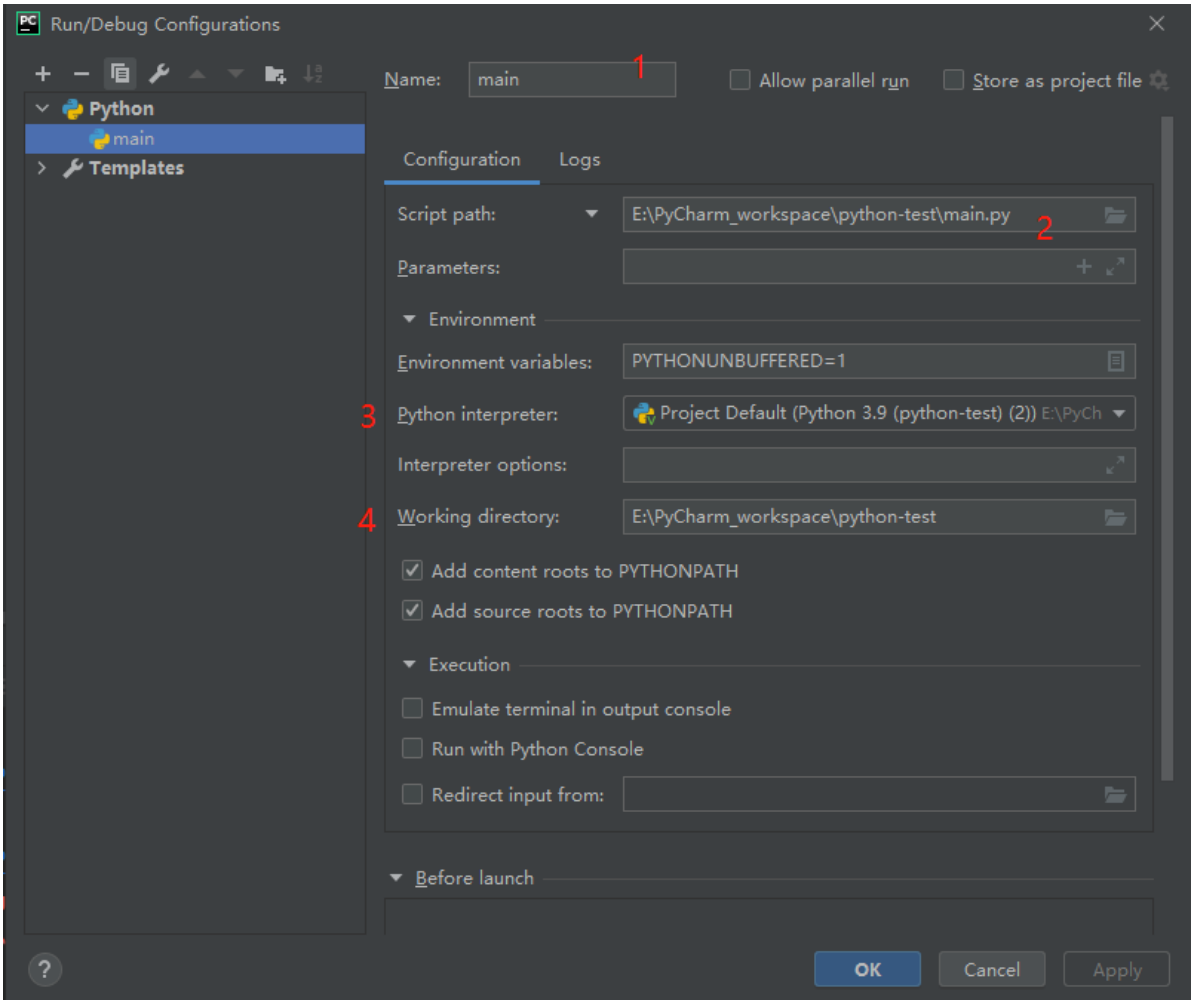


- `Refactor This..:` 重构当前
- `Rename:` 重命名，快捷键 `Shift + F6`
- `Move:` 移动文件，快捷键 `F6`
- `Copy:` 拷贝文件，快捷键 `F5`
- `Safe Detele:` 安全删除，快速删除py文件，快捷键 `Alt + Delete`

6、Run



- `Run 'xxx':` 运行当前文件
- `Debug 'xxx':` 通过Debug模式运行该文件
- `Run 'xxx' with Coverage:` 以统计覆盖的形式运行当前文件
- `Run ...:` 选择文件运行
- `Debug ...:` 选择文件Debug运行
- `Edit Configurations..:` 编辑配置内容

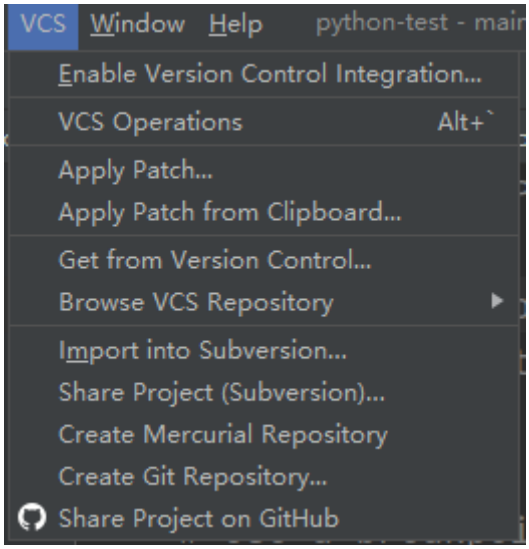


> 1、Name：可以自己随意起名
>
> 2、Script Path： 项目的文件路径
>
> 3、Python interpreter：Python解释器的路径
>
> 4、Workding directory：项目路径

7、Tools（工具）

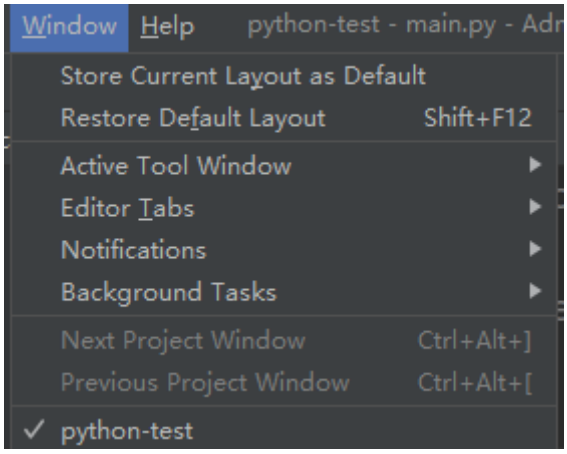
保存一些文件/项目模板。

8、VCS（版本控制）



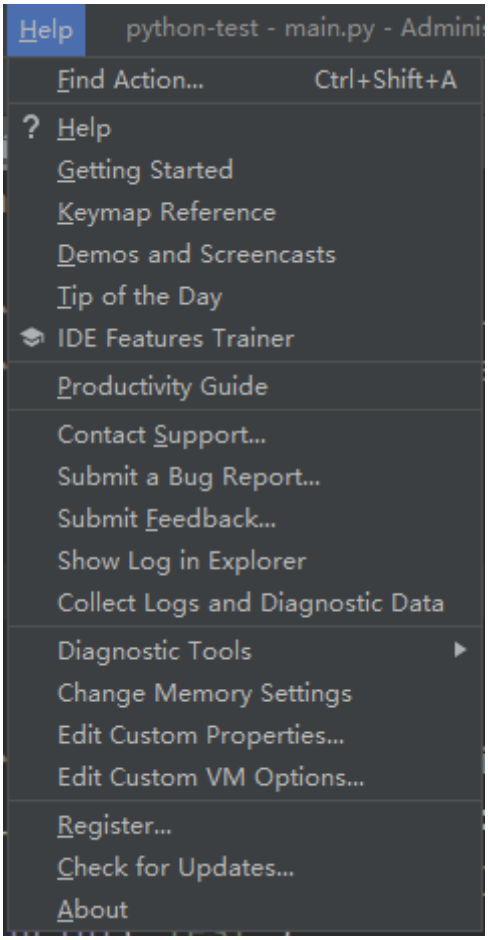
- `Enable Version Control Integration:` 选择相应的版本控制工具
- `VCS Operation:` 版本控制操作窗口
- `Get from Version Control... :` 从版本控制中获取（比如从GitHub上导入项目时，可在这个模块中完成）

9、Window（窗体）

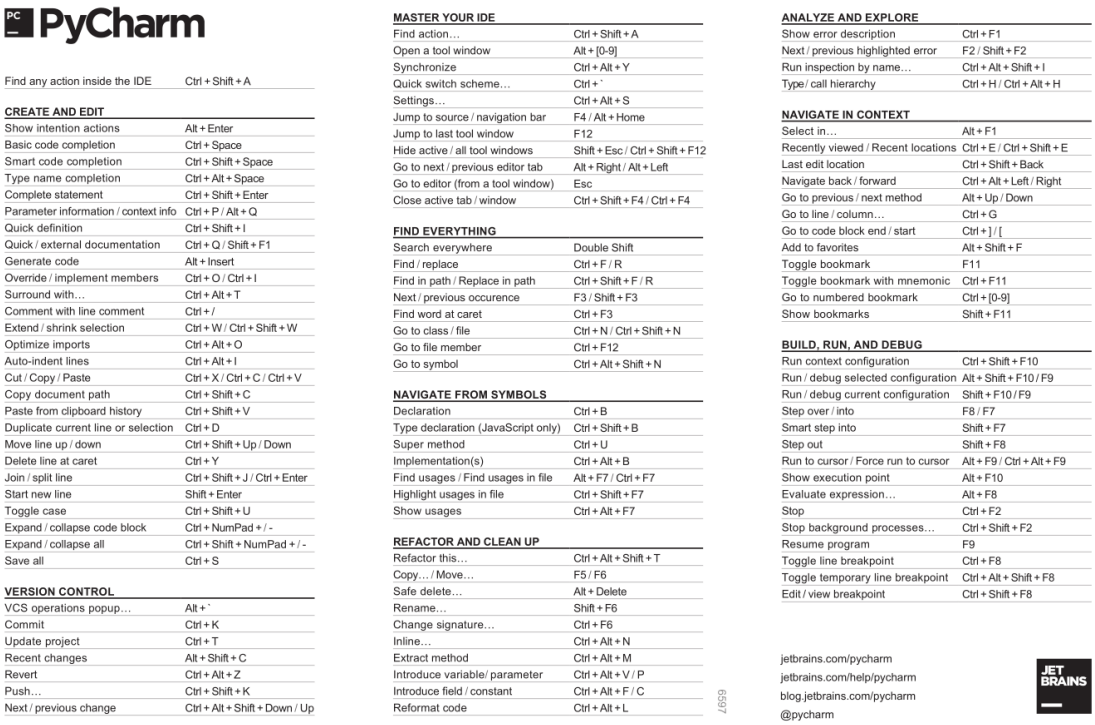


- `Store Current Layout as Default:` 存储当前PyCharm布局
- `Restore Default Layout:` 窗口布局复位（有时候窗口比较乱的时候，可以进行还原）
- `其他补充:` 主要就是控制窗口布局，以及tab显示的

10、Help



- `Find Action:`通过键入快捷键唤出想要的功能（非常强大，适合键盘流）
- `Keymap Reference:`查看快捷键清单（）



PyCharm

Find any action inside the IDE — Ctrl + Shift + A

CREATE AND EDIT

Show intention actions	Alt + Enter
Basic code completion	Ctrl + Space
Smart code completion	Ctrl + Shift + Space
Type name completion	Ctrl + Alt + Space
Complete statement	Ctrl + Shift + Enter
Parameter information / context info	Ctrl + P / Alt + Q
Quick definition	Ctrl + Shift + I
Quick / external documentation	Ctrl + Q / Shift + F1
Generate code	Alt + Insert
Override / implement members	Ctrl + O / Ctrl + I
Surround with…	Ctrl + Alt + T
Comment with line comment	Ctrl + /
Extend / shrink selection	Ctrl + W / Ctrl + Shift + W
Optimize imports	Ctrl + Alt + O
Auto-indent lines	Ctrl + Alt + I
Cut / Copy / Paste	Ctrl + X / Ctrl + C / Ctrl + V
Copy document path	Ctrl + Shift + C
Paste from clipboard history	Ctrl + Shift + V
Duplicate current line or selection	Ctrl + D
Move line up / down	Ctrl + Shift + Up / Down
Delete line at caret	Ctrl + Y
Join / split line	Ctrl + Shift + J / Ctrl + Enter
Start new line	Shift + Enter
Toggle case	Ctrl + Shift + U
Expand / collapse code block	Ctrl + NumPad + / -
Expand / collapse all	Ctrl + Shift + NumPad + / -
Save all	Ctrl + S

VERSION CONTROL

VCS operations popup…	Alt + `
Commit	Ctrl + K
Update project	Ctrl + T
Recent changes	Alt + Shift + C
Revert	Ctrl + Alt + Z
Push…	Ctrl + Shift + K
Next / previous change	Ctrl + Alt + Shift + Down / Up

MASTER YOUR IDE

Find action…	Ctrl + Shift + A
Open a tool window	Alt + [0-9]
Synchronize	Ctrl + Alt + Y
Quick switch scheme…	Ctrl + `
Settings…	Ctrl + Alt + S
Jump to source / navigation bar	F4 / Alt + Home
Jump to last tool window	F12
Hide active / all tool windows	Shift + Esc / Ctrl + Shift + F12
Go to next / previous editor tab	Alt + Right / Alt + Left
Go to editor (from a tool window)	Esc
Close active tab / window	Ctrl + Shift + F4 / Ctrl + F4

FIND EVERYTHING

Search everywhere	Double Shift
Find / replace	Ctrl + F / R
Find in path / Replace in path	Ctrl + Shift + F / R
Next / previous occurence	F3 / Shift + F3
Find word at caret	Ctrl + F3
Go to class / file	Ctrl + N / Ctrl + Shift + N
Go to file member	Ctrl + F12
Go to symbol	Ctrl + Alt + Shift + N

NAVIGATE FROM SYMBOLS

Declaration	Ctrl + B
Type declaration (JavaScript only)	Ctrl + Shift + B
Super method	Ctrl + U
Implementation(s)	Ctrl + Alt + B
Find usages / Find usages in file	Alt + F7 / Ctrl + F7
Highlight usages in file	Ctrl + Shift + F7
Show usages	Ctrl + Alt + F7

REFACTOR AND CLEAN UP

Refactor this…	Ctrl + Alt + Shift + T
Copy… / Move…	F5 / F6
Safe delete…	Alt + Delete
Rename…	Shift + F6
Change signature…	Ctrl + F6
Inline…	Ctrl + Alt + N
Extract method	Ctrl + Alt + M
Introduce variable/ parameter	Ctrl + Alt + V / P
Introduce field / constant	Ctrl + Alt + F / C
Reformat code	Ctrl + Alt + L

6697

ANALYZE AND EXPLORE

Show error description	Ctrl + F1
Next / previous highlighted error	F2 / Shift + F2
Run inspection by name…	Ctrl + Alt + Shift + I
Type / call hierarchy	Ctrl + H / Ctrl + Alt + H

NAVIGATE IN CONTEXT

Select in…	Alt + F1
Recently viewed / Recent locations	Ctrl + E / Ctrl + Shift + E
Last edit location	Ctrl + Shift + Back
Navigate back / forward	Ctrl + Alt + Left / Right
Go to previous / next method	Alt + Up / Down
Go to line / column…	Ctrl + G
Go to code block end / start	Ctrl +] / [
Add to favorites	Alt + Shift + F
Toggle bookmark	F11
Toggle bookmark with mnemonic	Ctrl + F11
Go to numbered bookmark	Ctrl + [0-9]
Show bookmarks	Shift + F11

BUILD, RUN, AND DEBUG

Run context configuration	Ctrl + Shift + F10
Run / debug selected configuration	Alt + Shift + F10 / F9
Run / debug current configuration	Shift + F10 / F9
Step over / into	F8 / F7
Smart step into	Shift + F7
Step out	Shift + F8
Run to cursor / Force run to cursor	Alt + F9 / Ctrl + Alt + F9
Show execution point	Alt + F10
Evaluate expression…	Alt + F8
Stop	Ctrl + F2
Stop background processes…	Ctrl + Shift + F2
Resume program	F9
Toggle line breakpoint	Ctrl + F8
Toggle temporary line breakpoint	Ctrl + Alt + Shift + F8
Edit / view breakpoint	Ctrl + Shift + F8

jetbrains.com/pycharm
jetbrains.com/help/pycharm
blog.jetbrains.com/pycharm
@pycharm

JET BRAINS

- `Tip of the Day:` PyCharm的每日小技巧
- `Edit Custom Properties:`在idea.properties中添加个人配置
- `Edit Custom VM Options:`在pycharm64.exe.vmoptions中添加启动配置
- `Register:`注册
- `Check for Update:`检查更新

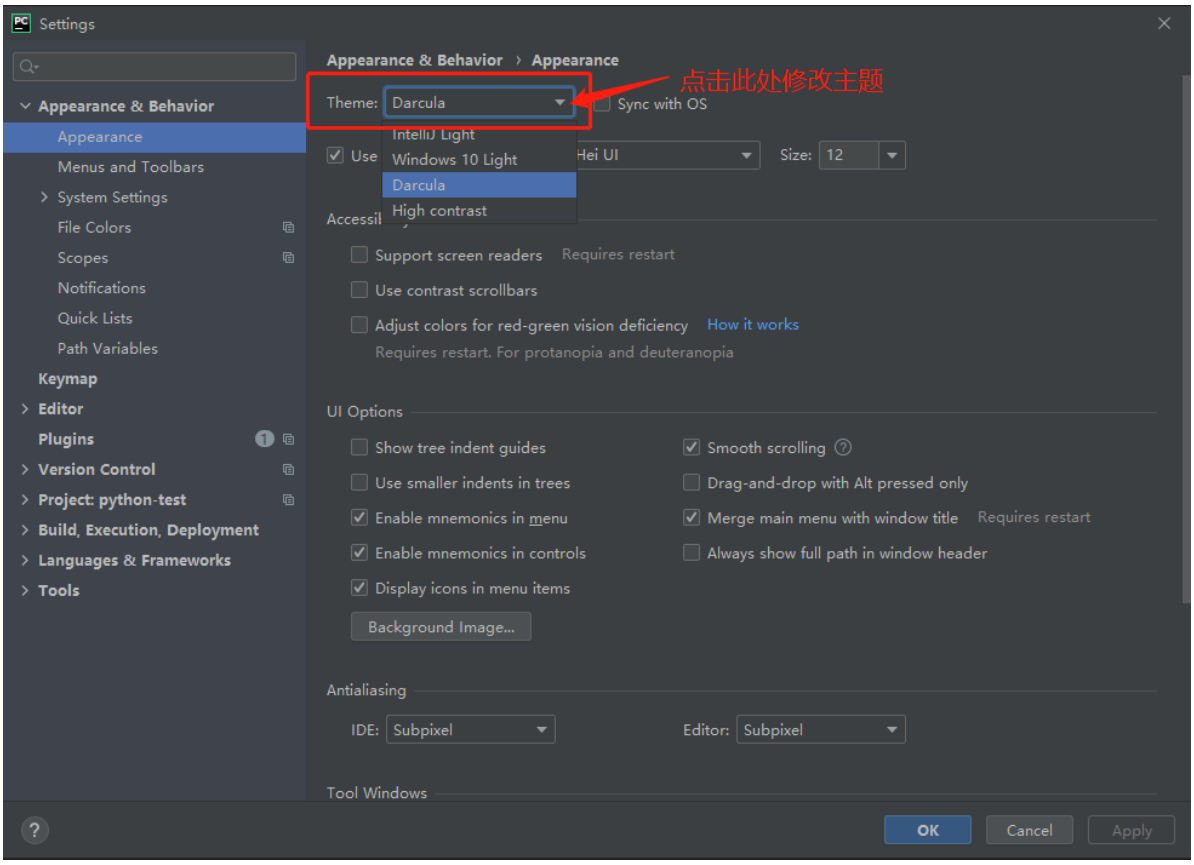
PyCharm基础配置

PyCharm安装完毕之后，我们需要修改一些常用的配置让视觉上更加享受，比如修改PyCharm的主题以及字体显示呢

基础配置在`File - > Settings`中进行

修改主题

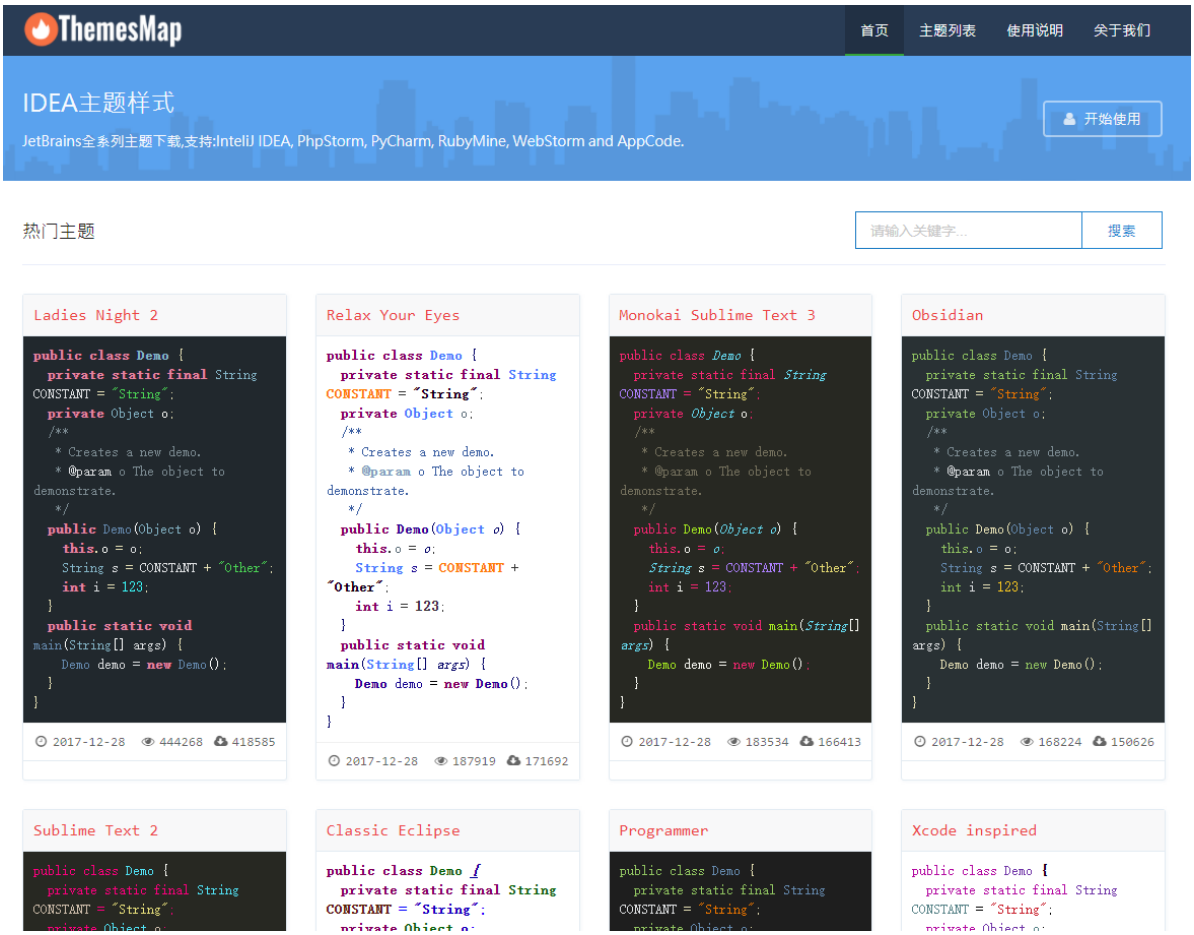
在`Settings`中 选择`Appearance & Behavior-> Appearance`，如下图所示



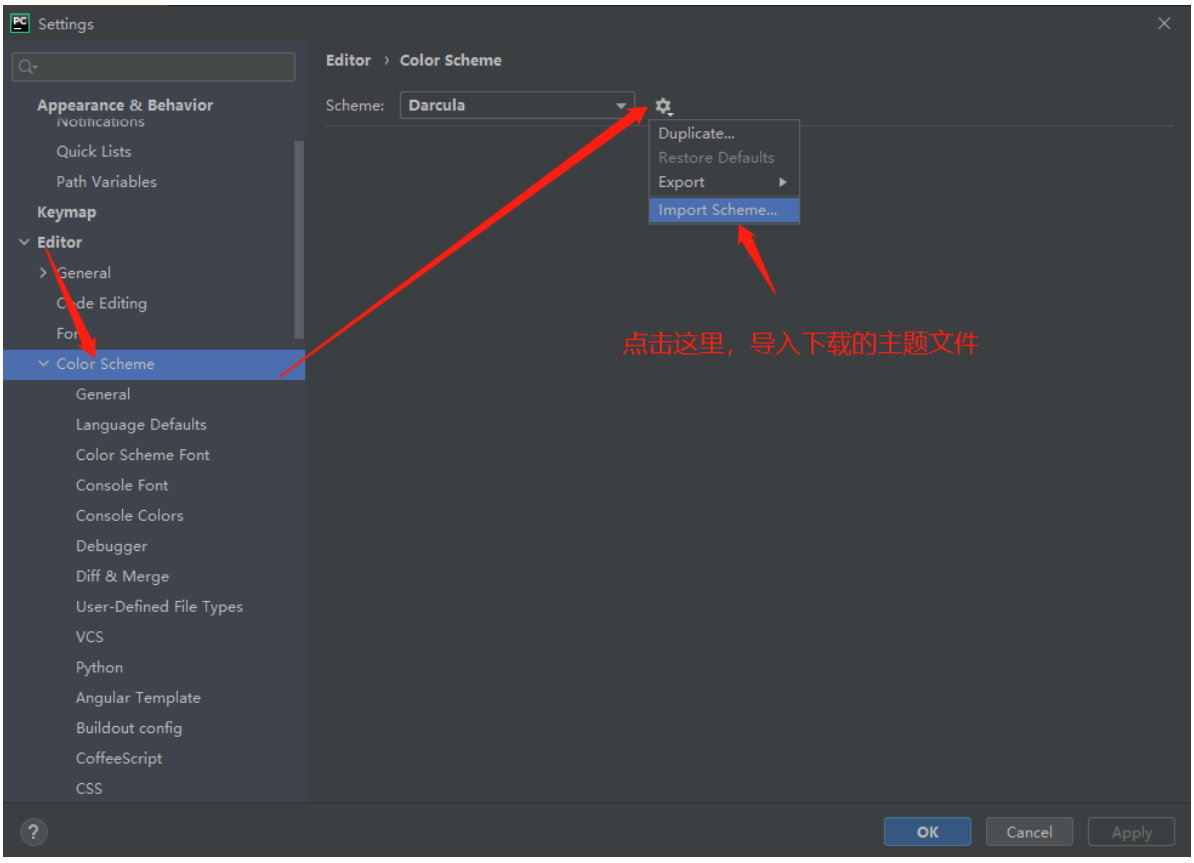
我个人习惯使用`Darcula`的黑色主题，除了默认的主题，我们也可以自己安装其他主题风格。

