

目录

前言	1.1
PyCharm简介	1.2
版本选择	1.2.1
Python项目开发	1.3
新建项目	1.3.1
设置Python版本	1.3.1.1
文件编码设置	1.3.1.2
调试项目	1.3.2
远程调试	1.3.2.1
调试心得	1.3.2.2
项目部署	1.3.3
智能之处	1.4
作为IDE本身	1.4.1
项目配置	1.4.2
编辑期间	1.4.3
调试期间	1.4.4
git支持	1.4.5
其他支持	1.4.6
REST	1.4.6.1
SciView	1.4.6.2
MongoDB	1.4.6.3
Markdown	1.4.6.4
Scientific Mode	1.4.6.5
全局设置	1.5
常用快捷键	1.5.1
CPython加速	1.5.2
问题和心得	1.6
附录	1.7
参考资料	1.7.1

最智能的Python的IDE：PyCharm

- 最新版本：`v1.0`
- 更新时间：`20200215`

简介

整理最智能的Python的IDE，PyCharm的各种好用功能，包括完整的Python项目开发的过程，从新建项目到项目调试，直到最后的项目部署和发布。包括期间涉及的Python版本的设置，文件编码设置等常见问题。以及详细阐述其智能体现在什么地方，包括了除了集成开发环境本身方便好用功能之外，还有项目配置、编辑Python文件期间的各种智能提示和自动处理，和调试期间的智能识别和显示，以及对git支持很人性化，以及其他一些方面的支持，比如REST、SciView、MongoDB、Markdown、ScientificMode等，最后总结一些常用的全局设置，和整理之前遇到的一些问题和经验心得。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- [crifan/most_intelligent_python_ide_pycharm: 最智能的Python的IDE：PyCharm](#)

如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook_template: demo how to use crifan gitbook template and demo](#)

在线浏览

- [最智能的Python的IDE：PyCharm book.crifan.com](#)
- [最智能的Python的IDE：PyCharm crifan.github.io](#)

离线下载阅读

- [最智能的Python的IDE：PyCharm PDF](#)
- [最智能的Python的IDE：PyCharm ePub](#)
- [最智能的Python的IDE：PyCharm Mobi](#)

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan_ebook_readme: Crifan的电子书的使用说明](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-01-17

00:51:28

PyCharm简介

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12 21:20:07

选择PyCharm版本

PyCharm有2个版本，关于如何选择，现在整理如下：

官网截图：

The screenshot shows the official PyCharm download page. At the top, there's a navigation bar with links like 'Tools', 'Languages', 'Solutions', 'Support', 'Store', and a search icon. Below the navigation is a main heading 'Download PyCharm'. Underneath, there are two large buttons: 'Professional' and 'Community', both with red outlines. The 'Professional' button is associated with a 'Windows' tab and a 'DOWNLOAD' button. The 'Community' button is associated with 'macOS' and 'Linux' tabs, and a 'DOWNLOAD' button. To the left of these buttons, there's a large 'PC' logo and some version information: 'Version: 2018.1.4', 'Build: 181.5087.37', and 'Released: May 31, 2018'. Below the 'Professional' and 'Community' buttons are sections for 'System requirements', 'Installation Instructions', and 'Previous versions'.

- 概述
 - Community =社区版=免费版
 - 轻量级，用于Python和科学计算的开发
 - Lightweight IDE for Python & Scientific development
 - 说明：虽然号称功能少一点，但是对于多数人，尤其是普通开发人员，影响不大，足够用了
 - Professional =专业版=收费版
 - 全部完整功能，用于Python和Web的开发
 - Full-featured IDE for Python & Web development
 - 结论
 - 建议：普通人选免费的社区版 Community version，足够用了。

社区版和专业版的PyCharm对比

对于两个版本的详细区别，整理如下表：

	收费的：Professional Edition专业版	免费的：Community Edition 社区版
	<ul style="list-style-type: none">• Web development with JavaScript, CoffeeScript,	

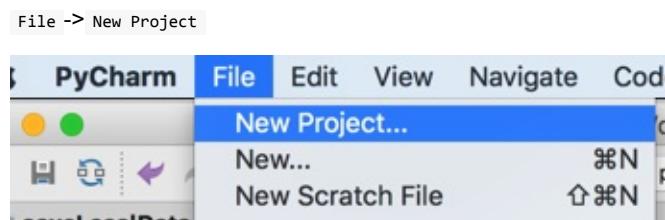
General comparison	<p>TypeScript, HTML/CSS and more</p> <ul style="list-style-type: none"> Frameworks: Django, Flask, Google App Engine, Pyramid, web2py Remote development capabilities: Remote run/debug, VM support Database & SQL support UML & SQLAlchemy Diagrams Scientific Tools <p>点评：后续可能会用到的功能点：</p> <ul style="list-style-type: none"> Flask, Django Remote run/debug SQLAlchemy Diagrams 	<ul style="list-style-type: none"> Intelligent Editor Graphical Debugger Refactorings Code Inspections Version Control Integration
Python, Frameworks & Tools	<ul style="list-style-type: none"> Cython Django AppEngine Flask Jinja2 Mako web2py Pyramid Profiler SQLAlchemy IPython Notebook Diagrams Remote interpreters, remote debugging, Vagrant, Docker Duplicate code detection Code coverage .po files support BDD support Profiler integration Thread Concurrency Visualization 	<ul style="list-style-type: none"> Core Python language support Code Inspections Refactoring Local debugger Test runners reStructuredText support PyQt PyGTK Package management Virtualenv/Buildout Python console
Platform	<ul style="list-style-type: none"> CSS/HAML/SASS/LESS/Stylus Database/SQL JavaScript and JS Debugger Perforce, TFS FTP/SFTP/FTPS remote host deployment TextMate bundles REST Client Puppet File watchers 	<ul style="list-style-type: none"> XML, HTML, YAML, JSON, RelaxNG Git, Mercurial, CVS, Subversion, GitHub IntelliLang Local terminal Task management

Python项目开发

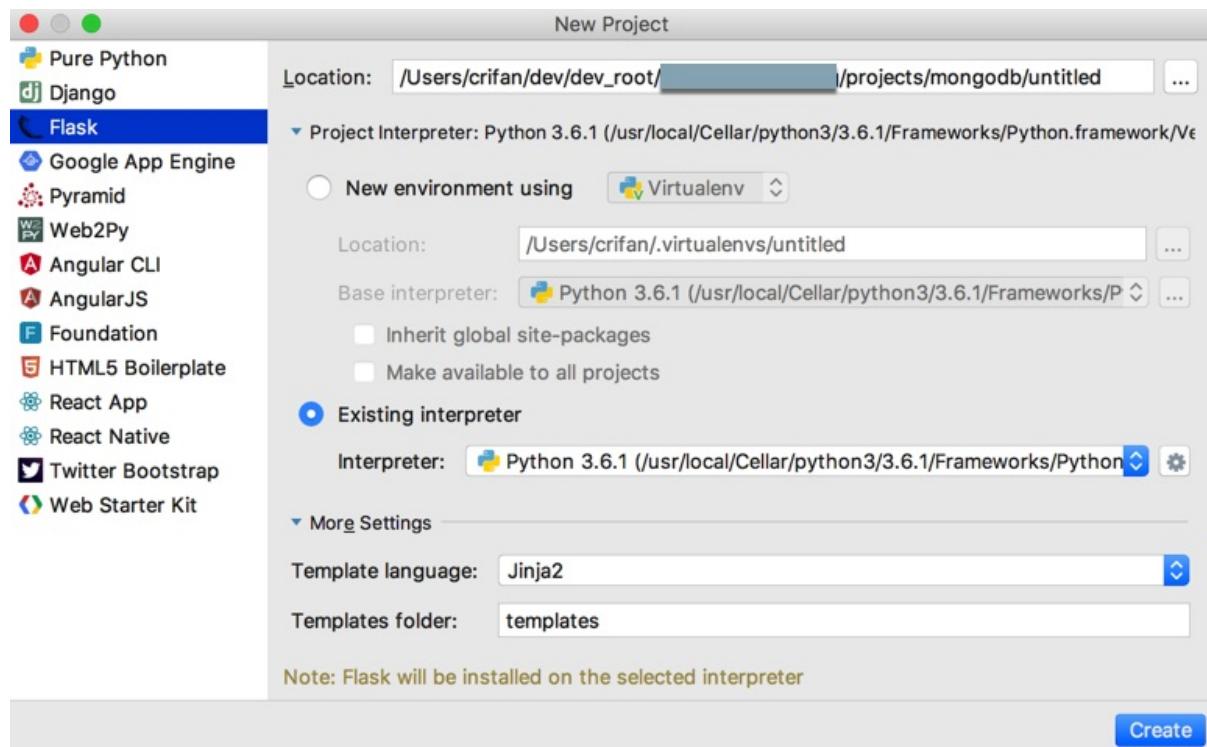
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12 21:25:27

新建项目

下面以创建Python的Flask项目为例，来介绍如何用PyCharm新建项目：



在 New Project 弹框中左边，选择项目是 Flask 的类型



再去确认Python的解释器：

如果已经是你希望的Python，则无需操作；

如果不是，则可以添加对应的Python：点击 Existing interpreter 的配置按钮，点击 All Local



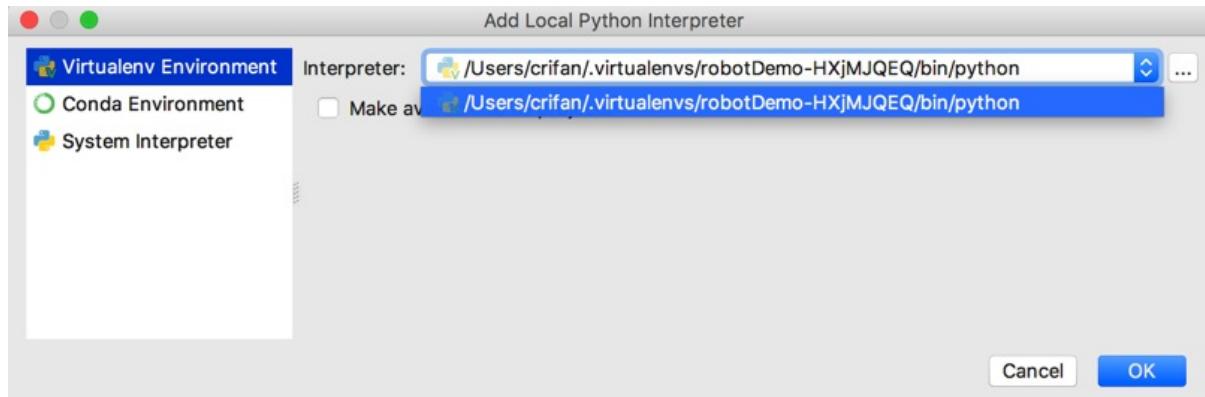
此处可以看到有提示

Note: Flask will be installed on the selected interpreter

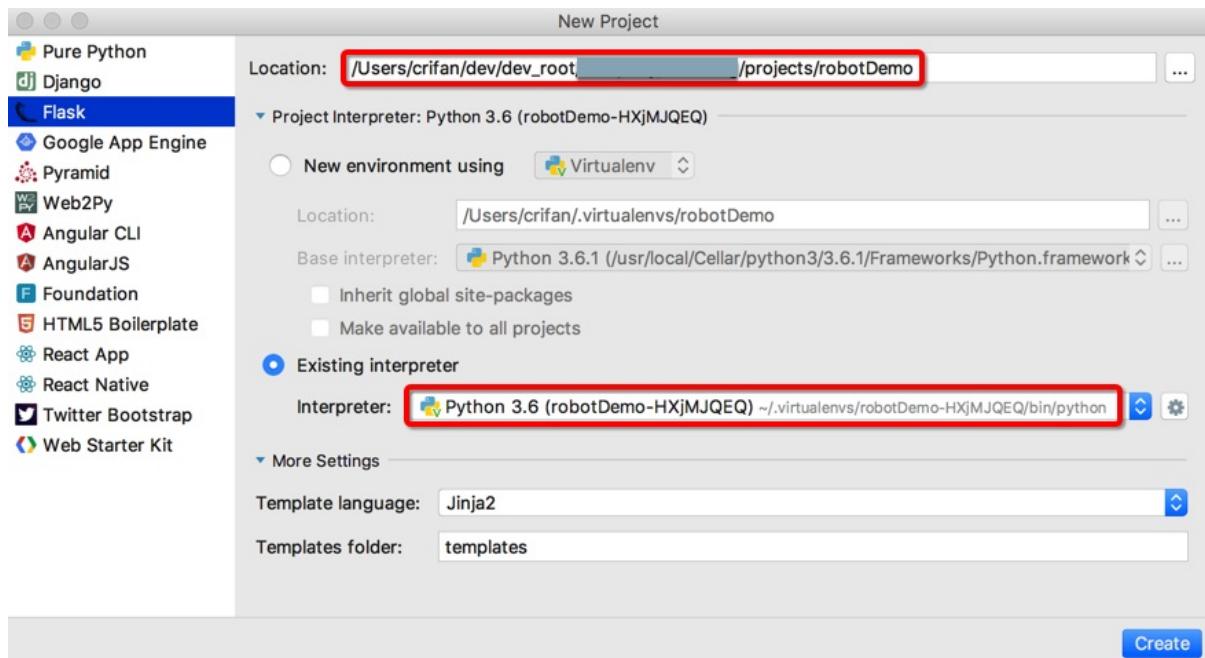
意思是：将会自动安装Flask到你所选择的虚拟环境中去。

-» 看来PyCharm针对于虚拟环境支持的很好了。

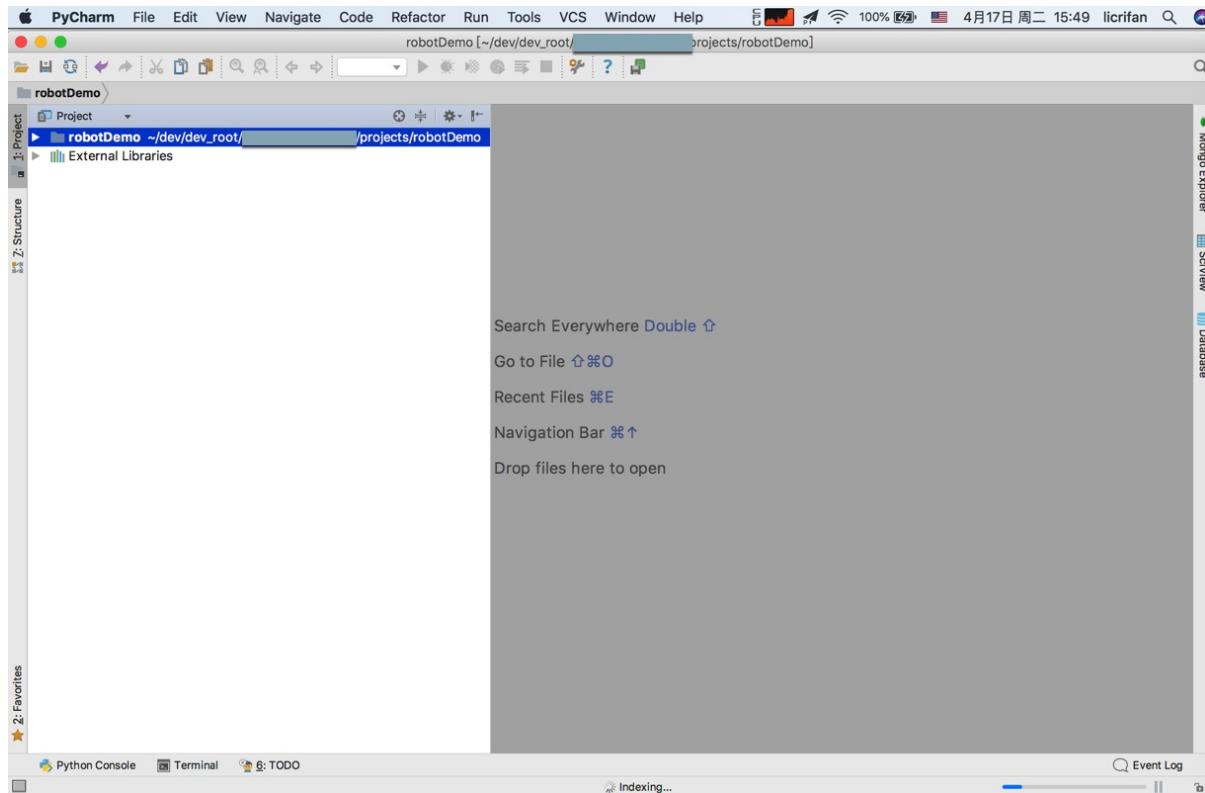
在弹框中选择对应的Python解释器版本：



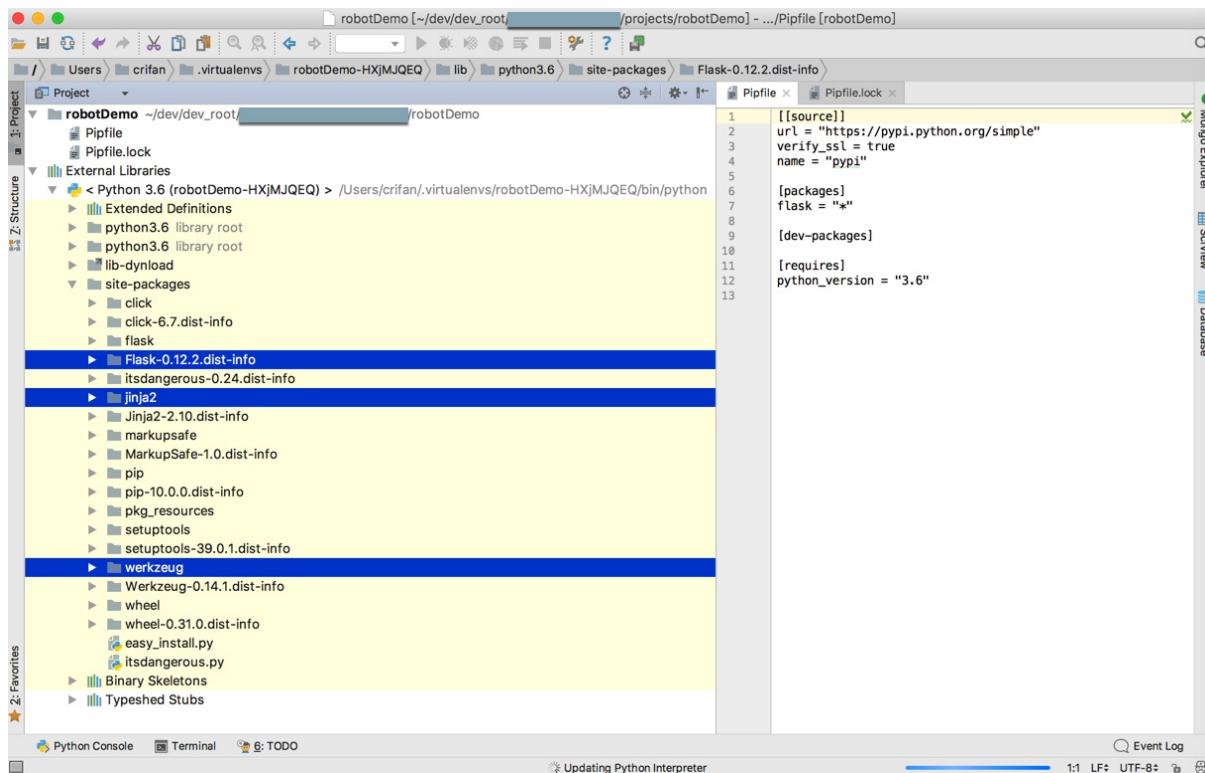
确定后，即可看到 Existing interpreter 是刚才所选的Python，以及同时设置好要项目要保存的路径：



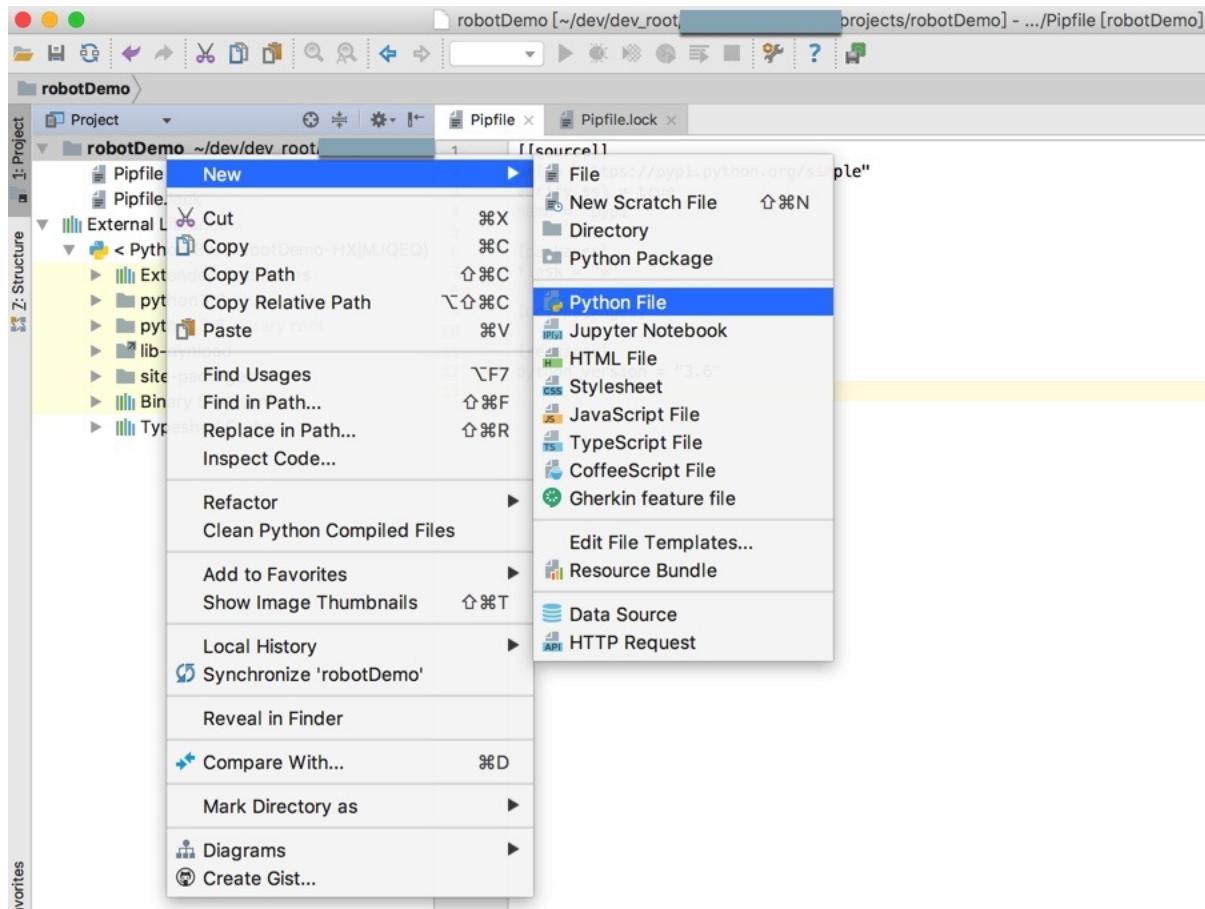
点击 Create 即可创建出项目，且底部右下角会显示正在建立索引indexing：



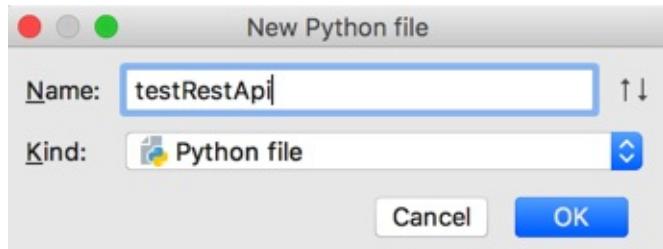
可以看到此虚拟环境中相关的python3的库都可以正常检测到：



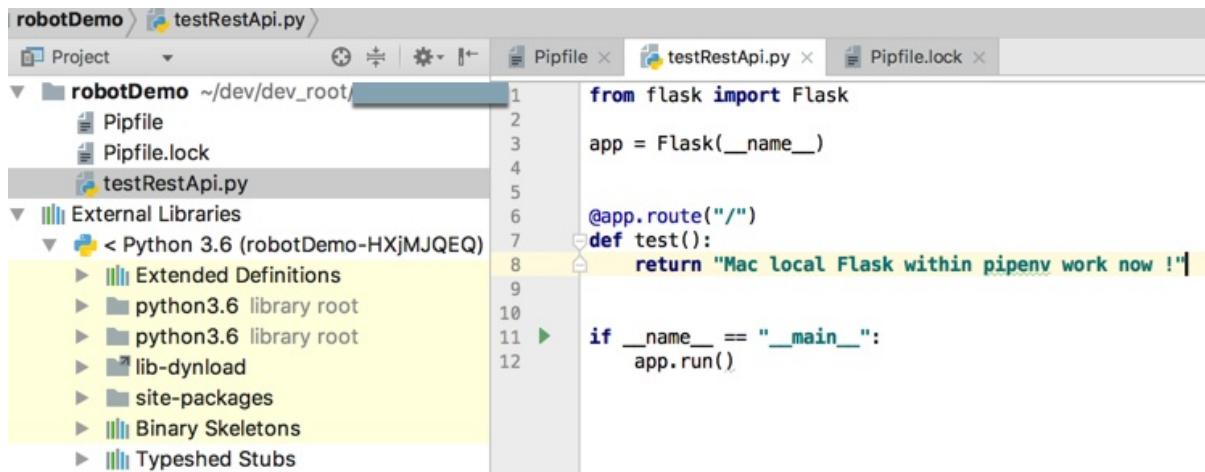
然后新建python代码：



输入文件名，点击OK：



即可创建出Python文件：



设置Python版本

新建Python项目后，记得要

- 选择 Python 版本
 - =设置Python环境
 - =设置Python解析器
 - =Python虚拟环境
 - (如果是虚拟环境的话)
- 尤其是
 - 当系统中安装多个版本的Python
 - 既有Python 2
 - 又有Python 3
 - 甚至有多个版本的Python 3
 - 或本身有个Python虚拟环境
 - 比如用 `virtualenv` / `pipenv` 等创建出来的Python虚拟环境