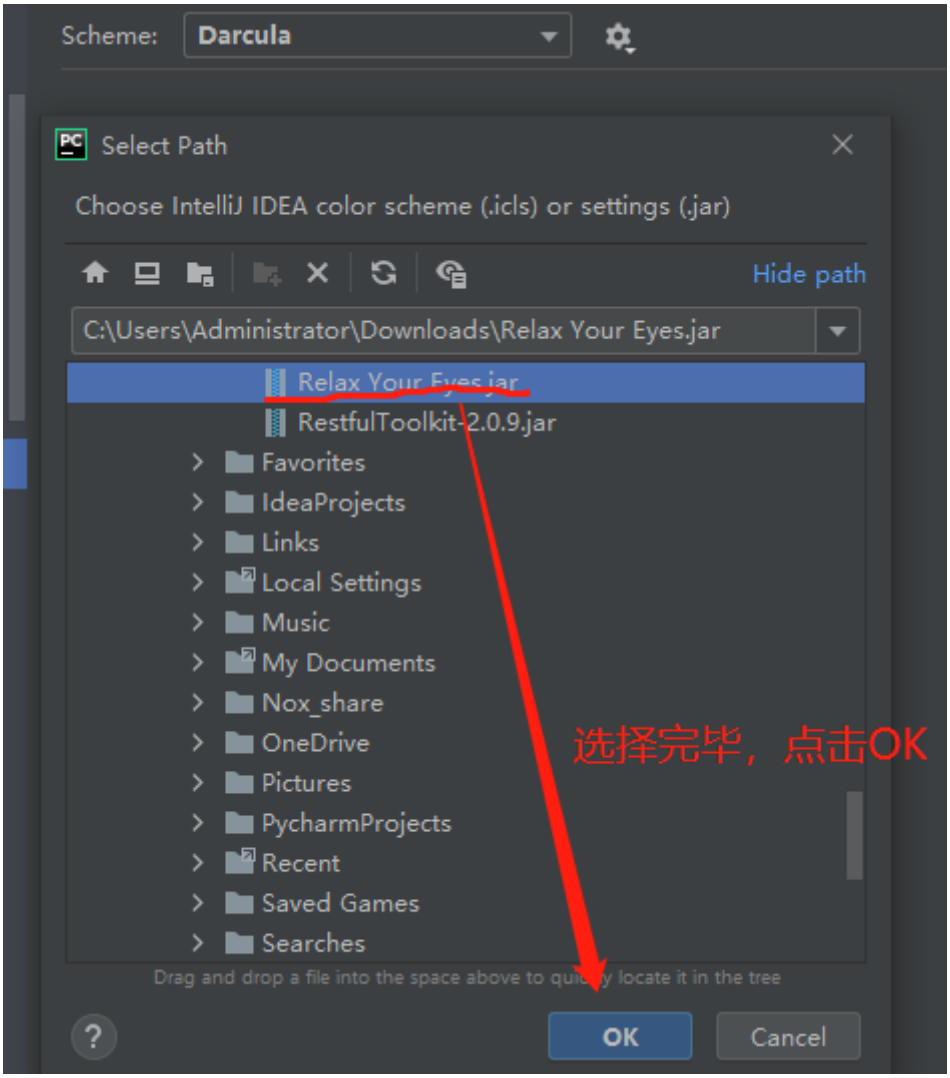
如何安装主题

主题可以从网站`http://www.themesmap.com/`上进行下载。



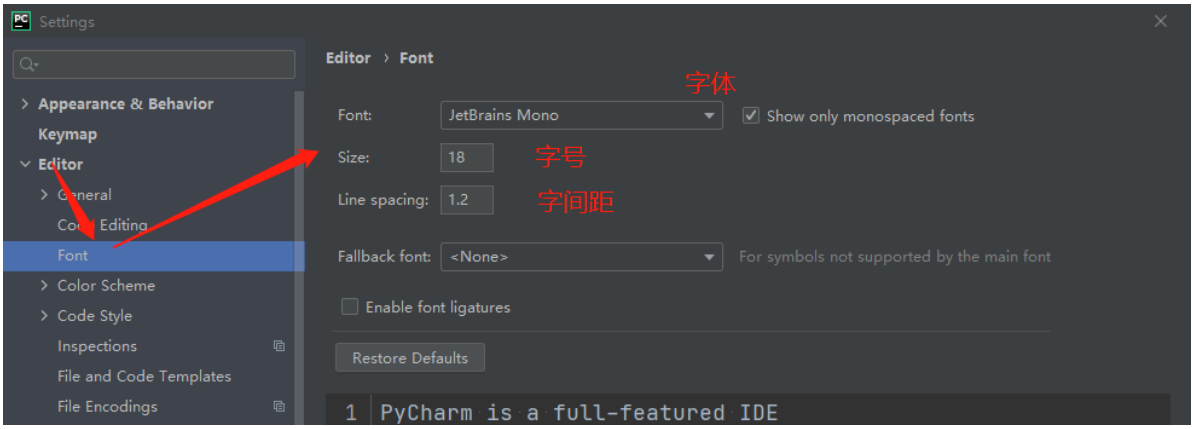
下载自己喜欢的主题，然后在`Settings`界面中`Editor -> color Scheme`中进行安装，如下图





设置字体

仍然是在`Settings`界面，在`Editor->Font`中修改字体及大小



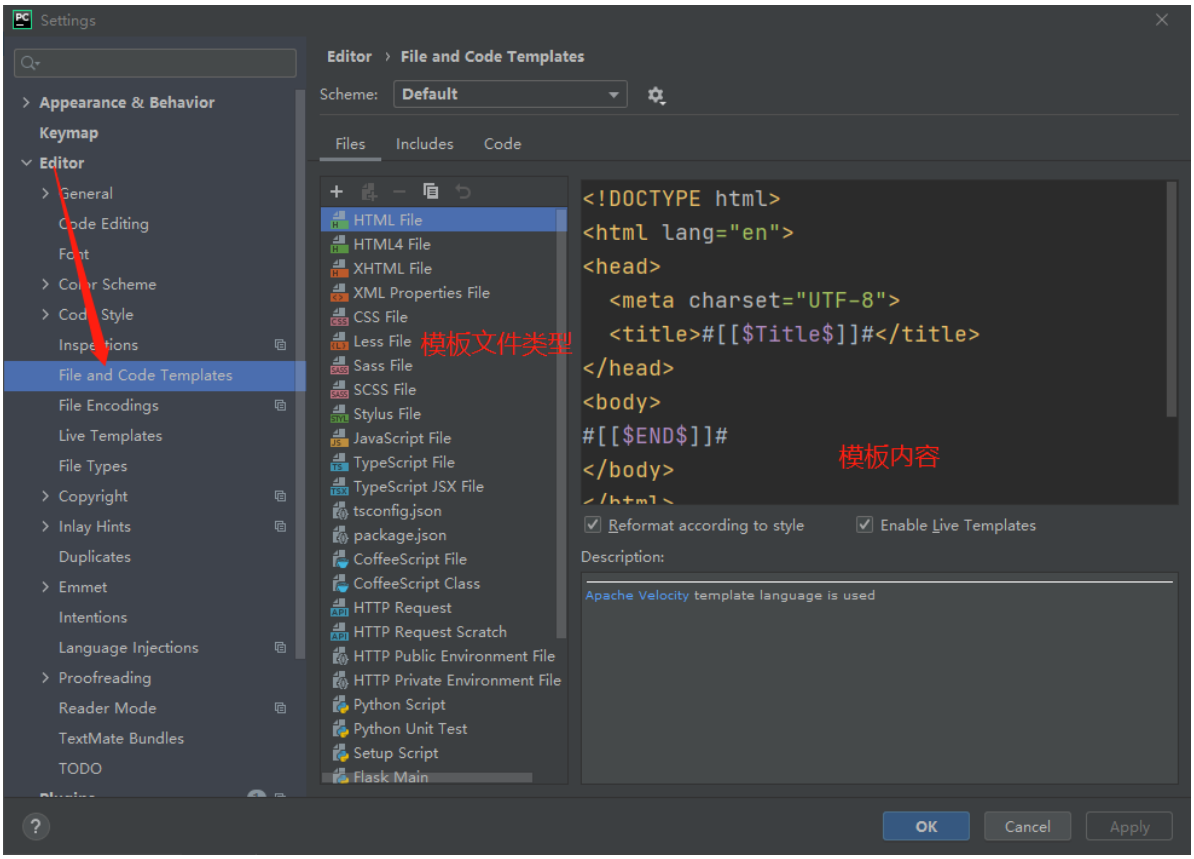
字体建议使用`JetBrains Mono`，它是JetBrains 公司开发的一款开源字体，也被称为最适合程序员的字体。

代码模板

通常情况，我们创建`.py`或者`.html`等文件时，需要指定一些信息，比如编码，开发人员信息，时间等..

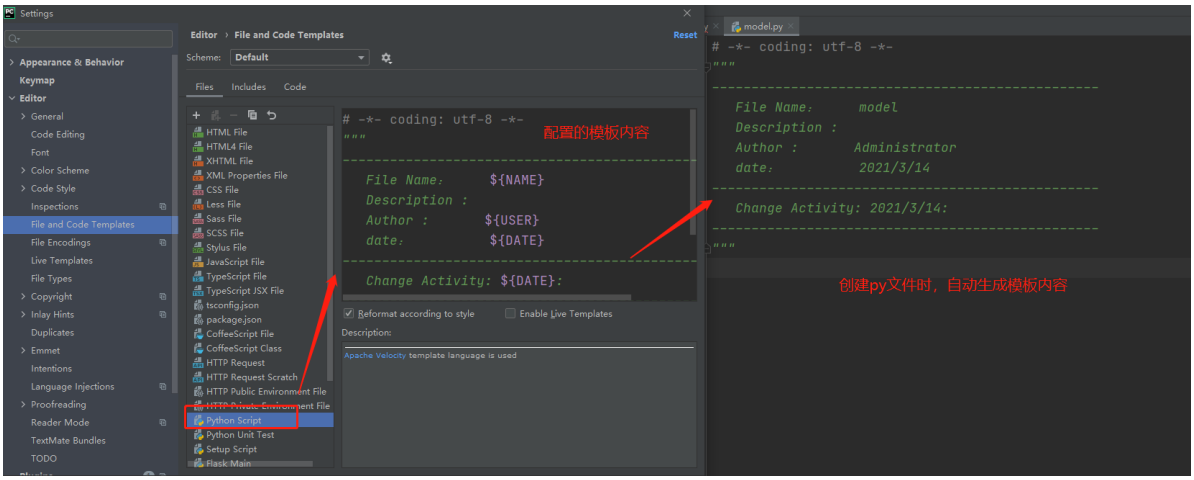
这个时候就需要修改一下对应的文件模板。

在`File - > Settings`下，找到`Editor - > File and Code Templates`



Example

我在`Python Script`中进行如下配置



模板其他变量信息

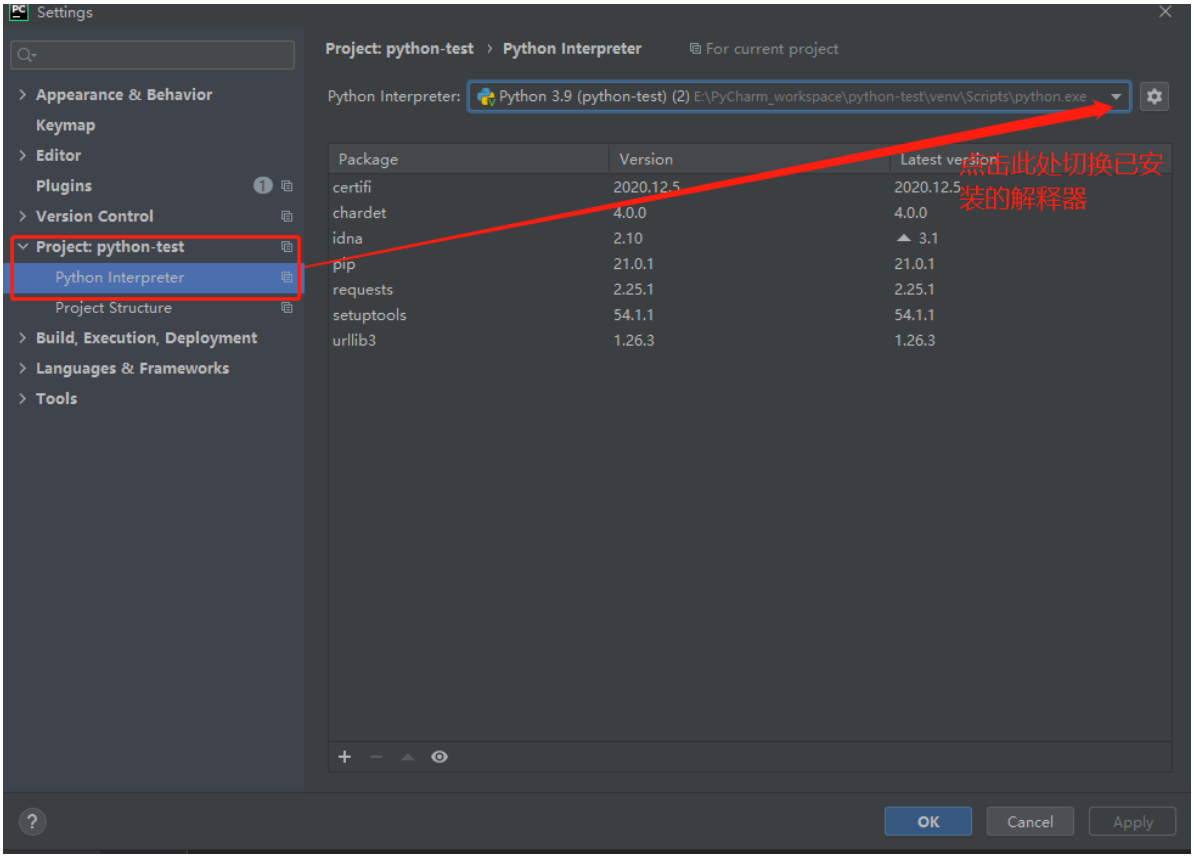
```
${PROJECT_NAME} - 当前Project名称;
```

```
${NAME} - 在创建文件的对话框中指定的文件名;

${USER} - 当前用户名;

${DATE} - 当前系统日期;

${TIME} - 当前系统时间;

${YEAR} - 年;

${MONTH} - 月;

${DAY} - 日;

${HOUR} - 小时;

${MINUTE} - 分钟;

${PRODUCT_NAME} - 创建文件的IDE名称;

${MONTH_NAME_SHORT} - 英文月份缩写, 如: Jan, Feb, etc;

${MONTH_NAME_FULL} - 英文月份全称, 如: January, February, etc;
```

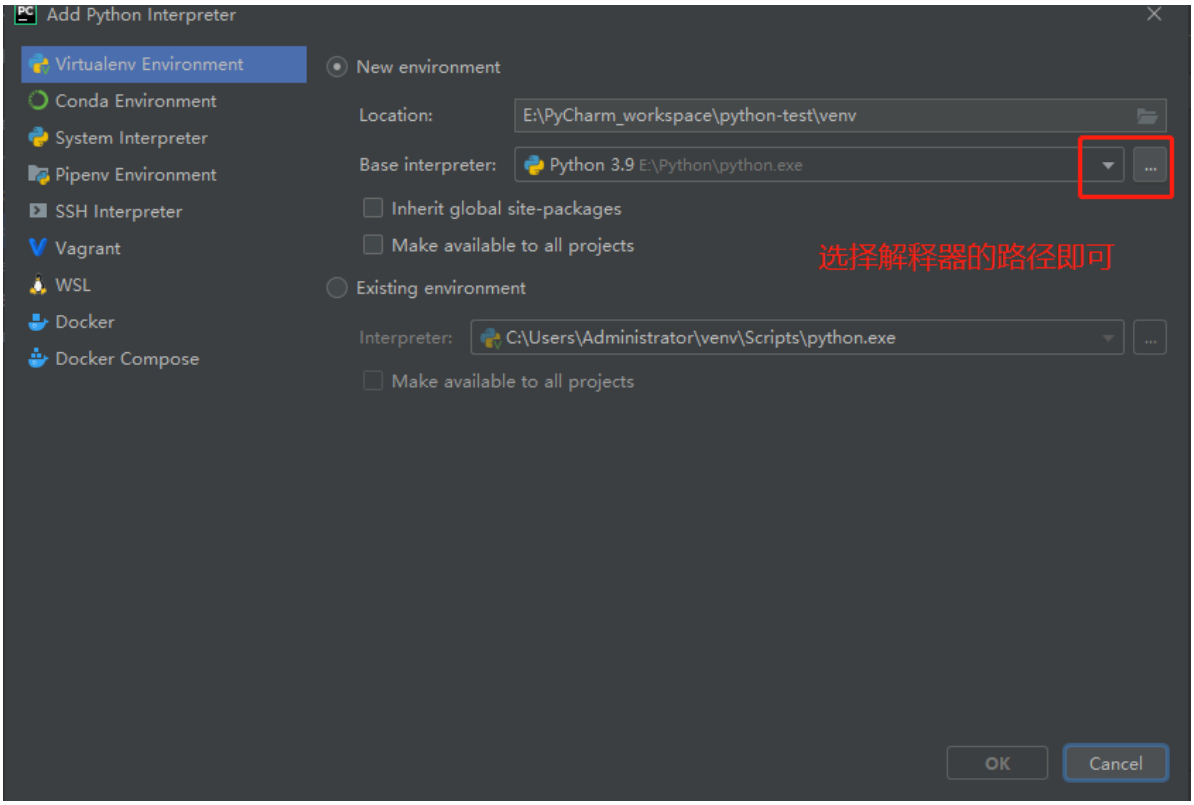
解释器配置

执行Python代码必须要用到解释器，在`环境配置`章节讲到过下载解释器。如果本地安装了多个解释器版本，在不同的项目中需要进行切换时。

可以在`Fiel -> settings`中的`Project 项目名 - > Project Interpreter`中进行设置，如下图



点击小齿轮，点击`add，选择其他版本的解释器即可。如何

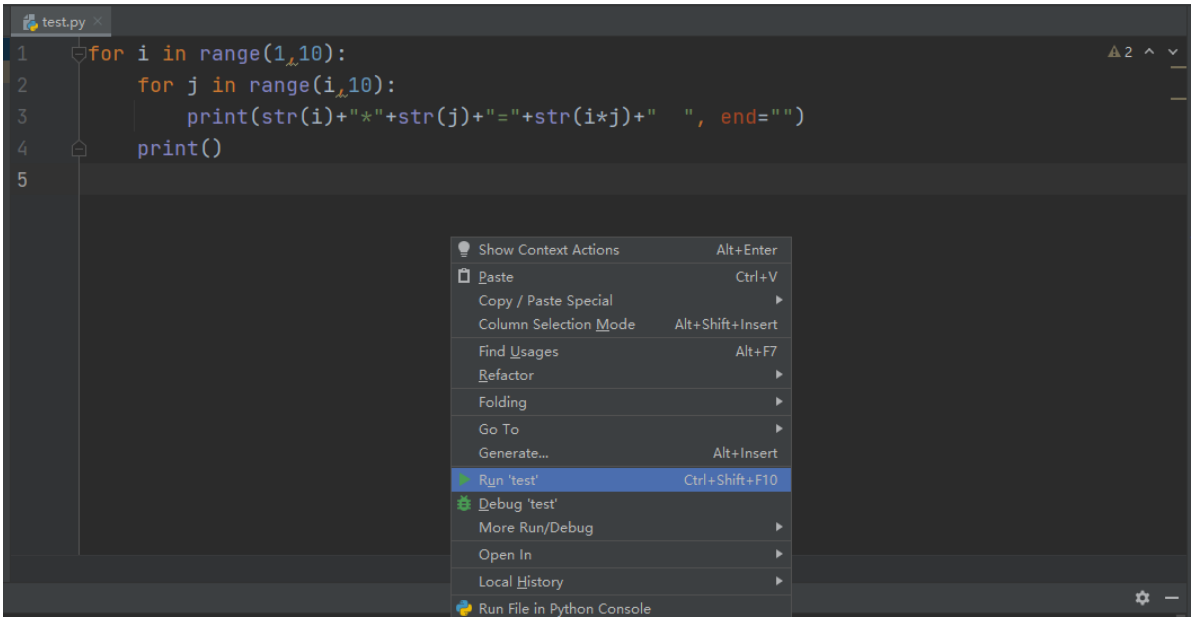


代码运行

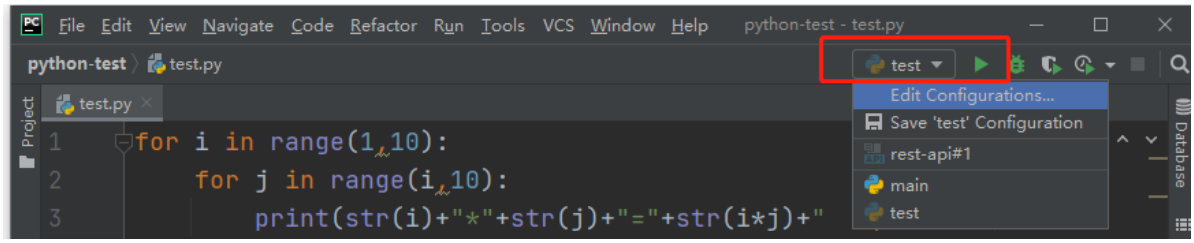
编写好代码肯定是需要去运行的，在Pycharm中运行代码的方式有以下

1、右键Run运行

直接在要运行的py文件中右键，点击`Run`即可， 或者使用快捷键`Ctrl + Shify + F10`

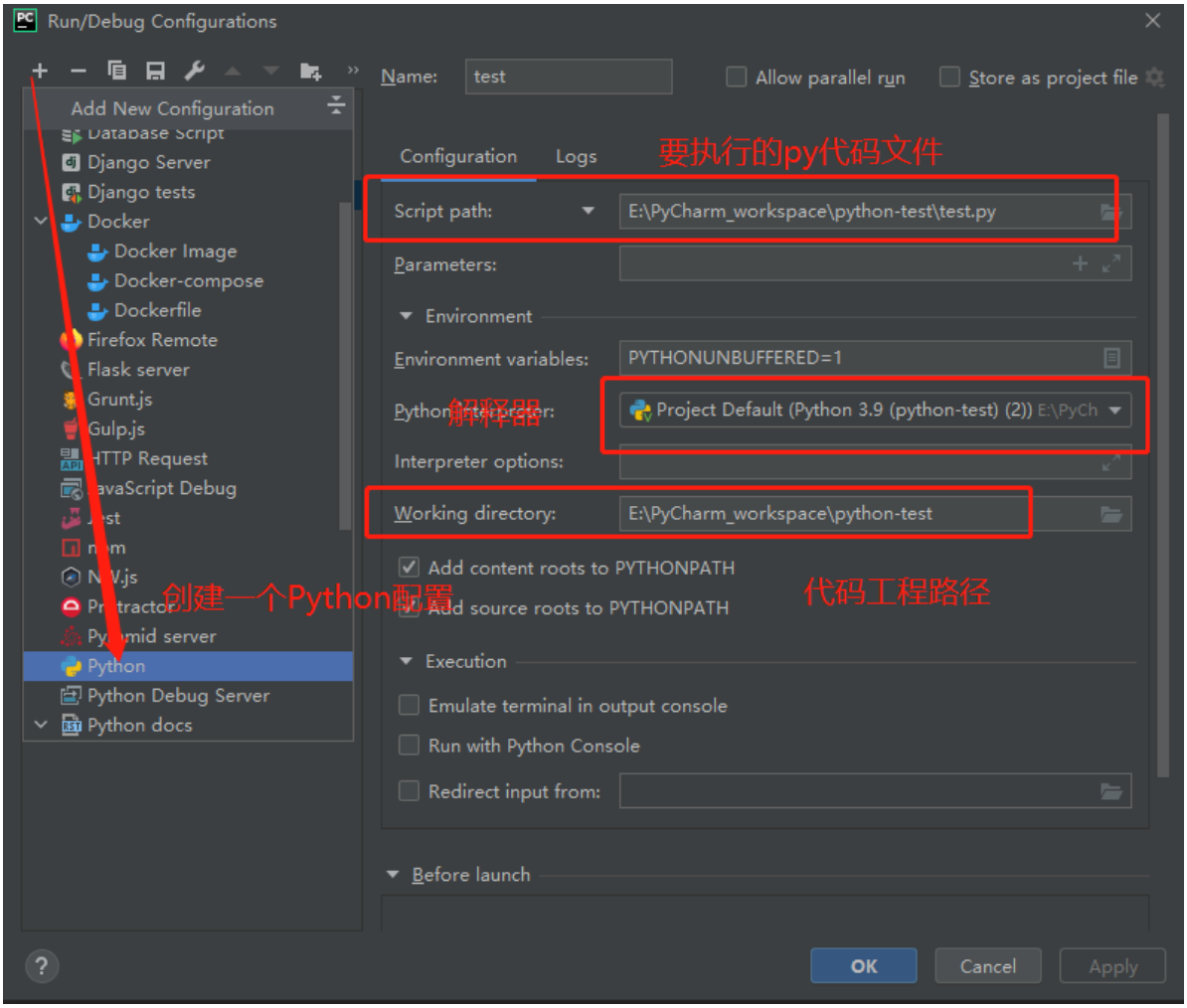


2、通过导航栏的Run执行



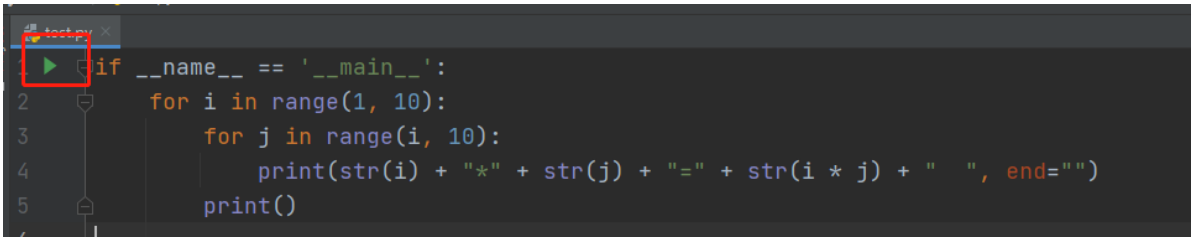
如果之前执行过某个程序，在这里是可以看到的，选择要执行的程序，点击绿色的三角箭头即可运行。

如果是一个新的程序，则点击`Edit Configurations`进行配置,这个在前面的`菜单栏 Run`章节中有讲过



3、通过main启动

如果程序中有`main`函数，会在左侧有个绿色的三角箭头，点击选择`Run 项目名`即可启动



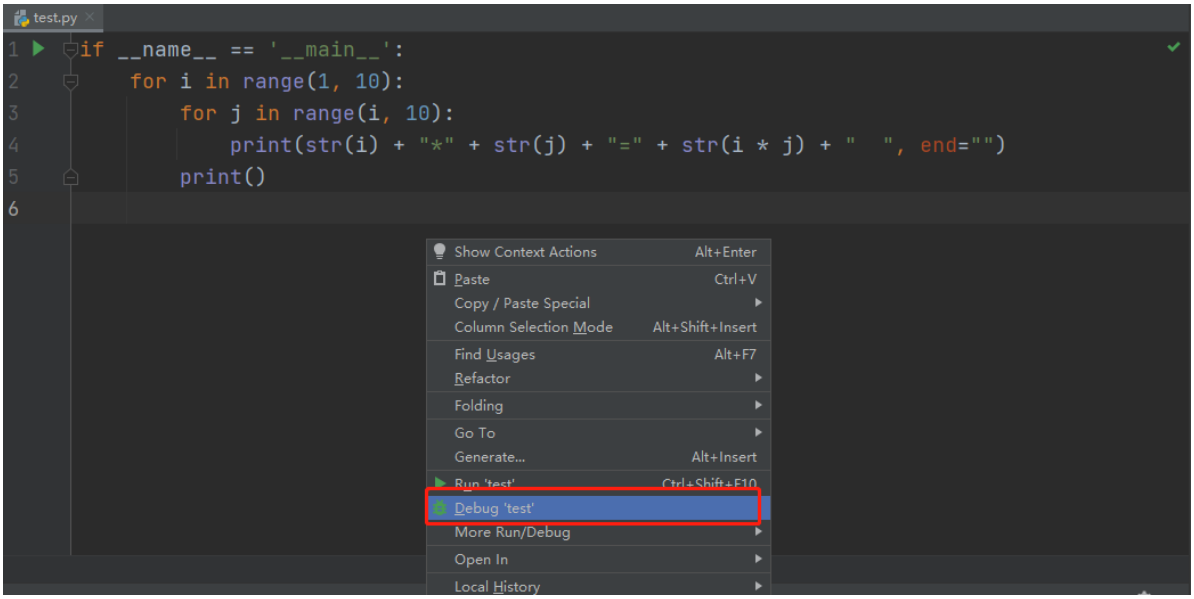
```python
if __name__ == '__main__':
    for i in range(1, 10):
        for j in range(i, 10):
            print(str(i) + "*" + str(j) + "=" + str(i * j) + "  ", end="")
        print()
```

DeBug运行/调试

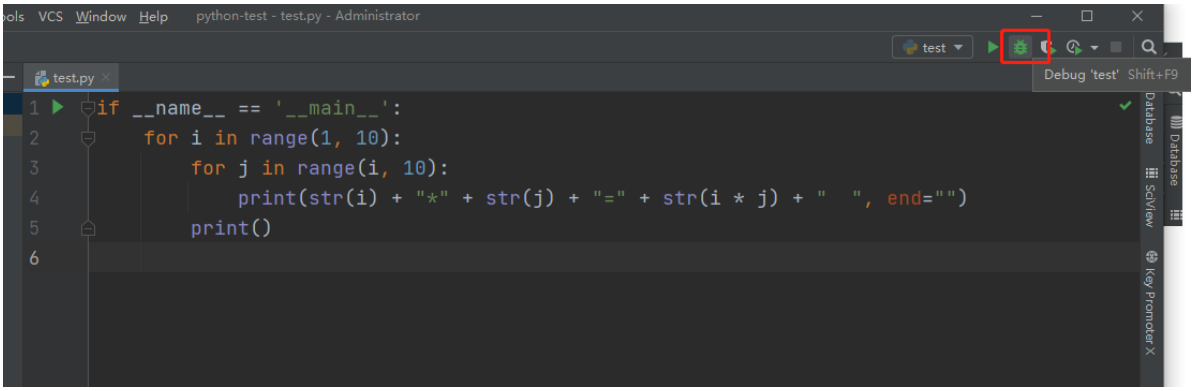
`Bug`大家都知道是程序中的错误，导致程序不能正常运行。而`DeBug`的字面意思就是解决Bug。

`DeBug`执行的方式也是有三种，与上面的`代码运行`章节类似，

1、右键DeBug



2、导航栏DeBug



点击导航栏绿色的蜘蛛图标即可DeBug启动。

3、通过main Debug执行

如果程序有`main`函数入口，可以点击左侧的绿色小三角，然后选择上图标识的`Debug 项目名`即可。

4、断点

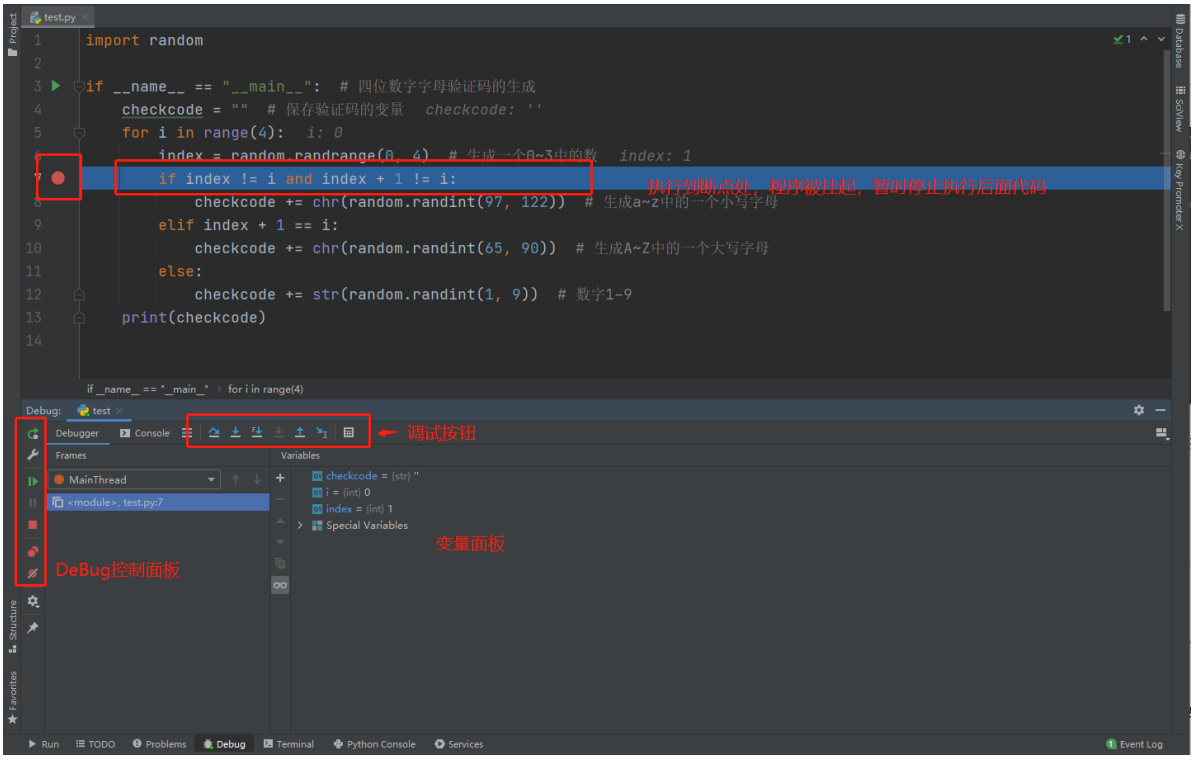
如果Debug的程序没有断点，则跟正常的执行没有区别。

> 断点：一个断点标注一个代码行，当程序执行到断点所在行时，会被挂起。我们可以查看项目中各参数的值，运行结果等信息

如何打断点?