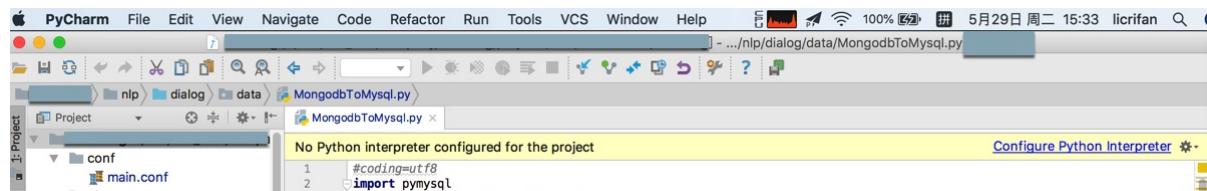
否则会出现：

- 无法正常调试
 - 找不到已安装的库

下面详细介绍一下，新建Python项目后后，如何设置对应的：

PyCharm中设置当前项目的Python解析器

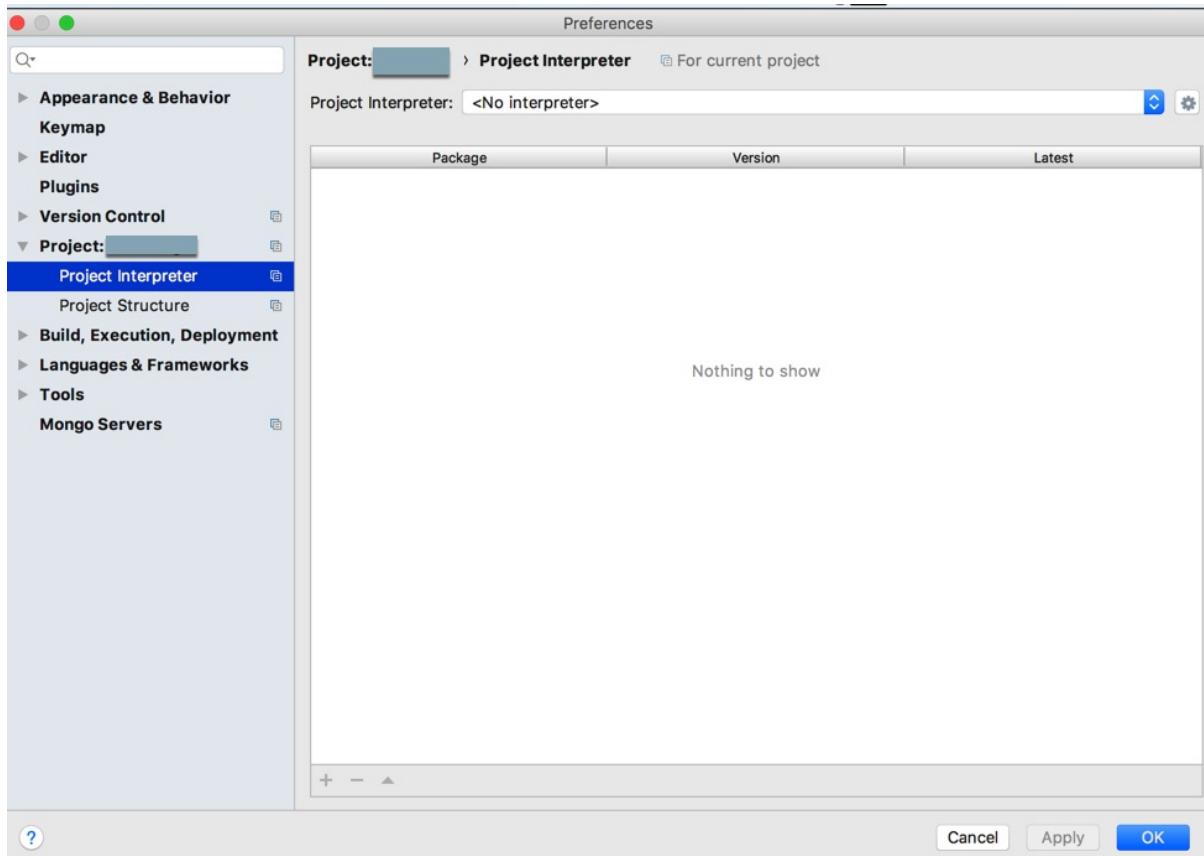
新建Python项目（或者打开一个已有Python代码的项目）后，会提示你设置python解析器的：



点击后，等价于：

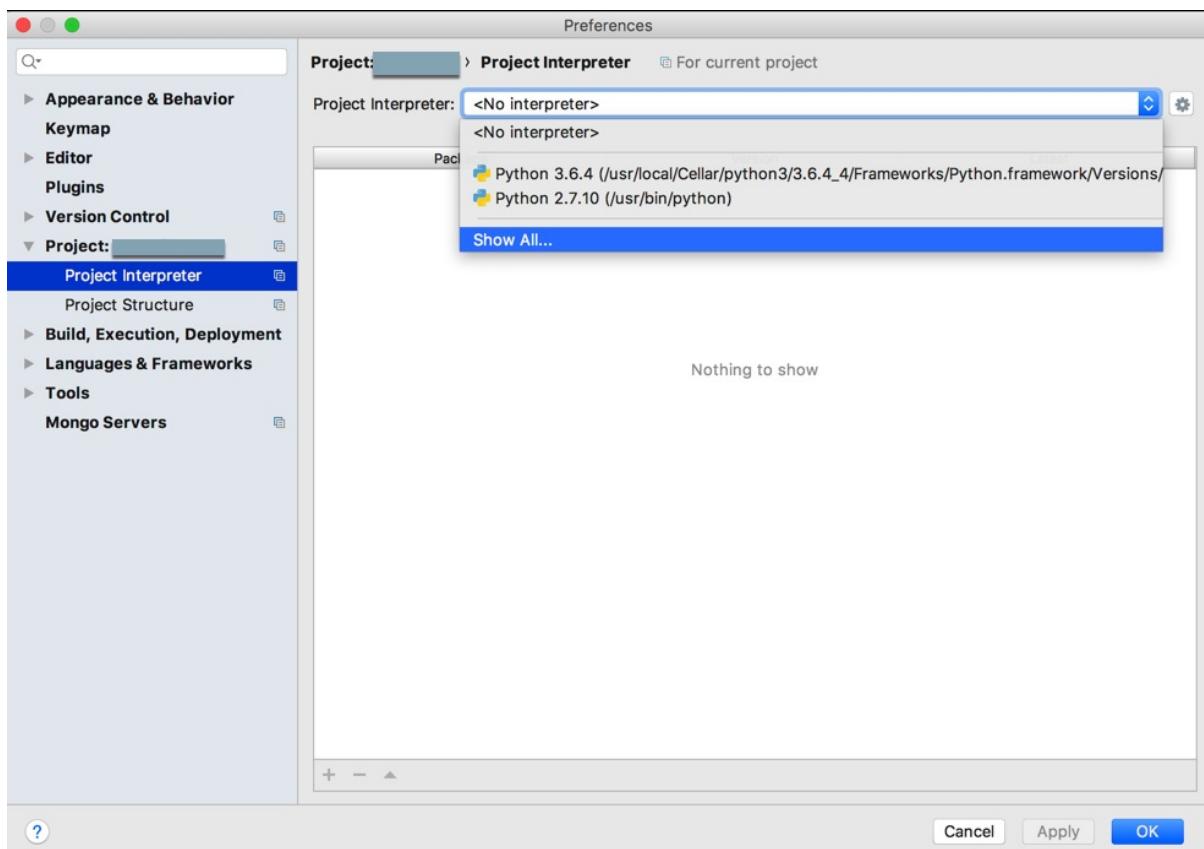
PyCharm -> Preferences -> Project xxx -> Project Interpreter



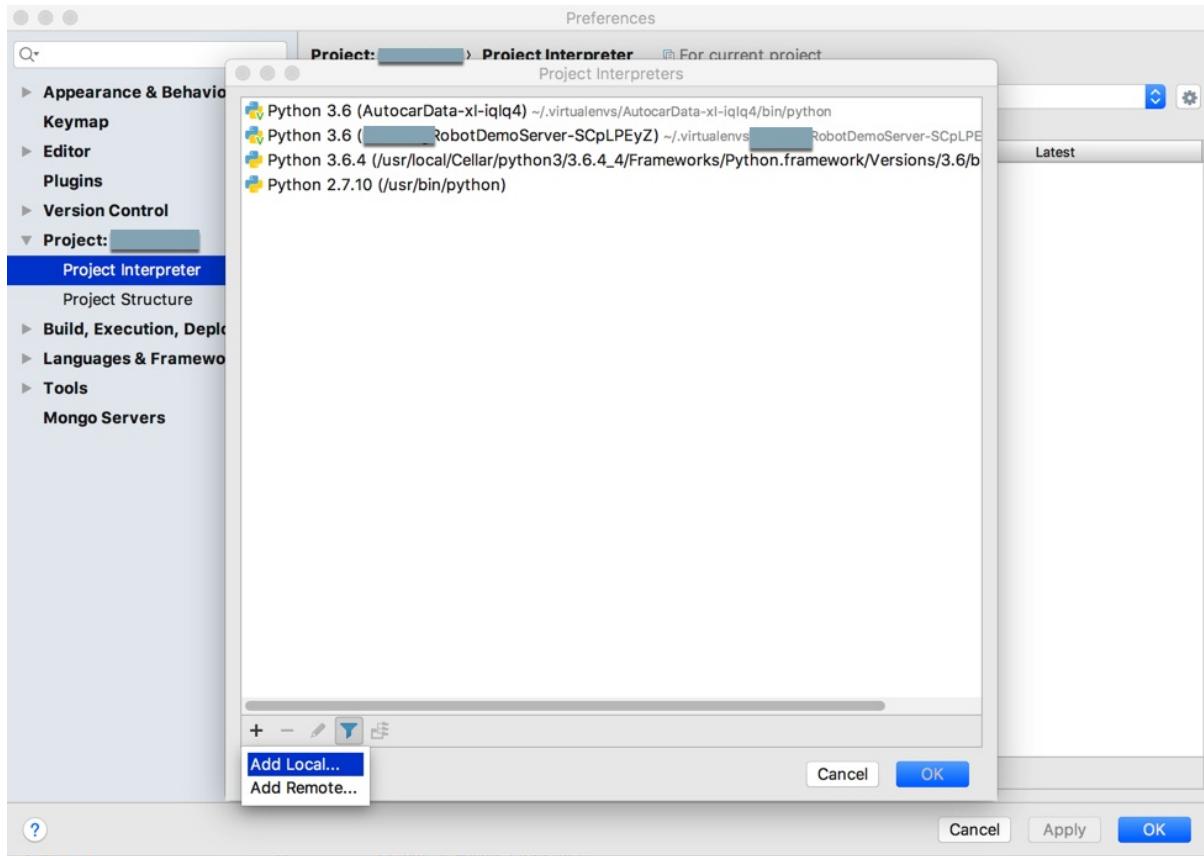


此处，通过 pipenv 添加了虚拟环境后，再去此处设置对应的 python解析器：

点击 Show All

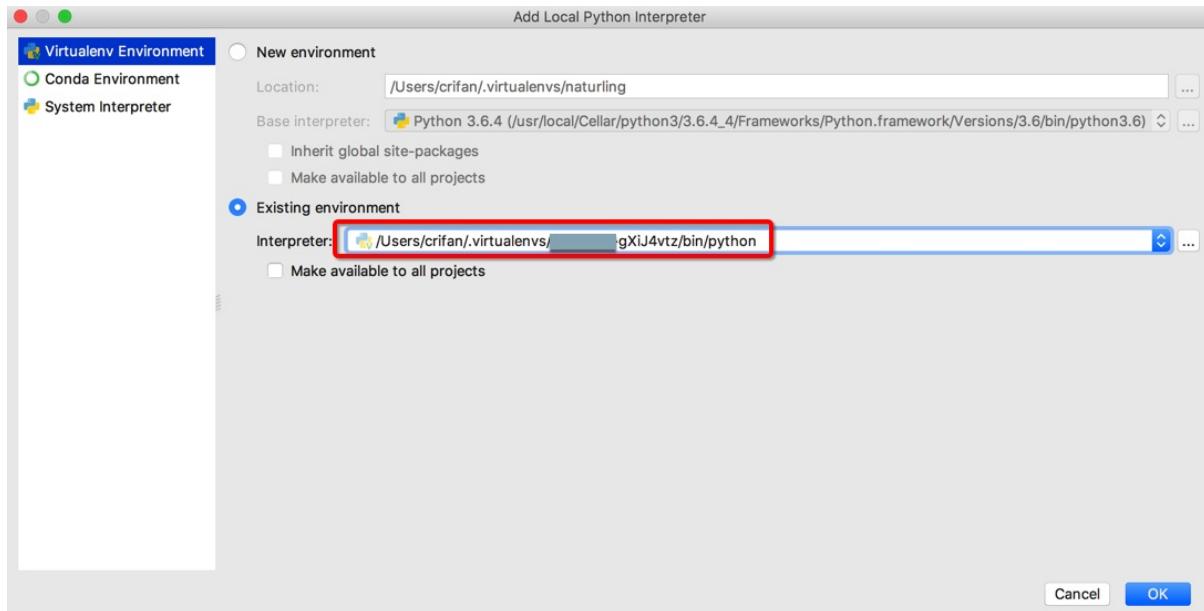


列表中没有刚新建的虚拟环境中的python解析器，所以去 Add local：

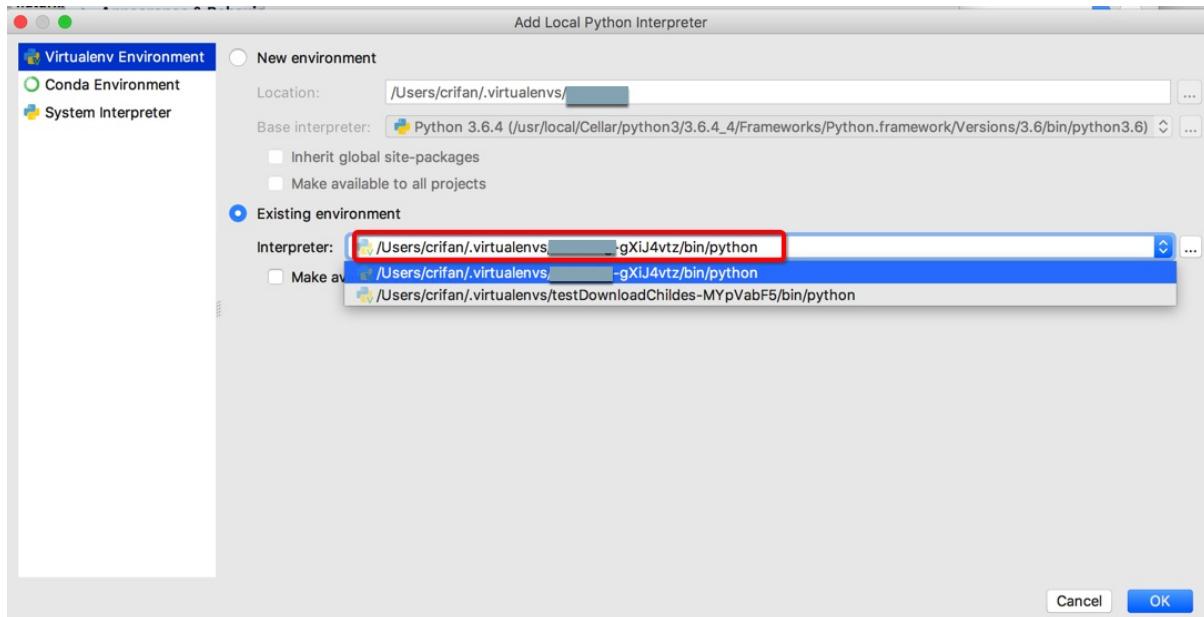


然后发现PyCharm中已经自动帮我们检测到并选择好了：

Existing environment 中的interpreter，是我们新建的virtualenv中的python解析器：



点击列表，还可以看到其他检测出来的虚拟环境的解析器：

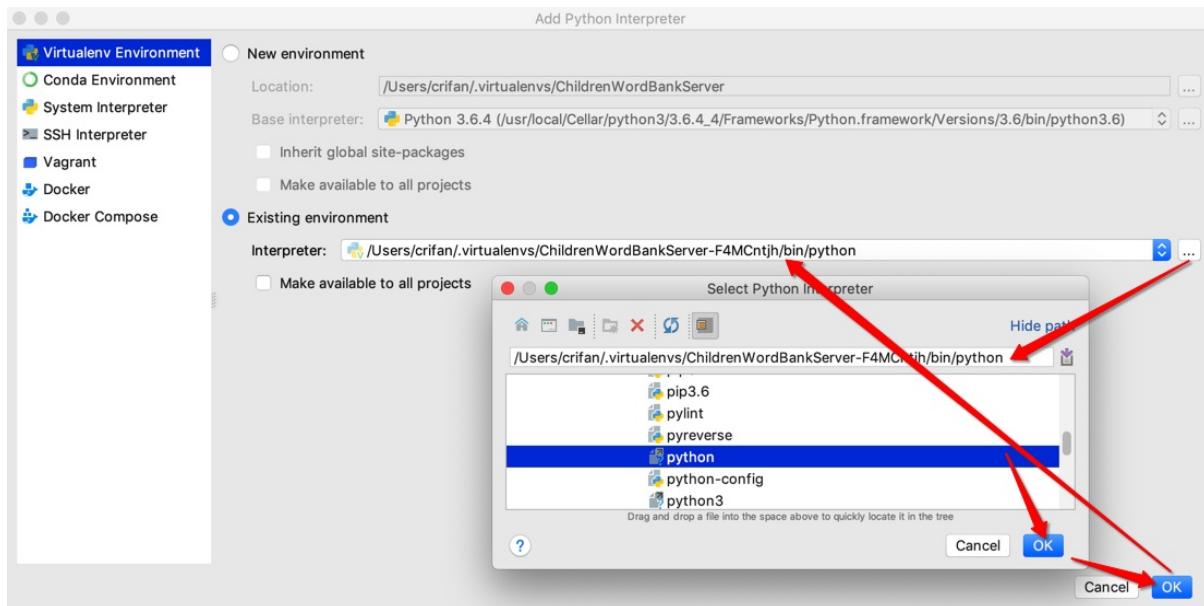


注：如果当前没有识别出你的Python版本，则需要自己手动输入路径去添加。

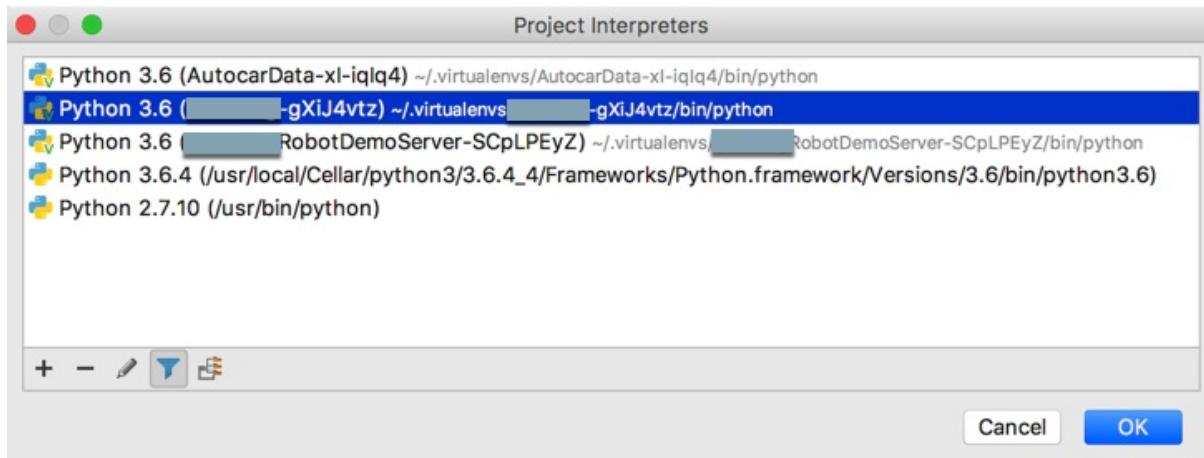
如果是 virtualenv 创建的虚拟环境，一般对应的路径都在：

```
/Users/YourUserName/.virtualenvs/YourProjectName-F4MCntjh/bin/python
```

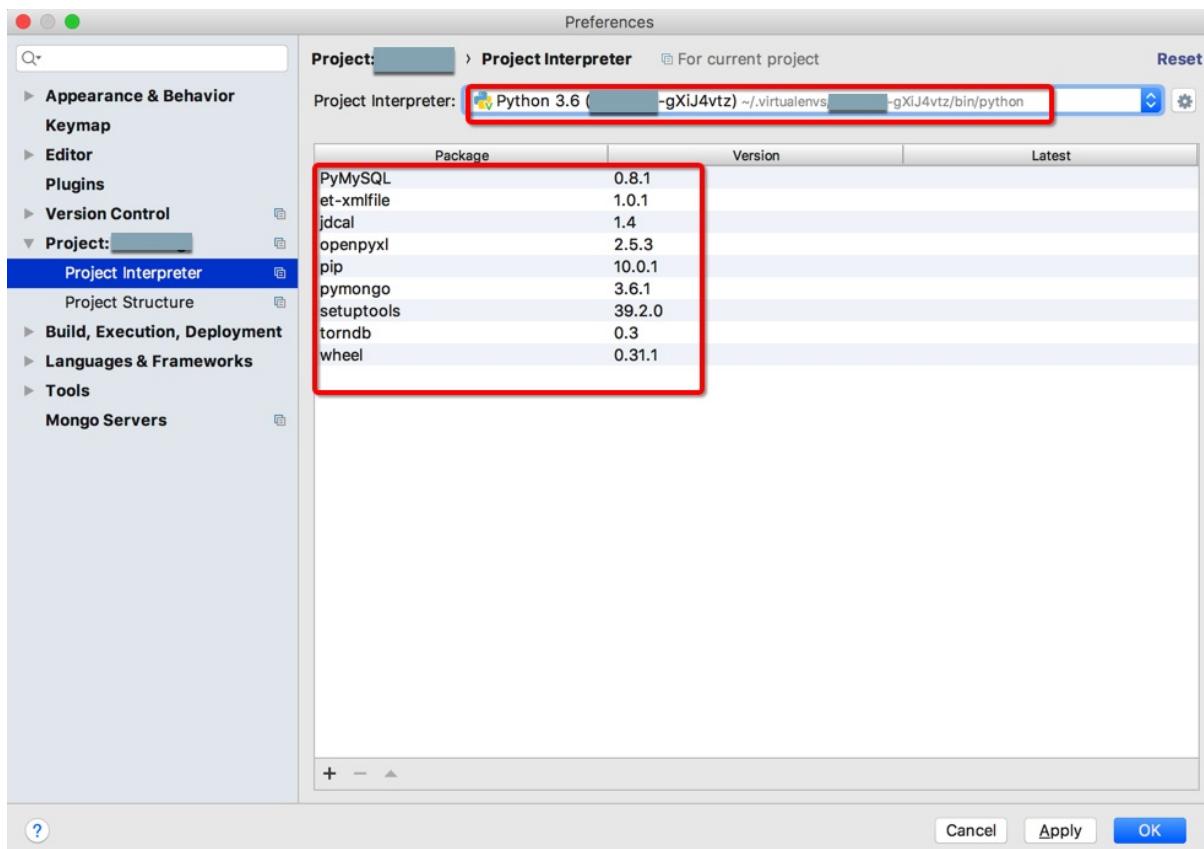
然后自己去选择：



确定后，返回添加页：

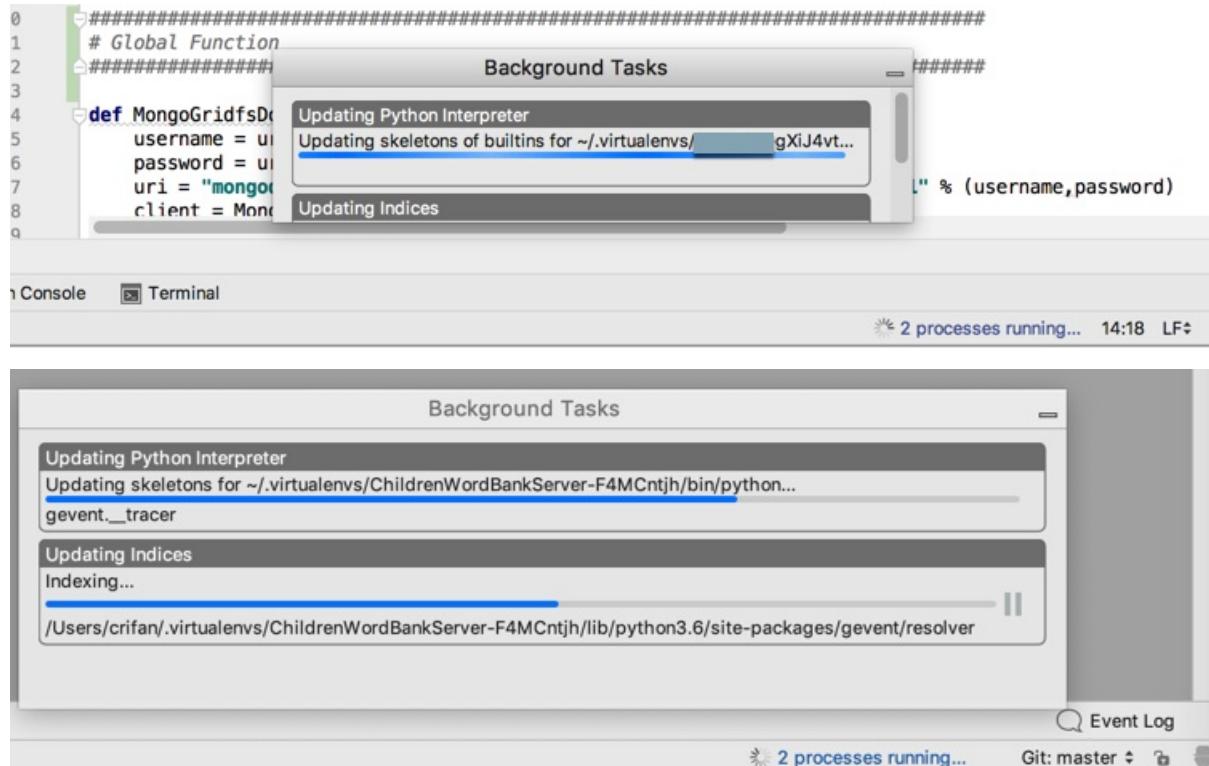


loading后，就可以找到当前python虚拟环境解析器中所安装的库了：

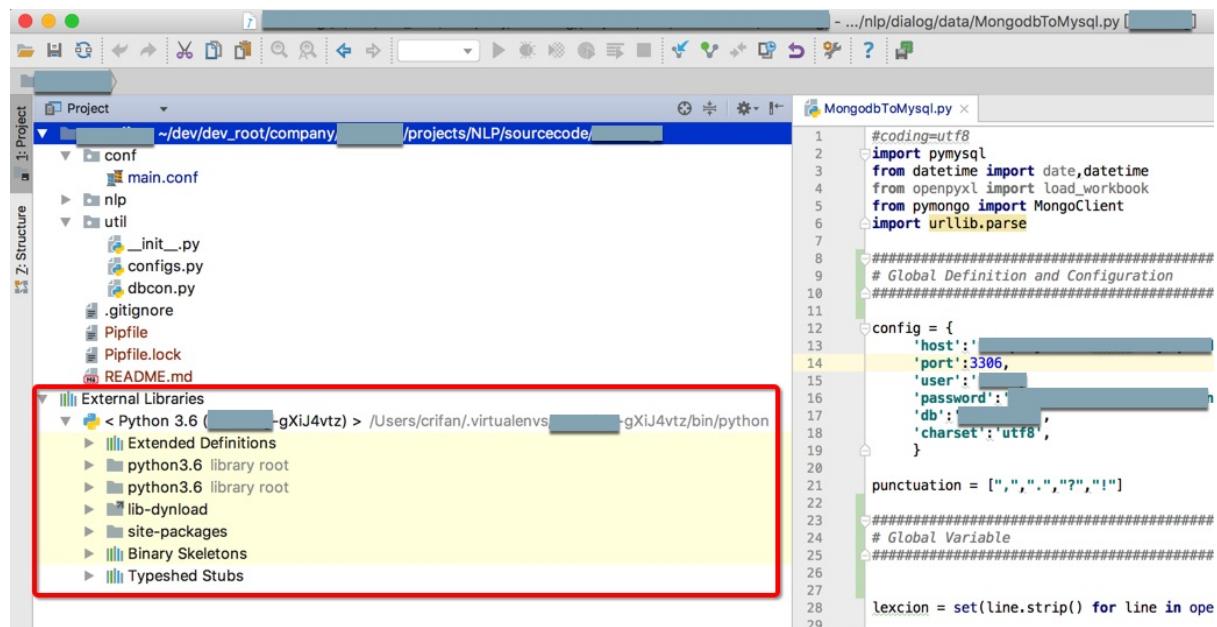


项目底部会显示，有后台任务去：

- 更新python解释器
- 重新建立索引|indexing



然后就可以在项目左边的文件结构中的 External Libraries 中看到当前Python解析器=当前Python版本，和已安装的相关库了：



crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12
22:55:46

文件编码设置

要声明好文件编码，否则PyCharm会报错

详见：

[【已解决】Mac中PyCharm中Python代码调试报错：SyntaxError Non-ASCII character \xe7 in file on line but no encoding](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15 22:57:17