如下图所示，在红框标注的地方单击一下就可以打上断点，**可以标记多个断点**，或者快捷键`Ctrl +F8`可快速在光标所在行打上断点。

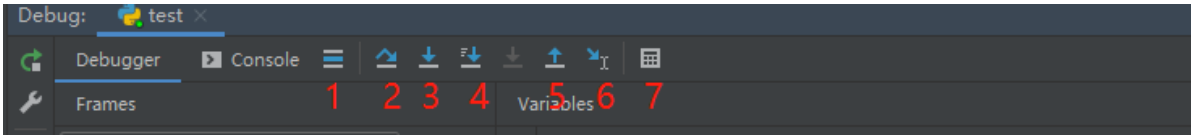
如下图，我在第7行打了一个断点，通过`DeBug`执行程序。



通过`变量面板`的信息我们可以得到以下信息

- 执行到第6行，`index`的值为 1，
- 当前循环为第一次循环，`i`的值为0
- checkcode 还未被赋值，依旧是空字符串`""`

5、调试按钮



调试面板如上图所示，每个按钮的意思如下，**将鼠标悬停在对应的图标上，可查看快捷键**

1、跳转到当前程序所执行的地方（快捷键Alt + F10）；比如你打开了很多窗口，当前界面在别的代码页面，只需要按一下这个按钮就会回到程序所执行的地方。

2、依次往下执行代码（快捷键F8）；不进入函数；顺序执行，如果某一行代码调用了别的函数，则不会进入那个函数

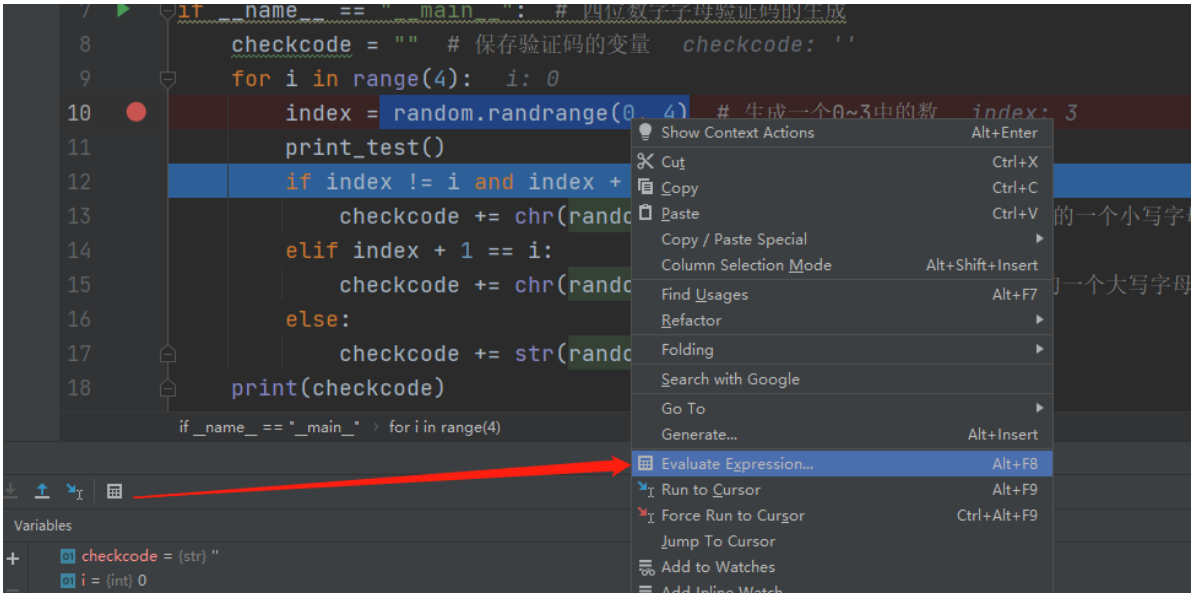
3、顺序执行，会进入函数（快捷键F7）；当某一行代码调用了其他函数，则会进入那个函数或者源码中。

4、顺序执行，进入自己写的代码中（快捷键Alt + Shift + F7）；与上面的区域是只会进入自己写的代码中，不会进入源代码中

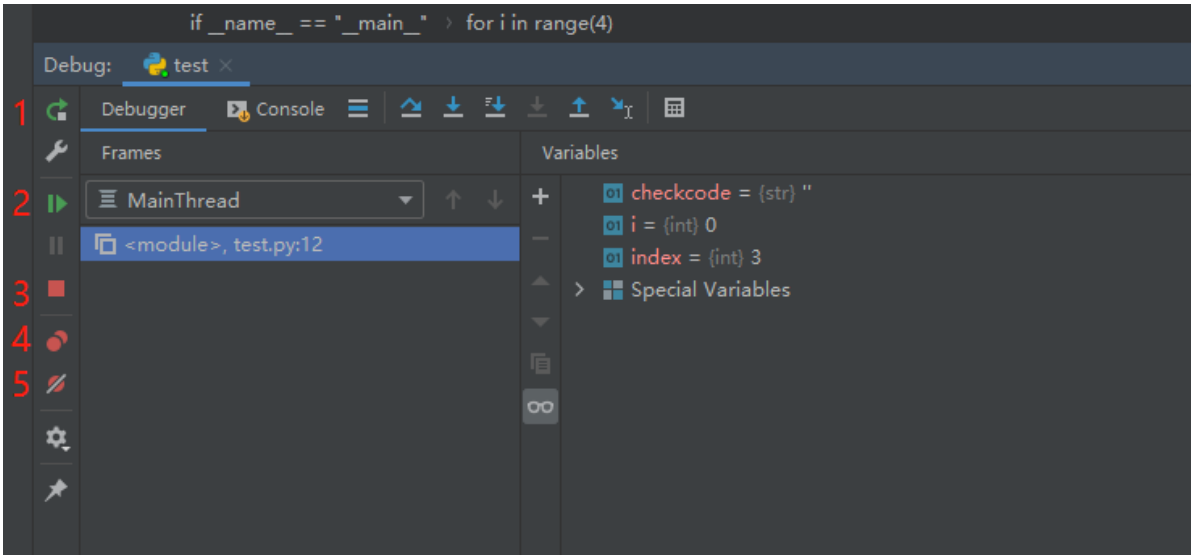
5、跳出当前函数（快捷键Shift + F8）；配合编号`3`、`4`当程序运行到函数体中，可以迅速跳出当前函数，回到程序执行的地方

6、运行到光标位置并暂停（快捷键Alt + F9）；此时你的光标在哪，程序就会运行到那个地方，并挂起（暂停）

7、计算表达式（快捷键Alt + F8）； 鼠标选中一段代码，点击`7`号按钮，或者右键选择`Evaluate Expression`，然后在弹窗中，点击`Evaluate`就可以计算出当前选中表达式的值。



6、DeBug控制面板



1、重新DeBug启动当前程序

2、跳过当前断点，直接运行到下一断点处，快捷键：F9

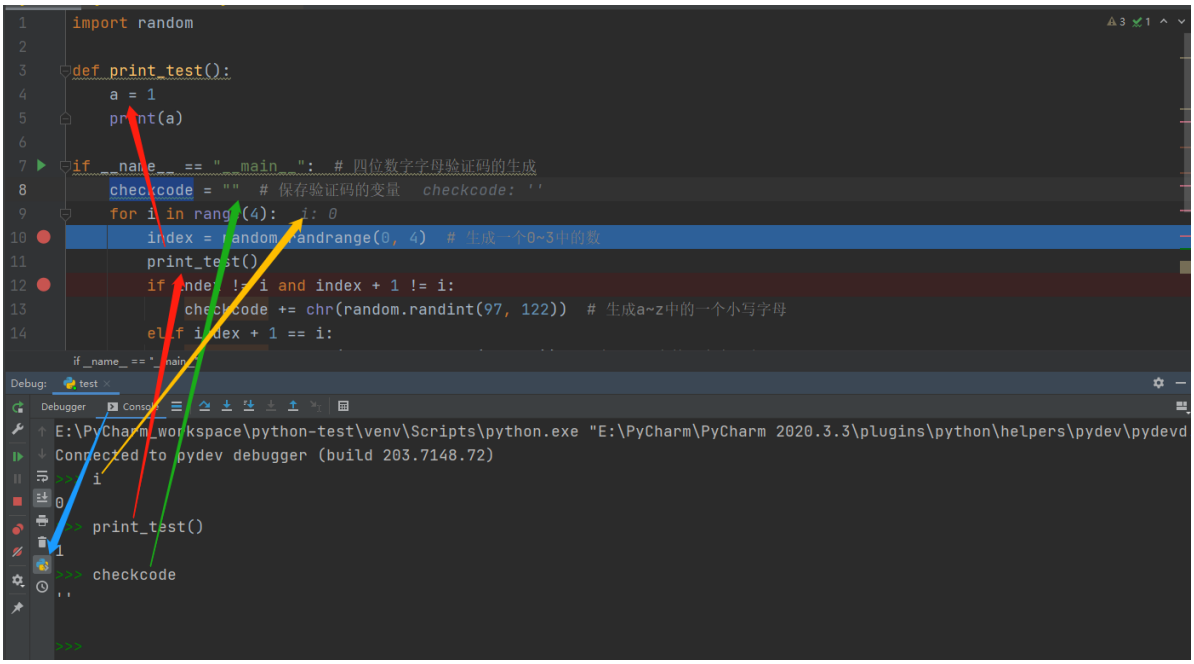
3、停止并关闭当前DeBug程序

4、查看当前所有设置的断点

5、使所有断点都失效（此时断点由红色变为灰白色）

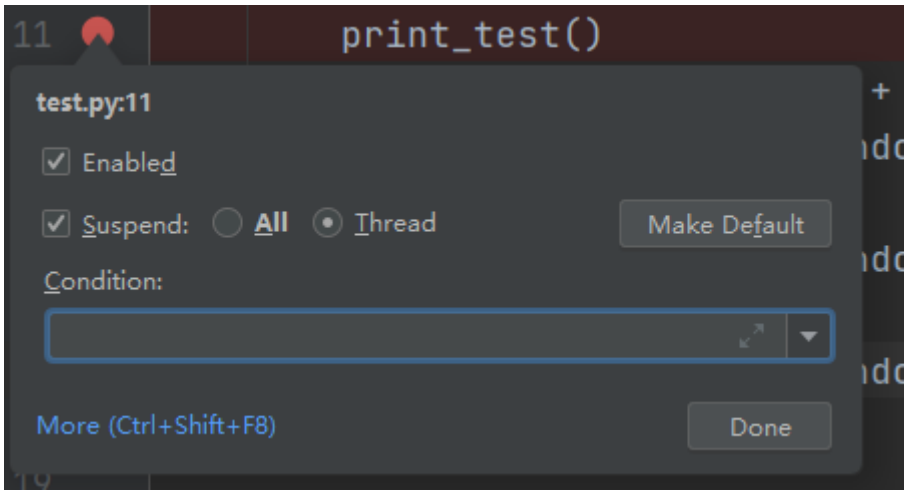
7、DeBug Console面板

在DeBug执行的时候，在`Console`中有个按 钮如下图蓝色箭头标注所示`Show DeBug Console`，可以在右侧终端中输入变量的值或者表达式，按回车之后，会返回相应的结果。



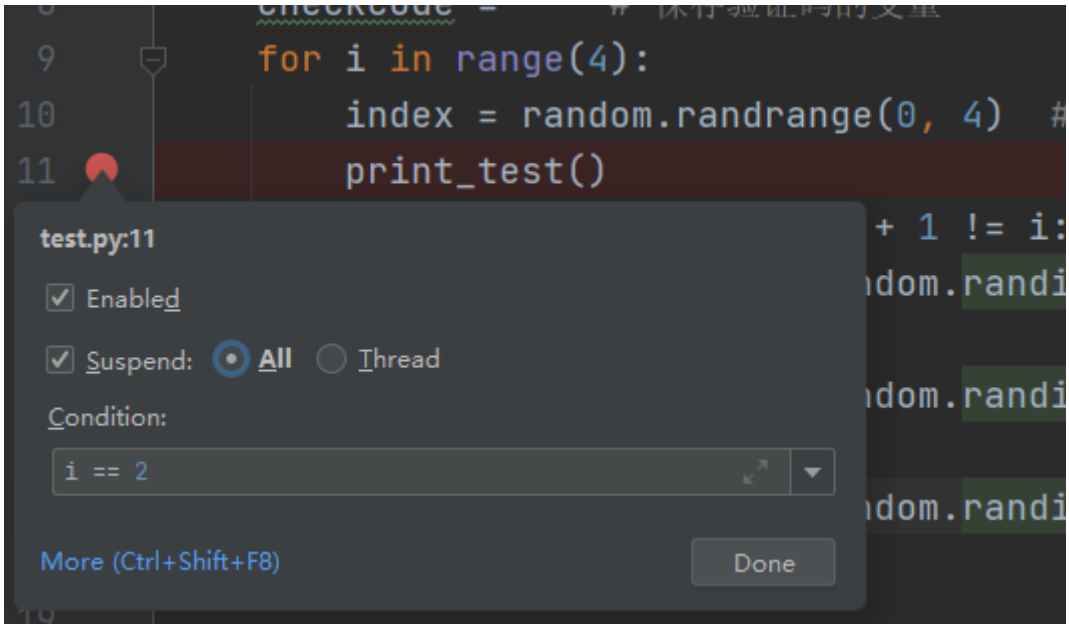
8、设置断点属性（过滤）

在断点的小红点上右键，我们会发现这样一个界面

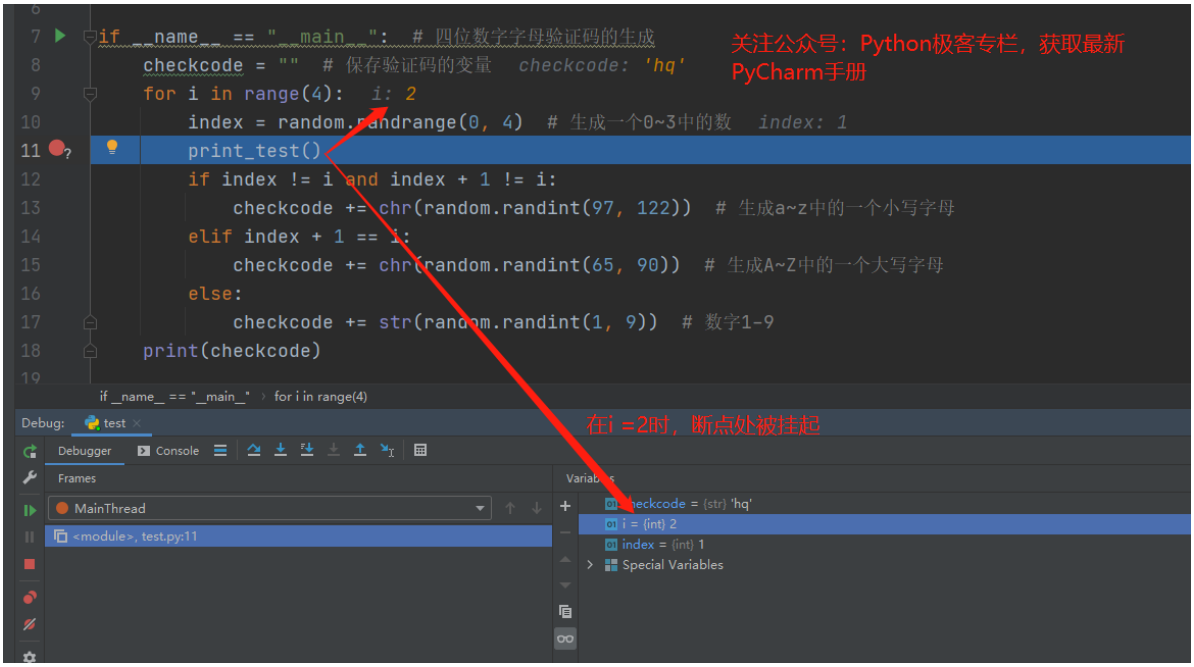


通过配置`Condition`中的内容，当程序符合`Conditioin`中的条件时，才会在当前断点暂停（挂起）

比如，上面代码，我设置`i = 2`时，在`print_test()`函数挂起，设置如下，然后点击`Done`设置生效



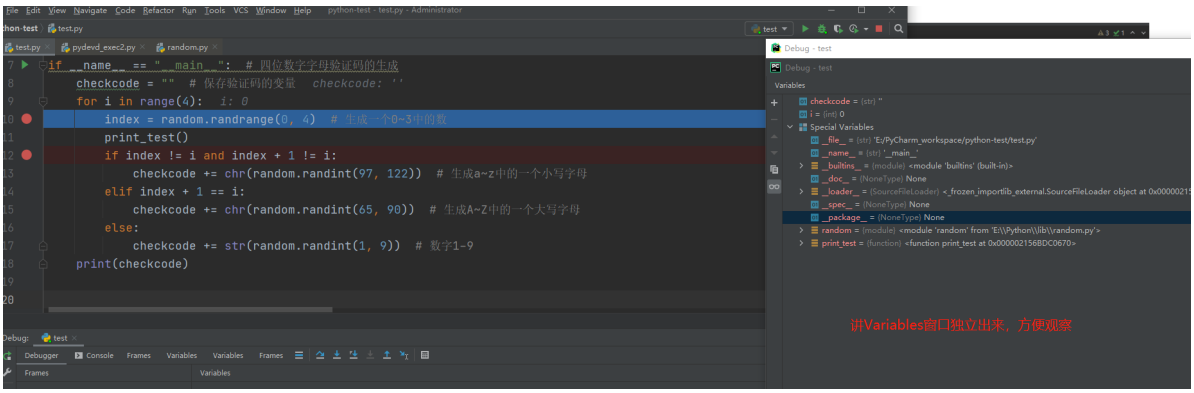
DeBug执行代码，效果如下



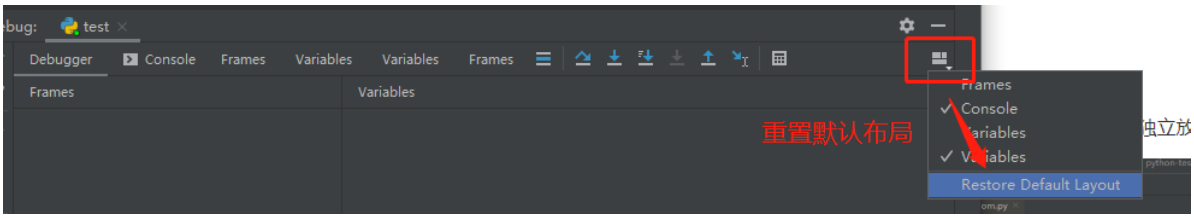
如果DeBug执行多线程的时候，可以指定线程名，专门针对某个线程进行DeBug.

9、Debug窗口悬浮

如果有多太显示器，或者想要将Debug窗口独立放大，可以点击相应标签，左键长按进行拖动



如果找不到相应窗口，或者布局比较乱，可以重置布局

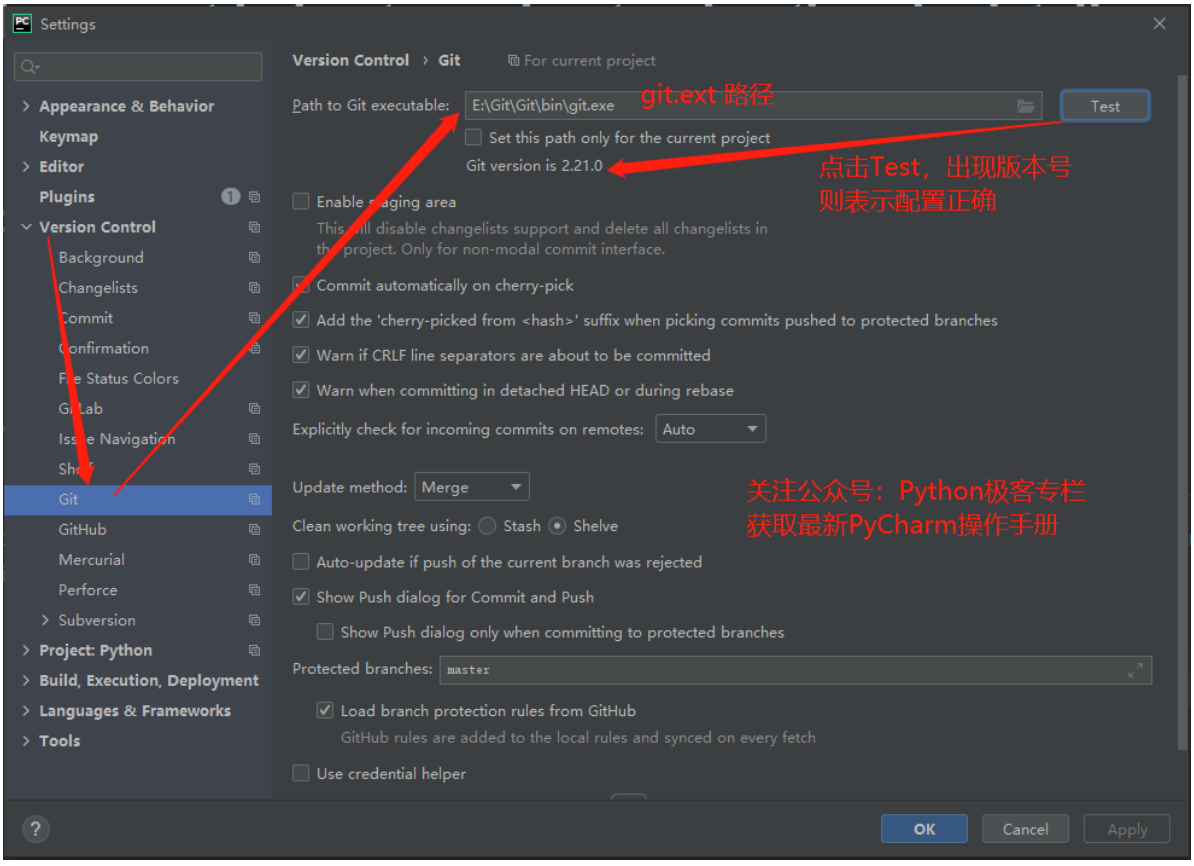


PyCharm版本控制

PyCharm集成了大部分流行的版本控制系统，如`Git`、`Subversion`、`Mercurial`、`Perforce`，以最常用的`Git`为例

Pycharm关联git

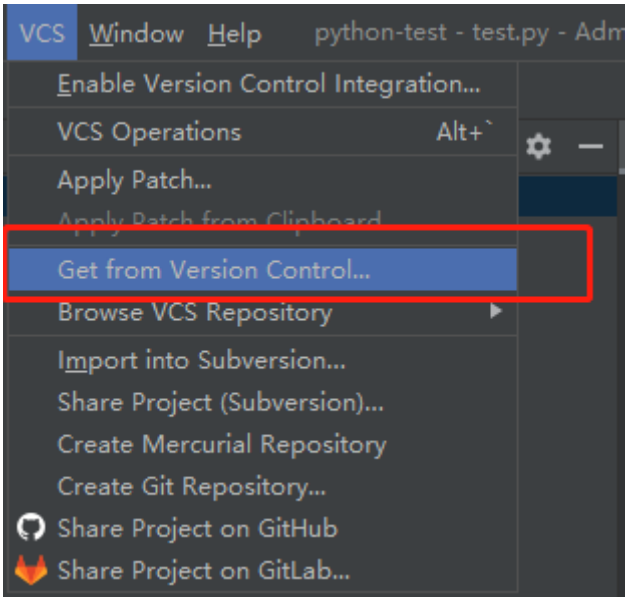
在`File - > settings`中，找到`Version Control -> Git`如下图所示,配置本地`git.exe`的路径