调试项目

创建完毕项目，添加完毕代码后，即可去调试代码。

添加调试配置

调试代码前，需要先添加和调试的配置。

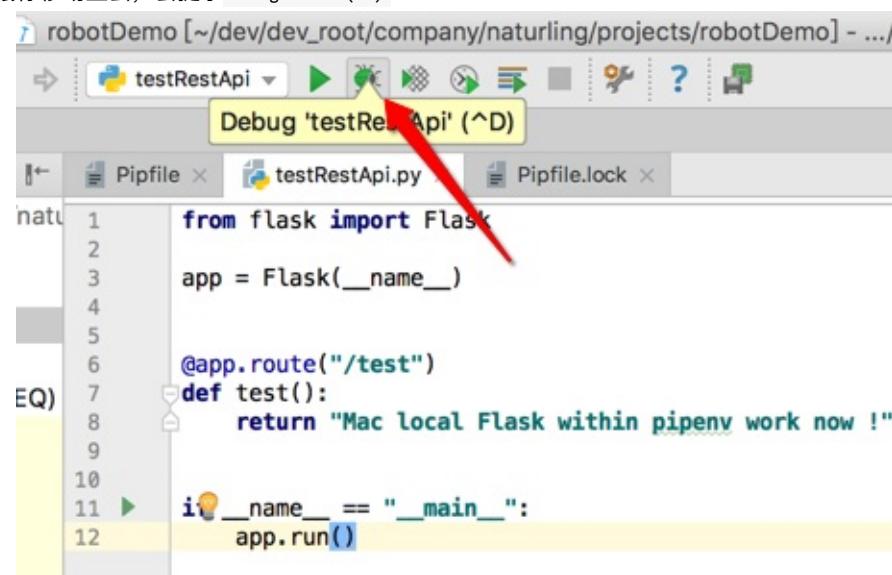
自动创建调试配置

对于多数情况下，其实可以用PyCharm的智能之处，自动生成调试配置，开始调试。

目前有2种方式：

1. 方式1：直接点击小爬虫一样的 Debug 的按钮：

- 且鼠标移动上去，会提示 Debug 'xxx' (^D)



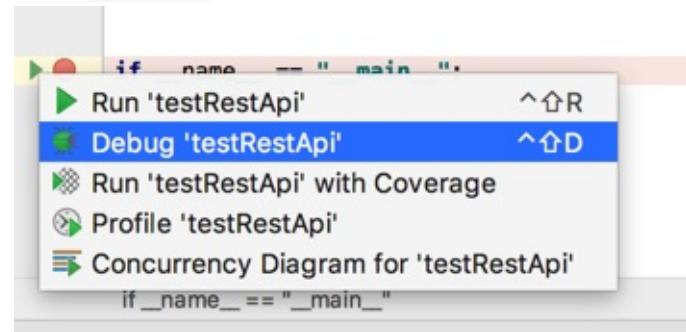
2. 方式2：点击 if __name__ == "__main__" 中的调试选项

- 对于代码中有：if __name__ == "__main__" 的情况，PyCharm能自动识别出来，在左边会多出一个绿色按钮，点击后，会出现多个选项：

A screenshot of the PyCharm IDE showing a code editor with Python code. A tooltip is displayed over the line `if __name__ == "__main__":` at line 11. The tooltip contains the following options:

- Run 'testRestApi'
- Debug 'testRestApi'
- Run 'testRestApi' with Coverage
- Profile 'testRestApi'
- Concurrency Diagram for 'testRestApi'

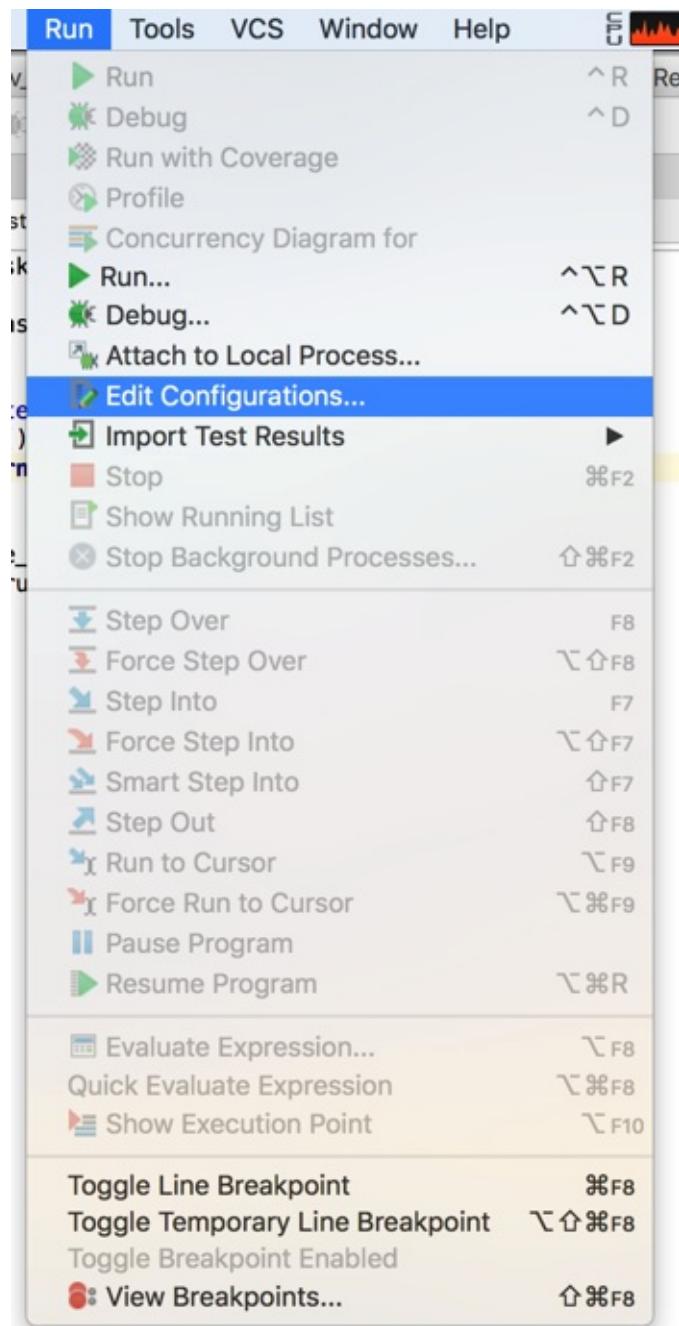
- 点击其中的 Debug 'xxx'



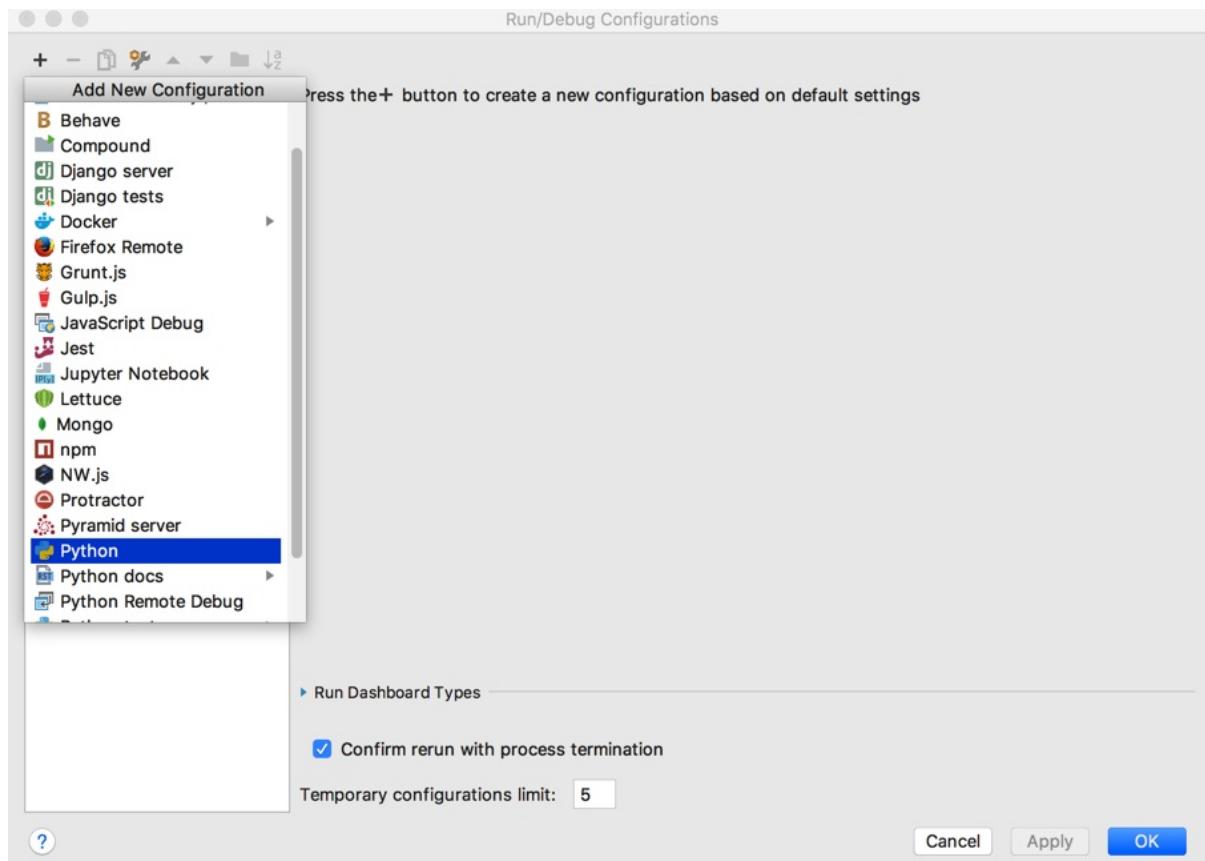
正常情况，即可生成调试配置，开始调试。

手动添加调试配置

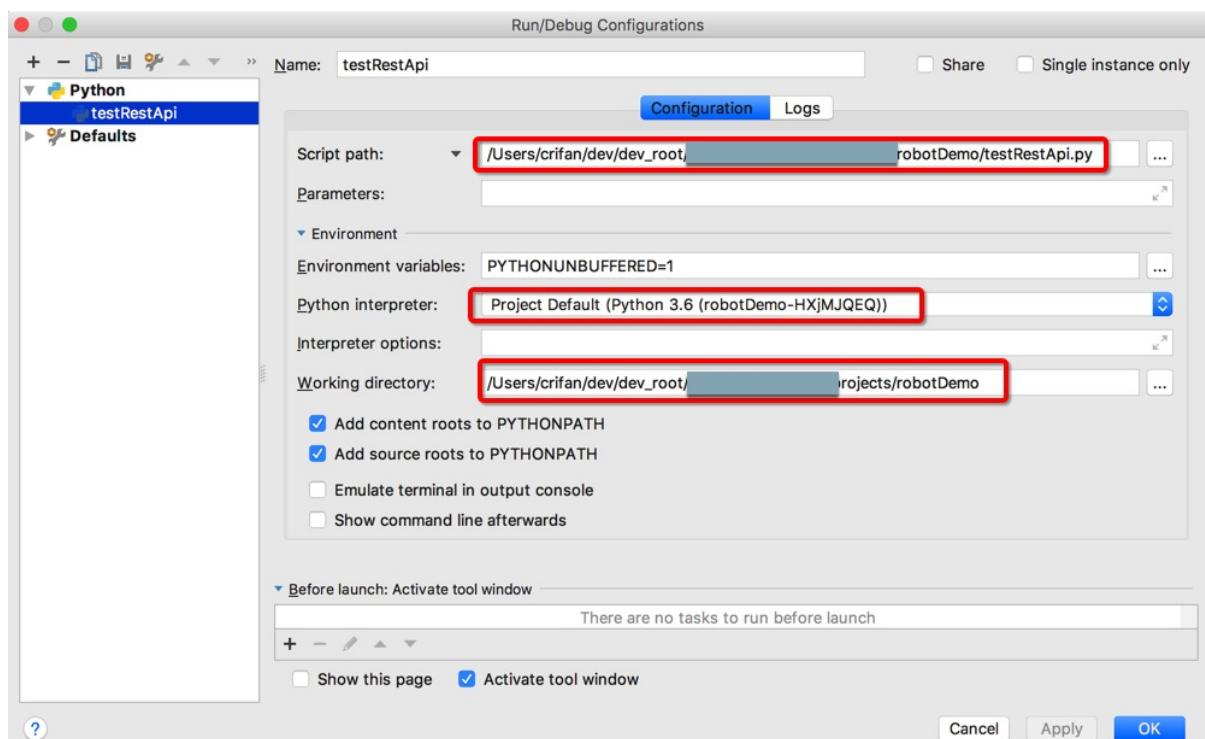
点击 Run->Edit Configurations :



点击 + 创建一个Python的调试配置：



然后输入对应的配置参数：



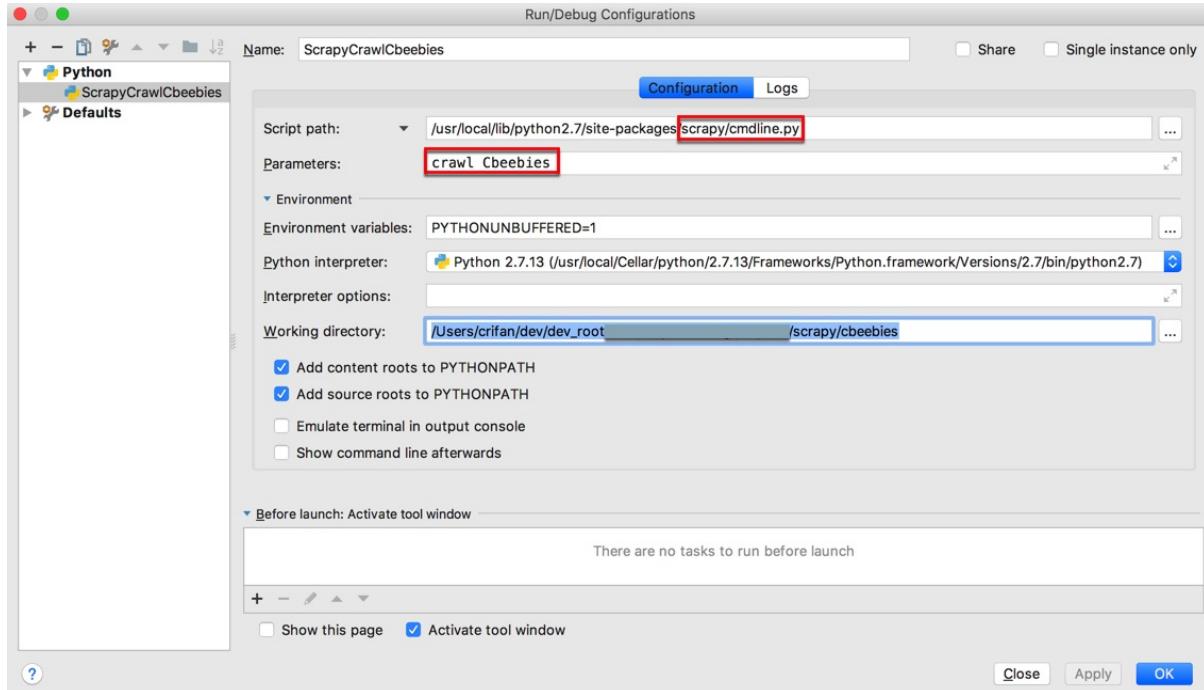
- Name : 随便设置个名字即可，比如此处的： testRestApi
- Script Path : /Users/crifan/dev/xxx/robotDemo/testRestApi.py
 - 你要调试的Python文件的完整路径
- Environment variables : PYTHONUNBUFFERED=1

- Working directory : /Users/crifan/dev/xxx/robotDemo
 - 一般设置为当前项目的根目录
 - PyCharm一般也会自动设置为你 Script Path 的值对应的文件所在的目录的

给Scrapy添加调试配置

对于一些相对特殊的项目，需要搞清楚配置要启动的python文件，和要传入的参数才可以。

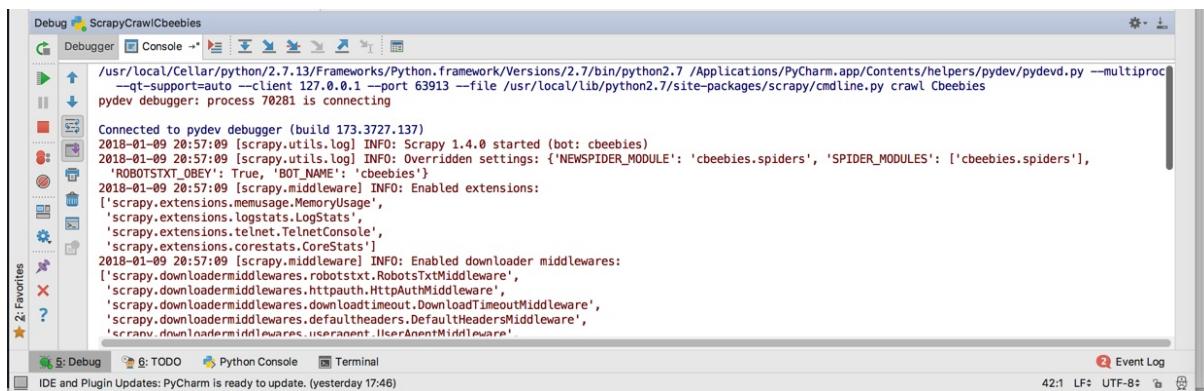
比如，Scrapy项目，就是这种，对应调试配置如下：



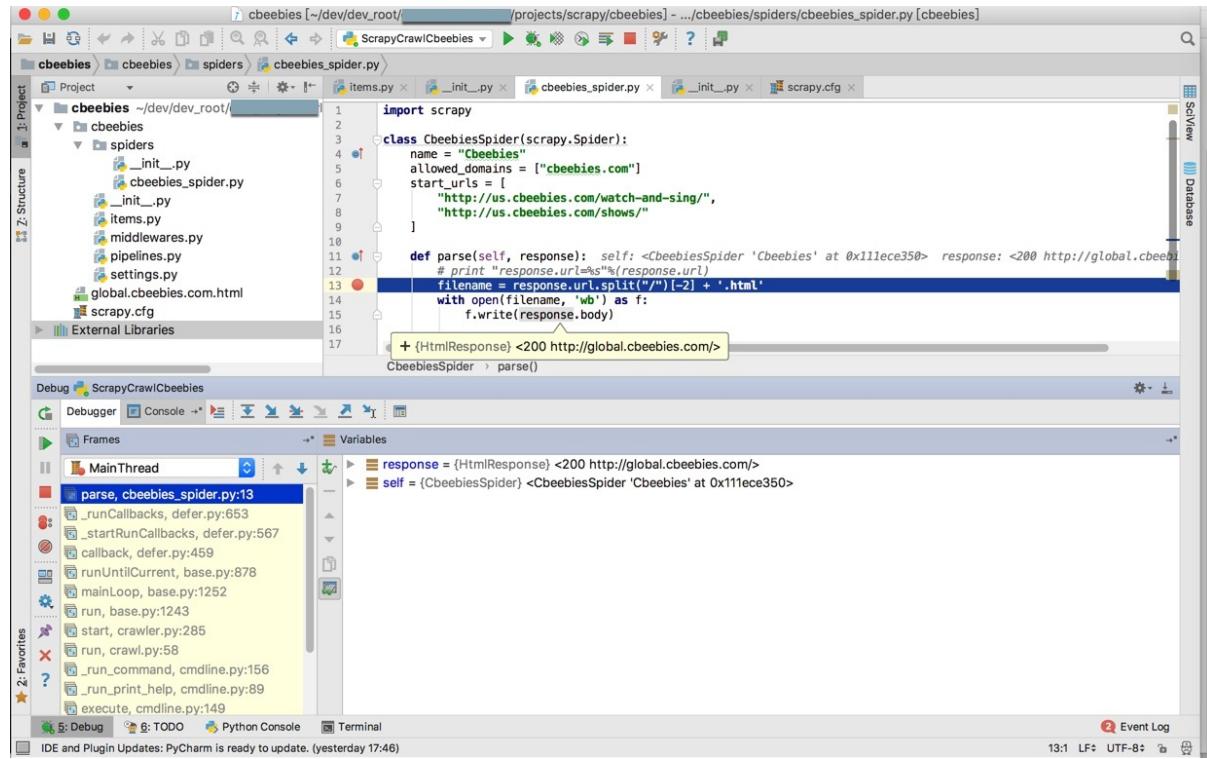
核心参数：

- Script Path : /your_python_version/site-packages/scrapy/cmdline.py
- Parameters : crawl yourScrapyProjectName

即可正常启动调试：



遇到对应断点可以停下来：



调试代码

上述调试配置弄好后，加上断点：

点击每行代码的最左边，行号的左边，即可给该行代码加上断点

然后即可开始调试：

```
from flask import Flask
app = Flask(__name__)
@app.route("/test")
def test():
    return "Mac local Flask within pipenv work now !"
if __name__ == "__main__":
    app.run()
```

Debug: testRestApi (1)

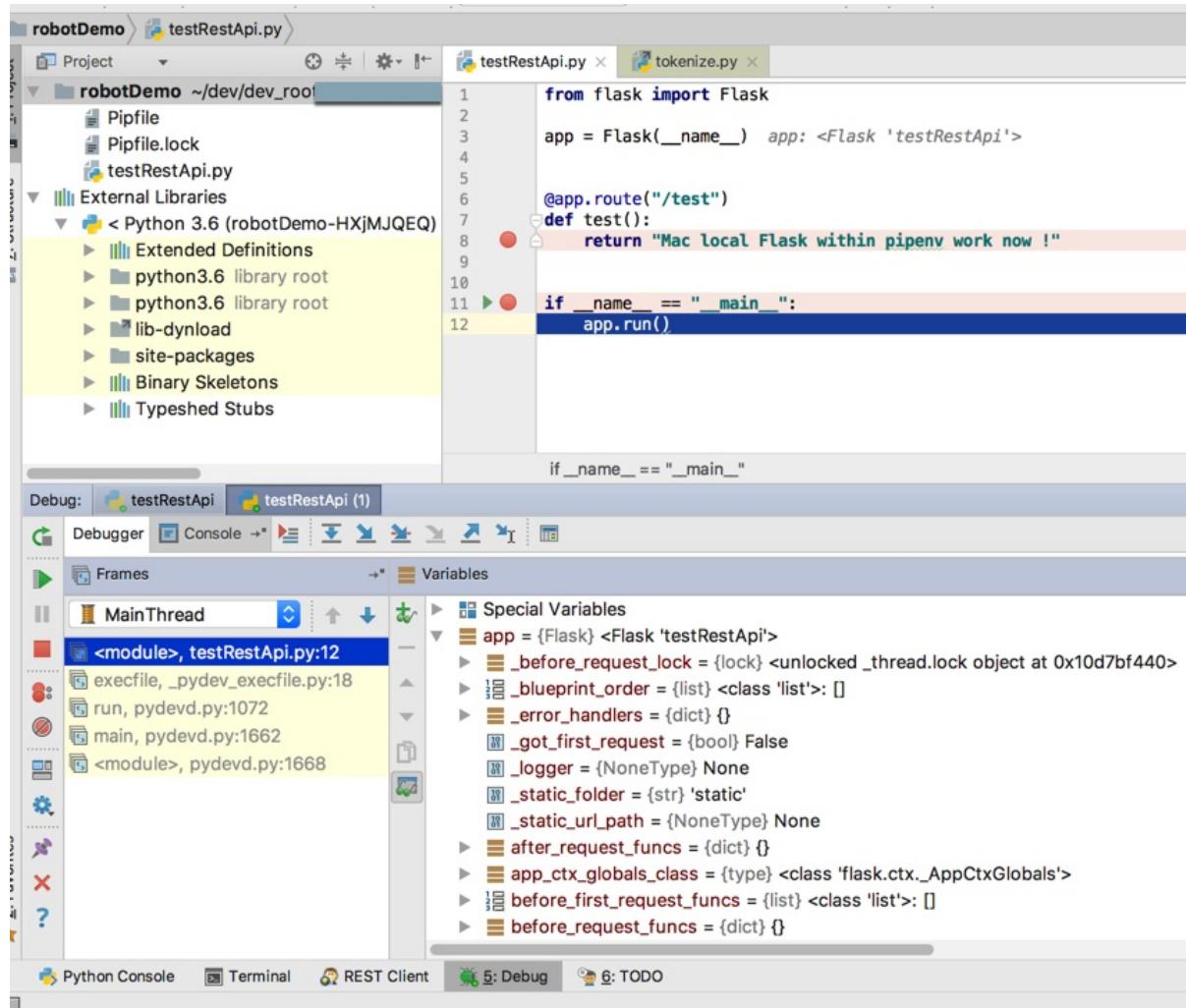
Frames:

- MainThread
- <module>, testRestApi.py:11
- execfile, _pydev_execfile.py:18
- run, pydevd.py:1072
- main, pydevd.py:1662
- <module>, pydevd.py:1668

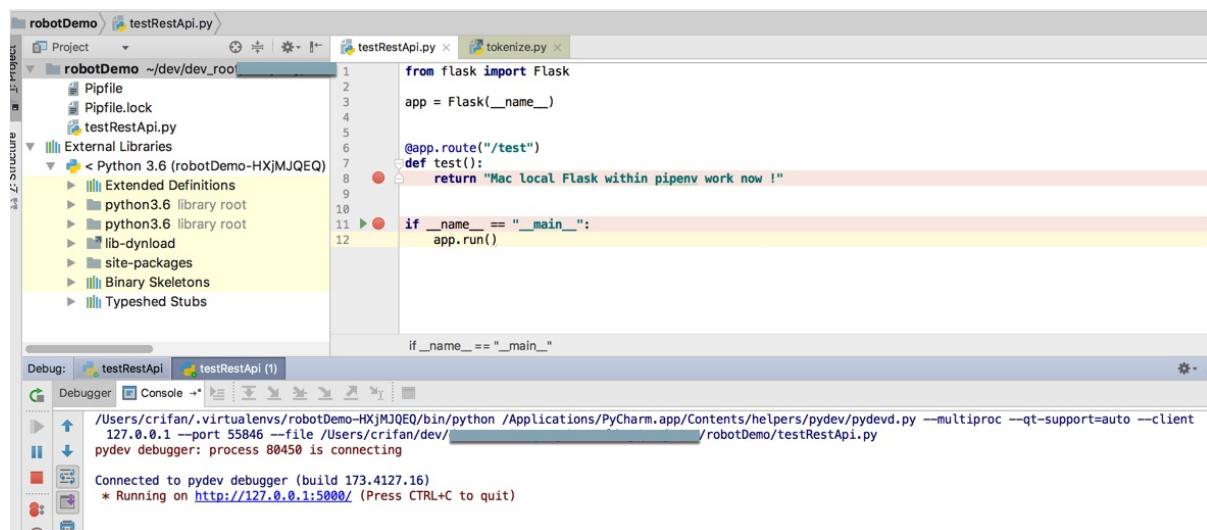
Variables:

- Special Variables
- app = {Flask} <Flask 'testRestApi'>

然后单步调试，进入下一行代码：



此处Flask的代码即可正常运行：



远程调试

目前没具体折腾过PyCharm的远程调试。

不过有找过一些资料：

- 使用PyCharm进行Python远程调试
- Pycharm的远程代码编辑 - kiwik's blog

抽空去试试。

不过，根据[版本选择](#)的解释，看来是需要：收费的专业版PyCharm，才支持远程调试 `remote debug`。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15
14:08:51

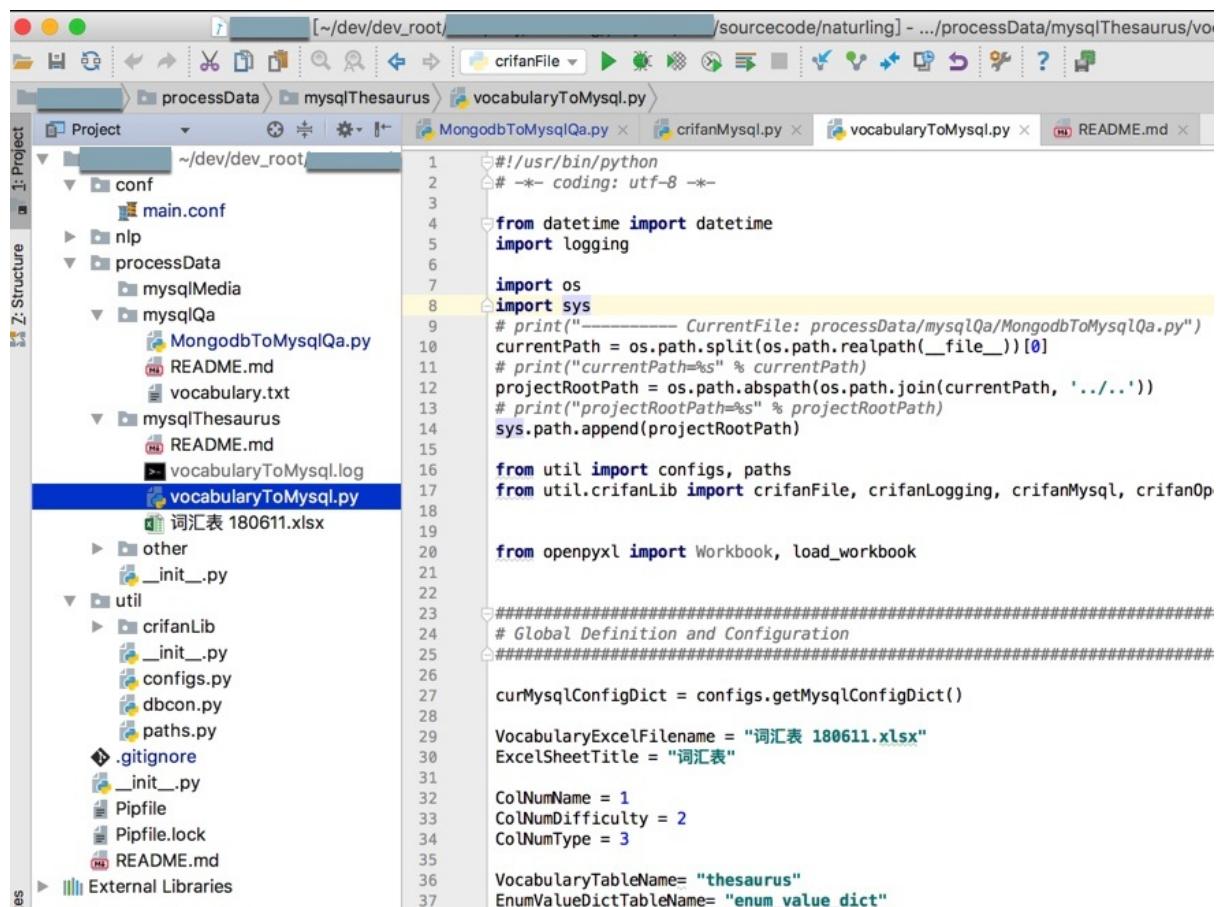
调试心得

在PyCharm调试期间，有些心得和注意事项，整理如下。

如果切换调试文件，记得更新Debug配置

背景是，在某个项目中，分别要单独调试某几个Python文件。

当时已经调试完毕其中一个了：



The screenshot shows the PyCharm interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the root directory `~/dev/dev_root/`. It includes:
 - `conf`: contains `main.conf`
 - `nlp`
 - `processData`: contains `mysqlMedia` and `mysqlQa` (which contains `MongodbToMysqlQa.py`, `README.md`, and `vocabulary.txt`).
 - `mysqlThesaurus`: contains `README.md`, `vocabularyToMysql.log`, and `vocabularyToMysql.py`.
 - `other`: contains `__init__.py`
 - `util`: contains `crifanLib` (with `__init__.py`, `configs.py`, `dbcon.py`, and `paths.py`), `.gitignore`, `__init__.py`, `Pipfile`, `Pipfile.lock`, and `README.md`.
- Code Editor:** The right panel displays the Python file `vocabularyToMysql.py` with the following content (lines 1-37):


```

1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from datetime import datetime
5  import logging
6
7  import os
8  import sys
9
10 # print("----- CurrentFile: processData/mysqlQa/MongodbToMysqlQa.py")
11 currentPath = os.path.split(os.path.realpath(__file__))[0]
12 # print("currentPath=%s" % currentPath)
13 projectRootPath = os.path.abspath(os.path.join(currentPath, '../..'))
14 # print("projectRootPath=%s" % projectRootPath)
15 sys.path.append(projectRootPath)
16
17 from util import configs, paths
18 from util.crifanLib import crifanFile, crifanLogging, crifanMysql, crifanOp
19
20 from openpyxl import Workbook, load_workbook
21
22 #####
23 # Global Definition and Configuration
24 #####
25
26 curMysqlConfigDict = configs.getMysqlConfigDict()
27
28 VocabularyExcelFilename = "词汇表 180611.xlsx"
29 ExcelSheetTitle = "词汇表"
30
31 ColNumName = 1
32 ColNumDifficulty = 2
33 ColNumType = 3
34
35 VocabularyTableName= "thesaurus"
36 EnumValueDictTableName= "enum_value_dict"
37 
```

然后切换到另外一个python文件去调试：

```

322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357

# NEW: get vocabulary from mysql 'thesaurus' table
getVocabularySql = "SELECT * FROM `%(VocabularyTableName)s`"
logging.info("getVocabularySql=%s", getVocabularySql)
getVocabularyOk, resultDict = connection.executeSql(getVocabularySql)
logging.info("getVocabularyOk=%s, resultDict=%s", getVocabularyOk, resultDict)
if getVocabularyOk:
    vocabularyRecordList = resultDict["data"] [0]
    for eachRecord in vocabularyRecordList:
        wordName = eachRecord["name"]
        wordName = wordName.strip()
        # gVocabularyList.append(wordName)
        gVocabularySet.add(wordName)

    countVocabularySet = len(gVocabularySet)
    logging.info("gVocabularySet=%s", gVocabularySet)
else:
    logging.error("Get vocabulary failed of sql: %s", getVocabularySql)

def main():
    # init logging
    crifanLogging.loggingInit(crifanFile.getInputFileBasenameNoSuffix() + ".log")
    logging.info("Logging initialized")

    # connection = pymysql.connect(**curMySQLConfigDict)
    connection = crifanMySQL.MysqlDb(config=curMySQLConfigDict)
    logging.info("Mysql connected: connection=%s", connection)

    initVocabularyList(connection)

    # gridfsFileGenerator = mongoGridfsGenerator()
    # insertMedia(connection, gridfsFileGenerator)

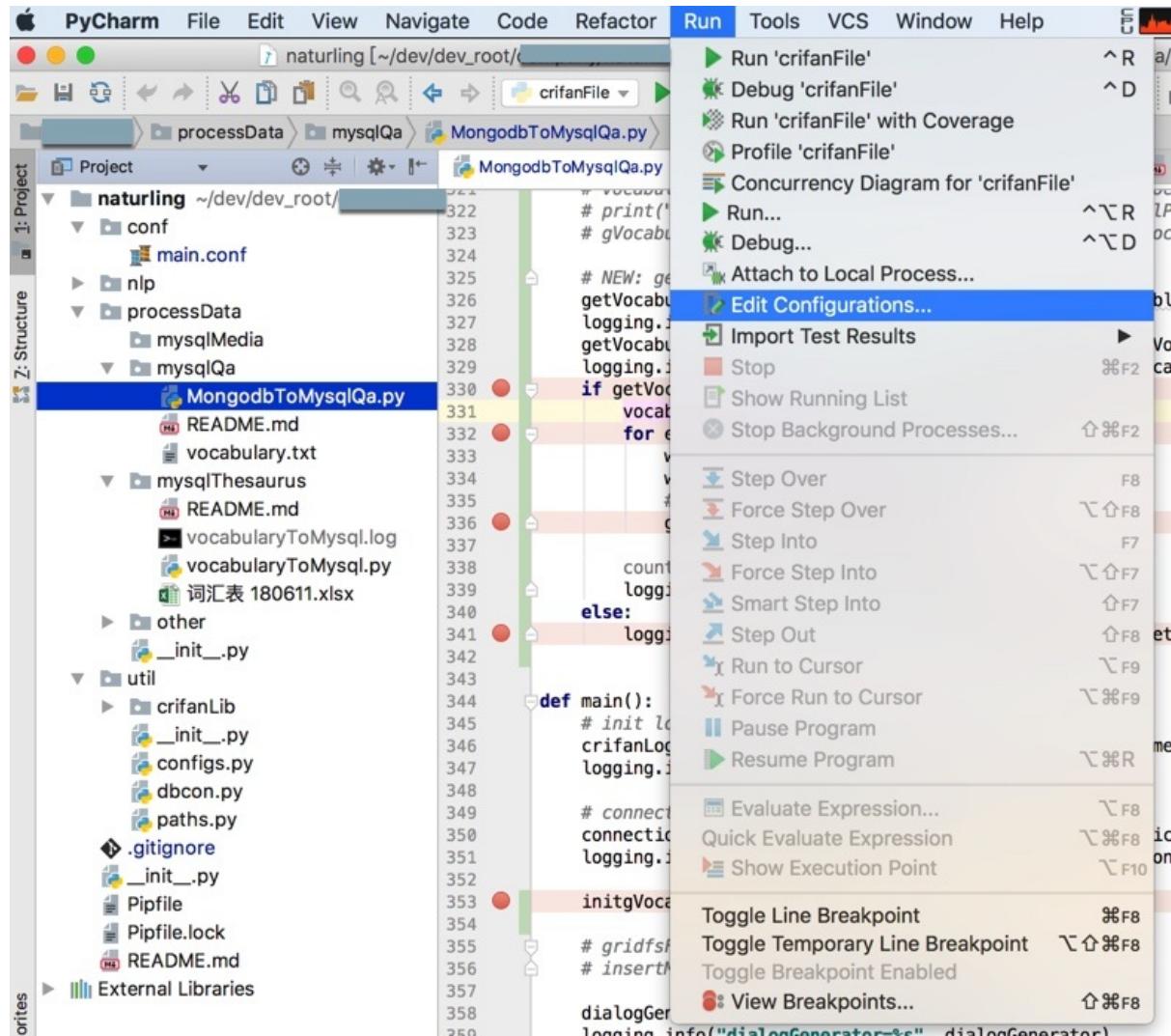
```

发现遇到很多奇怪的现象：

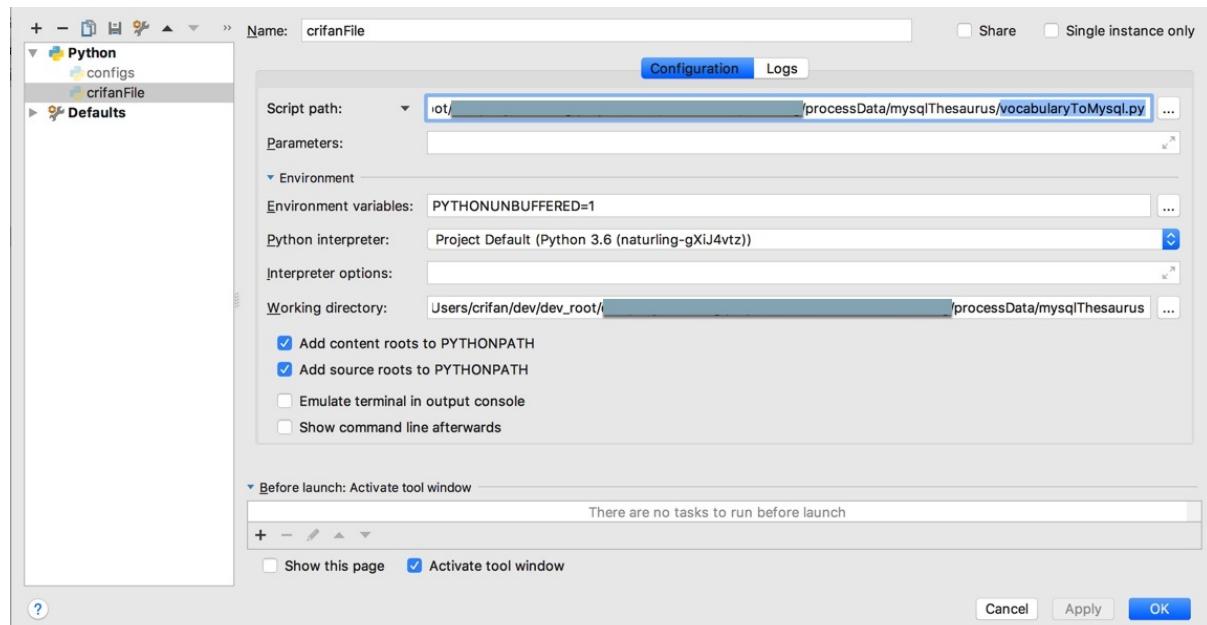
- log输出也会别的文件的log
- 新文件中的断点也始终无法执行到
- 等等

最后发现原因是：调试的始终是之前的文件，即没有更改调试配置

所以需要去：更新Debug配置

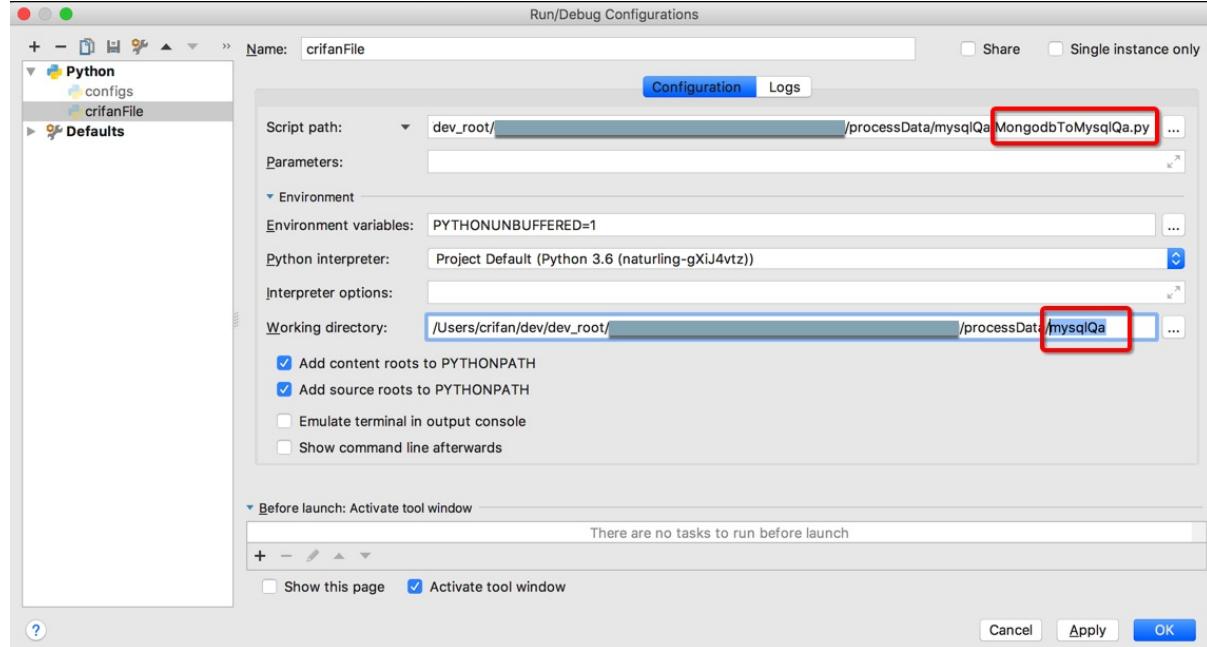


去把之前的要调试的文件：



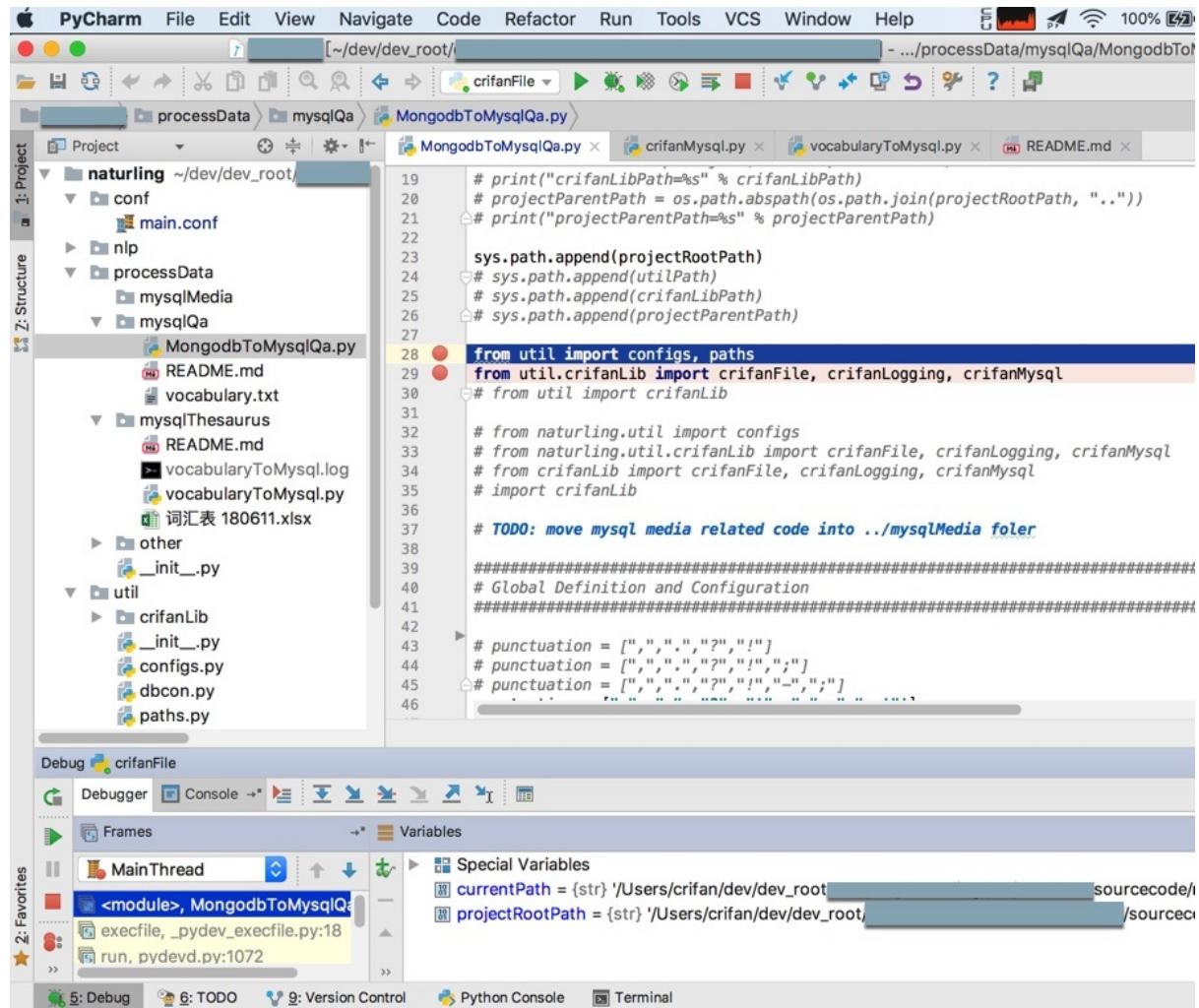
换成当前要调试的新的文件。

可以注意到：不仅要调试的文件名变了，对当前工作路径，也更新了：



(PyCharm很是智能和方便，考虑的很周到)

然后才能继续调试：



举例：没有更新调试导致文件找不到FileNotFoundException

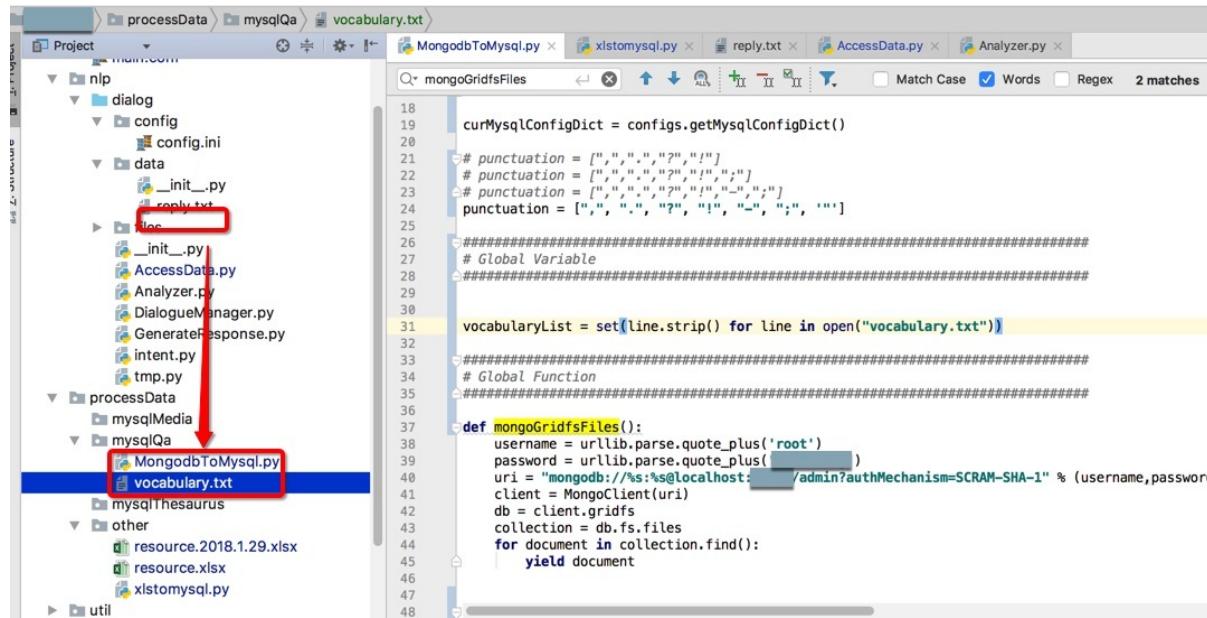
关于文件找不到：

[【已解决】PyCharm中调试pipenv虚拟环境中文件出错：FileNotFoundException Errno 2 No such file or directory pipenv run python py](#)

后记：

此处的vocabulary.txt和py文件MongodbToMysql.py是在同一个文件夹 nlp/dialog/data/ 下面的。

在移动了一次文件后：



都移动到了另外文件夹 processData/mysqlQa/MongodbToMysql.py 中了。

而之前的Debug调试设置中，是可以正常的执行代码：

```
vocabularyList = set(line.strip() for line in open("vocabulary.txt"))
```

去打开同文件夹下txt文件的。

但是在移动文件夹之后，再去调试，始终出错：

The screenshot shows the PyCharm IDE interface. On the left is a file tree with several projects and files. In the center is a code editor with Python code. A red arrow points from the text "vocabularyList = set(line.strip() for line in open('vocabulary.txt'))" in the code editor to the corresponding line in the stack trace below.

```

21 # punctuation = [",", ".", "?", "!"]
22 # punctuation = [",", ".", "?", "!", "-", "_"]
23 # punctuation = [",", ".", "?", "!", "-", "_"]
24 punctuation = [",", ".", "?", "!", "-", ";", "\n"]
25
26 ##### Global Variable #####
27 #####
28 #####
29 #####
30 #####
31 vocabularyList = set(line.strip() for line in open("vocabulary.txt"))
32 #####
33 #####
34 #####