1、导入GitHub项目及配置

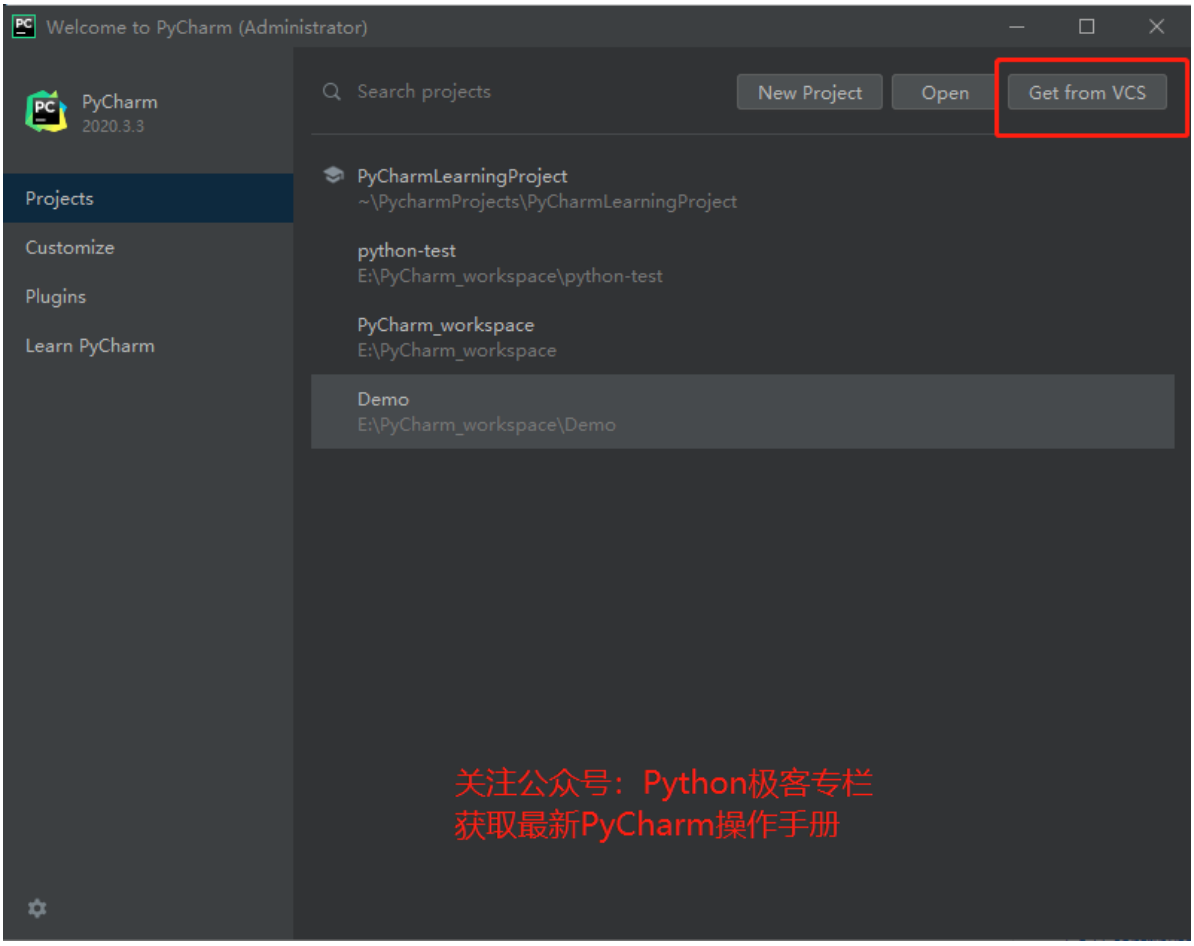
1、方法一

点击菜单栏中`VCS - > Get from Version Control`

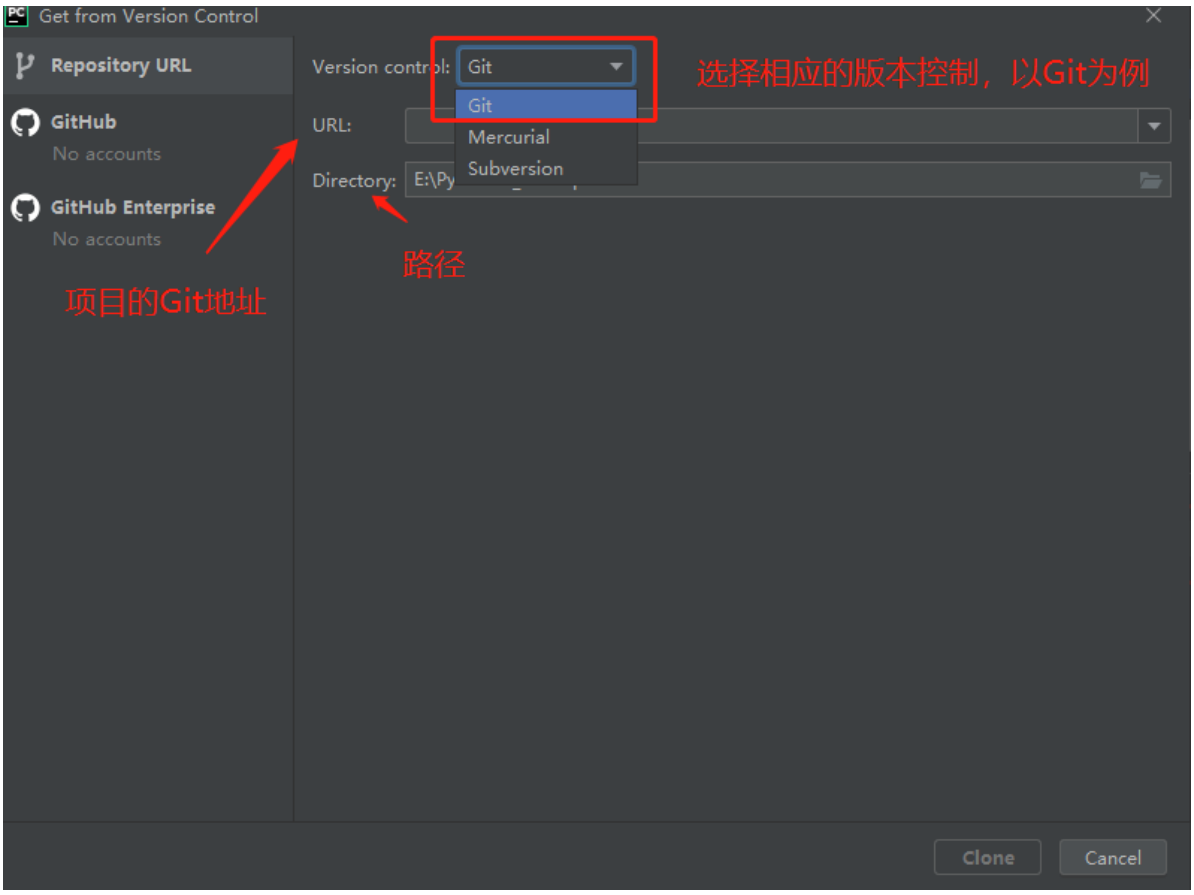


2、方法二

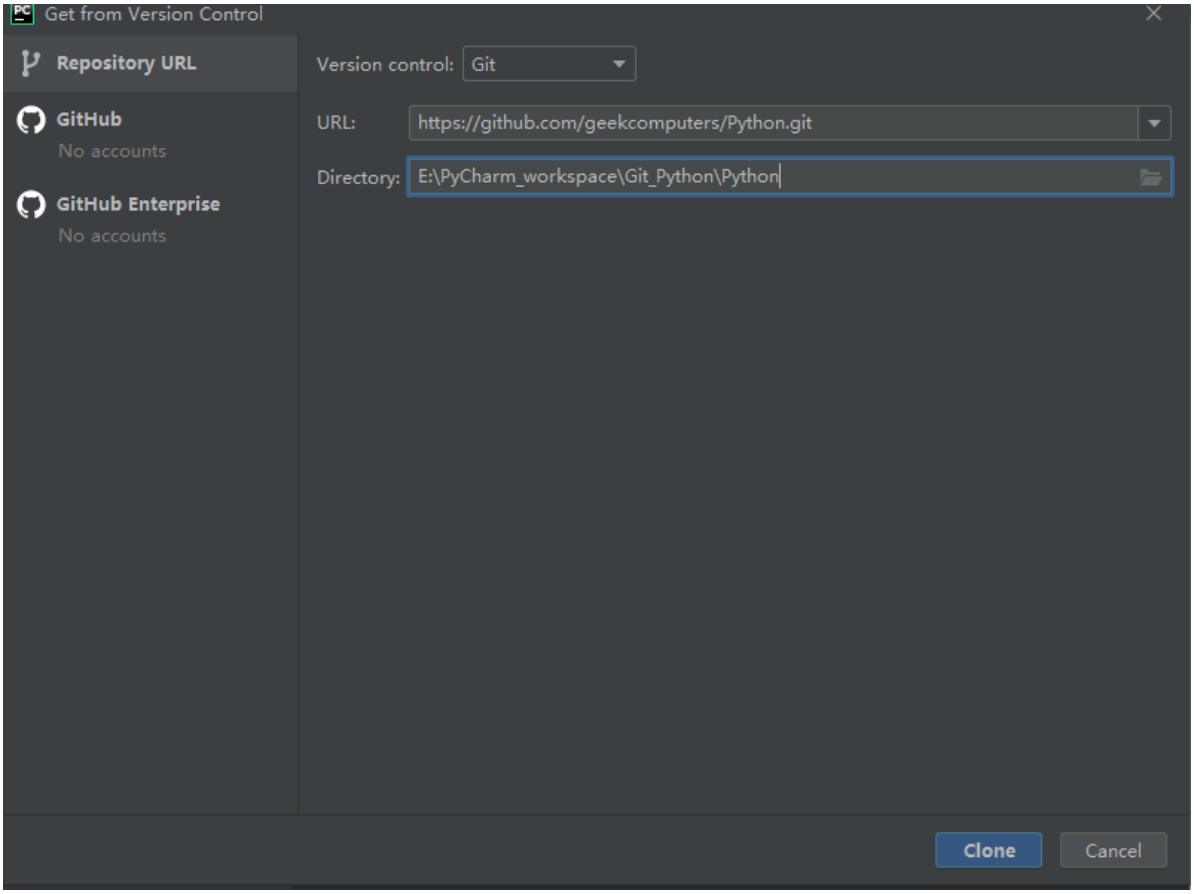
欢迎界面点击`Get from VCS`， 如果在项目中，可以点击`File-> close project`退出至欢迎界面



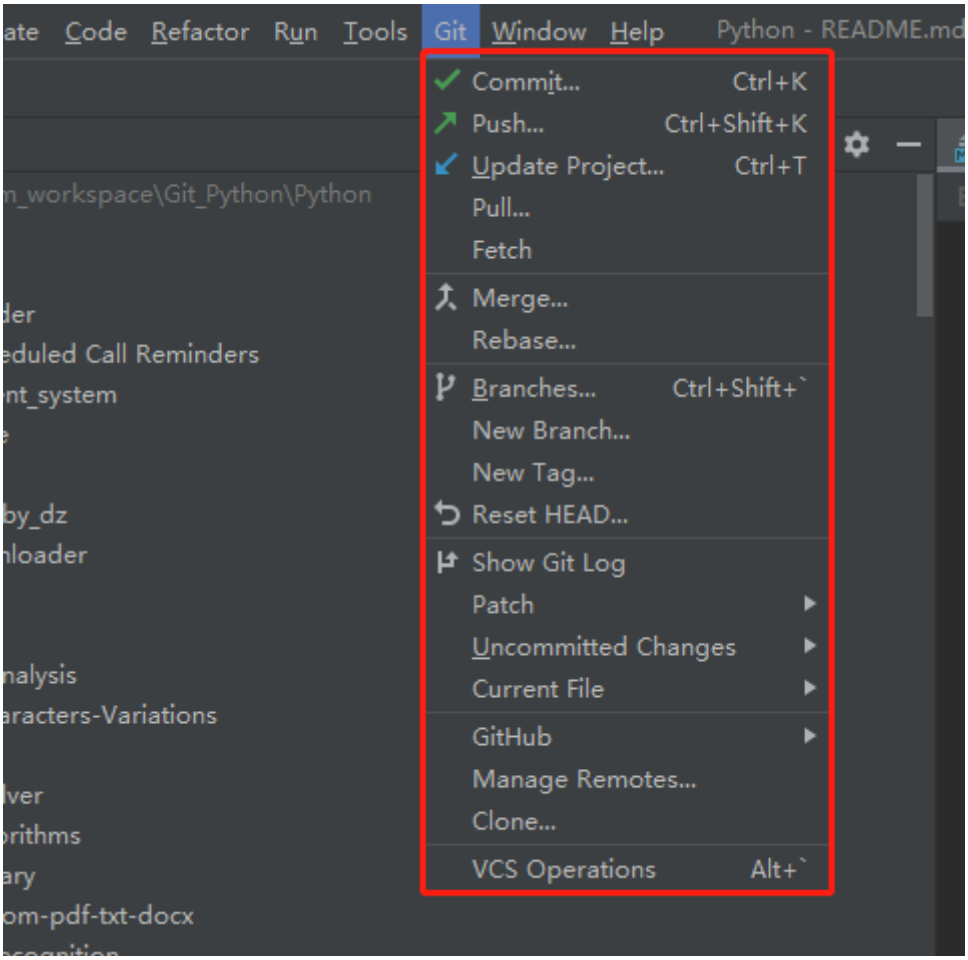
然后配置相关内容即可，如下图所示。



配置完毕后，点击`Clone`



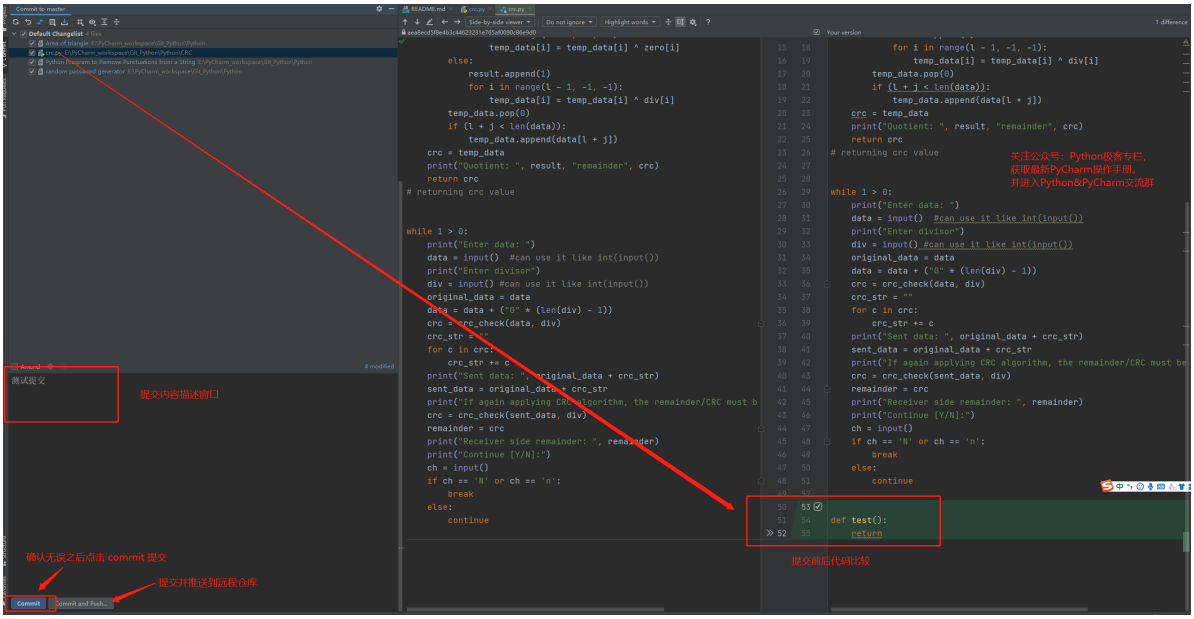
导入完毕，进入项目。我们会发现菜单栏的`CSV`不见了，取而代之的是`Git`



- `Commit：`将代码提交到本地仓库
- `Push：`将本地仓库代码提交到远程Git仓库
- **Pull**：将远程Git代码仓库同步到本地仓库
- `Branches：`查看项目的所有分支
- `New Branch：`创建新的分支
- `Show Git log:`查看Git的历史操作

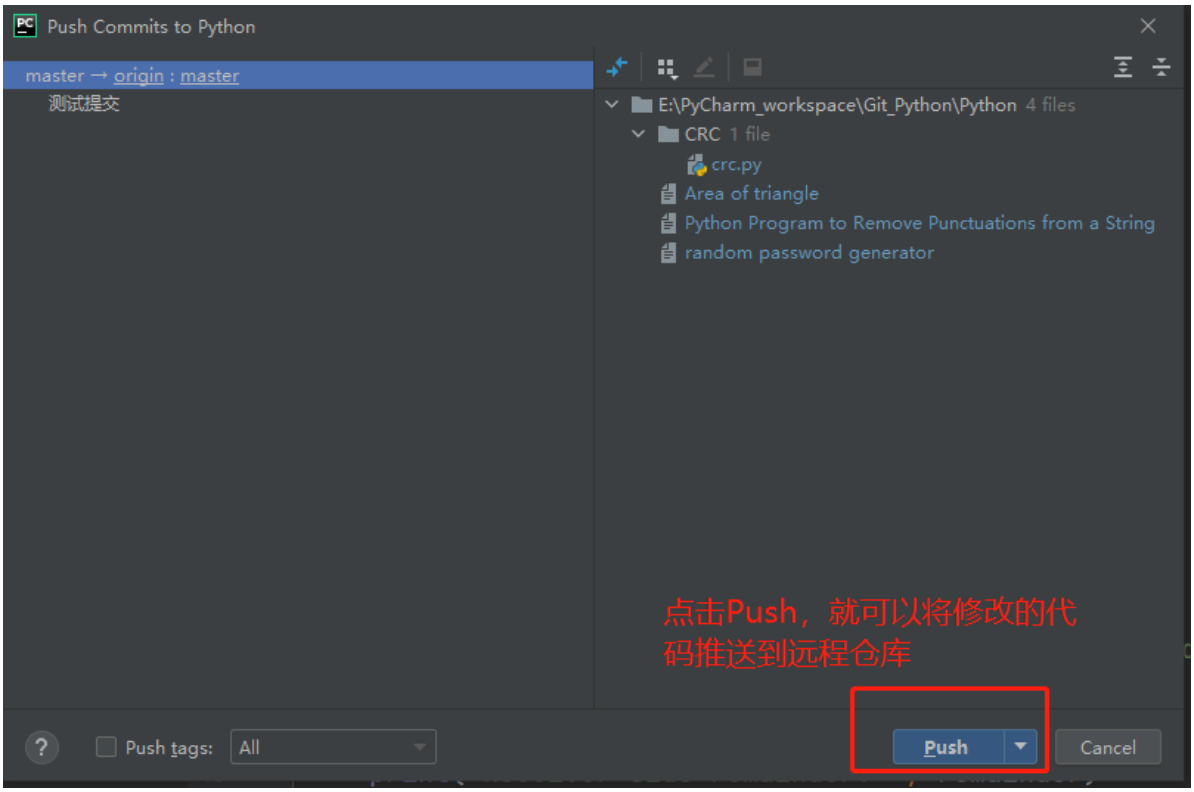
2、示例

在项目中的某个文件进行修改，然后点击`Git - > Commit`，快捷键`Ctrl + K`， 会出现如下界面

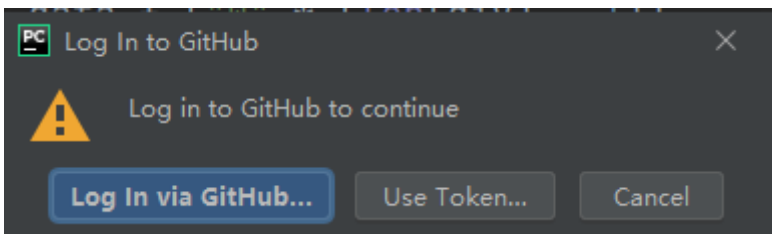


如果点击`Commit`需要再进行一次`Push`，或者直接点击`Commit and Push`操作完成。

`Push`路径在`Git-> Push`，快捷键`Ctrl + Shift + K`

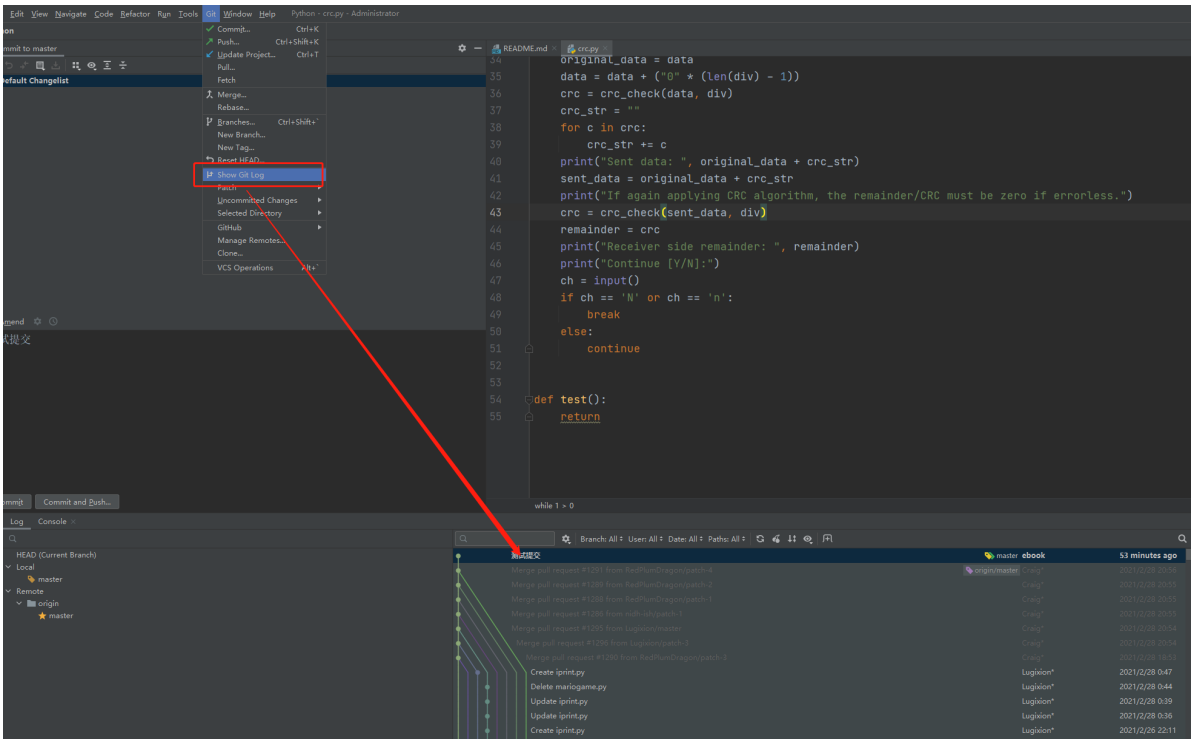


点击`Push`之后，会弹出一个验证窗口



点击`Log In via GitHub`会在浏览器中弹出一个验证窗口。点击`Authorize in GitHub`即可。

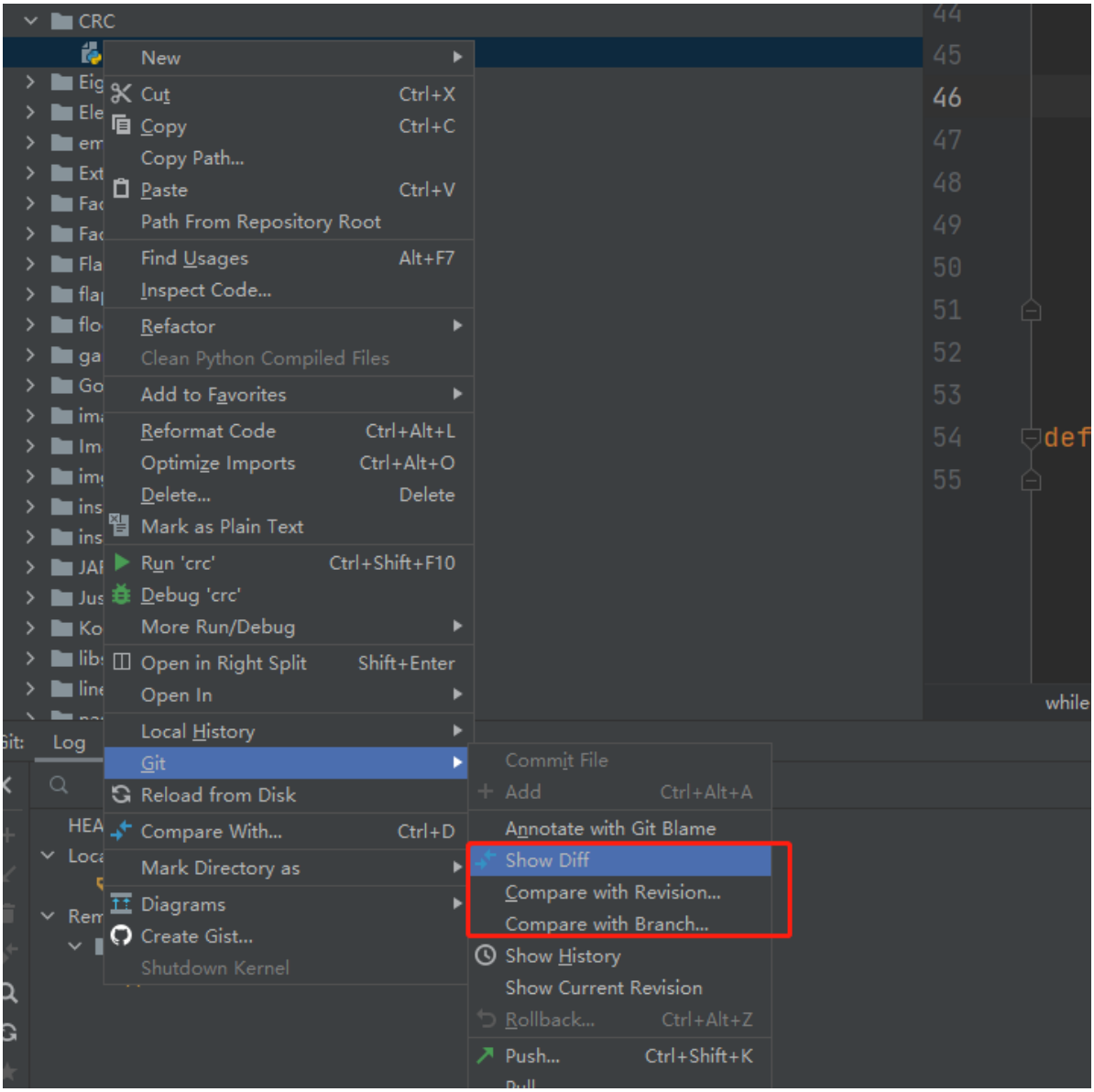
当前修改记录，可以在`Show Git Log`中查看



3、差异比较

方法一：

如果要比较某个`py`文件的差异，或者和历史版本进行比较，可以在文件上右键选择`Show Diff`、`Compare with ...`

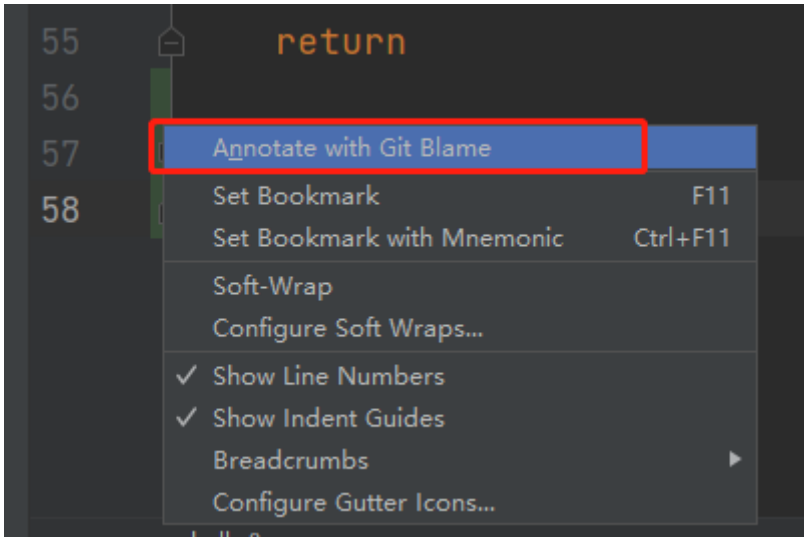


方法二：

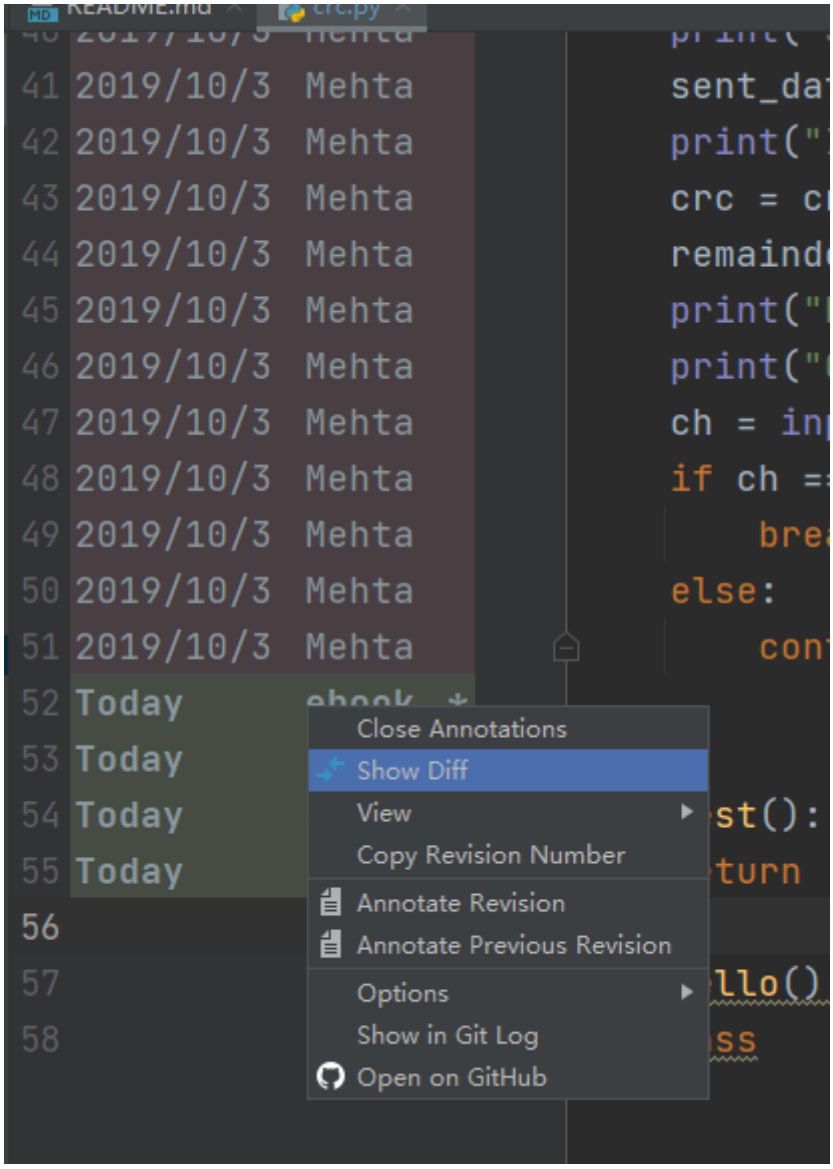
在修改的代码左侧有绿色的线条，表示这段代码被修改过

```
46          print("Continue [Y/N]:")
47          ch = input()
48          if ch == 'N' or ch == 'n':
49              break
50          else:
51              continue
52
53
54      def test():
55          return
56
57      def hello():
58          pass
```