```

```

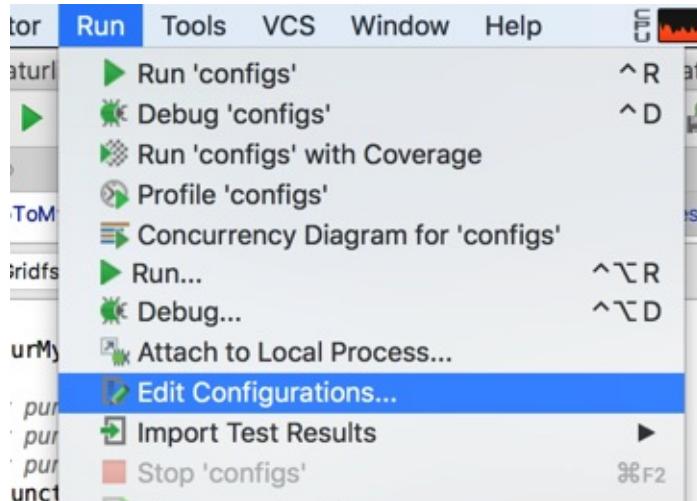
Traceback (most recent call last):
  configParser=<configparser.ConfigParser object at 0x105b88400>
  File "/Applications/PyCharm.app/Contents/helpers/pydev/pydevd.py", line 1668, in <module>
    main()
  mysqlConfigDict={'host': '127.0.0.1', 'port': 3306, 'user': 'root', 'password': 'crifan_mysql', 'db': 'naturling', 'charset': 'utf8'}
  File "/Applications/PyCharm.app/Contents/helpers/pydev/pydevd.py", line 1667, in main
    globals = debugger.run(setup['file'], None, None, is_module)
  File "/Applications/PyCharm.app/Contents/helpers/pydev/pydevd.py", line 1672, in run
    pydev_imports.execfile(file, globals, locals) # execute the script
  File "/Applications/PyCharm.app/Contents/helpers/pydev/pydev_imps/_pydev_execfile.py", line 18, in execfile
    exec(compile(contents+"\n", file, 'exec'), glob, loc)
  File "/Users/crifan/dev_root/_projects/NLP/sourcecode/processData/mysqlQa/MongodbToMysql.py", line 31, in <module>
    vocabularyList = set(line.strip() for line in open("vocabulary.txt"))
FileNotFoundError: [Errno 2] No such file or directory: 'vocabulary.txt'

Process finished with exit code 1

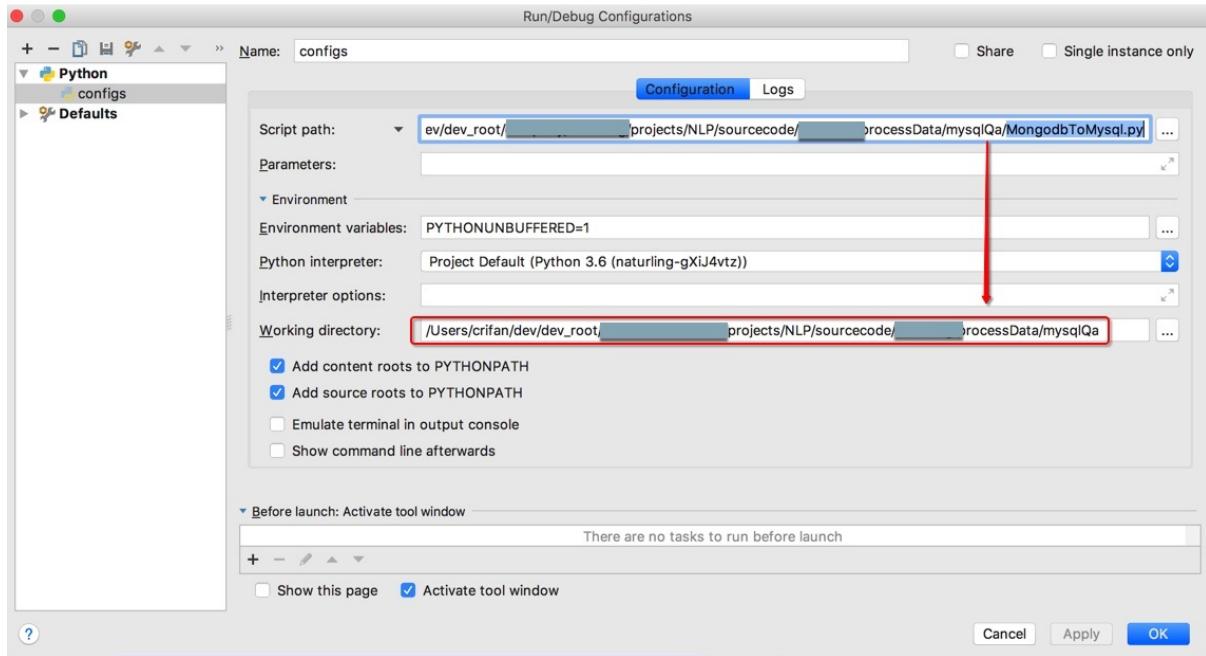
```

找了找原因，发现是：

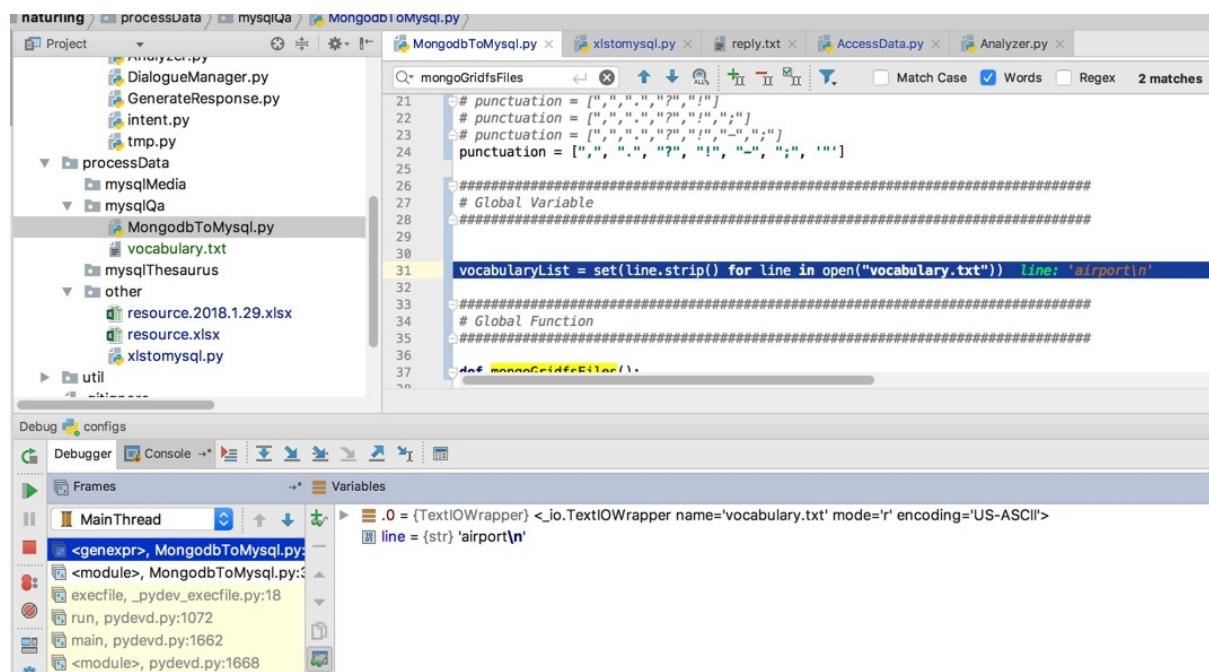
在移动文件之后，之前的Debug配置中的当前目录，没有变化，所以去改为py所在的文件夹：



(通过单独选择py文件，自动会设置 Working directory，或者手动输入，都可以)



然后就可以解决找不到文件的问题了：



crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15
18:42:33

项目部署

开发和调试完毕项目后，会涉及到发布和部署项目代码。

PyCharm对于项目部署、代码上传也有很好的支持。

比如想要把本地调试好的项目代码：

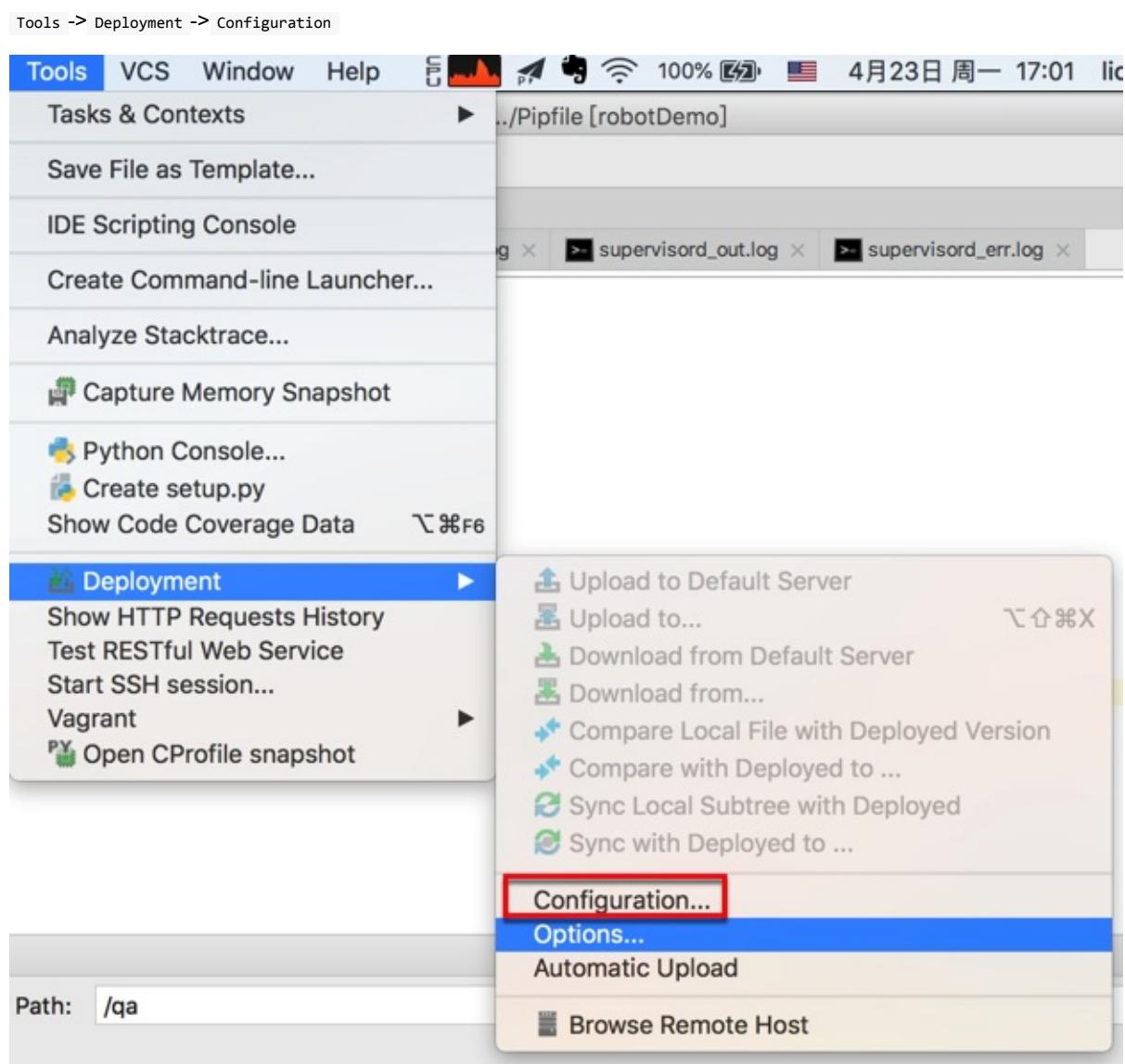
```
/Users/crifan/dev/dev_root/company/xxx/xxxRobotDemoServer
```

上传到服务器对应位置：

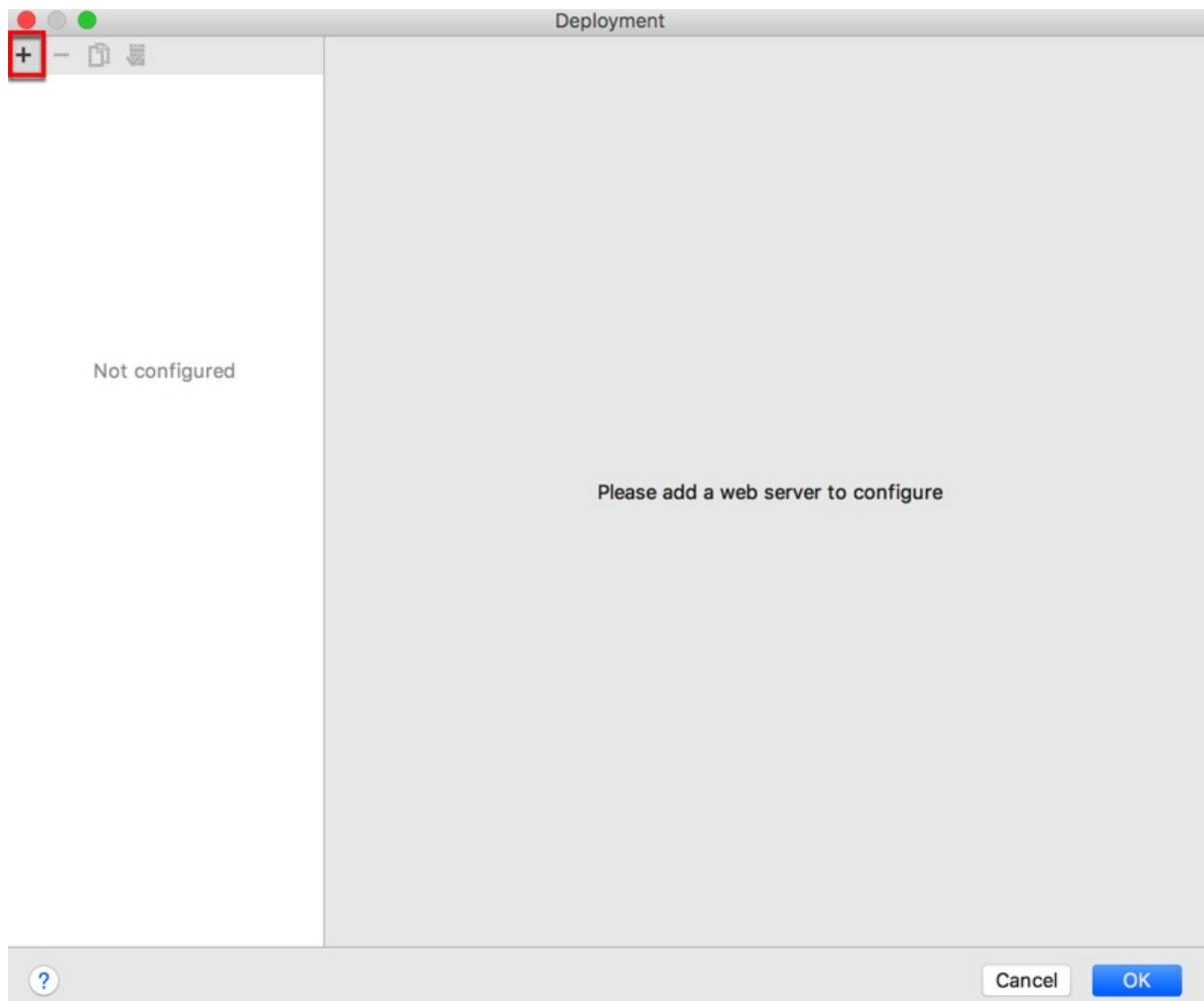
```
/yyy_20180101/web/server/robotDemo
```

添加和配置Deployment参数

先去新增一个 Deployment 配置：

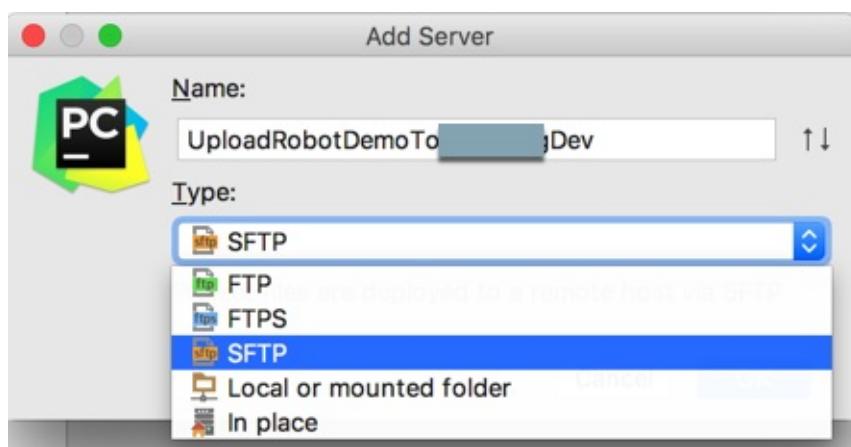


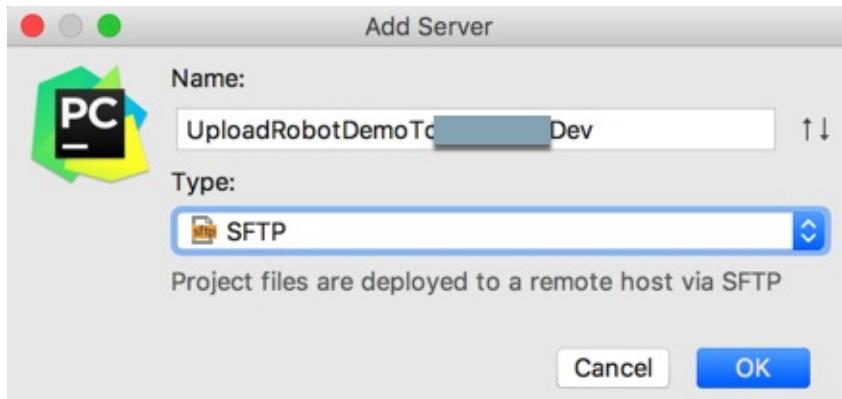
然后点击 + 加号去添加：



Add Server 中，输入名字后，选择类型是：SFTP

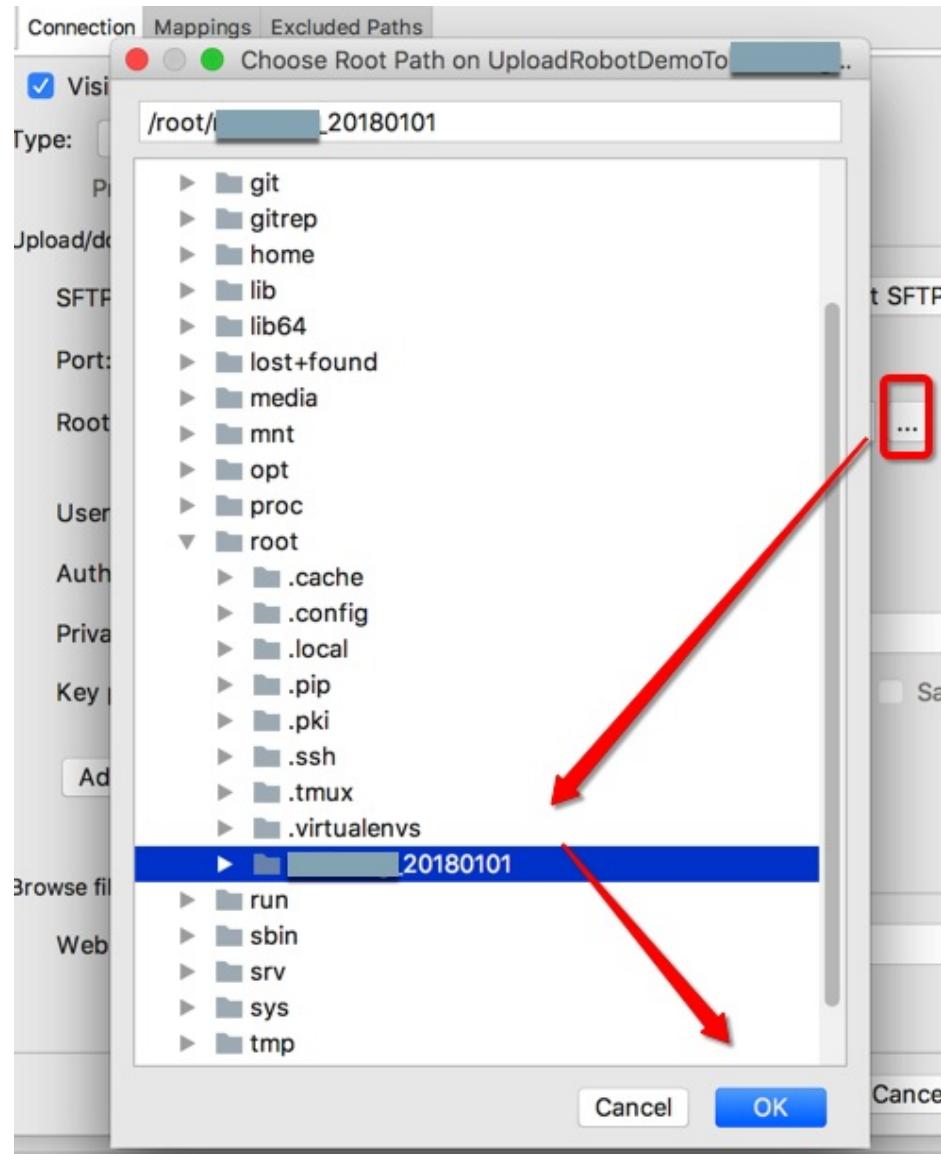
- 说明：此处由于服务器中没有配置FTP服务器，所以用只要支持 SSH 就内置支持的 SFTP





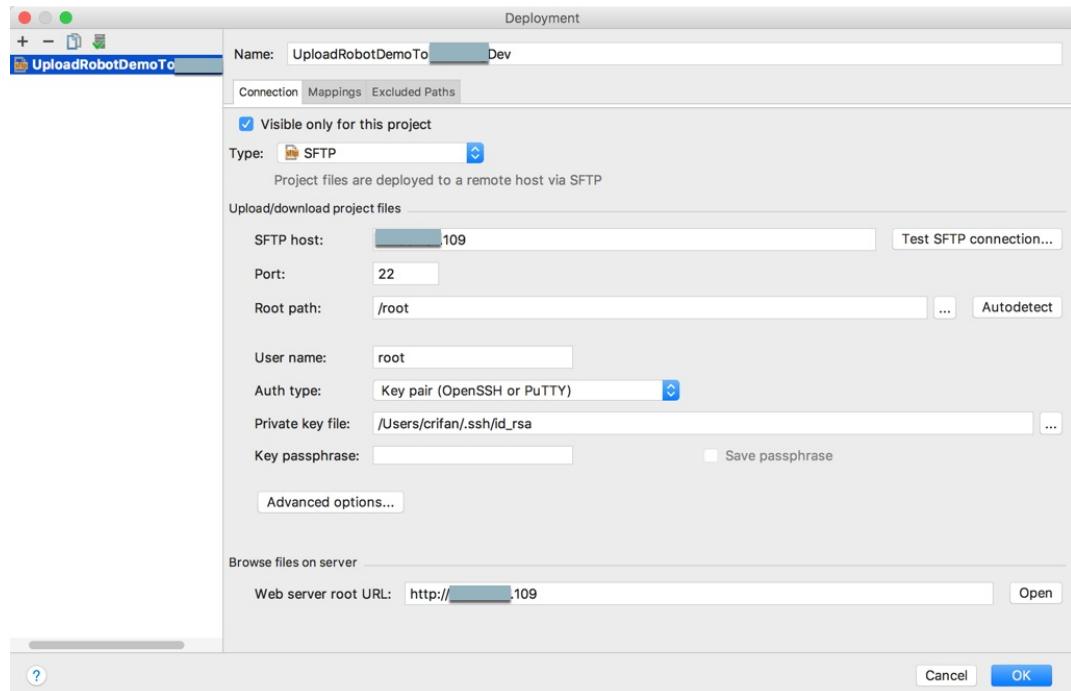
对应的 PyCharm 的 Deployment 设置是：

- Connection
 - 详细配置：
 - SFTP Host : xx.xx.xx.109
 - 服务器的IP地址
 - Port : 22
 - 默认一般就是 22
 - Root Path : /root
 - 选择之前，记得要先去设置好登录方式（用户名+密码，还是用户名+auth）
 - 注意路径的设置：
 - 可以点击三个点，去选择对应的路径



- 也可以点击 Auto Detect 会自动
 - (默认)设置为当前用户的默认目录
 - 此处用户是root, 所以默认路径是: /root
- User name : root
 - SSH的用户名, 一般都是 root
- Auth type : Key Pair (OpenSSH or PuTTY)
 - 密码模式: 之前用的多的是, 用户名加密码
- Private key file : /Users/crifan/.ssh/id_rsa
 - key文件模式: 后面别人用了私钥文件, 则选择 Key Pair (OpenSSH or Putty)
 - Private key file : 选择之前自己创建好的, 可以用于SSH登录的rsa的私钥文件
 - 比如此处放在了: /Users/crifan/.ssh/id_rsa

◦ 如图

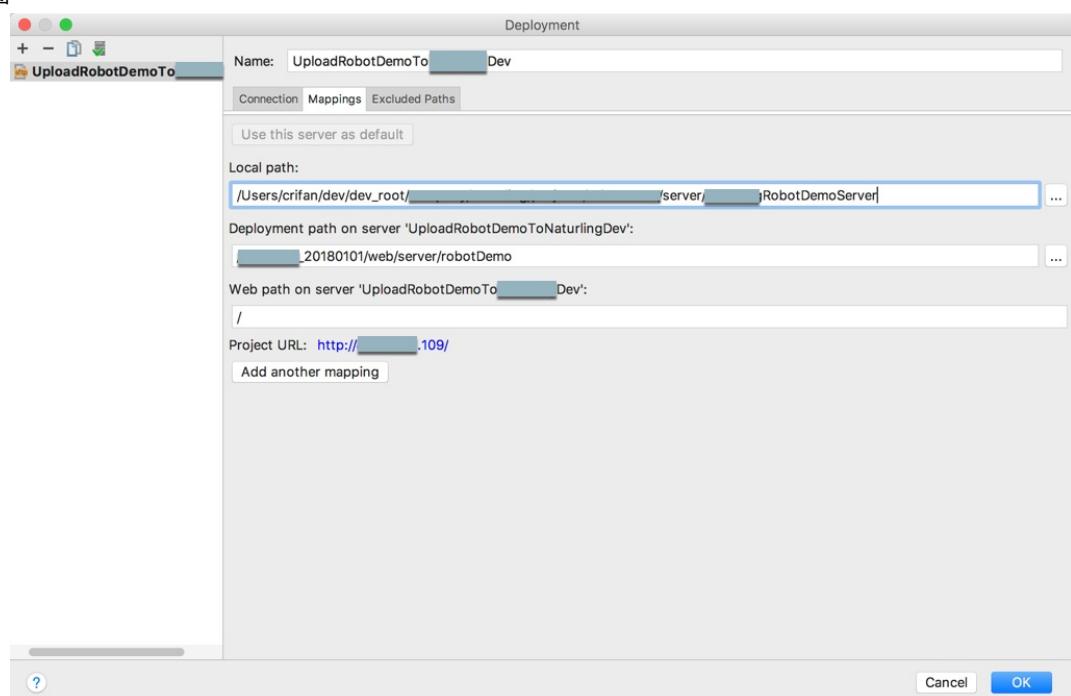


- Mapping :

- 详细配置：

- Local Path : /Users/crifan/dev/dev_root/company/xxx/xxxRobotDemoServer
 - 自己本地的目录，写绝对路径，好理解
 - Deployment Path on server : /yyy_20180101/web/server/robotDemo
 - 注意：此处的路径是相对于 connection 中的 Root Path 来说的
 - 我此处的 Connection 中的 Root Path 是 /root
 - 但是此处还是 / 开头的：/naturling_20180101/web/server/robotDemo
 - 其实更简单省事的办法是：点击三个点，自己选择列出来的路径，即可
 - Web Path on server : /

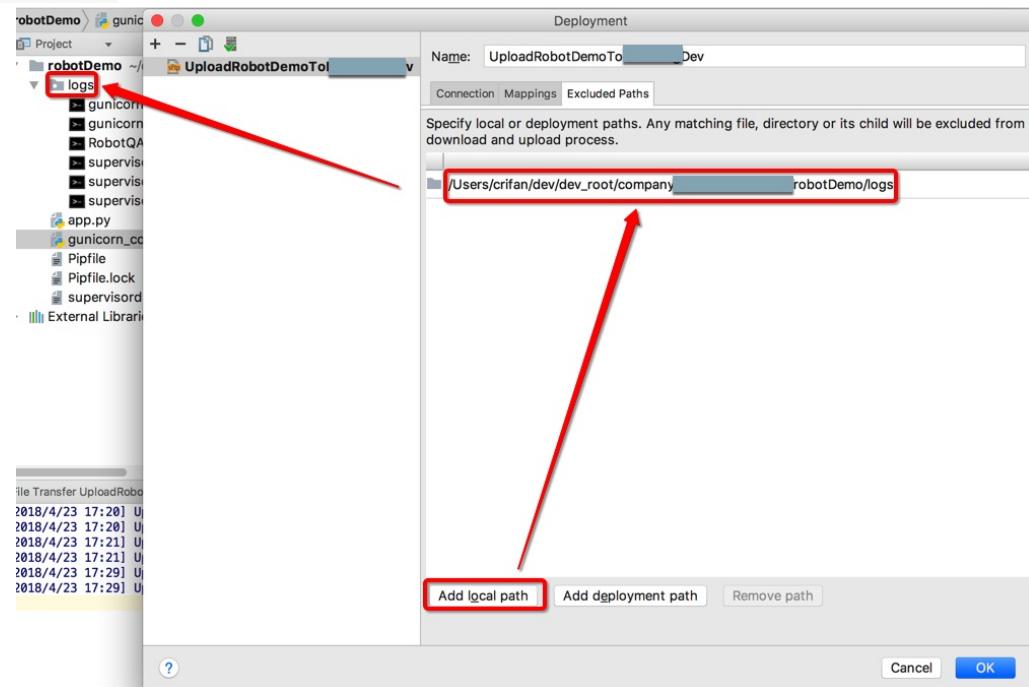
- 如图：



- Excluded Path

- 设置步骤

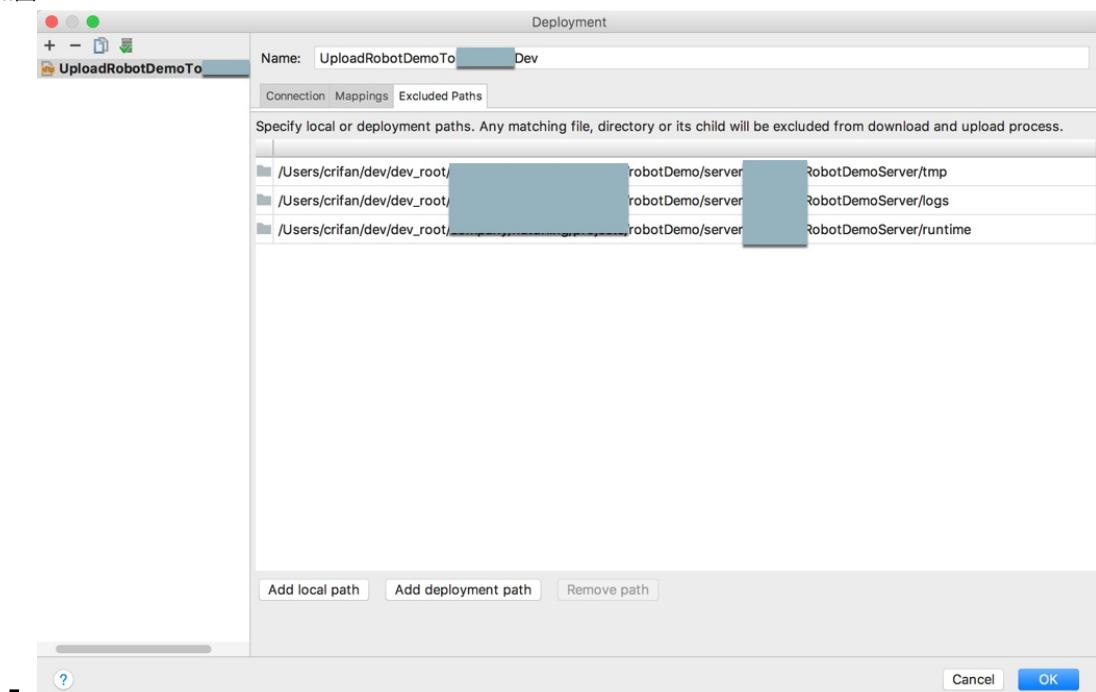
- Add local Path -> 然后输入对应的本地的要排除的路径



- 详细配置：

- /Users/crifan/dev/dev_root/xxx/xxxRobotDemoServer/tmp
- /Users/crifan/dev/dev_root/xxx/xxxRobotDemoServer/logs
- /Users/crifan/dev/dev_root/xxx/xxxRobotDemoServer/runtime

- 如图：

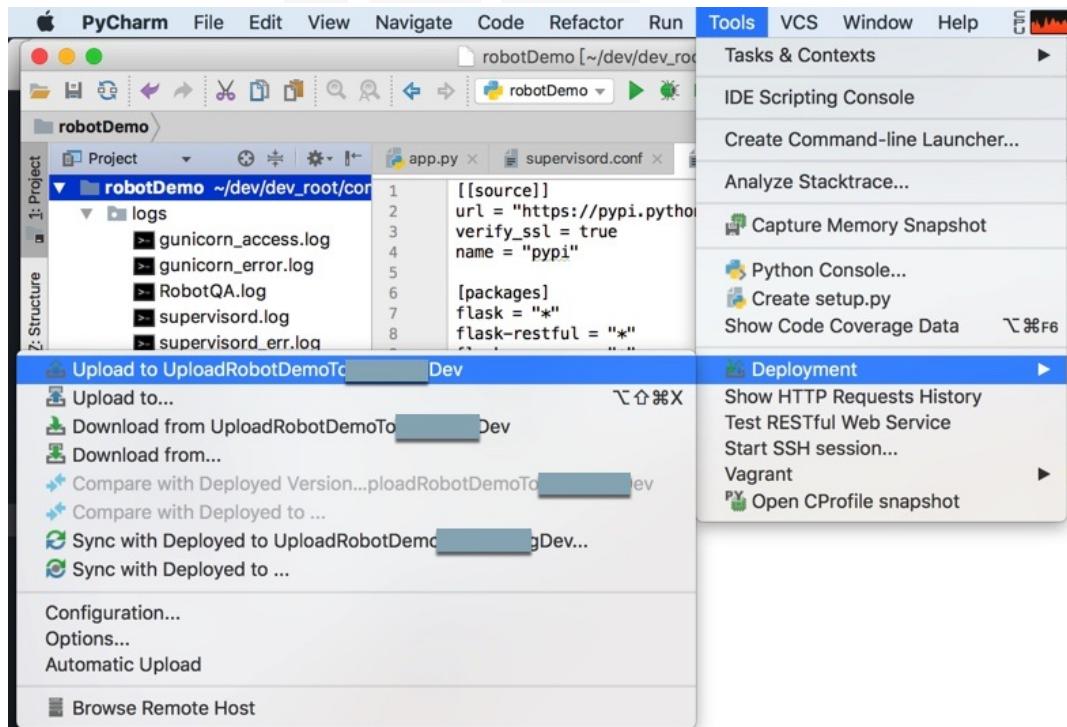


如何使用=同步文件到服务器

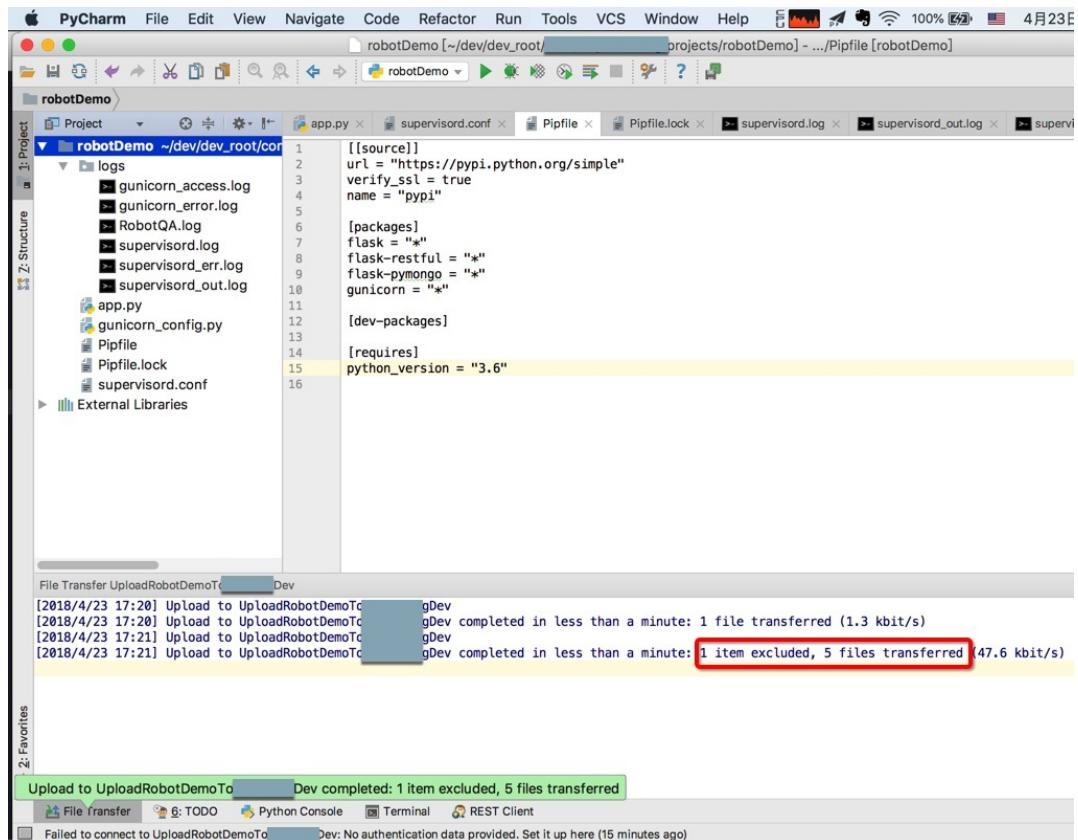
之后就可以：

- 同步整个项目（有改动的部分）

- 先点击项目根目录后，再去点击 Tools -> Deployment -> Upload to xxx

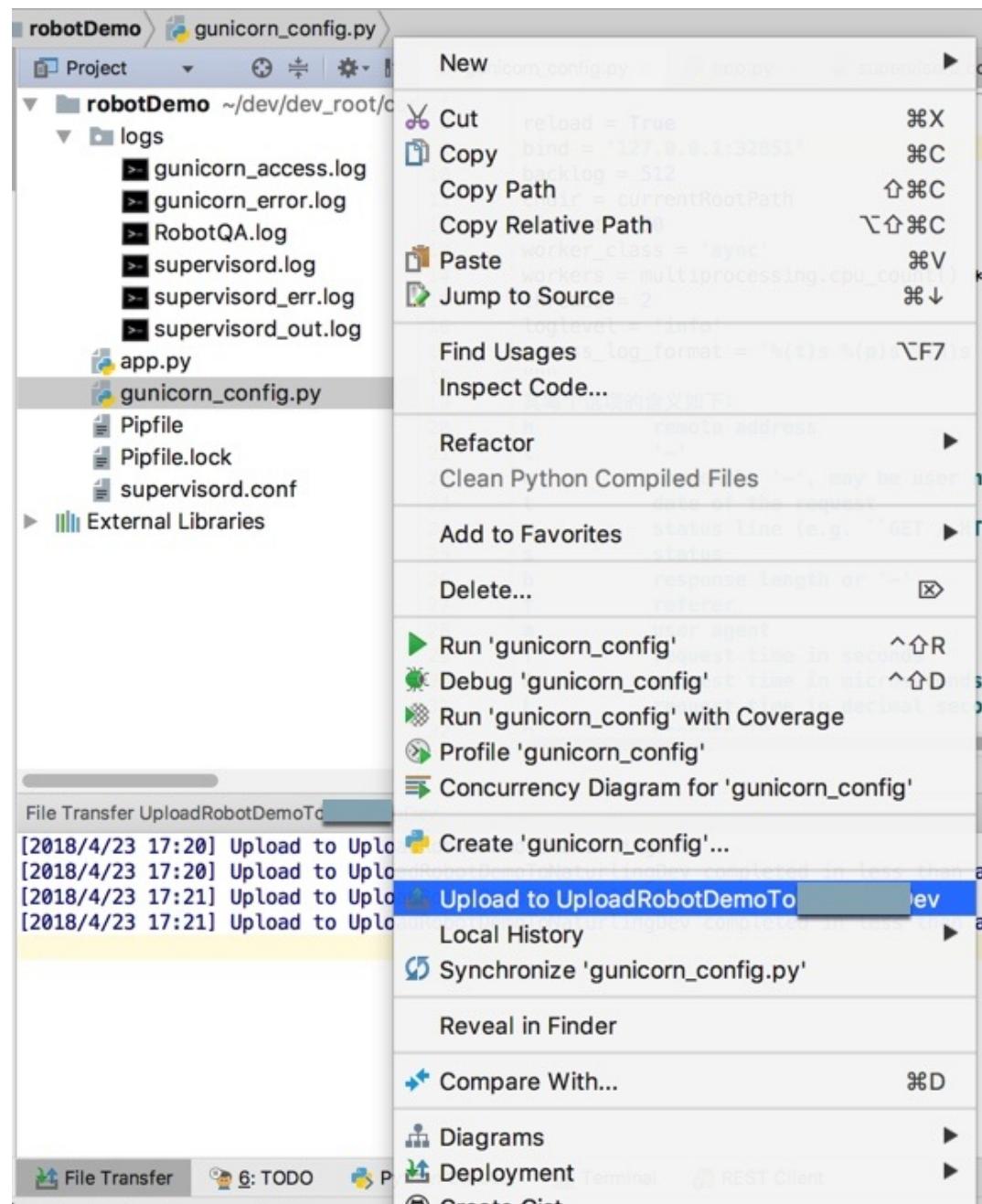


- 即可上传整个项目的所有文件（去除排除掉的）

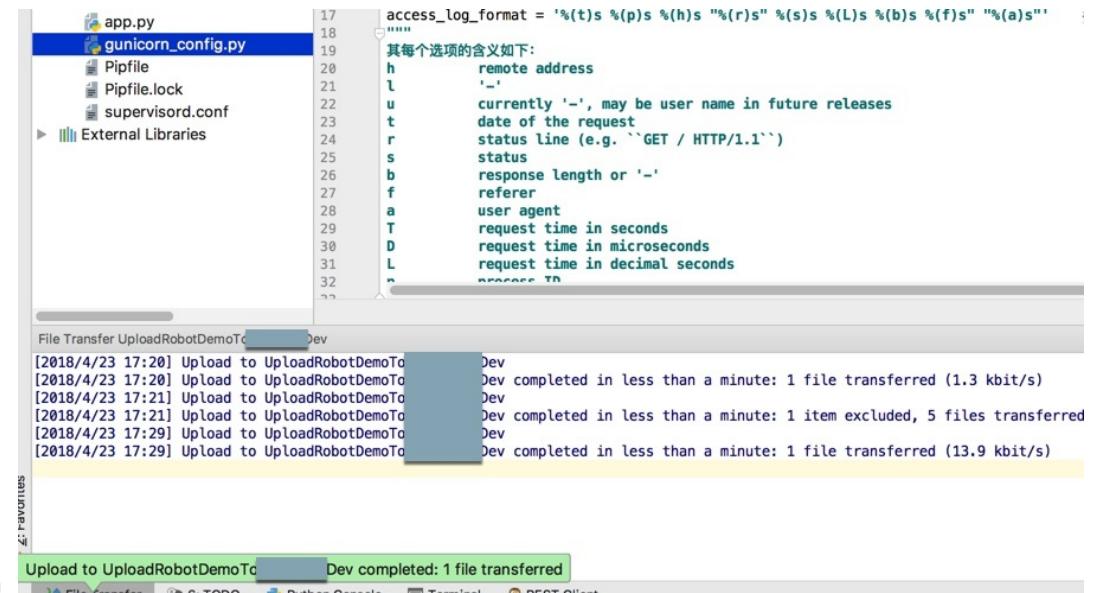


- 同步单个文件

- 可以（在修改了某单个文件后）右击该单个文件，选择 Upload to xxx，也可以直接单独上传该文件



- 上传后的提示



The screenshot shows a code editor with several files listed in the sidebar: app.py, gunicorn_config.py, Pipfile, Pipfile.lock, supervisord.conf, and External Libraries. The gunicorn_config.py file is open, and a tooltip is displayed over the log format string. The tooltip lists the meanings of each variable:

```
access_log_format = '%(t)s %(p)s %(h)s "%(r)s" %(s)s %(L)s %(b)s %(f)s" "%(a)s"\n' +\n    ....\n    其每个选项的含义如下:\n    h      remote address\n    l      '\n    u      currently '-', may be user name in future releases\n    t      date of the request\n    r      status line (e.g. ``GET / HTTP/1.1``)\n    s      status\n    b      response length or '-'\n    f      referer\n    a      user agent\n    T      request time in seconds\n    D      request time in microseconds\n    L      request time in decimal seconds\n    n      process ID
```

Below the code editor, a terminal window shows a file transfer history:

```
[2018/4/23 17:20] Upload to UploadRobotDemoTo Dev\n[2018/4/23 17:20] Upload to UploadRobotDemoTo Dev completed in less than a minute: 1 file transferred (1.3 kbit/s)\n[2018/4/23 17:21] Upload to UploadRobotDemoTo Dev\n[2018/4/23 17:21] Upload to UploadRobotDemoTo Dev completed in less than a minute: 1 item excluded, 5 files transferred\n[2018/4/23 17:29] Upload to UploadRobotDemoTo Dev\n[2018/4/23 17:29] Upload to UploadRobotDemoTo Dev completed in less than a minute: 1 file transferred (13.9 kbit/s)
```

A message bar at the bottom of the terminal window says "Upload to UploadRobotDemoTo Dev completed: 1 file transferred".