在绿色线条上右键，



选择`Annotate with Git Blame`，可以看到所有历史记录。



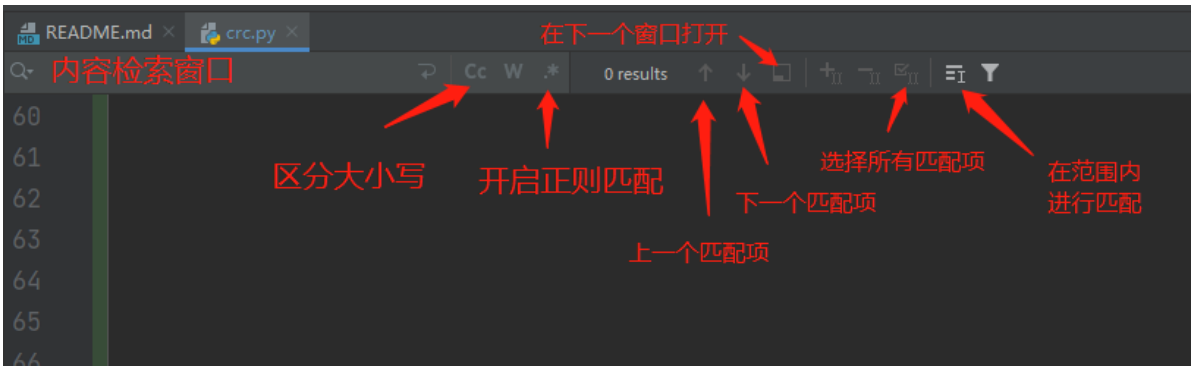
然后选择想要查看的，右键`Show Diff`即可查看到差异变化。

PyCharm搜索（文件、函数、内容）

Pycharm对搜索有很强大的支持，非常方便我们在项目中搜索某个关键词，或者函数等等

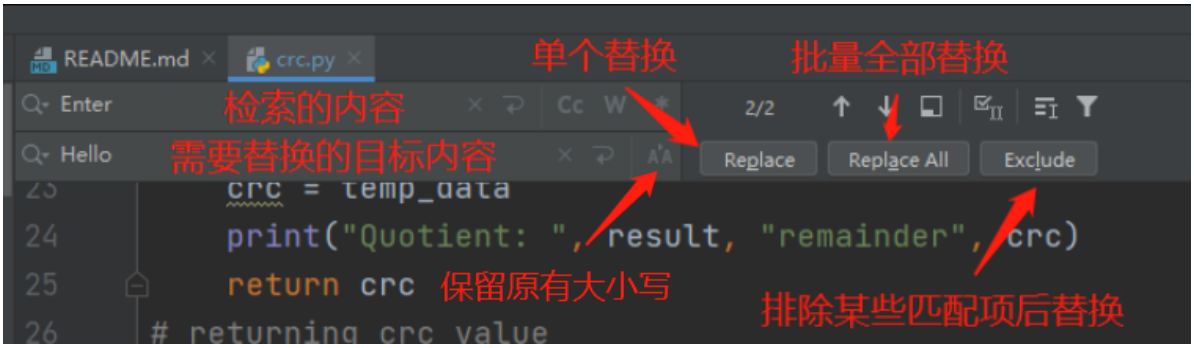
1、文件内检索

在文件内`Ctrl + F`， 如下图所示

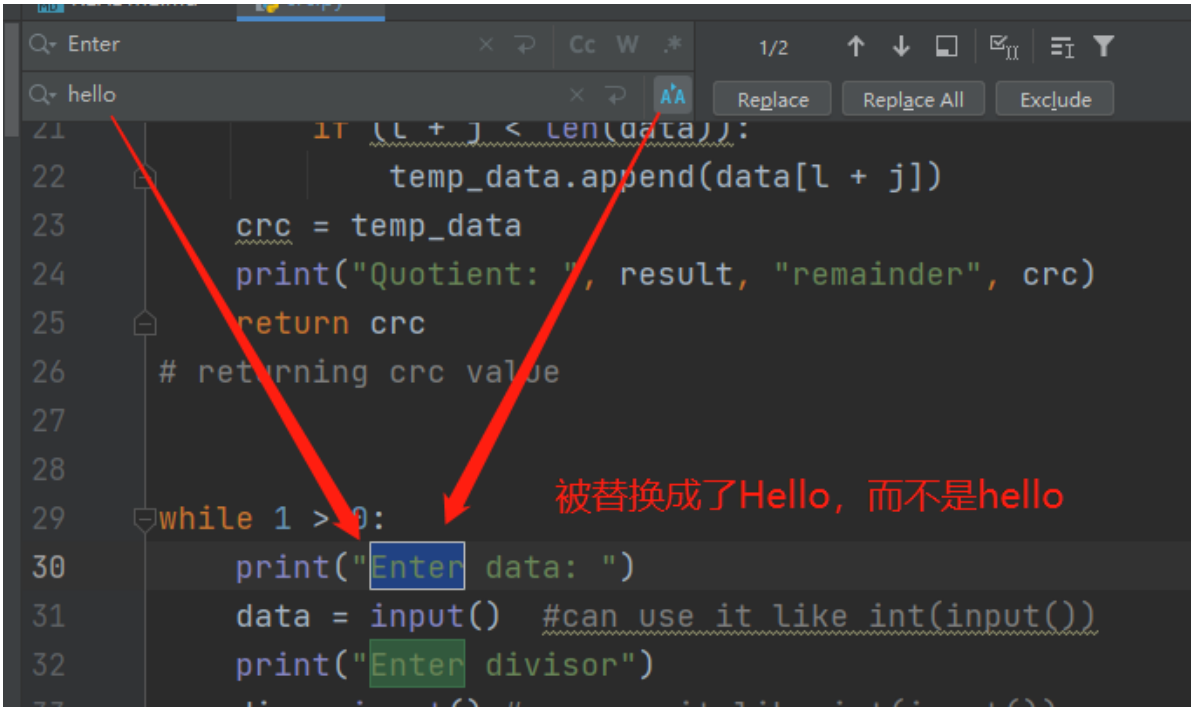


2、文件内替换

快捷键`Ctrl + R`， 将搜索到的内容替换成目标内容。



说明：保留原有大小写，比如原来的首字母是大写，替换之后仍旧保留首字母大写。如下所示

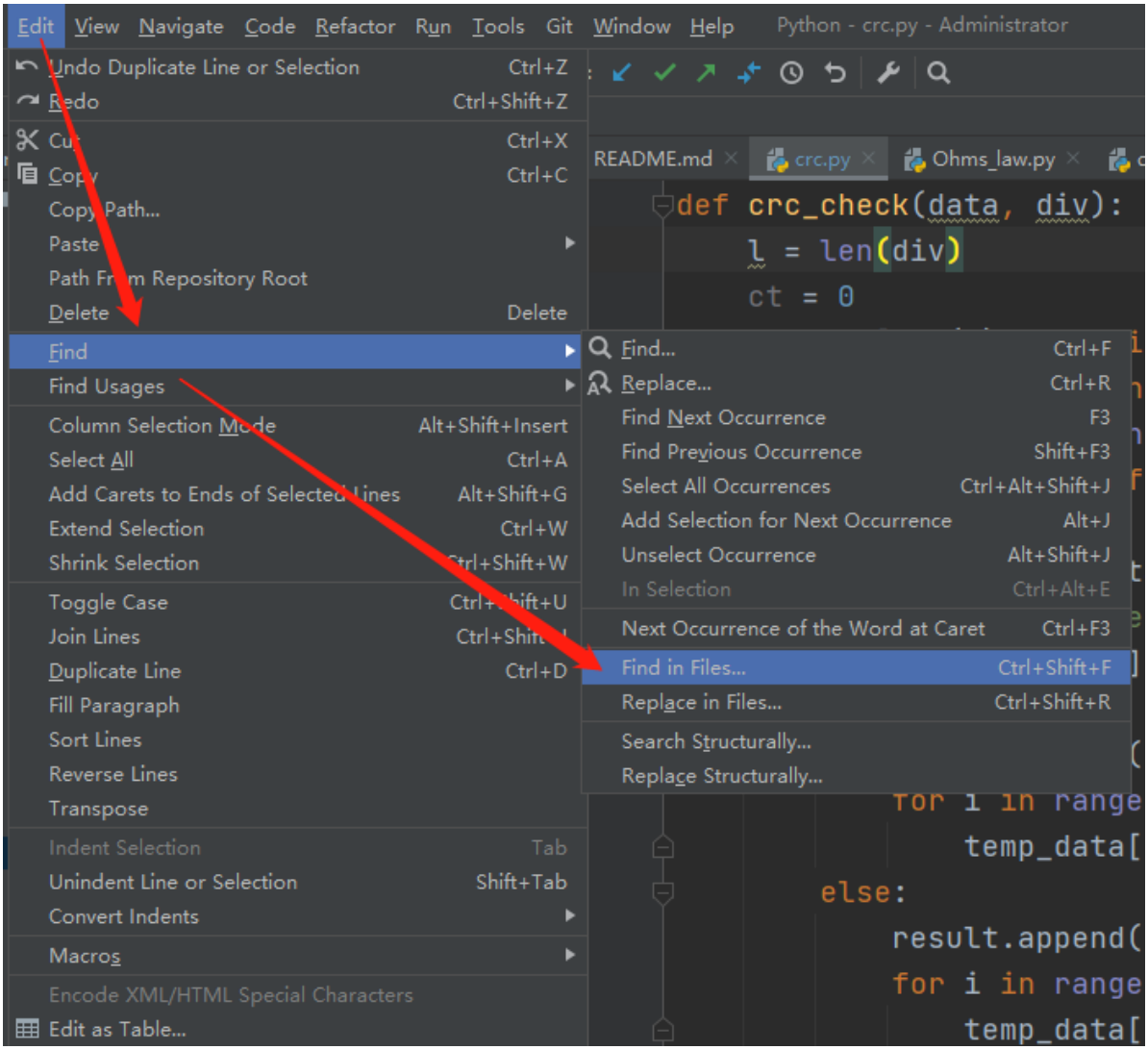


3、项目中查找

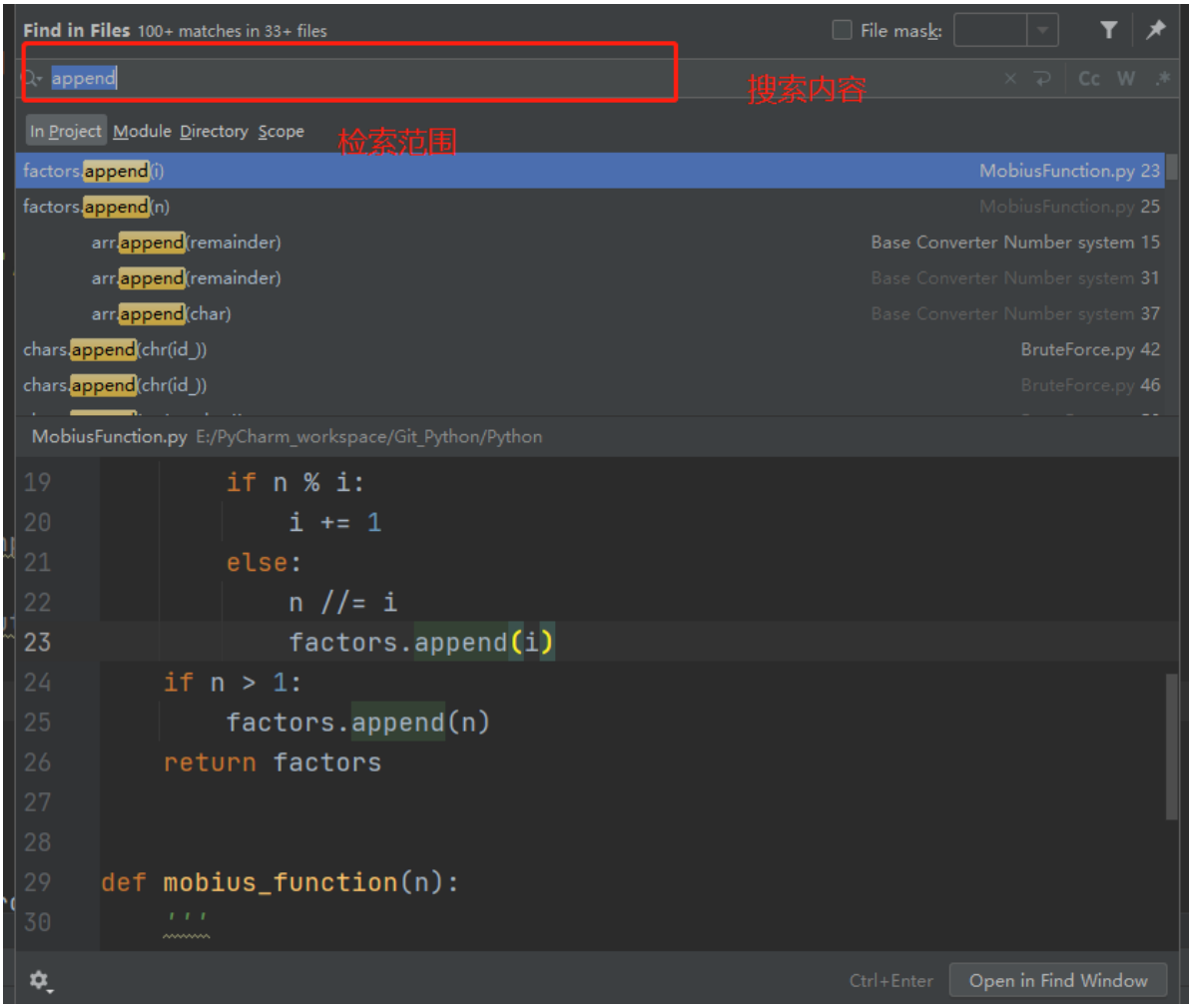
1、Ctrl + Shift + F

该快捷键容易冲突，比如本地如果安装了搜狗输入法，可以先将对应的快捷键关闭再使用。

或者通过菜单栏进入， 如下图

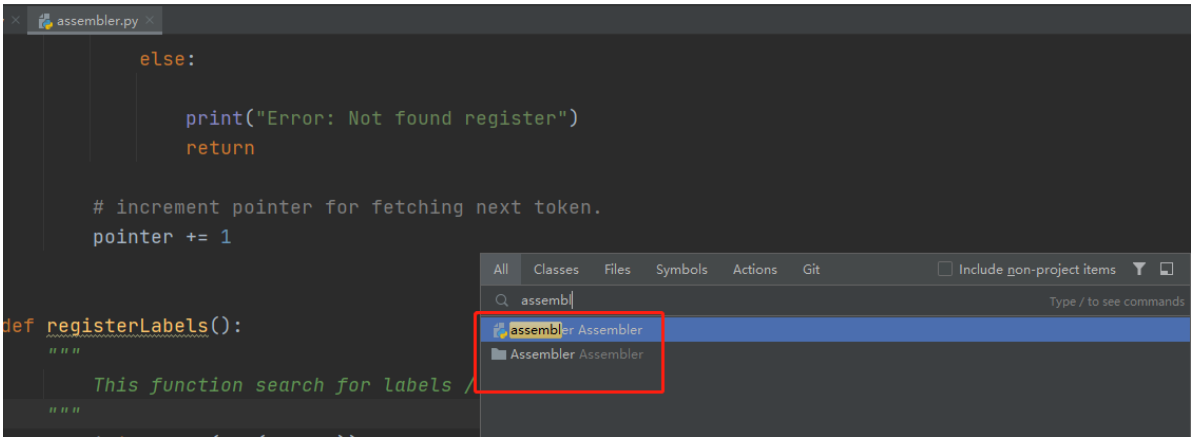


界面如下，可以检索出项目中的所有复合条件的结果。

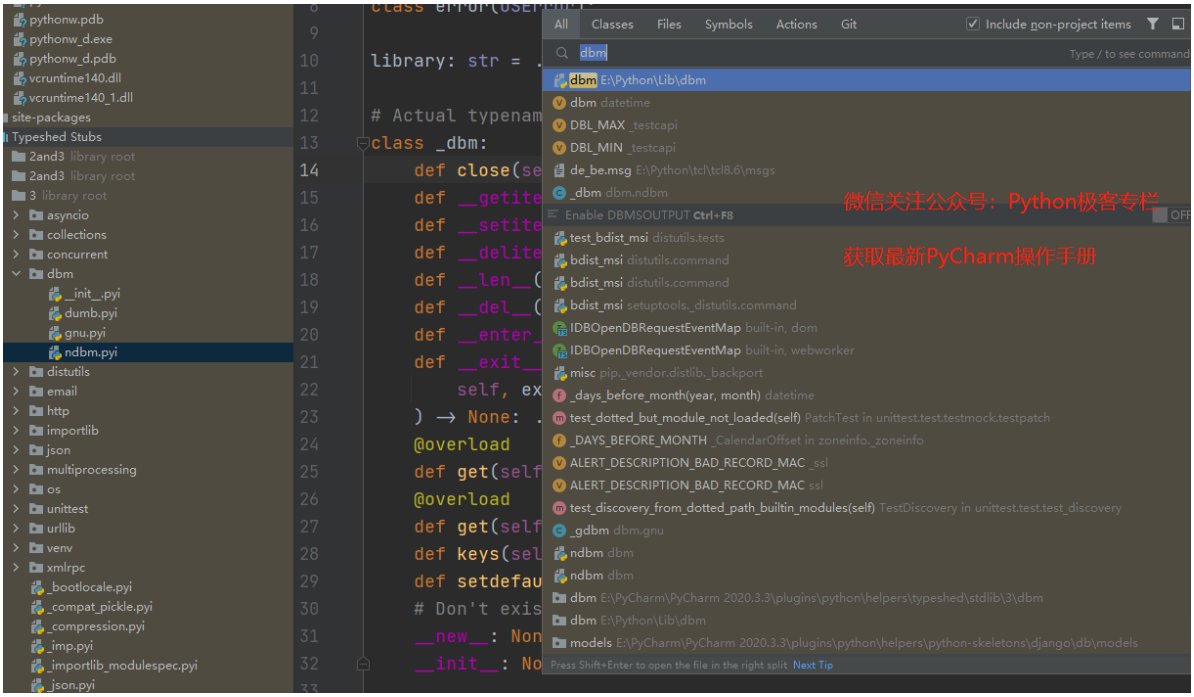


2、Shift + Shift

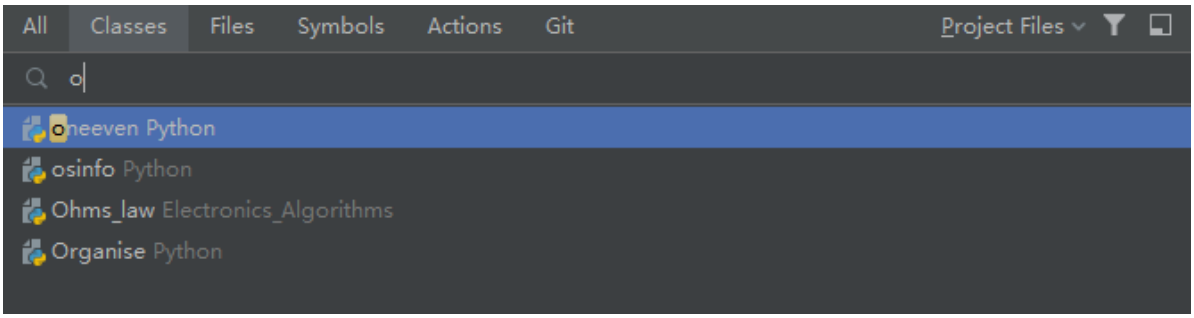
快捷键双击`Shift`，可以更精确的查找到类名/函数名/文件名



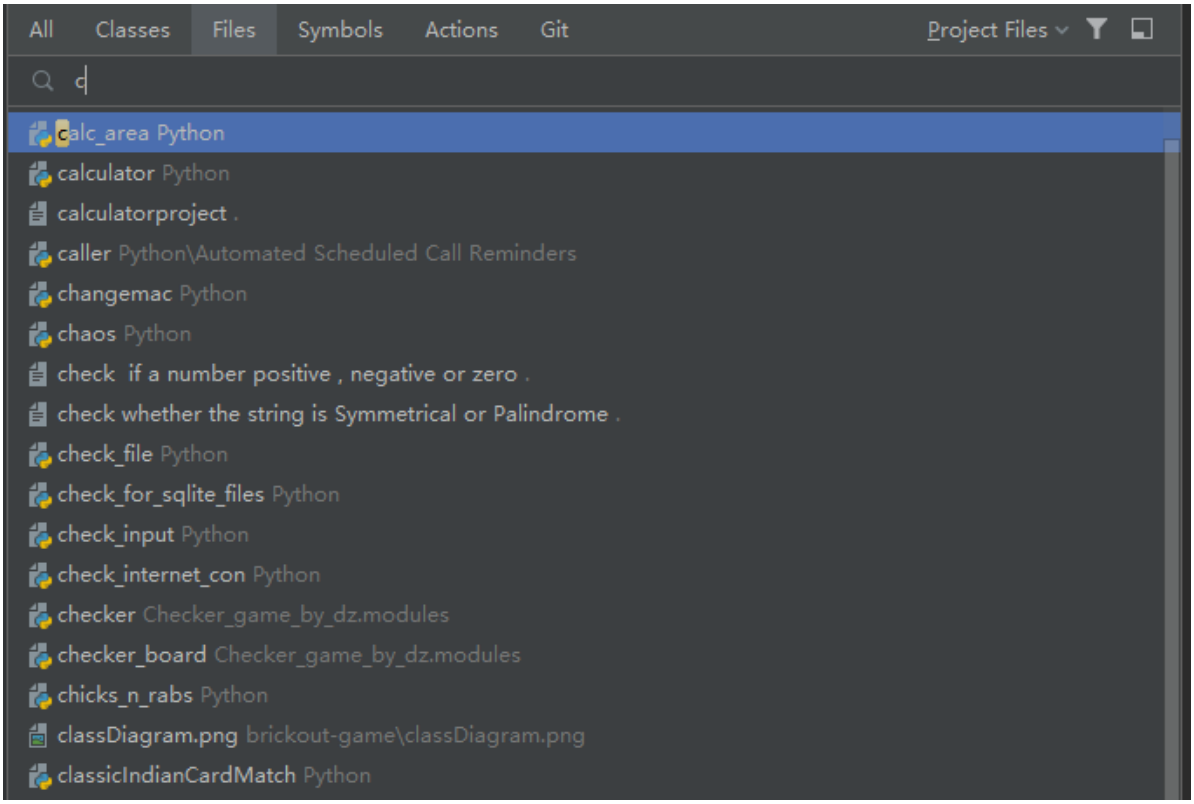
勾选`Include non-project items`，可以搜索项目代码之外的内容，比如引入的库