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2020-02-15
16:32:51

智能之处

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12 21:26:47

作为IDE本身

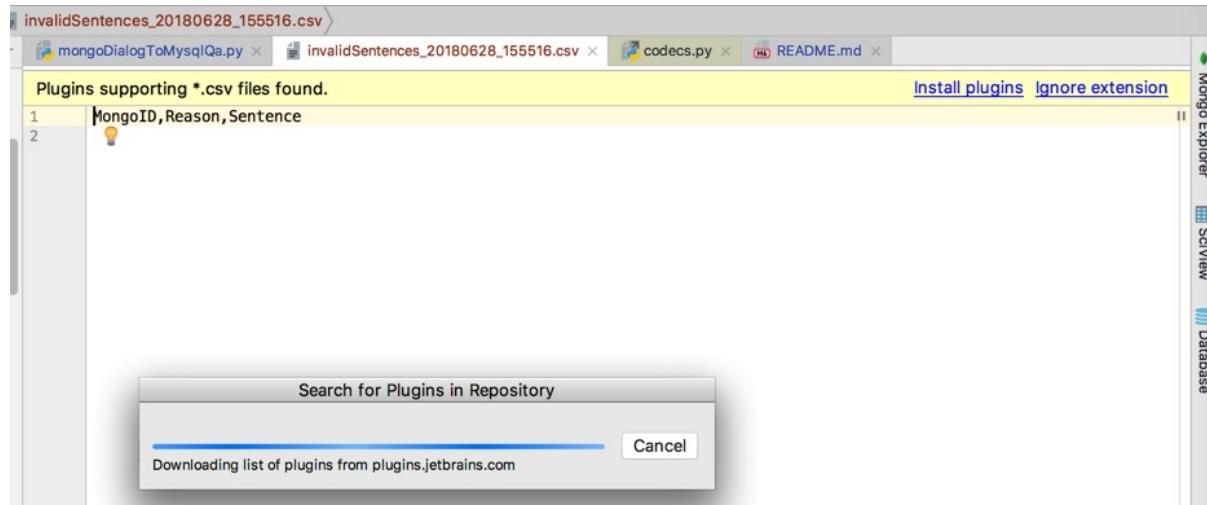
PyCharm作为一个IDE本身，功能方面也做的很好了。下面举例说明。

自动识别文件类型并提示安装相关插件

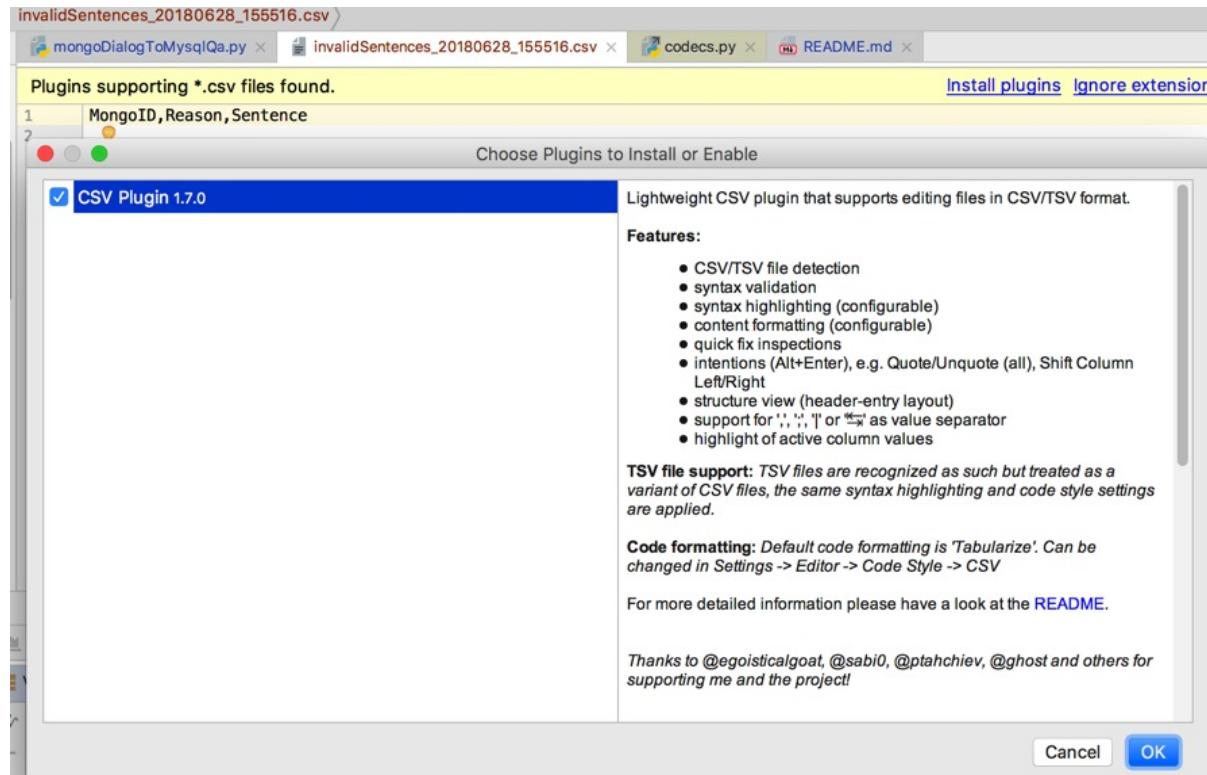
和之前总结的[VSCode](#)中一样：

当打开不认识类型的文件（应该是通过后缀名推测的）

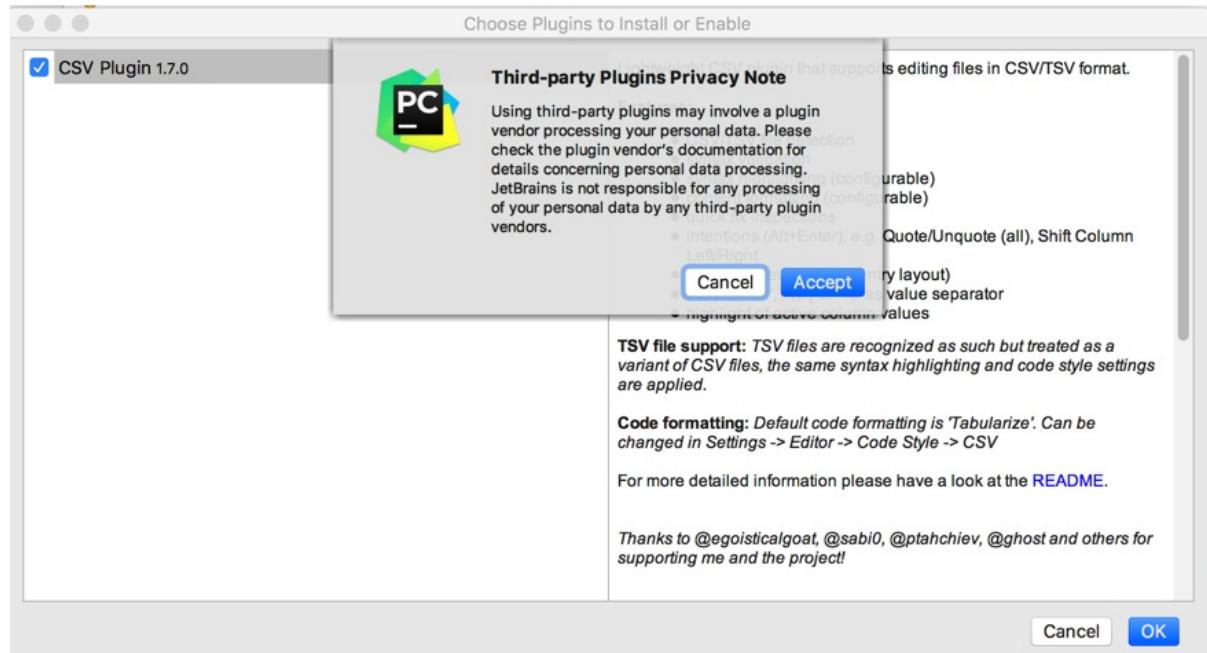
然后就会提示安装对应插件，以便于更好的显示（比如代码高亮）：



根据提示去安装 csv 文件的插件 CSV Plugin：

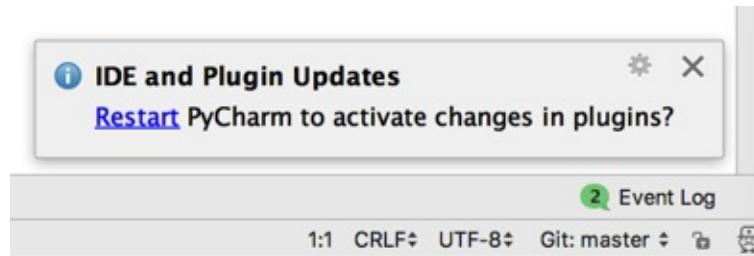


此处是第三方的插件（不是PyCharm官方的），所以会弹框提示 Third-party Plugins Privacy Note：

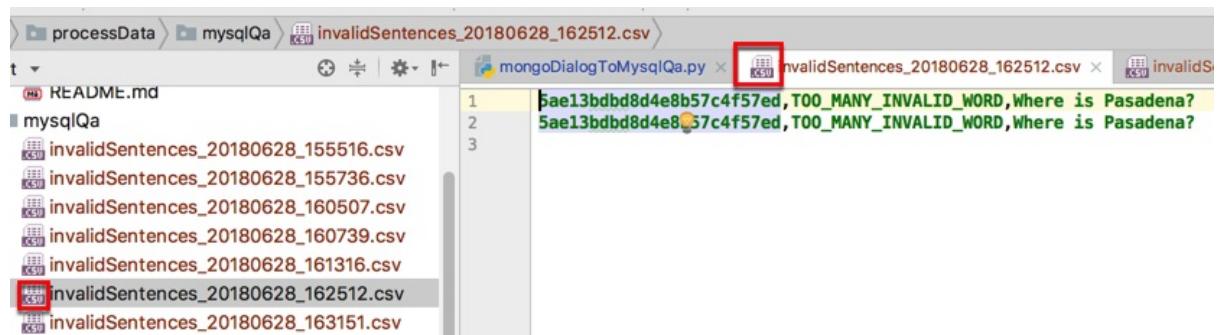


点击 Accept 即可。

插件安装完毕后会提示你重启：

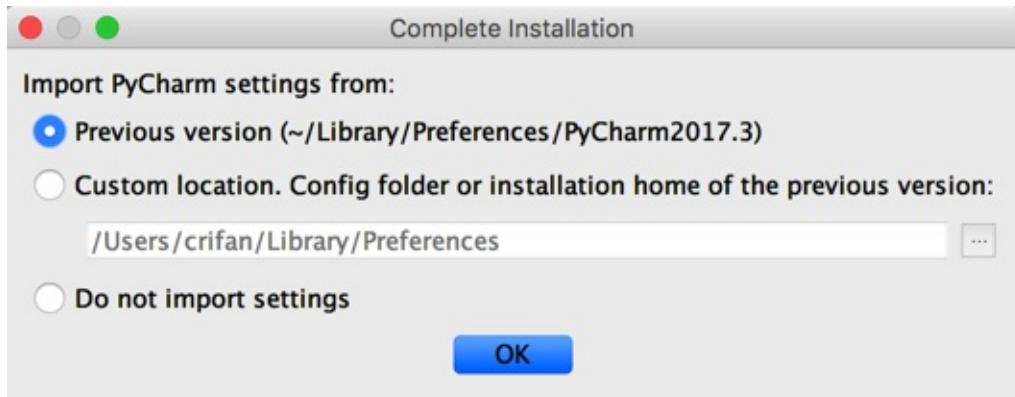


点击 Restart 重启后，项目列表中文件有了新的图标，且文件内容可以代码高亮：



支持导致之前版本的配置

版本升级后，智能监测到之前版本，并导入之前的配置：



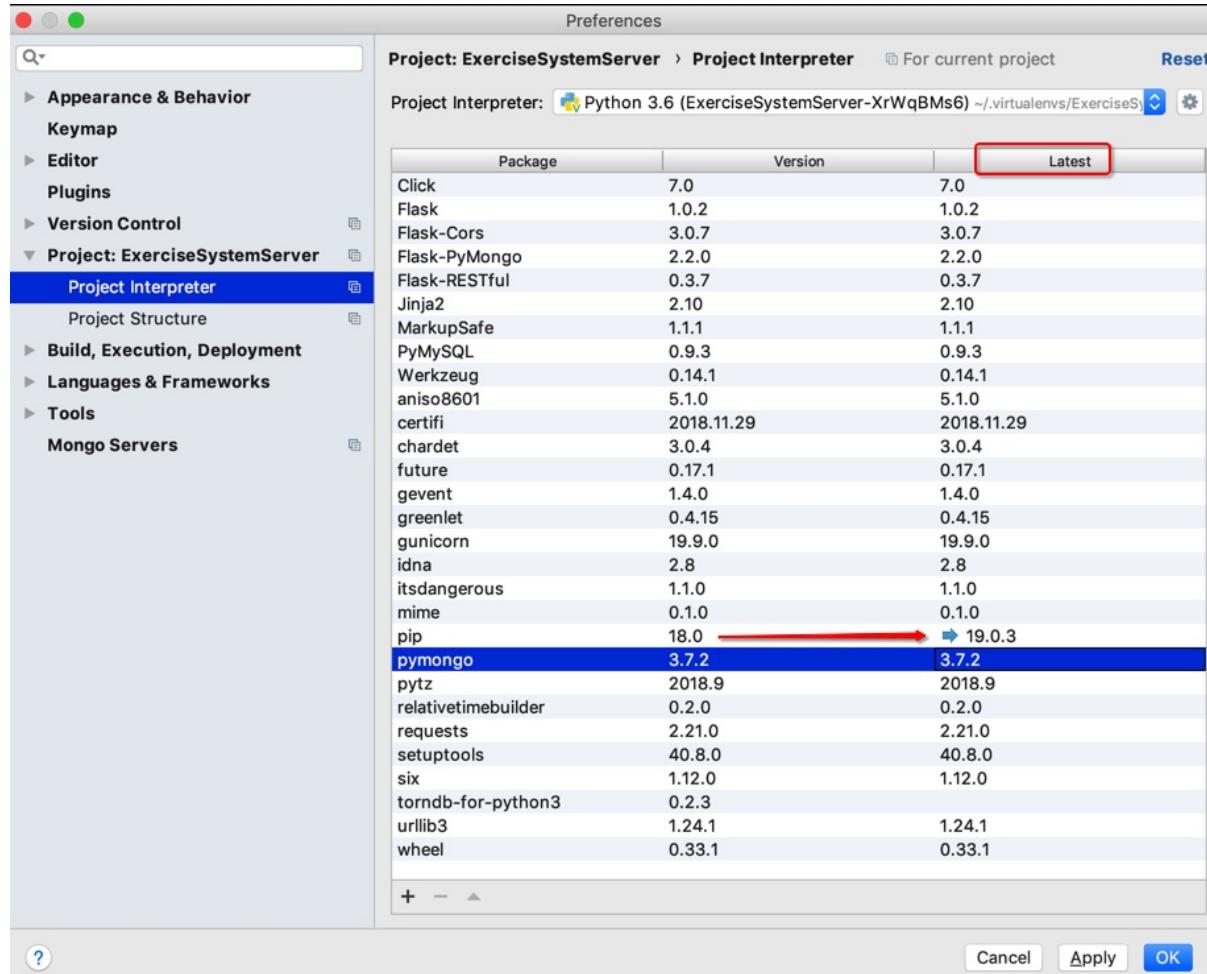
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-15 11:55:45

项目配置

作为项目的配置方面，PyCharm也有很多智能之处。

显示当前安装的库的可升级的版本

Python的虚拟环境中所用的库，除了显示当前已安装版本外，甚至还会列出最新版本是多少，甚至检测出有哪些可升级的库



crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-13
22:40:24

编辑期间

PyCharm的智能之处，在静态的代码文件编辑期间，也有很多的体现。

下面举例说明：

变量名重复

变量错误提示之 命名重复

此处拷贝上面一行后，修改了后面的值后，但是忘了及时修改前面的变量名了，然后变量名变黄色警告提示出错了：

```
Redeclared xxx defined above without usage
106      ######
107      # Exercise
108      #####
109
110      ENDPOINT_EXERCISE = "exercise"
111      ENDPOINT_UNIT = "unit"
112      ENDPOINT_UNIT = "storybook"
113
Redeclared 'ENDPOINT_UNIT' defined above without usage more... (⌘F1)
115
```

-» 然后及时改为正常的变量名，即可：

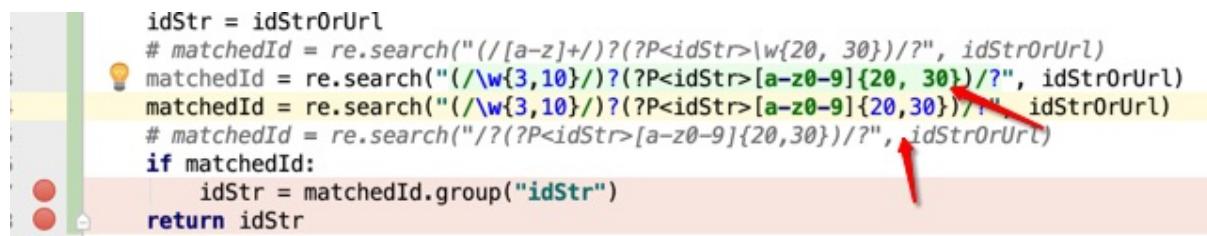
```
105      ######
106      # Exercise
107      #####
108
109
110      ENDPOINT_EXERCISE = "exercise"
111      ENDPOINT_UNIT = "unit"
112      ENDPOINT_STORYBOOK = "storybook"
113
```

-» 很是贴心，防止笔误导致变量出错。

正则表达式语法错误则显示普通字符串颜色

折腾 【已解决】 Python的正则re的search查找不到值期间

- 都能检测出Python的 re 的 {m,n} 的格式，从而显示出正确的蓝色数字
 - 而不小心写错成中间有多余空格： {m, n}，都会显示出字符串的绿色

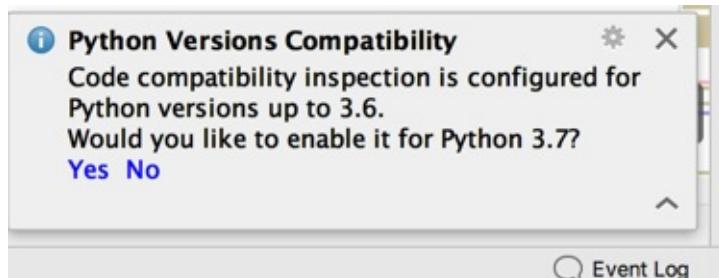


```
idStr = idStrOrUrl
# matchedId = re.search("(/[a-z]+/)?(P<idStr>\w{20, 30})/?", idStrOrUrl)
matchedId = re.search("(/\w{3,10}/)?(P<idStr>[a-zA-Z]{20, 30})/?", idStrOrUrl)
matchedId = re.search("(/\w{3,10}/)?(P<idStr>[a-zA-Z]{20,30})/?", idStrOrUrl)
# matchedId = re.search("/(P<idStr>[a-zA-Z]{20,30})/?", idStrOrUrl)
if matchedId:
    idStr = matchedId.group("idStr")
return idStr
```

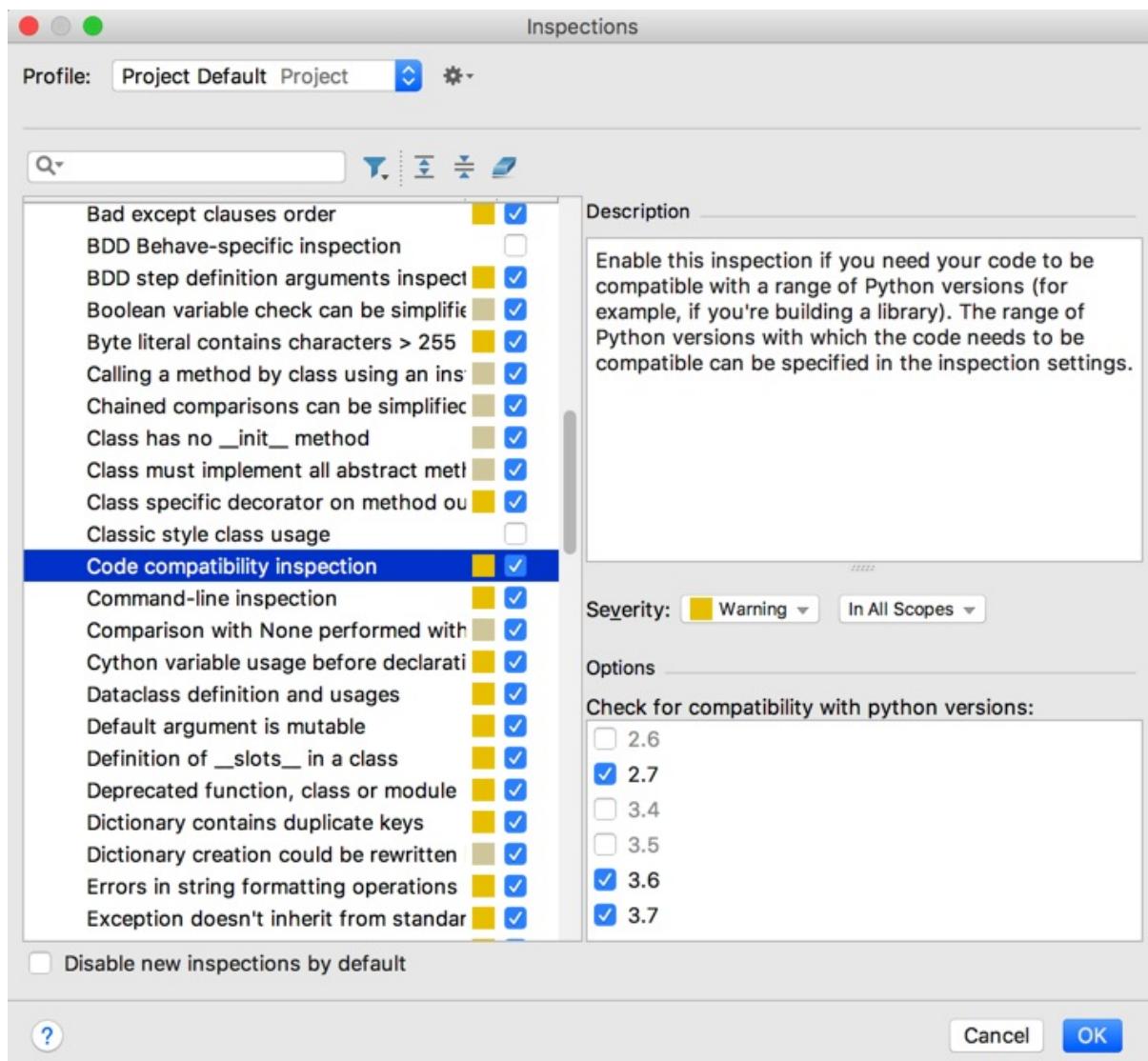
真的很智能->从颜色显示上就帮你容易识别出错误的正则的语法。

Python版本兼容性提示

之前遇到提示：



点击 Yes



变量提示： Shadows name from outer scope

之前遇到代码

```
def xxx():
    for (curIdx, eachProductDict) in enumerate(curLoopProductList):
        logging.info("[{}] eachProductDict={}", curIdx, eachProductDict)
        (processedInfoDict, toProcessInfoDict) = orderSingleMsStoreProduct(driver, gCfg, eachProductDict)
        processedOrderTime = processedInfoDict["orderNumber"]
```

出现提示：

```
Shadows name from outer scope
This inspection detects shadowing names defined in outer scopes
```

```

399     def orderSingleMsStoreProduct(driver, gCfg, productInfoDict):
400         """
401             do auto order for single ms store product
402
403             @type productInfoDict: dict
404             @type gCfg: dict
405
406             :param productInfoDict: single product info dict
407             examples:
408                 {
409                     "productId": "https://www.microsoft.com/en-us/store/d/dell-xps-13-xps-9360-laptop-pc/8q17384grz37/GV5L",
410                     "productName": "",
411                     "orderNumber": 3,
412                     "expectPrice": 699
413                 },
414                 {
415                     "productId": "https://www.microsoft.com/en-us/store/d/dell-inspiron-15-i5570-5364slv-pus-laptop/8nksc7",
416                     "productName": "Dell Inspiron 15 i5570-5364SLV-PUS Laptop",
417                     "orderNumber": 8,
418                     "expectPrice": 399
419                 }
420
421             :return:
422         """
423
424
425         # init
426         processedInfoDict = productInfoDict
427         toProcessInfoDict = productInfoDict
428         processedInfoDict["orderNumber"] = a
429
430     Shadows name 'toProcessInfoDict' from outer scope more... (⌘F1) dict["orderNumber"]
431     logging.debug("initiated processedInfoDict=%s, toProcessInfoDict=%s", processedInfoDict, toProcessInfoDict)
432

```

具体含义是：

函数内部的变量，如果和函数被调用的外部的变量一样的话，就被 PyCharm 中叫做 shadows name

这样的话，容易引发不容易觉察到的，由于函数内部和外部的变量名一致而引发的一些问题：

比如：内部函数名引用时不小心写错了时，就会导致其实调用了外部变量名，从而导致逻辑错乱。

所以解决办法是：

确保函数内部和外部的变量名不要重复，这样就不会导致可能由此导致的错误了。

后来代码做了改动：

在外部也有的一个全局变量 processedInfoDict 的情况下，把函数内部的 processedInfoDict 变量改名，比如改为 hasProcessedInfoDict：

```

processedInfoDict = {"xxx": "yyy"}

def xxx():
    for (curIdx, eachProductDict) in enumerate(curLoopProductList):
        logging.info("[{}] eachProductDict={}", curIdx, eachProductDict)
        (hasProcessedInfoDict, needToProcessInfoDict) = orderSingleMsStoreProduct(driver, gCfg, eachProductDict)
        processedOrderTime = hasProcessedInfoDict["orderNumber"]
    ...

```

就可以了消除警告了。

调试期间

PyCharm的调试功能很强大，也很智能。

调试时实时打开项目外的文件

某次调试项目，遇到import导入了当前项目之外的文件：



继续下一步调试 Step Into 的话，则可以打开，当前项目之前的文件去继续调试：



很是方便。

动态显示变量的属性的值

对于代码：

```
if not connection.isConnected:
```

isConnected 是对象变量 connection 的属性值，PyCharm都支持显示：

当鼠标移动到 isConnected 上时，可以自动检测到 connection.isConnected 是个整体，然后动态显示出其值：



还是很好用的。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15
11:50:09

git支持

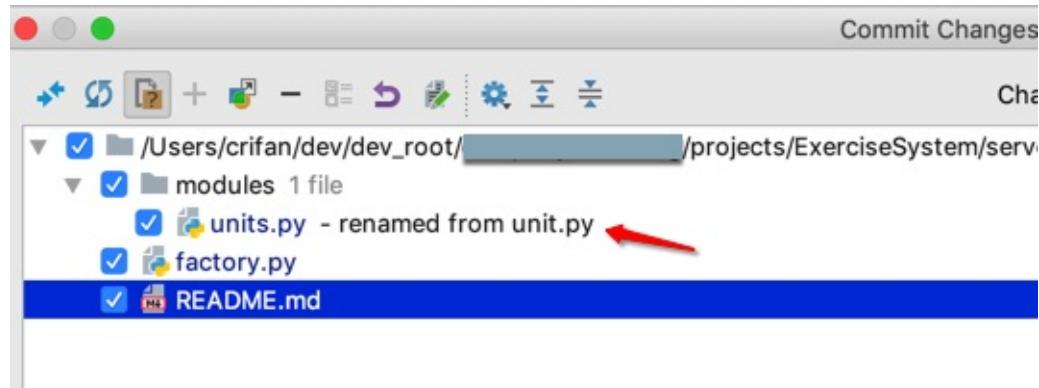
PyCharm对于代码版本管理工具：git

支持的也很好。有很多细节功能值得一提，体现了智能之处：

文件名改动提示

git提交代码时，文件名改动都可以检测出来并提示你：

某次把 `unit.py` 改为了 `units.py`，然后commit提交代码之前，就可以看到对应提示：



很是人性化。

检测出`.gitignore`排除的文件残留在git记录的项目中

当去编辑 `.gitignore`，加上 `.idea/` 后，目的是，排除和项目代码无关的PyCharm的一些配置文件

此时，PyCharm 会检测到此改动，并且提示你：

`.gitignore`中发现排除了一些文件

但是却（由于之前没排除，而加入）存在项目中

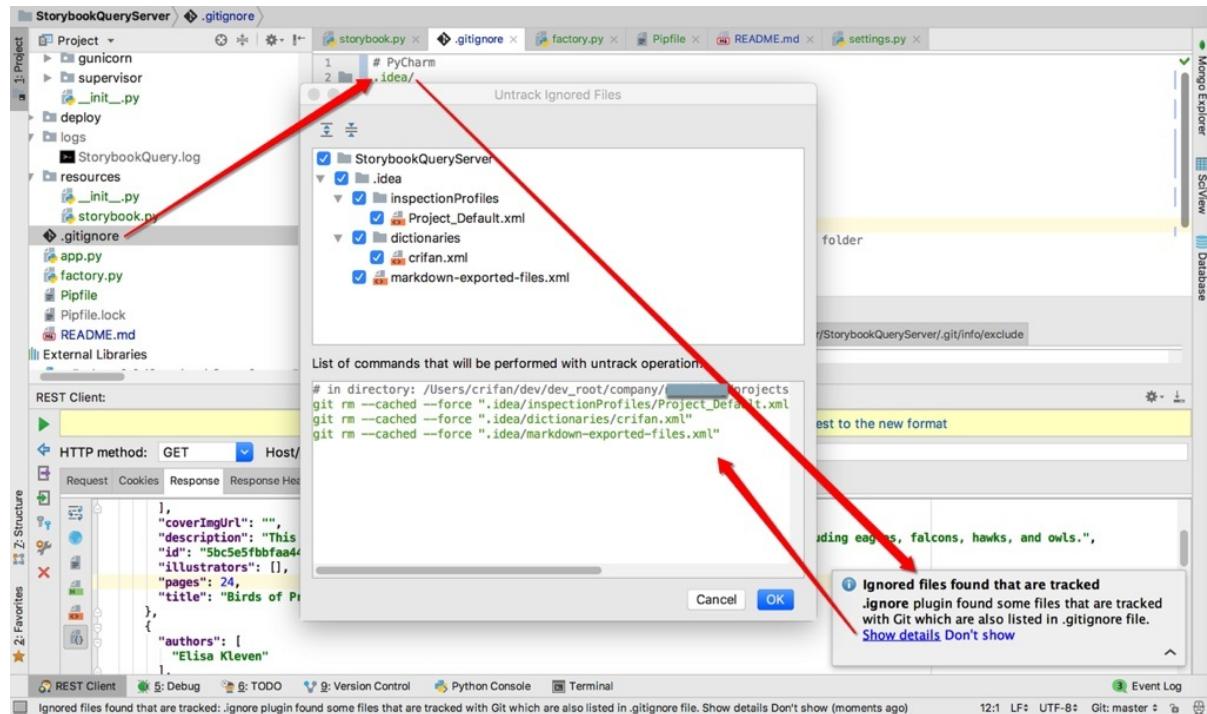
所以会问你如何操作

点击查看详情

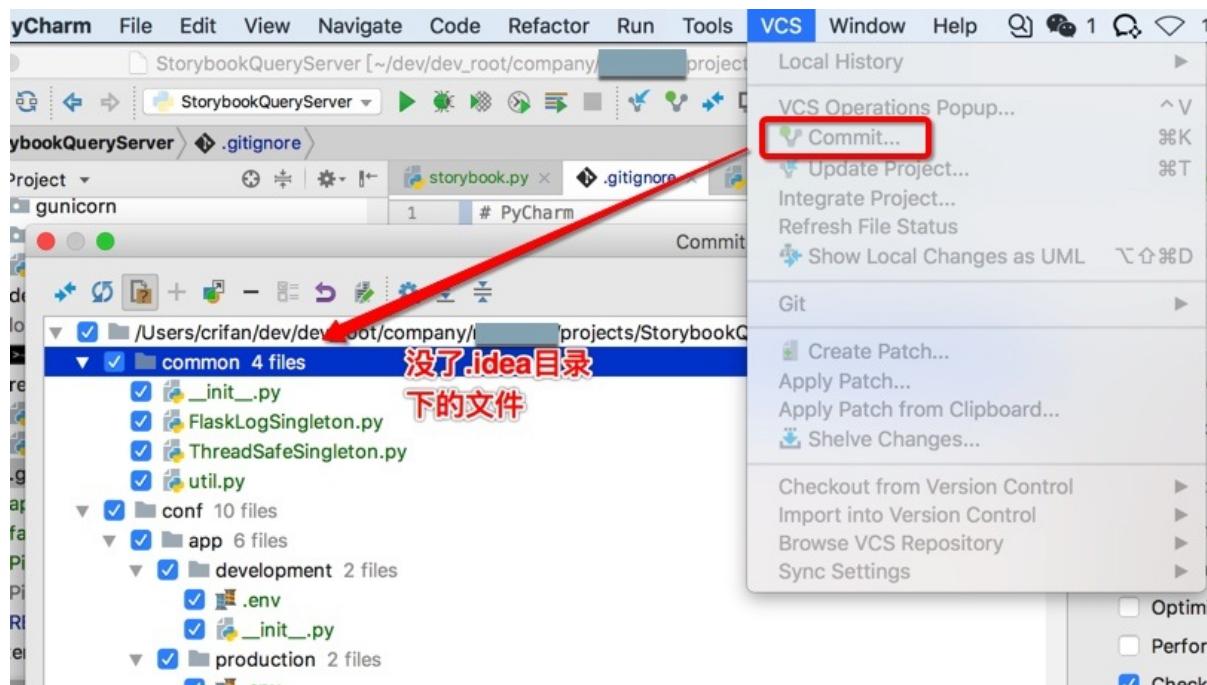
此时，会自动列出代码：

```
# in directory: /Users/crifan/dev/dev_root/xxx/StorybookQueryServer
git rm --cached --force ".idea/inspectionProfiles/Project_Default.xml"
git rm --cached --force ".idea/dictionaries/crifan.xml"
git rm --cached --force ".idea/markdown-exported-files.xml"
```