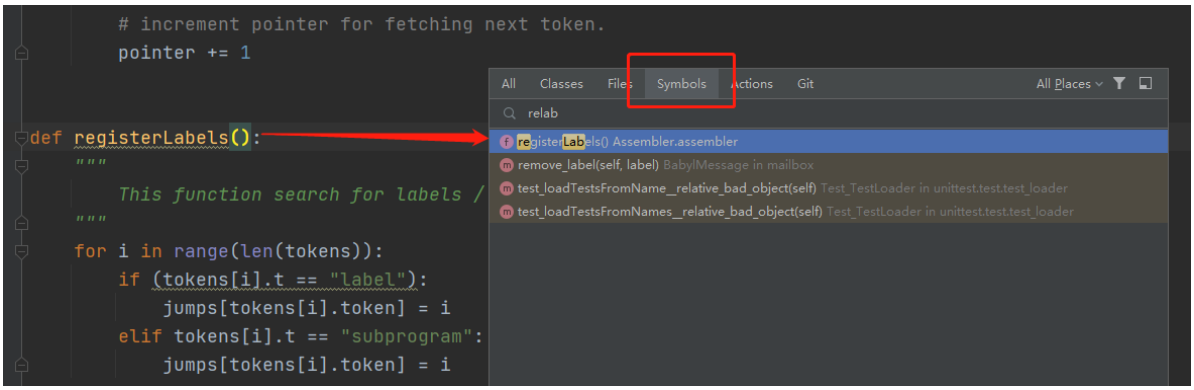
`Classes`搜索并跳转特定的类，快捷键`Ctrl + N`。



`Files`可以快速跳转到文件，比如我输入`c`，就会检索出所有与C相关的文件，快捷键`Ctrl + Shift + N`

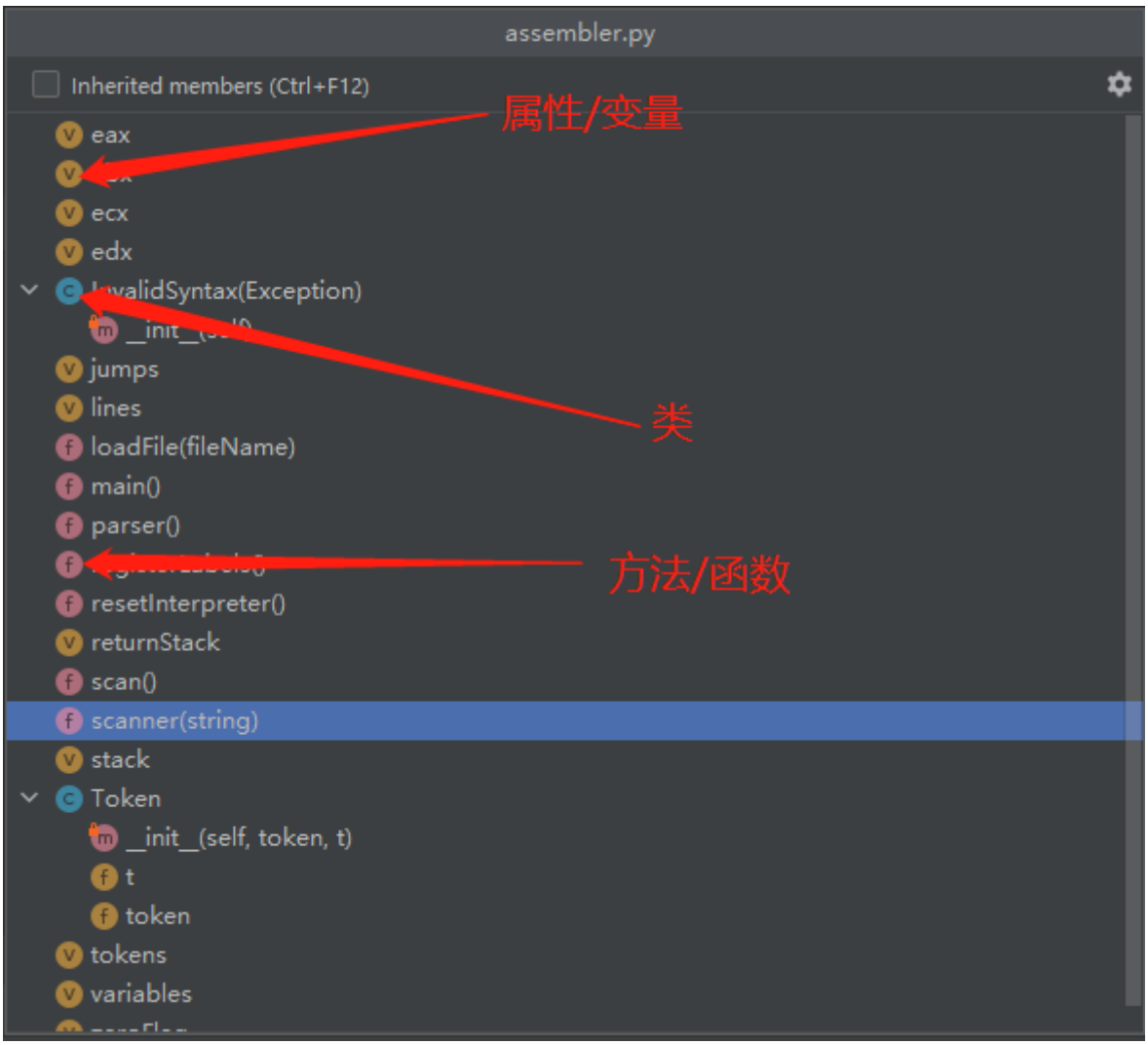


另外`Symbols`的模糊查询也非常实用。当记不清完整的关键词时，可以进行模糊搜索。快捷键`Ctrl + Alt + Shift + N`如下图所示

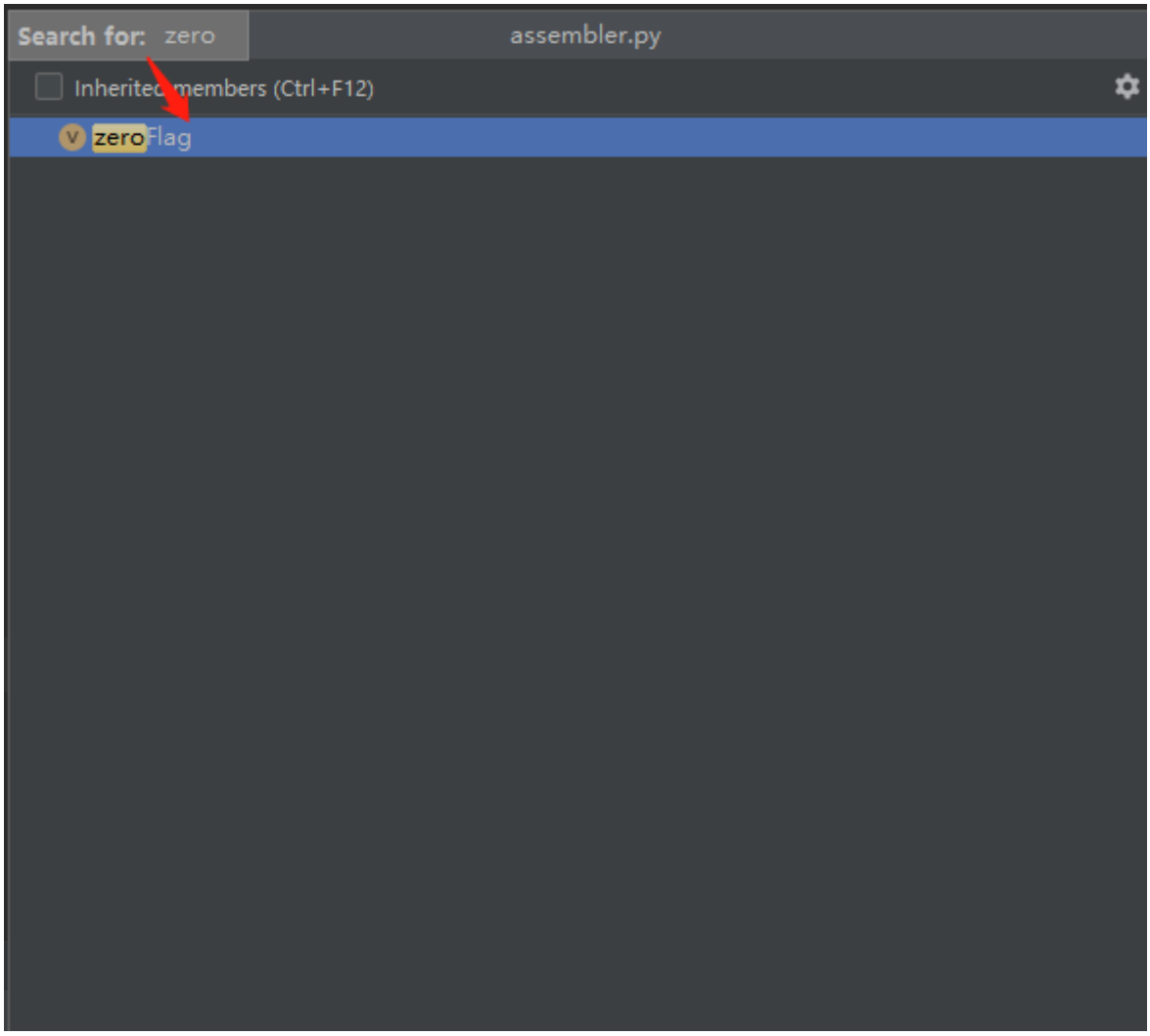


4、当前类、方法、属性列表

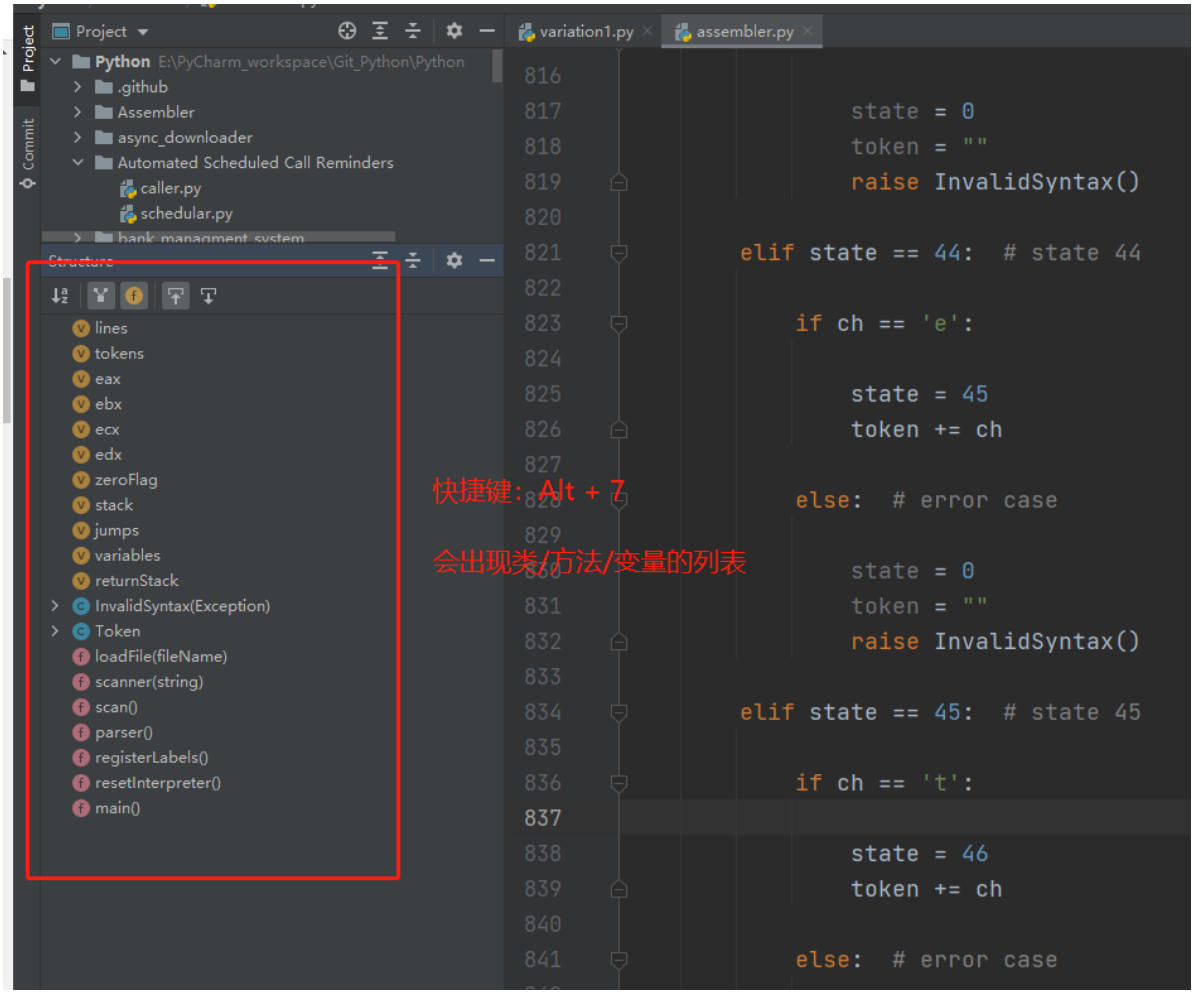
快捷键`Ctrl + F12`， 可以把当前文件中的所有属性、类、方法都显示出来



直接输入关键字，就可以检索出符合条件的属性/类/方法，并且可以定位到相关位置

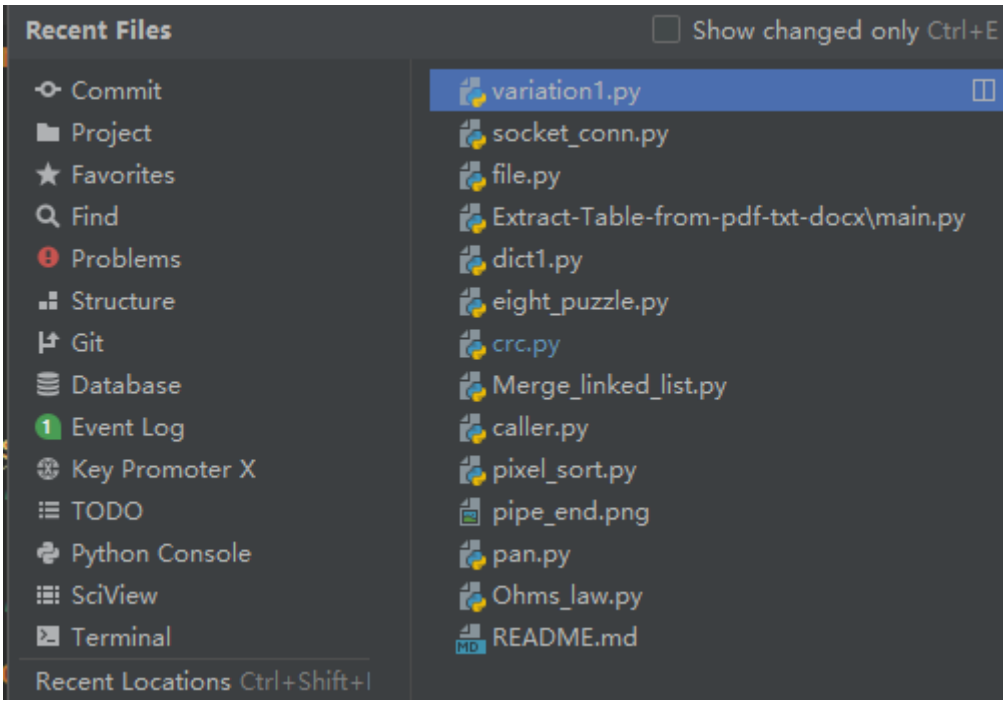


同样的功能，也可以通过`Alt + 7`来实现，如下图所示。同样也是直接直接输入关键字进行搜索。



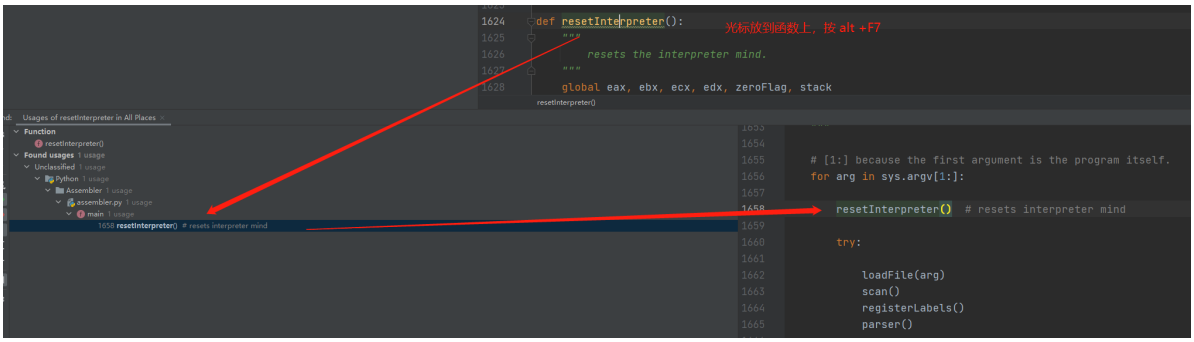
5、查看最近修改的文件

快捷键`ctrl + e`。可以查看最近修改的文件



6、查看函数的调用关系

当一个函数不知道被哪些地方调用的时候，可以通过快捷键`Alt + F7`进行查看，效果如下图



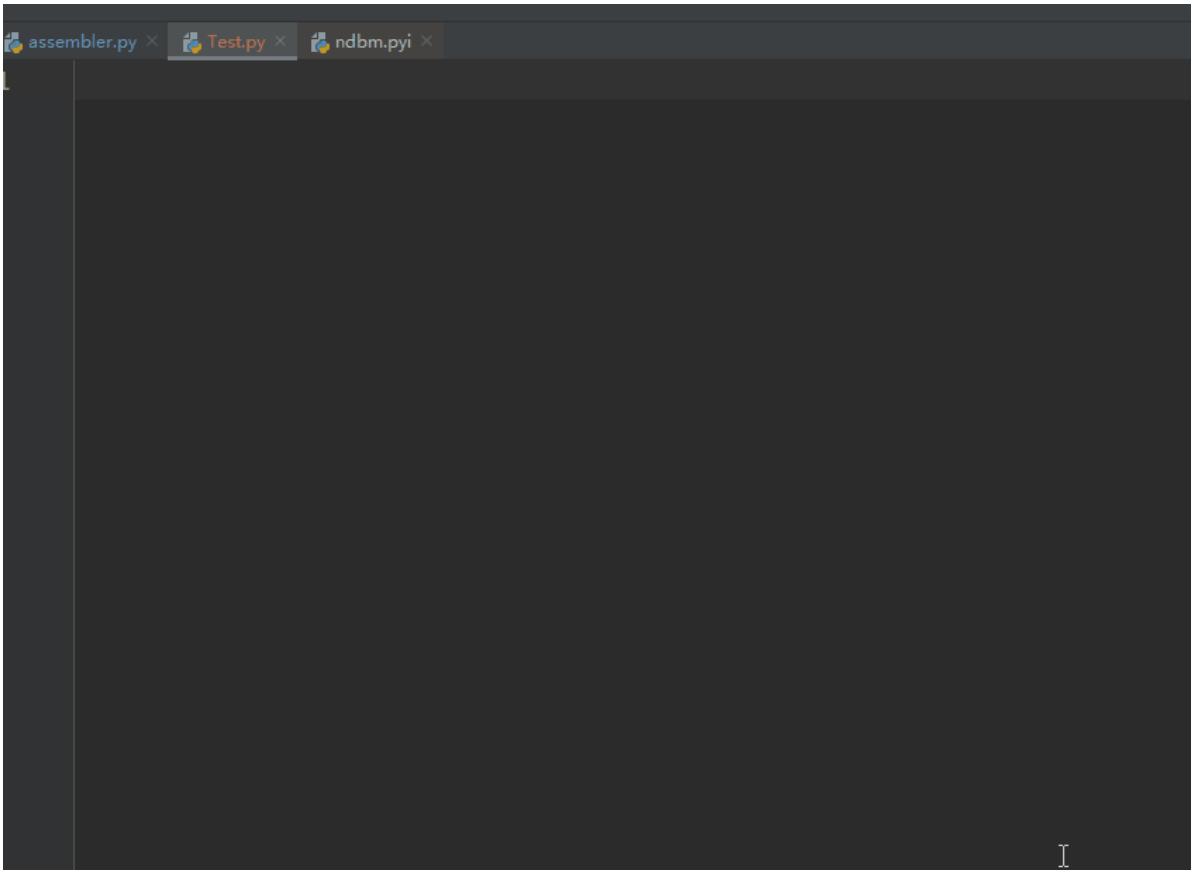
常用小技巧

1、快速导入依赖

平时我们导入第三方库时，会回到文件的开头写上`import xxx`。

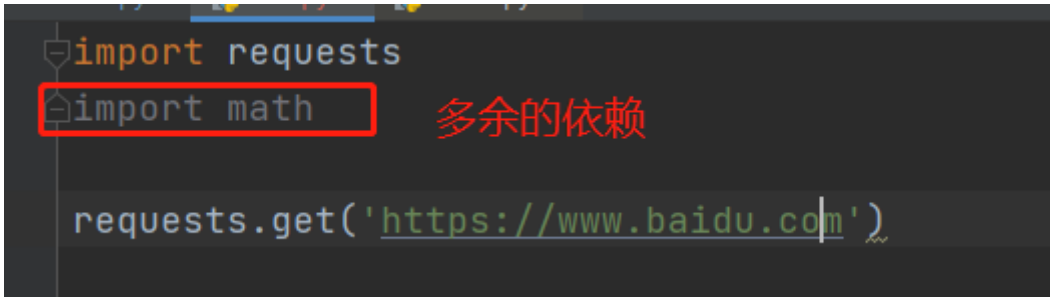
其实可以通过快捷键`Alt + Enter`进行快速导入，但是前提是你本地已经安装过这个库。

键盘敲入`Alt + Enter`或者点击下图中的红色小灯泡，选择`import this name`然后选择对应的包即可

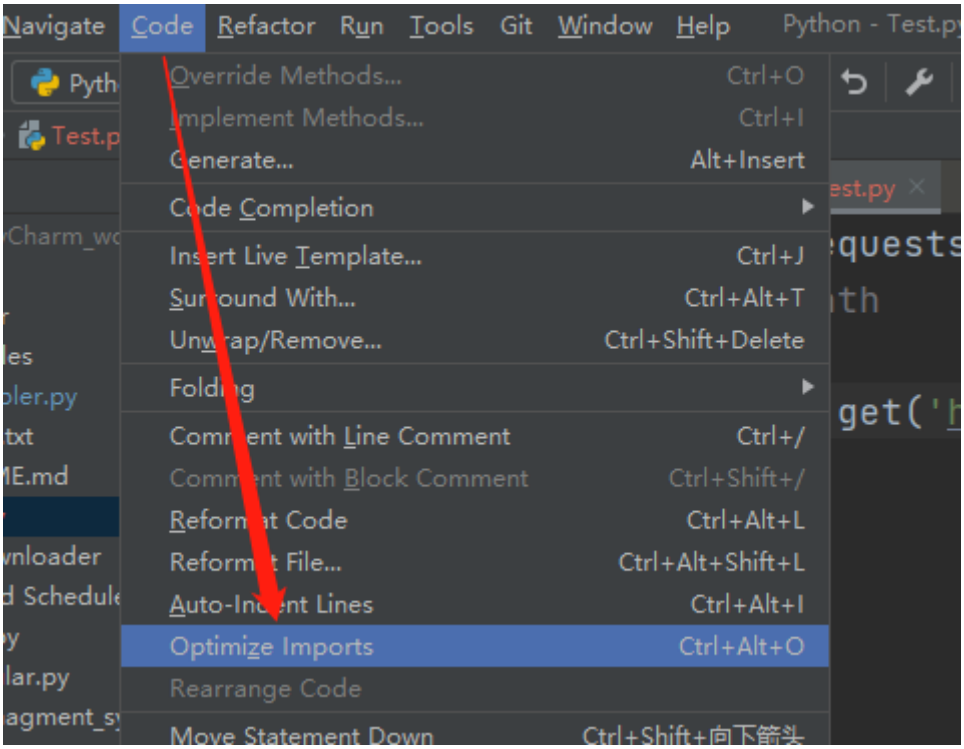


2、快速删除冗余依赖

有时候项目中有些无用的依赖，会呈灰色现实。

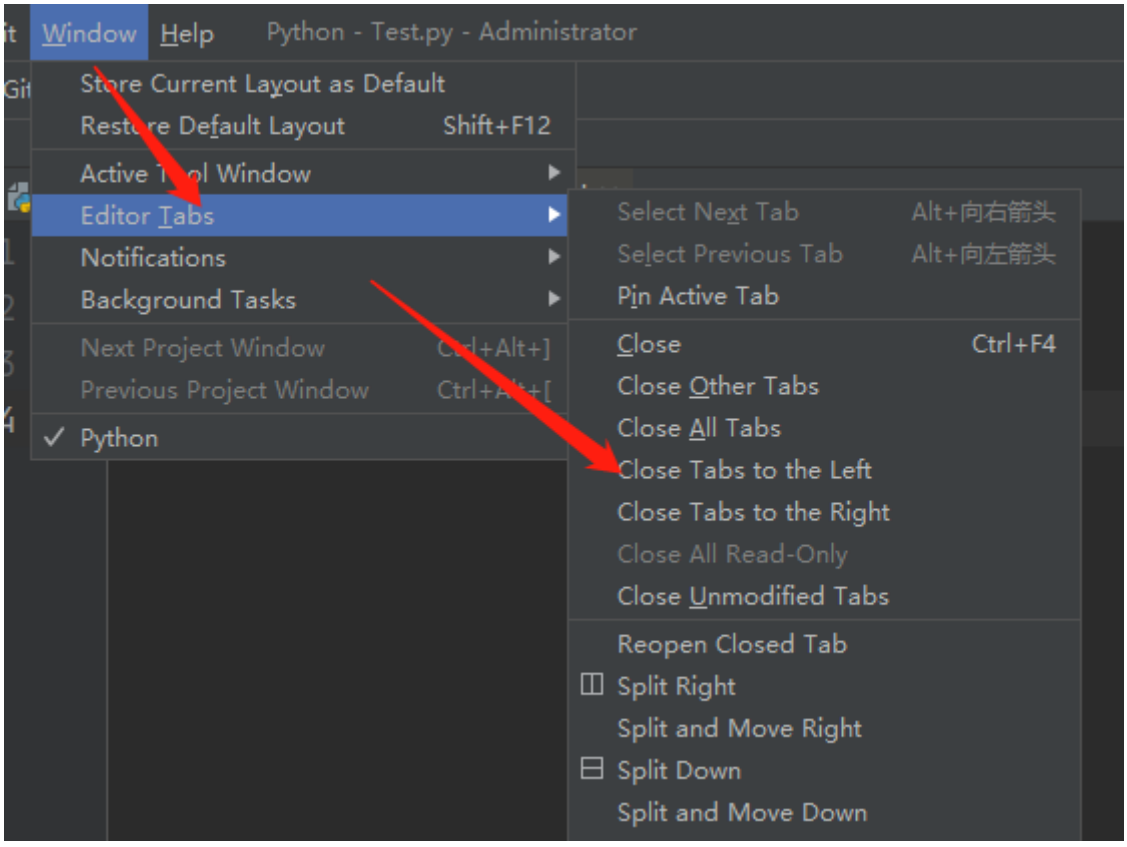


这个时候不需要一个个手动删除了，只需要点击`Code - > Optuimize Imports`即可，当然也可以通过快捷键`Ctrl + Alt + O`来快速清除

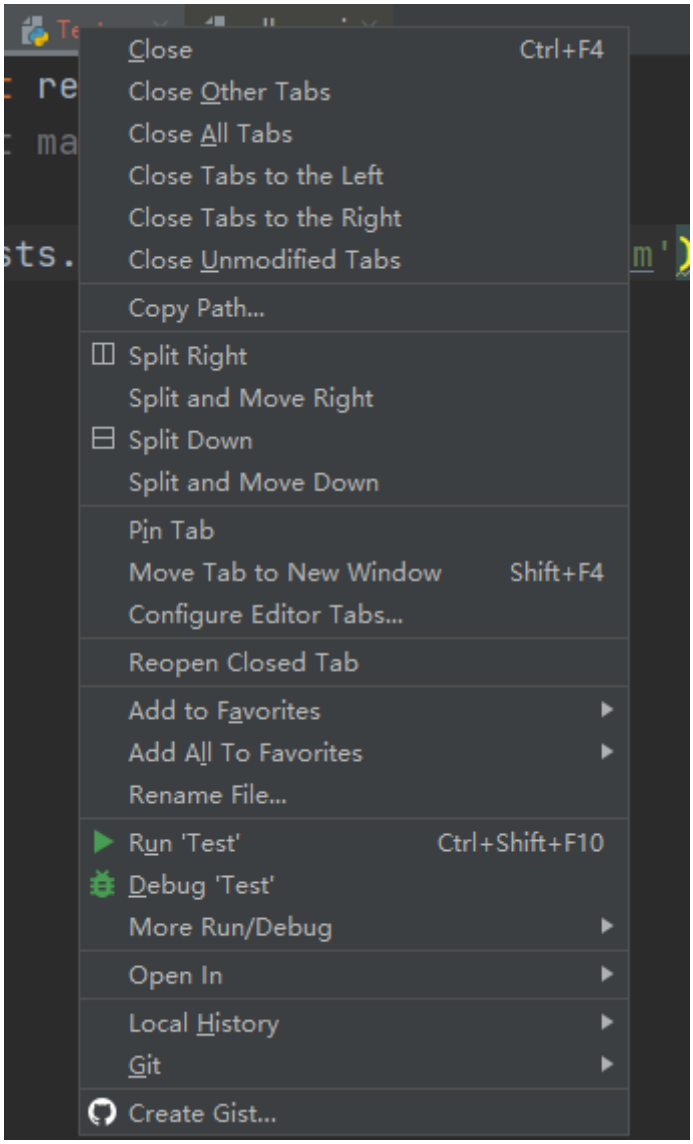


3、编辑器窗口管理

前面提到在菜单栏中可以操作窗口

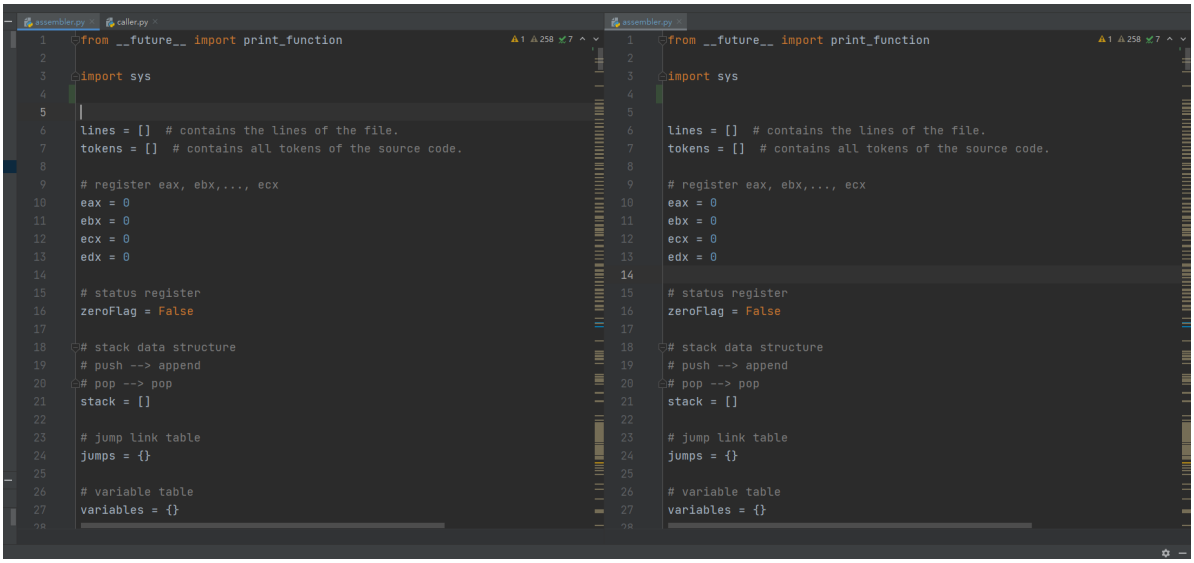


但是这样操作很麻烦，直接在代码的选项卡上右键就可以操作。

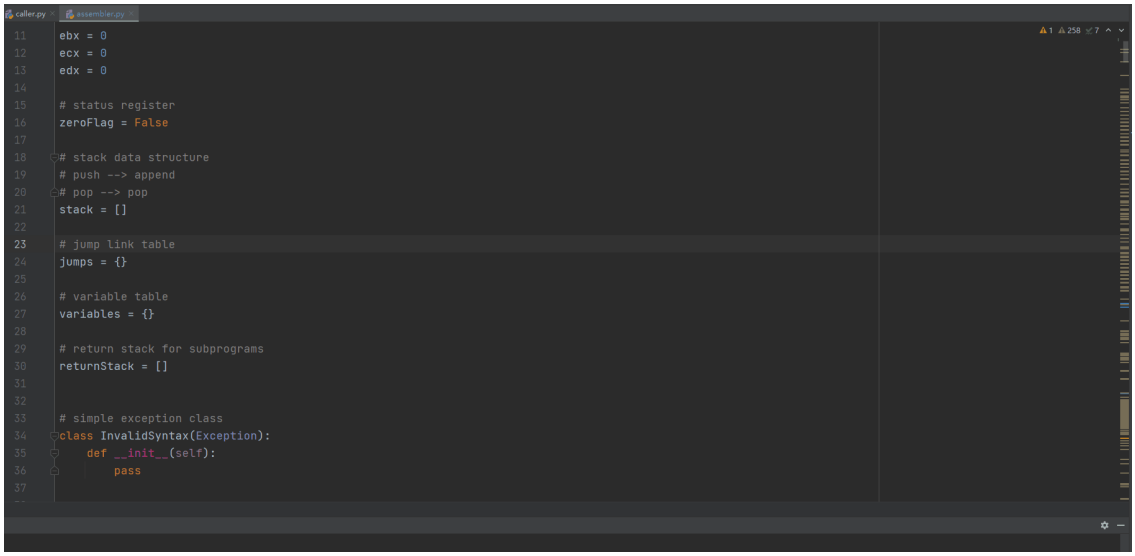


重点介绍几个

- `Close`：关闭当前窗口，
- `Close Other Tabs:` 关闭当前窗口之外的所有窗口
- `Close Tabs to the Lef:` 关闭当前窗口左侧的所有窗口
- `Close Tabs to the Right:` 关闭当前窗口右侧的所有窗口
- `close unmodified tabs:`关闭没有修改过的窗口
- `Copy Path...:` 复制文件路径，可以选择文件的绝对路径/文件名 复制
- `Split Right:` j将当前选项卡分割到右侧（并不会从当前窗口移除），可以分割成两个或多个窗口



- `Split and Move Right:` 将当前选项卡窗口移除，并分割到右侧，也可以鼠标长按选项卡进行拖动实现



4、任意代码块折叠

通常情况下，遇到代码块，PyCharm会有折叠的标识

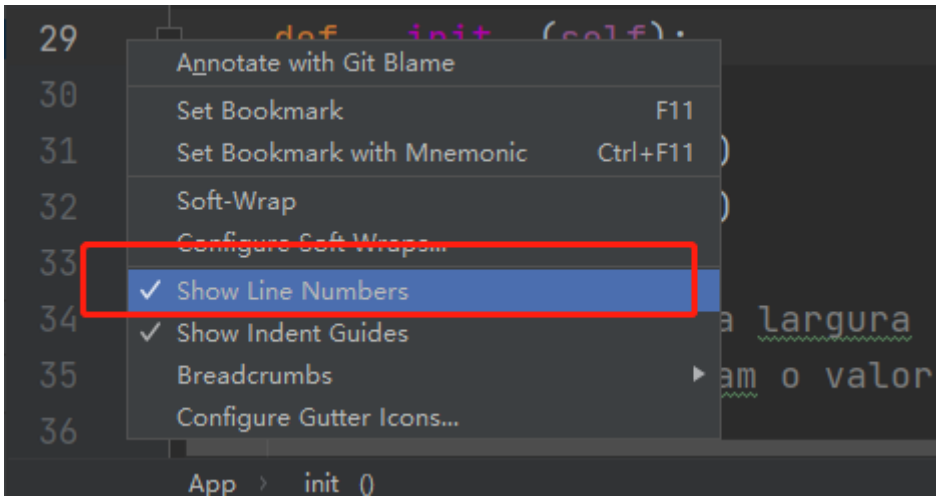
```
class Node:
    def __init__(self,state,depth = 0,moves = None,optimizer=0):
        self.object_ = '''
            Parameters:
                state: State of Puzzle
                depth: Depth of State in Space Search Tree
                moves: Moves List to reach this state from initial state
                optimizer: Used for UCS Only
                    0 - Manhattan Distance
                    1 - Hamming Distance
                    2 - Combination of 0 and 1

            Returns: Node Object
        '''
        self.state = state
        self.size = len(state)
        self.depth = depth
        self.optimizer = optimizer
        if moves is None:
            self.moves = list()
        else:
            self.moves = moves


    def getAvailableActions(self):
        '''
            Parameters: Current State
```

如果对于任意代码块改如何折叠呢？比如我想要折叠12~18行的内容。

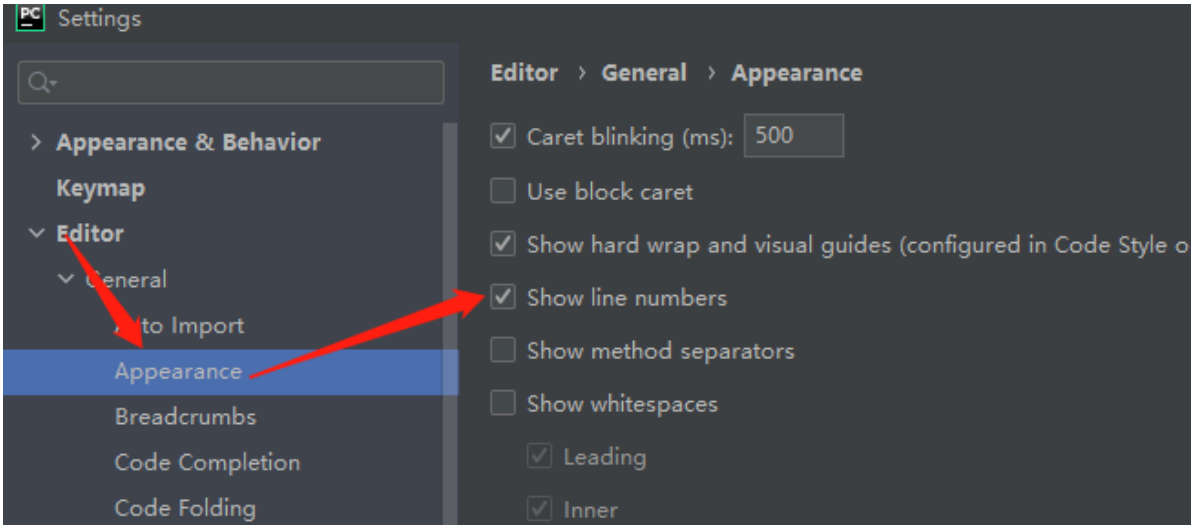
只需要选中相关代码在菜单栏`Code - > Folding - > Fold Selection/Remove region`中操作即可。

5、设置显示行数和分隔符

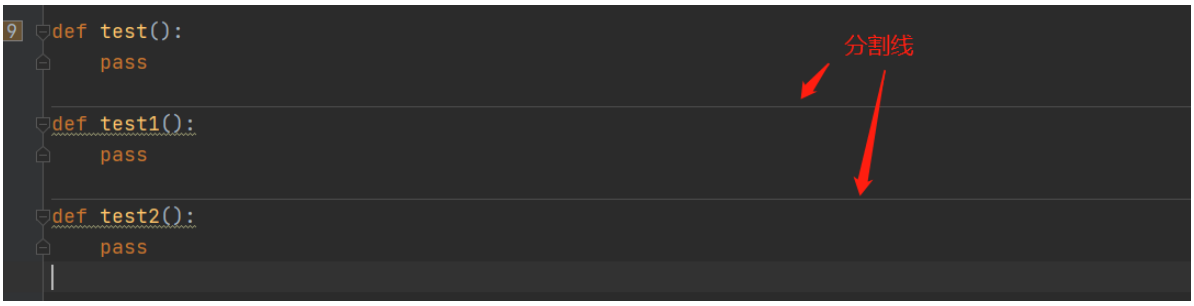
显示行数，可以直接在编辑侧边栏右键，选择`Show Line Numbers`



或者在菜单栏中进行设置，`File -> settings`中选择`Editor-> General-> Appearance`，如下图标识，勾选`Show line numbers`。

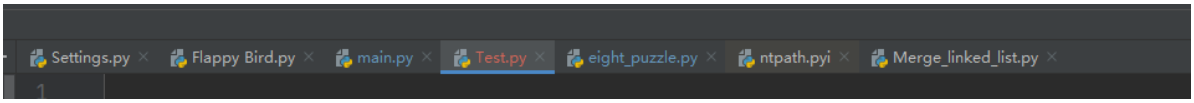


上图如果勾选了`Show method separators`则每个方法会有分割线提示，效果如下：



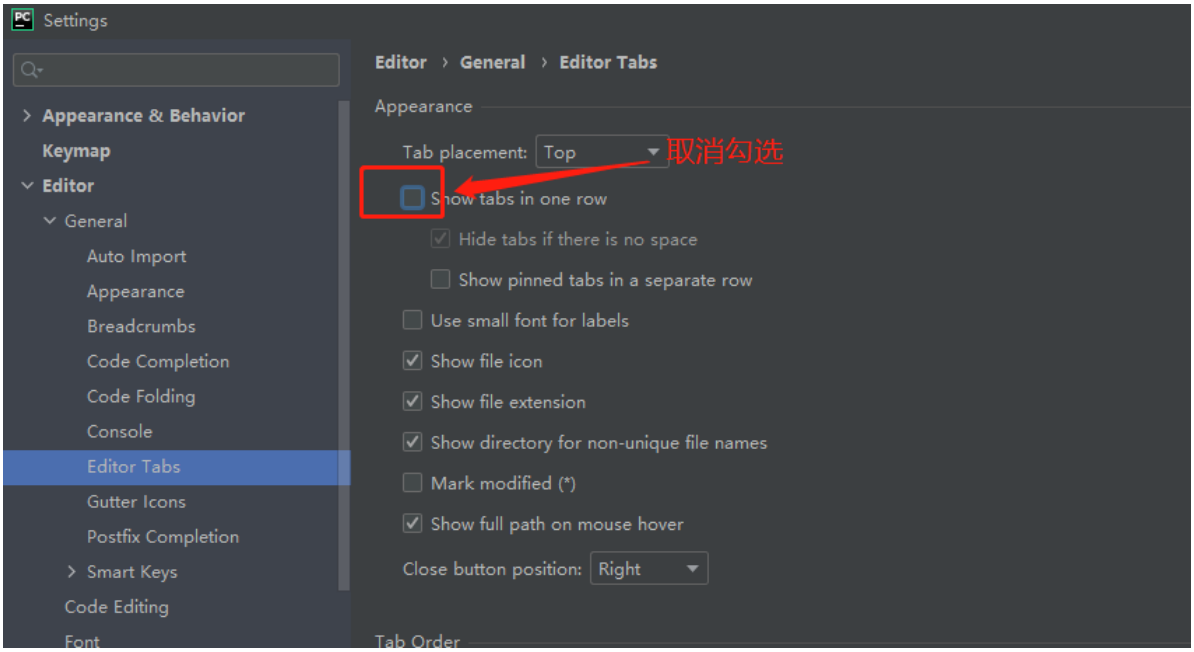
6、多行标签显示

默认情况，打开多个文件时，标签是一行显示的，如下图

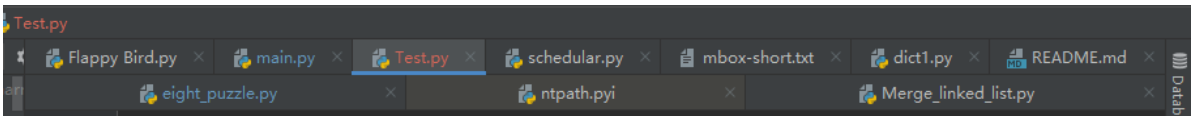


当打开的选项卡过多时，多余的会自动被折叠，不方便查看。

解决：在`File->settings`中，选择`Editor->Editor Tabs`取消勾选`Show tabs in one row`即可



设置完毕后，效果如下：



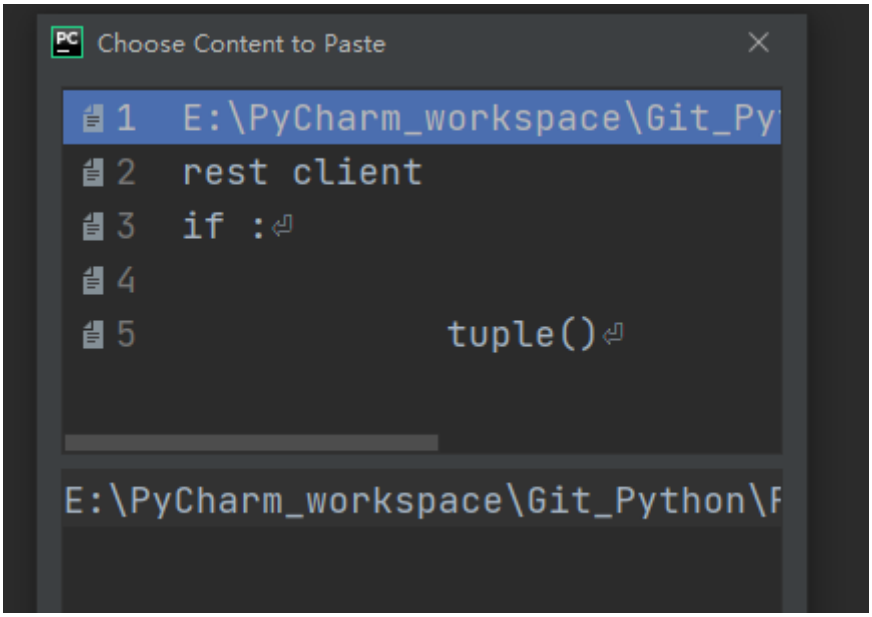
7、快速补全：，切换下一行

通过使用快捷键`Ctrl + Shift + Enter`快速补全`:`，并切换下一行。 不光是补全`:`，也可以进行收尾

8、粘贴板历史

CV大法好，`Ctrl + C/V`大家也都会用。但是如果想看历史复制的记录该怎么办呢?

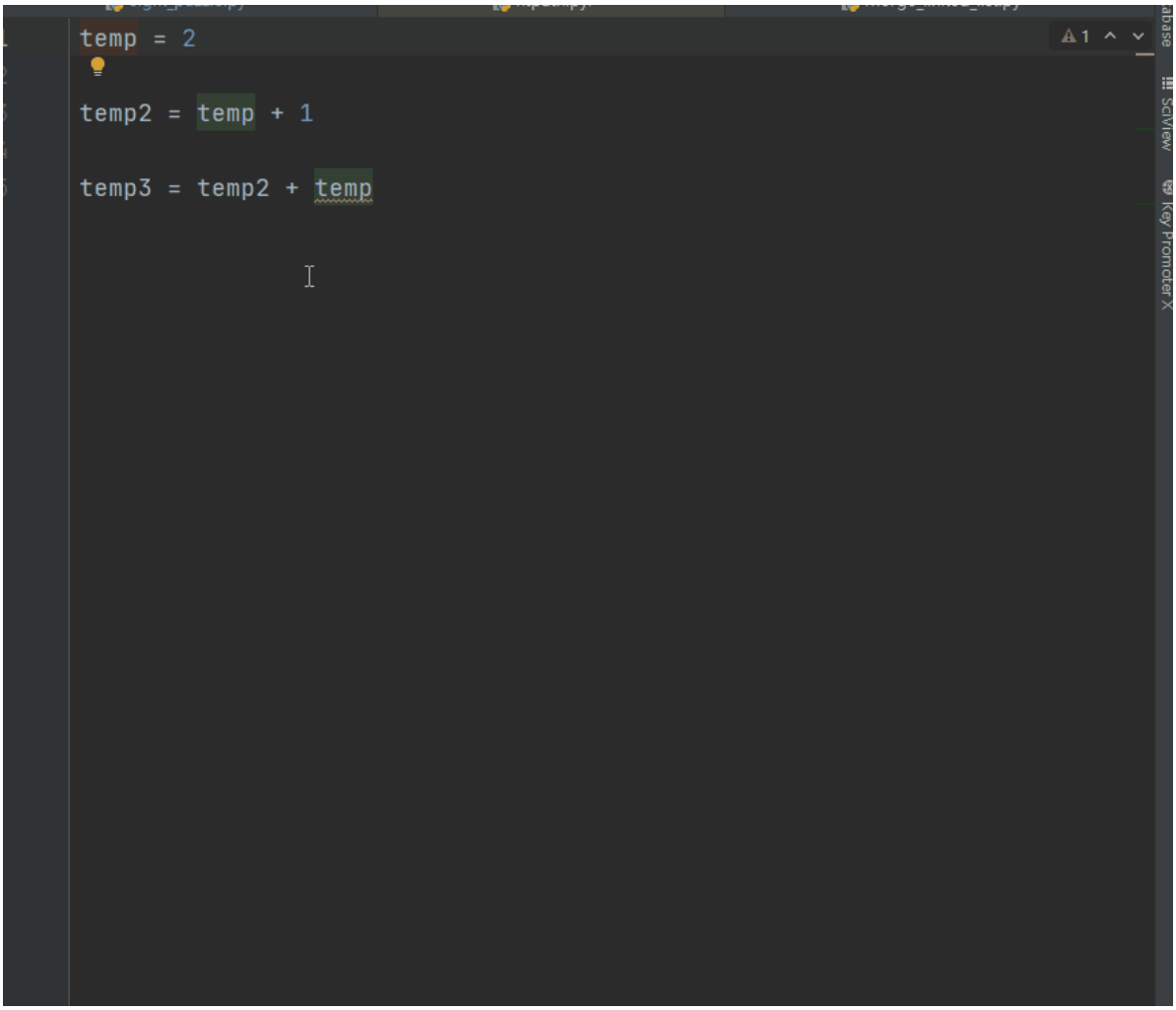
很简单，使用快捷键`Ctrl + Shift +V`，然后在弹窗中，就可以看到历史复制记录啦



9、批量重命名

有时候粗心大意，发现一个变量名写错了，等到发现的时候已经被引用了一大堆。一个个的去修改容易出错不说还很麻烦。

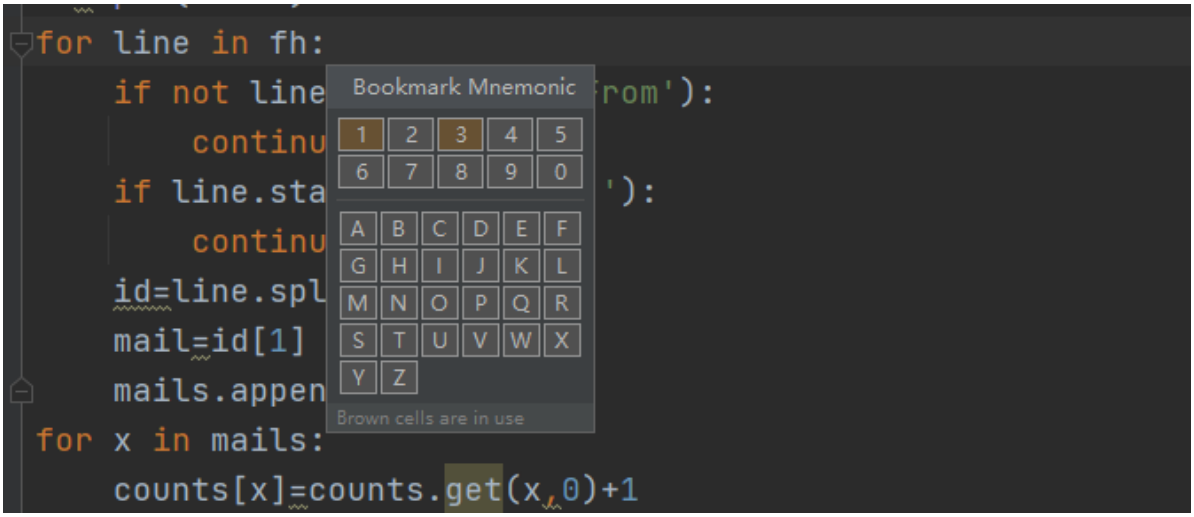
这个时候只需要将光标放到变量上，按下快捷键`Ctrl + F6`， 就可以批量重命名了。



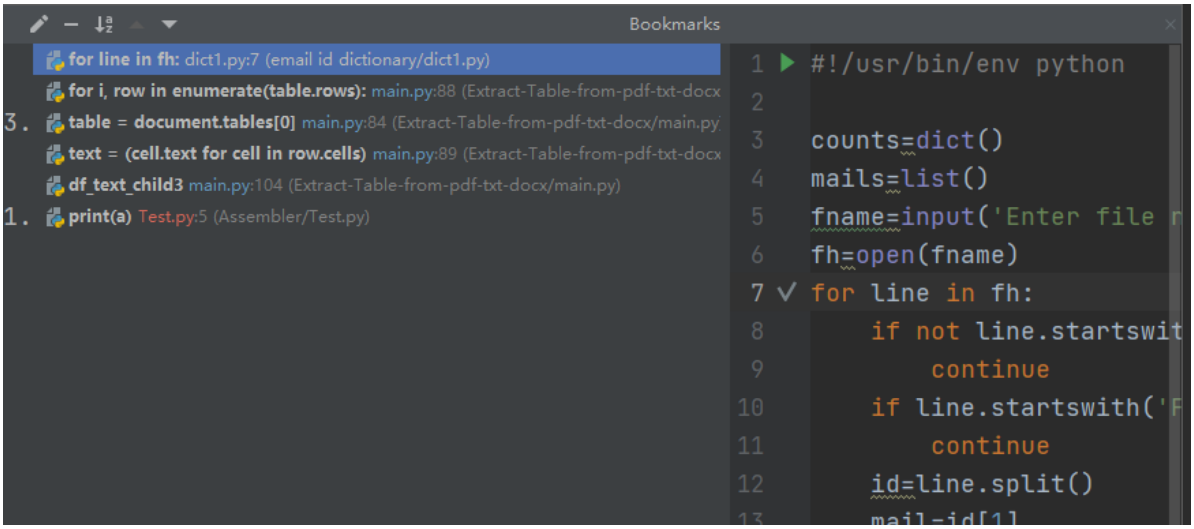
10、PyCharm的书签功能

PyCharm也可以像浏览器那样，对关键代码打上标签。对某段代码进行标记，后面需要查阅的时候就会方便很多。

打书签，快捷键`Ctrl +F11`， 选择对应的数字键，可以通过`Ctrl + 数字键`快速定位到书签处



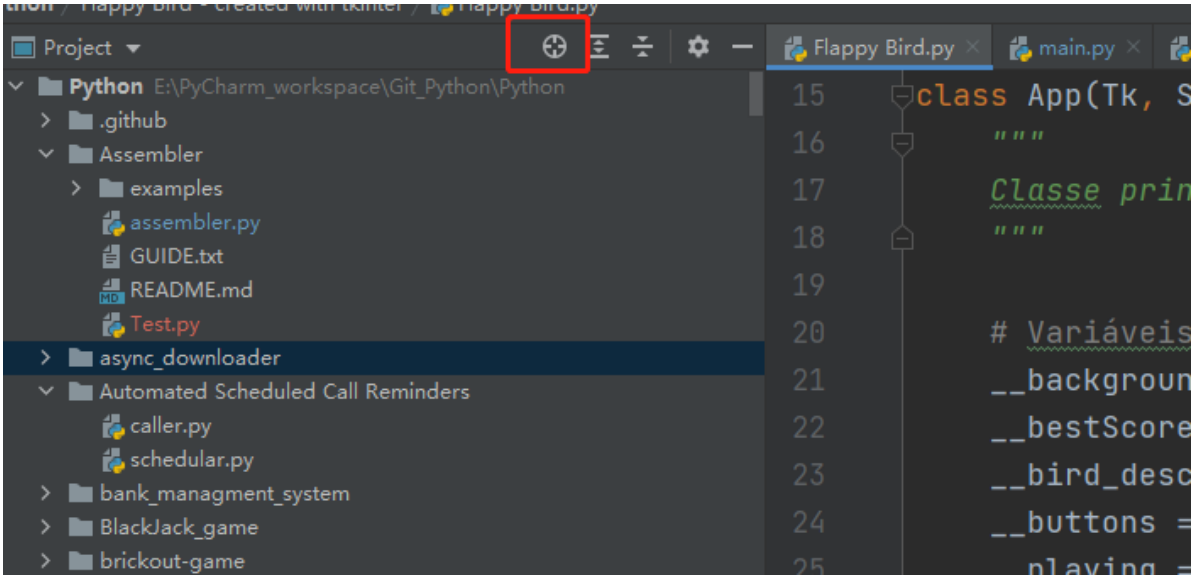
书签列表，快捷键`Shift + F11`，展示所有书签



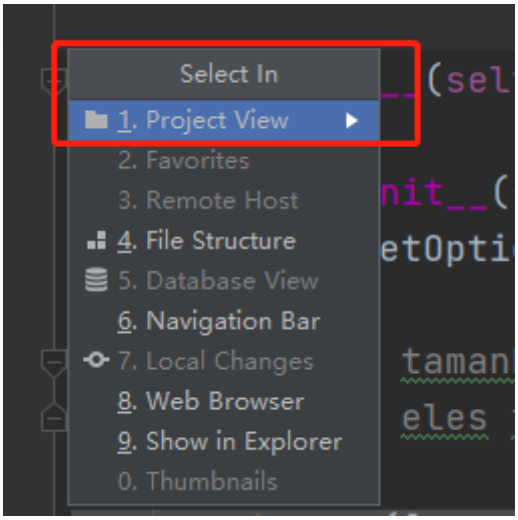
11、快速定位位置

如何快速定位代码在`Project`中的位置，有两种方法。

直接点击项目结构区的小圆圈，如下图



第二种方法，通过快捷键`Alt + F1`， 选择`Project New`

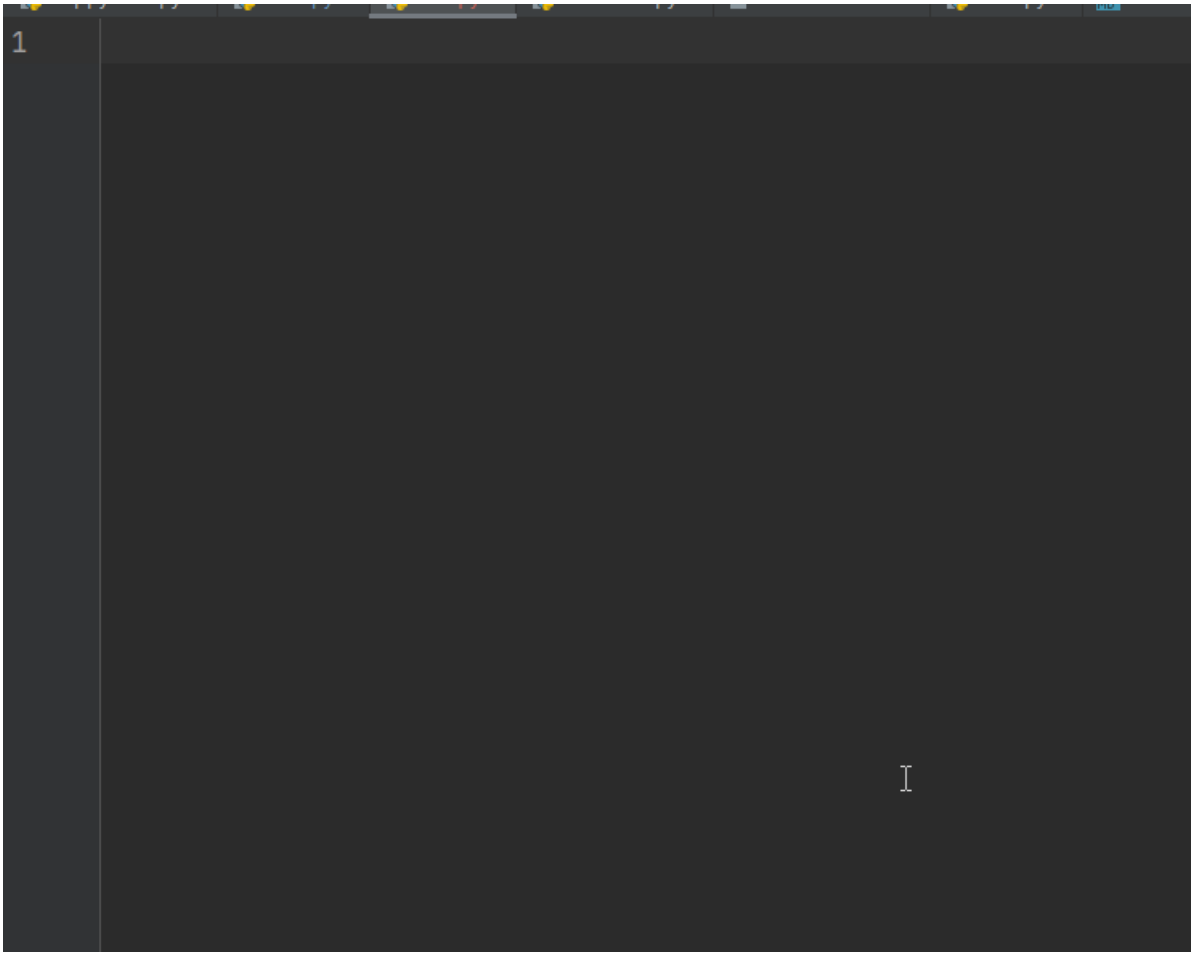


12、自动生成常用语法

假设有个变量`a`。

生成if判断可以直接`a.if`

类似的还有`a.print`、`a.while`



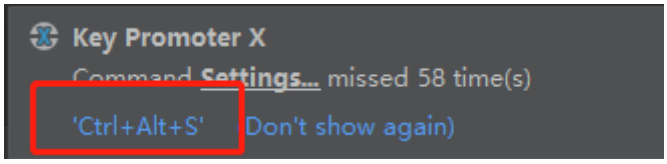
必备插件

以下所有插件在公众号：Python极客专栏，后台回复【插件】

1、Key Promoter X（快捷键）

用来提示快捷键的插件，帮助我们尽可能的摆脱鼠标操作

在用鼠标进行操作是，插件会自动提示相应功能对应的快捷键。

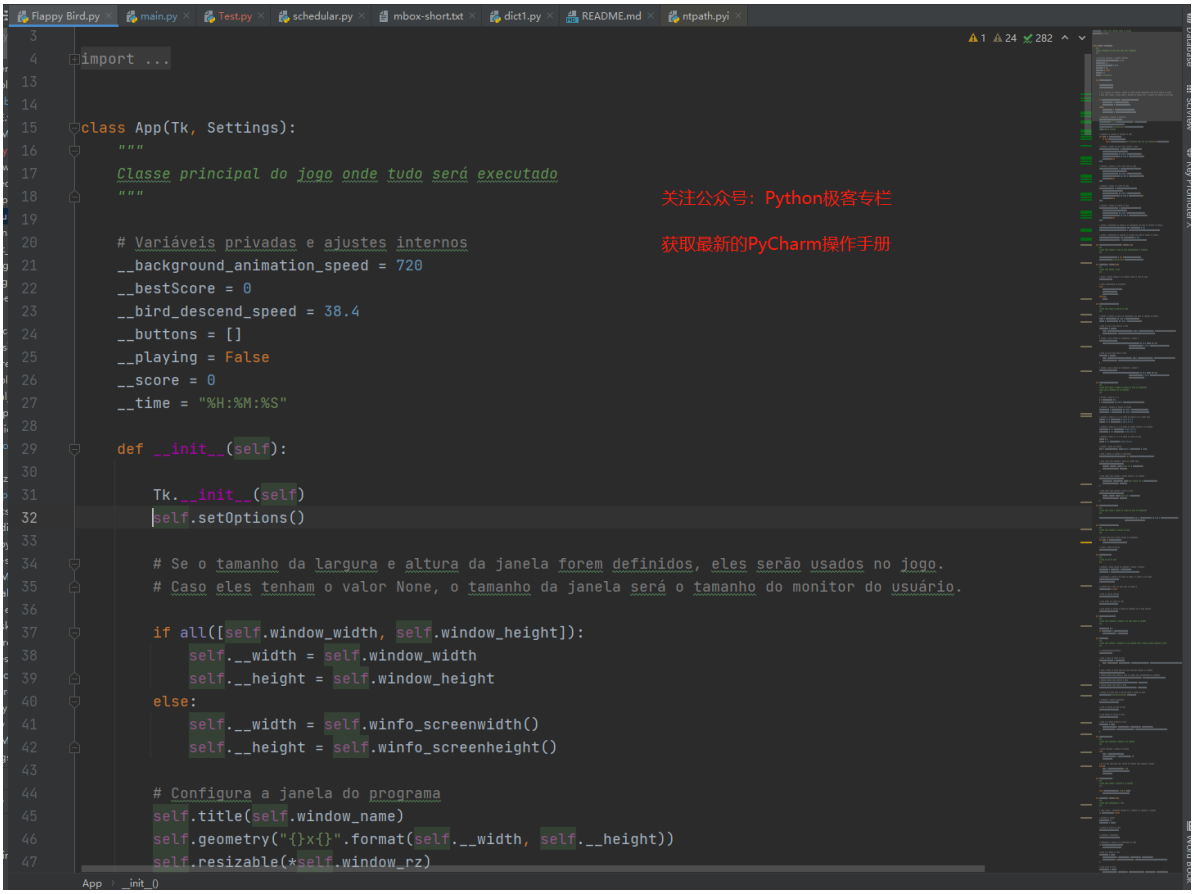


2、Translation（翻译）

一款翻译插件，支持谷歌/有道/百度翻译。支持中英文互译。再也不用担心看不懂英文，和命名变量/函数了。

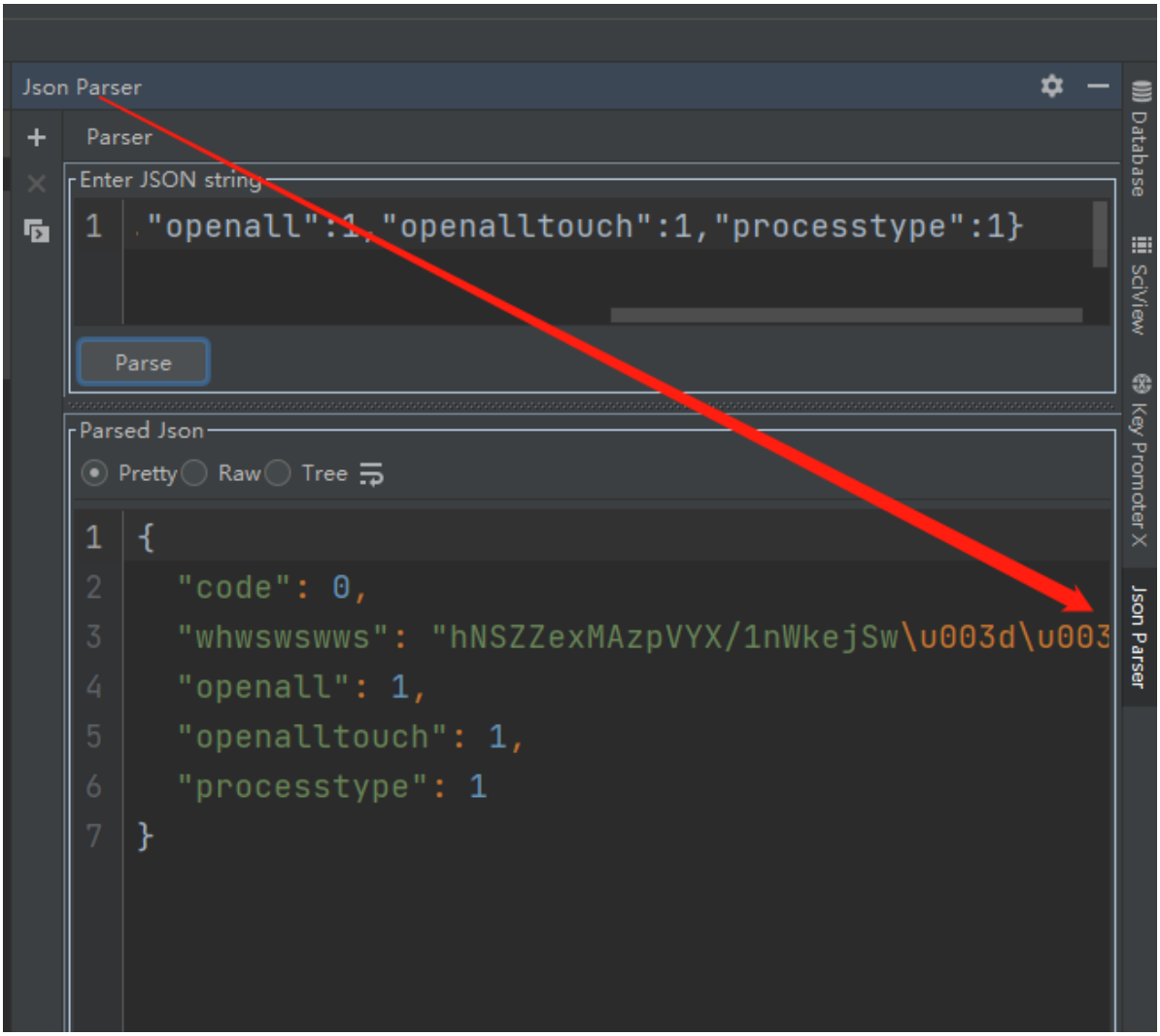
3、CodeGlance（缩略图）

对于较长的代码文件，有这个缩略图插件会非常方便。 插件生效后 效果如下



4、Json Parser（Json格式化）

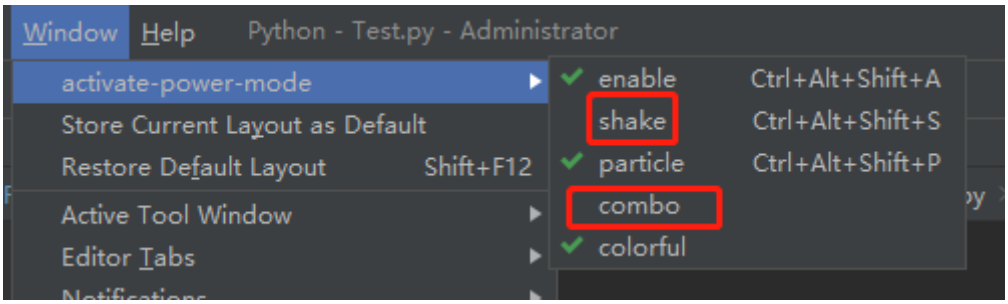
一个支持在PyCharm内部进行Json验证和格式化的轻量级插件，省去了来回切换浏览器格式化校验的麻烦。



如果找不到入口可以在左下角如下图所示

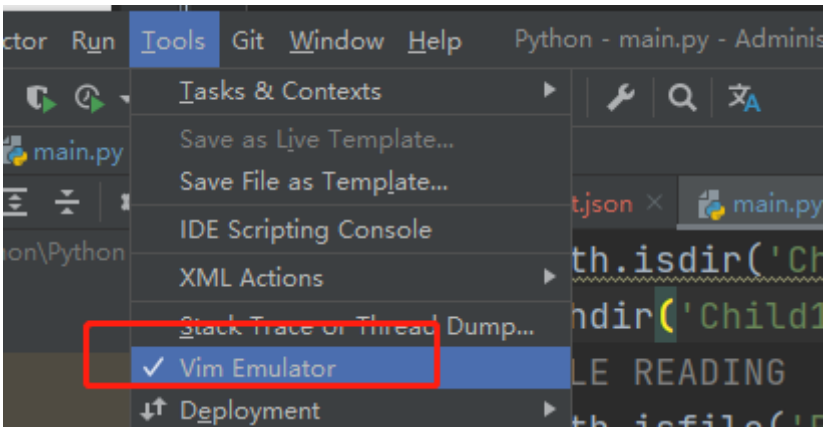
5、activate-power-mode（花里胡哨）

安装完毕之后，每次敲击字符都有动画效果。 可以把combo/shake 取消勾选，这样效果会好一些



6、ideaVim（vim编辑器）

安装完毕之后，在`Tools -> Vim Emulate`中进行开启和关闭



这款插件如果对Vim不熟悉的刚开始用回很不习惯，但是一旦熟悉之后，可以完全脱离鼠标进行编码了。不光效率看起来也很装X。

- `ESC`：切换Vim模式
- `i`： 切换到编辑模式

方向键：`k`上 / `j`下 / `h`左 / `l`右

词组正向跳转：`W`

词组反向跳转：`B`

正向跳转到指定字符：`f+字符`，例如`fh`正向跳转到h字符

反向跳转到指定字符：`F+字符`，例如`Fh`反向跳转到h字符

跳转文件头部：`gg`

跳转至文件结尾：`G`

跳转到指定行的开头：`行号+G`，例如`10G`

复制当前行：`yy`

粘贴：`p`

删除当前行：`dd`

删除选中区域：`d`

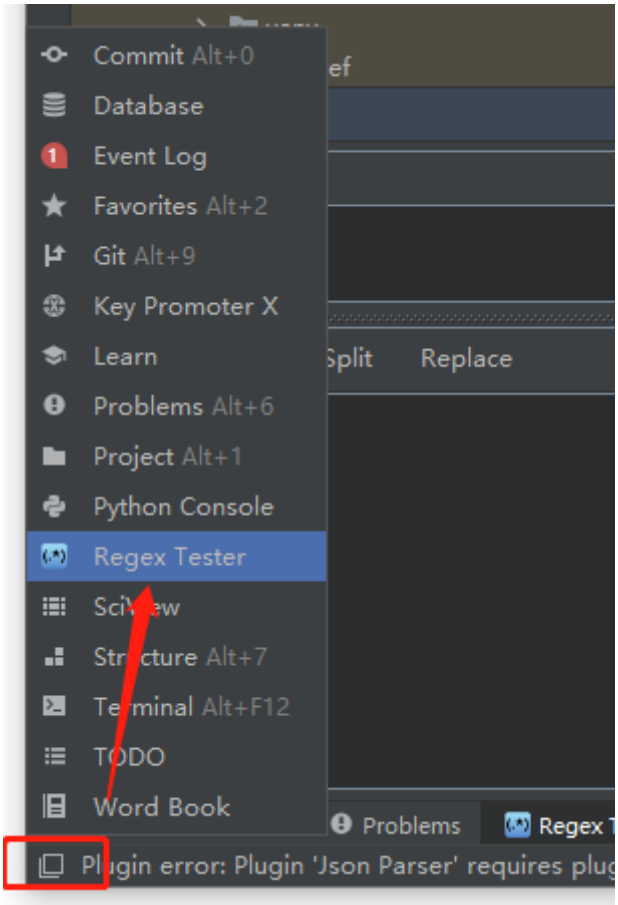
撤销：`u`

正向按字符删除： `x`

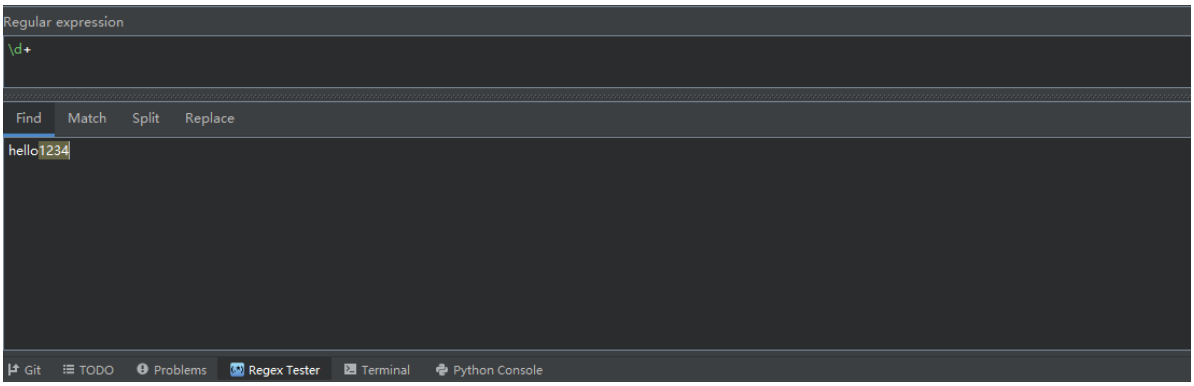
从光标处删除至指定字符： `df指定字符`

7、Regex Tester in PyCharm（正则）

用来测试正则表达式的插件，安装完毕之后在左下角的小矩形框中可以找到



界面如下，上面输入的是正则表达式，下面输入需要查找的字符串。

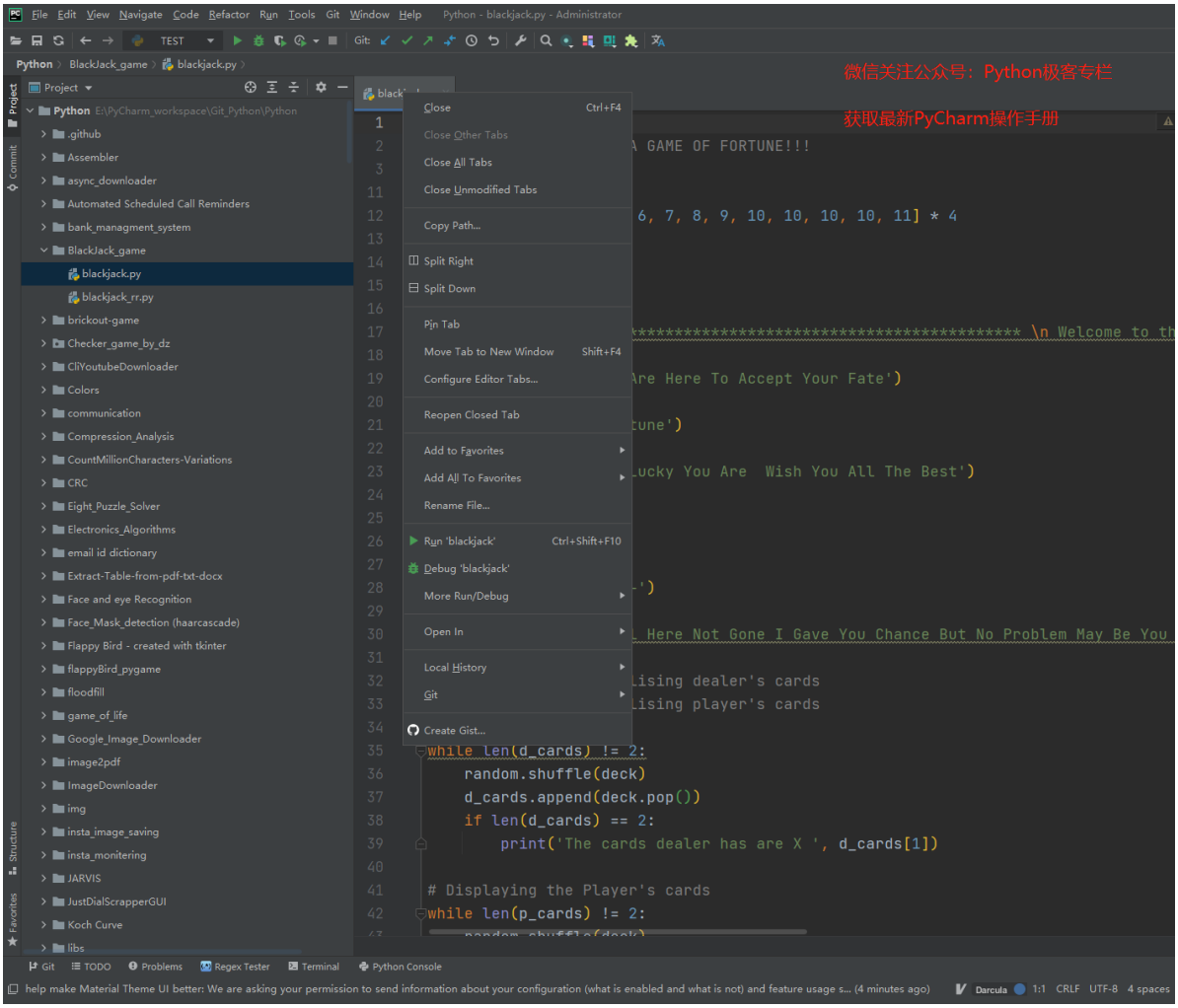


8、Rainbow Brackets（彩虹色）

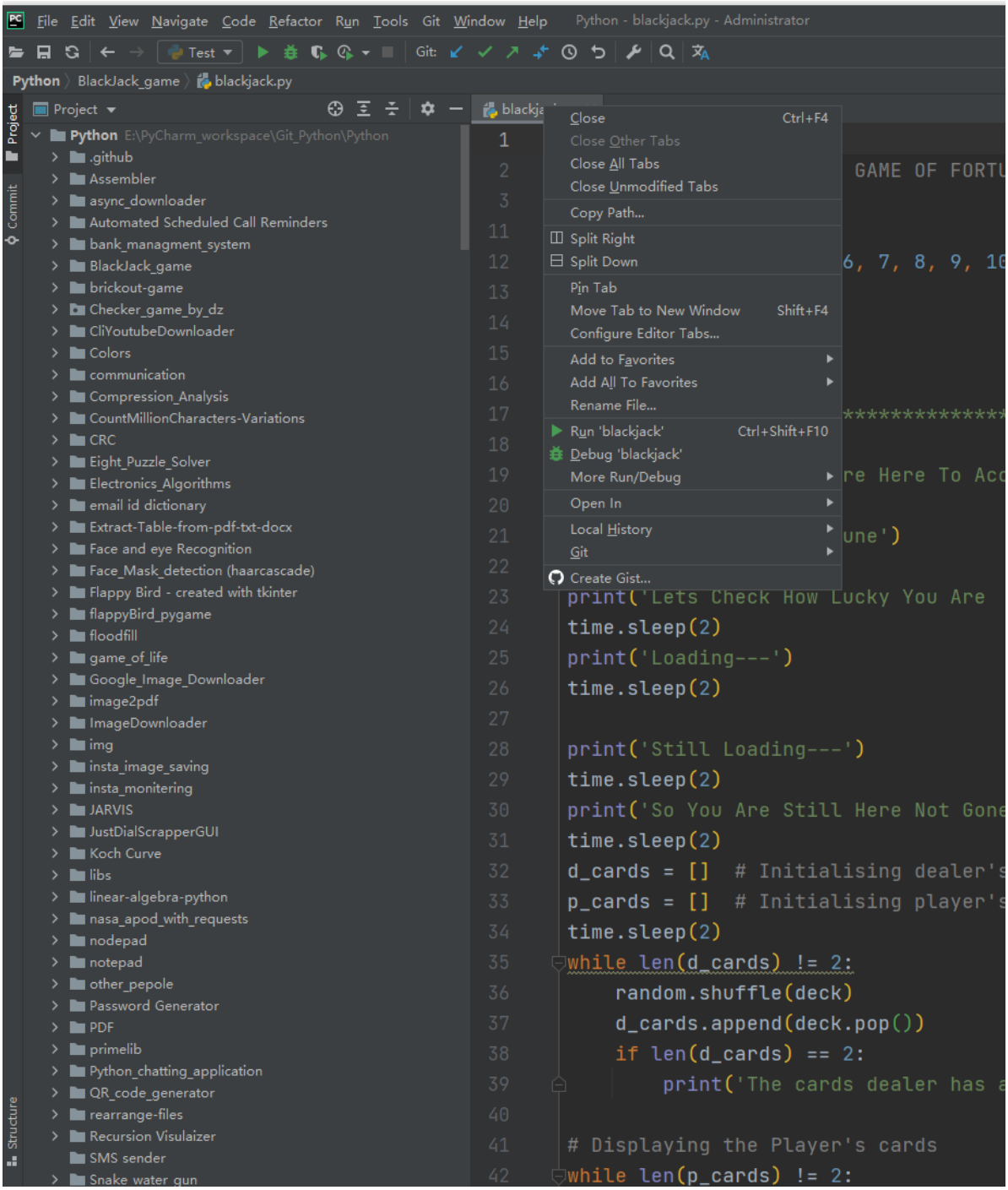
可以让代码块之间很清晰的显示出各种颜色。匹配的括号是相同的颜色，并且实现选中代码高亮显示。可以对代码的排错有帮助。

9、Material Theme（炫酷主题）

如果大家不喜欢扁平化的主题风格，可以尝试一下这种主题，简洁大方。安装之后PyCharm的界面会变成这样



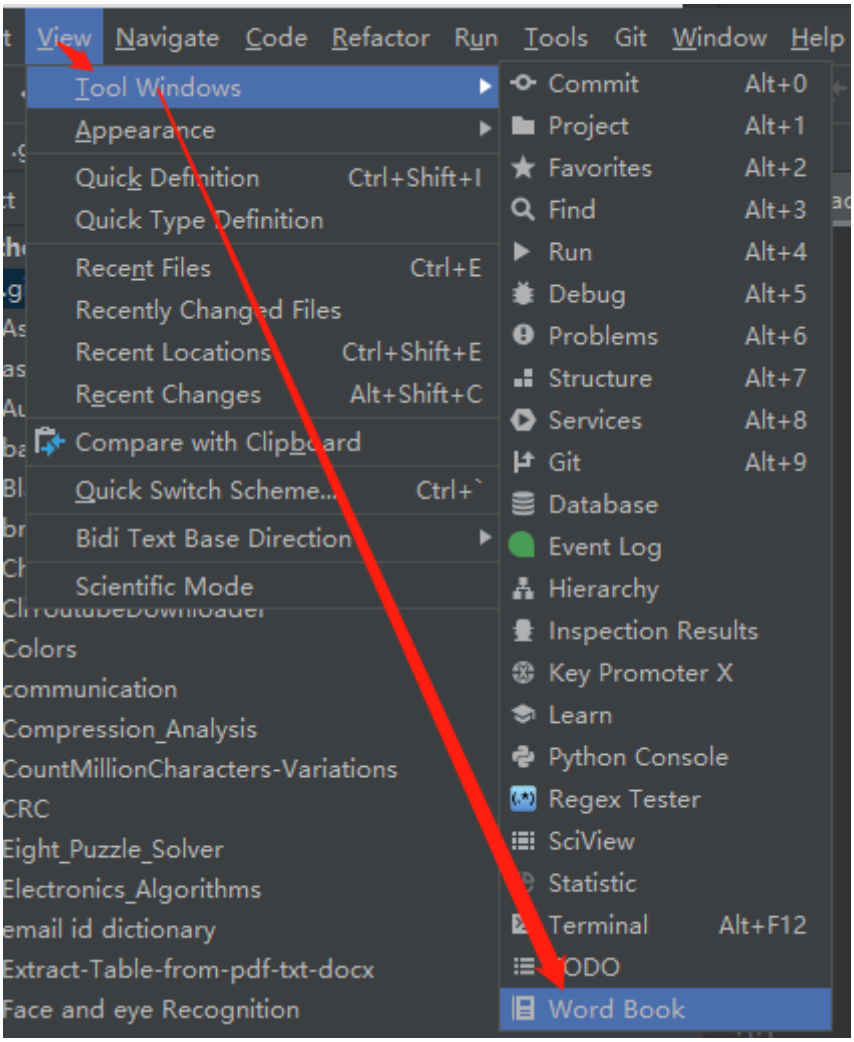
安装之前是这样的，大家可以比较一下。



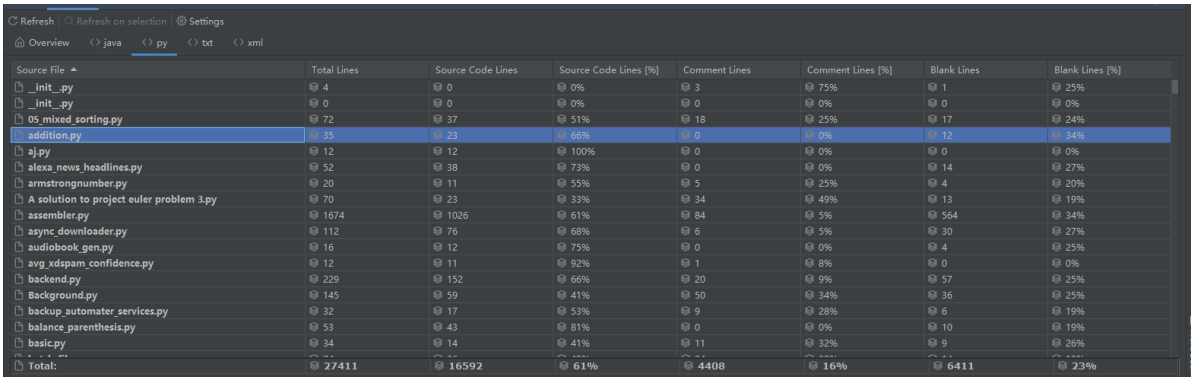
10、Statistic（代码统计）

可以统计当前项目中代码的行数和大小等信息

插件安装完毕，在`View -> Tool Windows -> Statistic`中开启



统计效果图如下



结尾

虽说学Python没有规定一定要使用PyCharm，但是不得不说PyCharm确实好用。希望能够帮助到那些刚开始使用PyCharm又束手无策的同学。

本手册有很多不足之处，欢迎斧正。

后续也会不断的进行完善和迭代，将第一时间发布在公众号上。**欢迎关注公众号：Python极客专栏**

赞赏支持

如果本手册能够帮助你快速入手PyCharm，请作者喝杯咖啡吧~ 后续会继续完善更新！ 一起加油！