你点击OK后，即可从git本地缓存中删除这些之前误操作保存到git中的文件



然后再去commit提交时：



即可发现，已经去除了刚才还存在的 .idea 文件夹中的那些文件

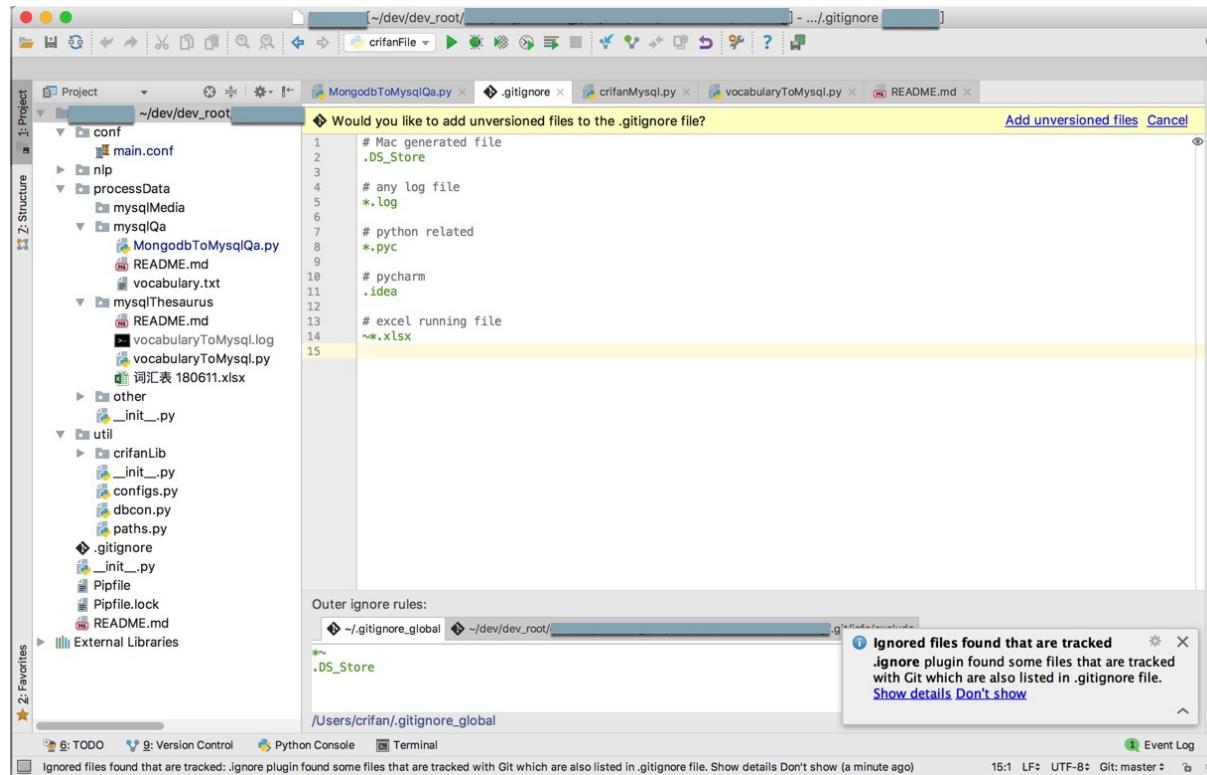
-» 由此自动帮你检测出问题，还给出问题的解决方案的IDE，才叫真的智能。

对于 .gitignore 的智能支持

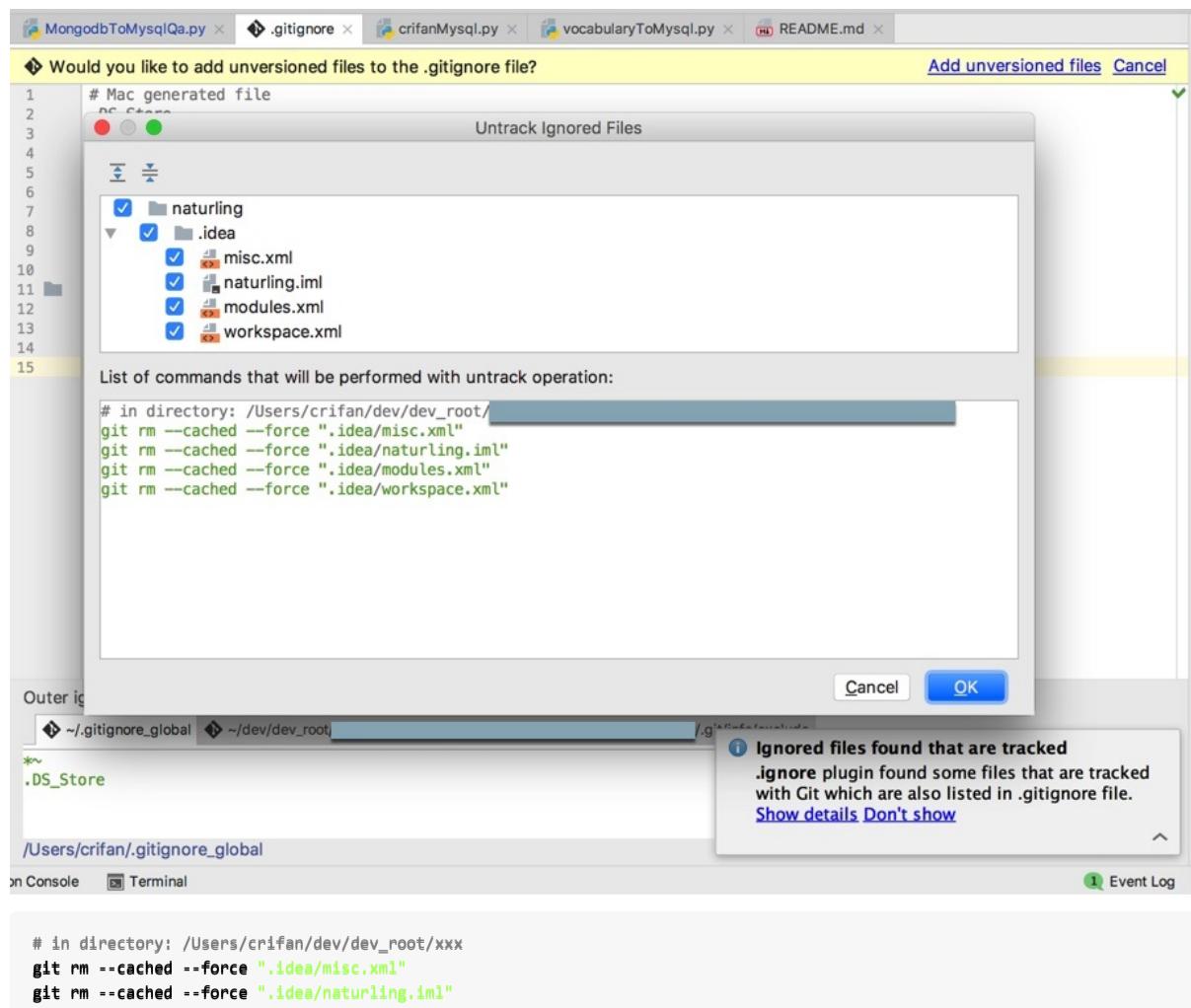
安装了 .ignore 插件后，自动检测出并提示：

```
Would you like to add unversioned files to the .gitignore file?
```

意思是：虽然有些文件之前被放到了.gitignore中，但是却也被加入到了git中了：



点击 Add unversioned files , 弹框 Untrack Ignored Files :



```
git rm --cached --force ".idea/modules.xml"
git rm --cached --force ".idea/workspace.xml"
```

意思是：之前已经把PyCharm的项目配置 .idea 目录加到git管理中了，但是此处 .gitignore 中却又排除管理了：

```
# pycharm
.idea/
```

所以有点自相矛盾了。

而此处的处理是：继续把 .idea/ 加入到项目管理中。

而把 .gitignore 中改为：

```
# pycharm
#.idea/
```

即可。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15
14:40:02

其他支持

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12 21:28:53

REST

PyCharm自带 REST Client，可用于测试RESTful API：

The screenshot shows the PyCharm IDE interface. On the left, the Project structure is visible for a project named 'robotDemo'. Inside the project, there are files like Pipfile, Pipfile.lock, testRestApi.py, and tokenize.py. The 'External Libraries' section shows Python 3.6 dependencies. In the center, the code editor displays a Python file named testRestApi.py:

```
from flask import Flask
app = Flask(__name__)
@app.route("/test")
def test():
    return "Mac local Flask within pipenv work now !"
if __name__ == "__main__":
    app.run()
```

A red arrow points from the 'External Libraries' section towards the code editor. At the bottom, the REST Client interface is shown, with the following configuration:

- HTTP method: GET
- Host/port: http://127.0.0.1:5000
- Path: /test

Below the host field, there are tabs for Request, Cookies, Response, and Response Headers.

测试效果：

The screenshot shows the PyCharm IDE interface. The top navigation bar displays the project name "robotDemo" and the file "testRestApi.py". The left sidebar shows the project structure with files like Pipfile, Pipfile.lock, testRestApi.py, and various Python library dependencies under "External Libraries". The main code editor window contains the following Python code:

```

from flask import Flask
app = Flask(__name__)
@app.route("/test")
def test():
    return "Mac local Flask within pipenv work now !"
if __name__ == "__main__":
    app.run()

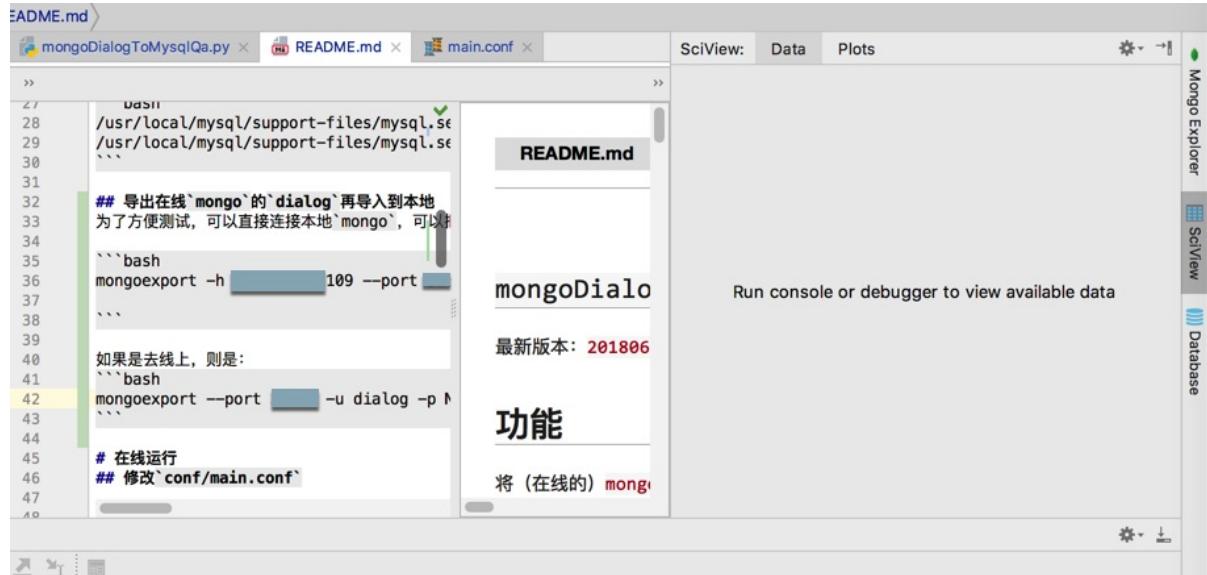
```

Below the code editor is a "REST Client" panel. It shows a successful GET request to "http://127.0.0.1:5000/test" with the response body "Mac local Flask within pipenv work now !". The bottom navigation bar includes tabs for "Python Console", "Terminal", "REST Client" (which is selected), "Debug", and "TODO".

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2020-02-14 21:45:21

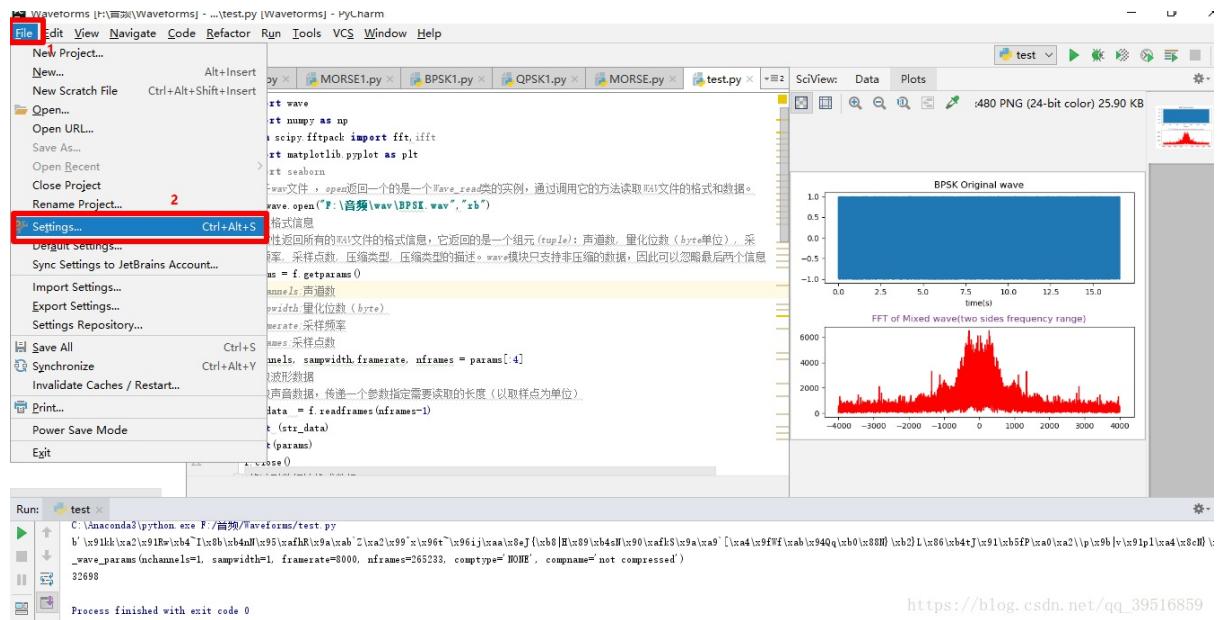
SciView

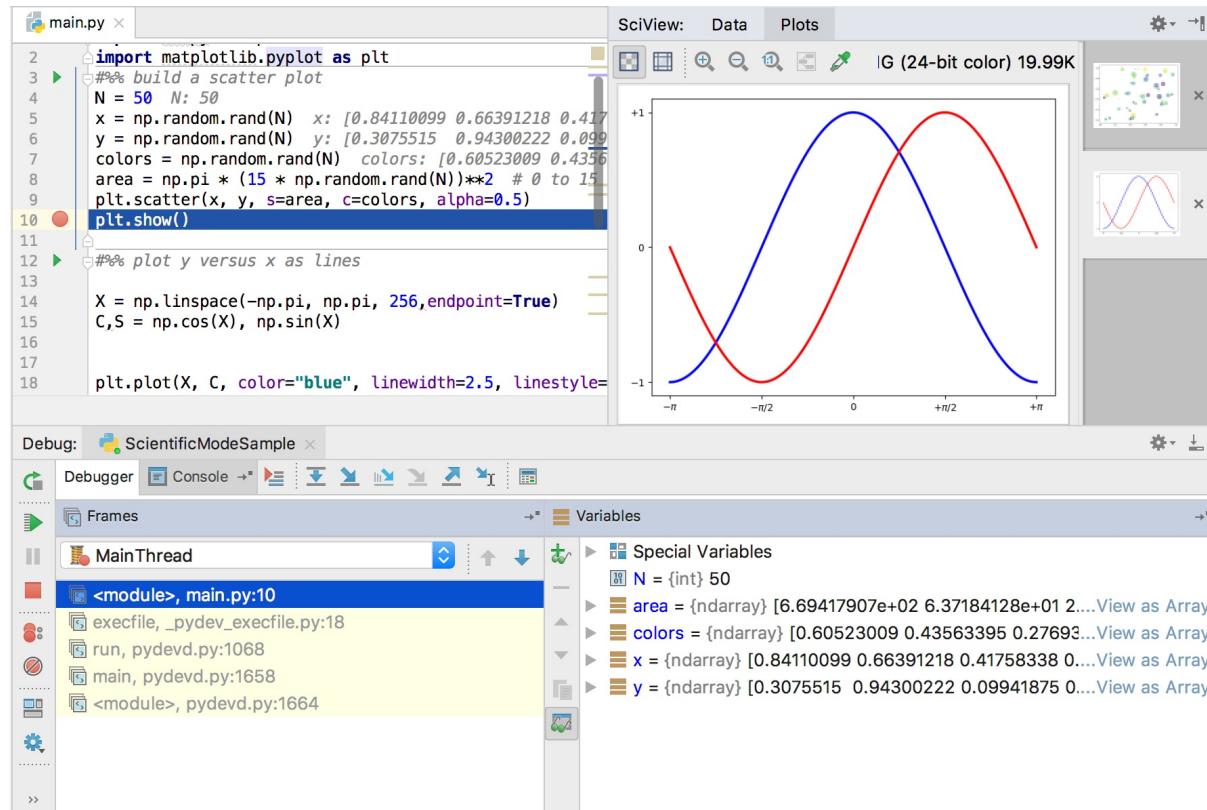
无意间发现PyCharm中（在Mongo窗口的下面）有个： SciView



查了下，得知是用来显示科学计算Python Scientific 相关的图表的

即： SciView 是 PyCharm 支持的科学计算 Python Scientific 模式中的功能之一，用来显示科学计算相关数据的图表的：





crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2020-02-15
11:37:26

MongoDB

PyCharm中有插件 `mongo4idea`，可以支持方便的操作 `MongoDB`。

详见：

- 【已解决】用PyCharm写Python的MongoDB代码并调试
- 【已解决】PyCharm中安装MongoDB的插件：[mongo4idea](#)
- 【已解决】用PyCharm的MongoDB插件连接远程MongoDB数据库
- 【已解决】PyCharm连接远程添加security的authorization的MongoDB出错：[com.mongodb.MongoCommandExceptions: Command failed with error 13](#)

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2020-02-15

18:47:12

Markdown

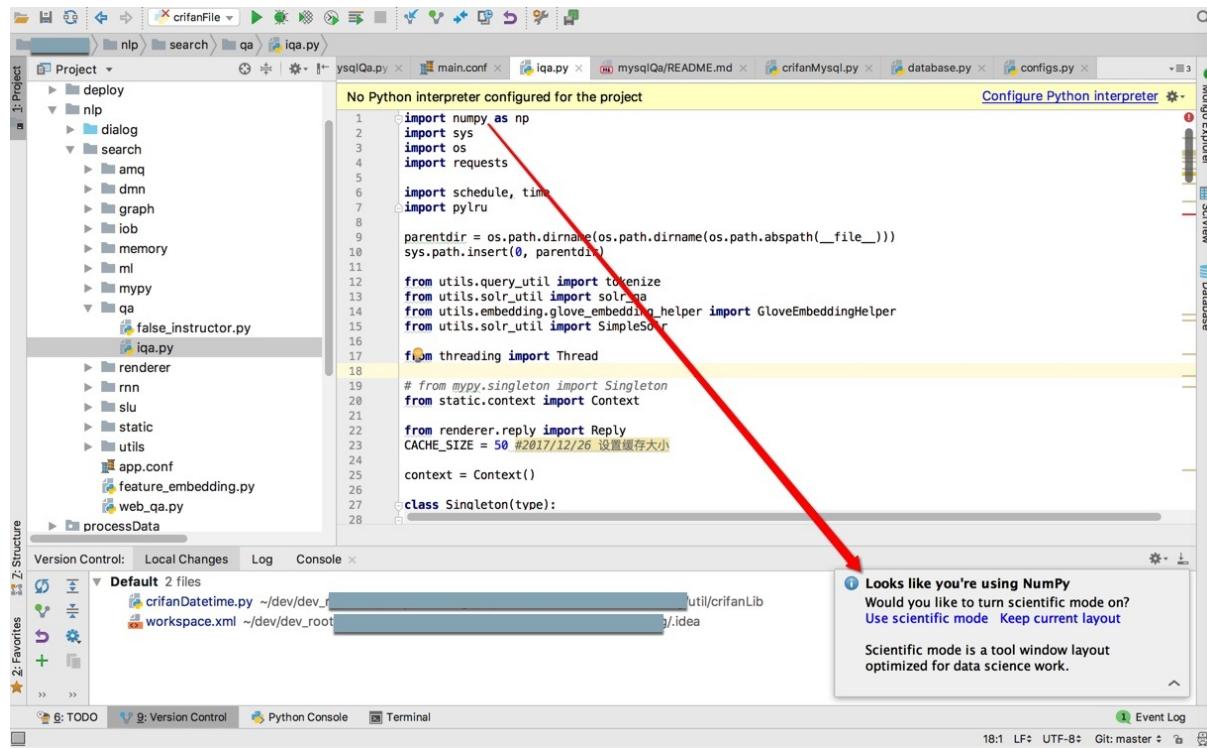
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12
21:29:44

Scientific Mode

PyCharm还针对，科学计算方面，提供了一个 Scientific Mode

之前遇到过，自动检测到代码用到了科学计算中最常用的库之一： Numpy

就提示是否要切换到 科学计算模式：



不过此处暂时不切换。

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新： 2020-02-14
22:04:48

全局设置

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-12
21:30:23

常用快捷键

此处整理 PyCharm 中常用的一些快捷键。

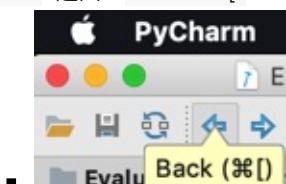
光标移动：前进/后退(返回)

- 返回=光标返回=移动到 返回到 前一次光标 鼠标 的位置
- 前进=光标前进=移动到 前进到 后一次光标 鼠标 的位置

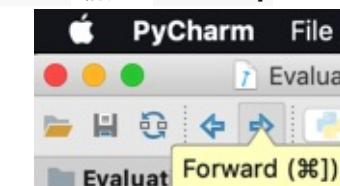
的快捷键是：

- Mac

- Back =返回: Command + [



- Forward =前进: Command +]



跳转到页首=文件开始处=页面顶部

PyCharm 文件最开始 页面开始 页面顶部

即 跳转 页首 页面开始

的快捷键：

- Win

- Control + Home / End : 跳转到文件的开始/末尾

- Mac

- Command + Fn + ←左键 / 右键→: 跳转到文件的开始/末尾

- 说明

- Mac中此处 Command 对应着Win中的 Control

- Mac中没有（直接的）Home/End键，而以实际上是：

- Fn + ←左键 = Home

- Fn + 右键→ = End

CPython加速

详见：

[【记录】PyCharm中安装Cython去加速调试](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-15
14:56:40

问题和心得

已知问题

PyCharm也有一些小问题。整理如下：

log文件的语法高亮

记得，之前默认就已支持log日志文件的语法高亮显示的。

但是后来不知道何时何故log文件失去语法高了

去折腾过

[【未解决】PyCharm中添加log文件支持语法高亮显示](#)

但是并没有解决。

抽空再去找原因。

偶尔会一直indexing重建索引

PyCharm偶尔会有个bug：

一直处于indexing，且CPU占用率很高

解决办法：

`File -> Invalidate Caches/Restart ->重启PyCharm`

一般均可解决。如果还不行，再多试试几次。

详见：[【已解决】PyCharm一直在重新indexing建立索引](#)

不支持在Markdown中粘贴图片

PyCharm中在Markdown中，不支持粘贴图片：

[【未解决】PyCharm中markdown编辑器支持粘贴图片](#)

其实这个问题，严格意义上不算是PyCharm的问题，而是属于Markdown的插件的问题。

但是与之对比，[VSCode](#)中支持Markdown中粘贴图片，很是好用：

[Paste Image · 史上最好用的编辑器：VSCode](#)

[【已解决】VSCode中插件Paste Image的粘贴图片快捷键Command+Alt+V失效 – 在路上](#)

PyCharm心得

此处整理一些使用PyCharm的经验和心得。

更改文件类型使其语法高亮

macOS中 `conf` 文件之前被识别为 `Text`，没有语法高亮：

```

[rc]
1 ;ROBOTDEMO_PROJECT_ROOT=/robotDemo
2 ;ROBOTDEMO_VIRTUALENV_ROOT=/root/.local/share/virtualenvs/robotDemo-dwdcgdaG
3
4 [program:redis]
5 directory=/robotDemo
6 command=/usr/bin/redis-server
7
8 autostart=true
9 autorestart=true
10
11 stdout_logfile=/logs/redis-%(program_name)s-stdout.log
12 stdout_logfile_maxbytes=2MB
13 stdout_logfile_backups=10
14
15 stderr_logfile=/logs/redis-%(program_name)s-stderr.log
16 stderr_logfile_maxbytes=2MB
17 stderr_logfile_backups=10
18
19 [program:robotDemo_CeleryWorker]
20 command=/root/.local/share/virtualenvs/robotDemo-dwdcgdaG/bin/celery worker -A app.celeryApp
21 directory=/robotDemo
22 autostart=true
23 autorestart=true
24
25 stdout_logfile=/logs/celery-worker-%(program_name)s-stdout.log
26 stdout_logfile_maxbytes=2MB
27 stdout_logfile_backups=10
28
29 stderr_logfile=/logs/celery-worker-%(program_name)s-stderr.log
30 stderr_logfile_maxbytes=2MB
31 stderr_logfile_backups=10
32
33
34
35 [program:robotDemo_CeleryBeat]
36 command=/root/.local/share/virtualenvs/robotDemo-dwdcgdaG/bin/celery beat -A app.celeryApp -s /root/naturling_2
37 directory=/robotDemo
38 autostart=true
39 autorestart=true
40
41 stdout_logfile=/logs/celery-beat-%(program_name)s-stdout.log
42

```

The screenshot shows the PyCharm IDE with the supervisord_server.conf file open. The code is displayed in a monospaced font. Several lines of code are highlighted in blue, which typically indicates syntax errors or unsupported features in the current file type configuration. The project structure on the left shows files like ai, debug, logs, runtime, tmp, and various Python scripts and configuration files.

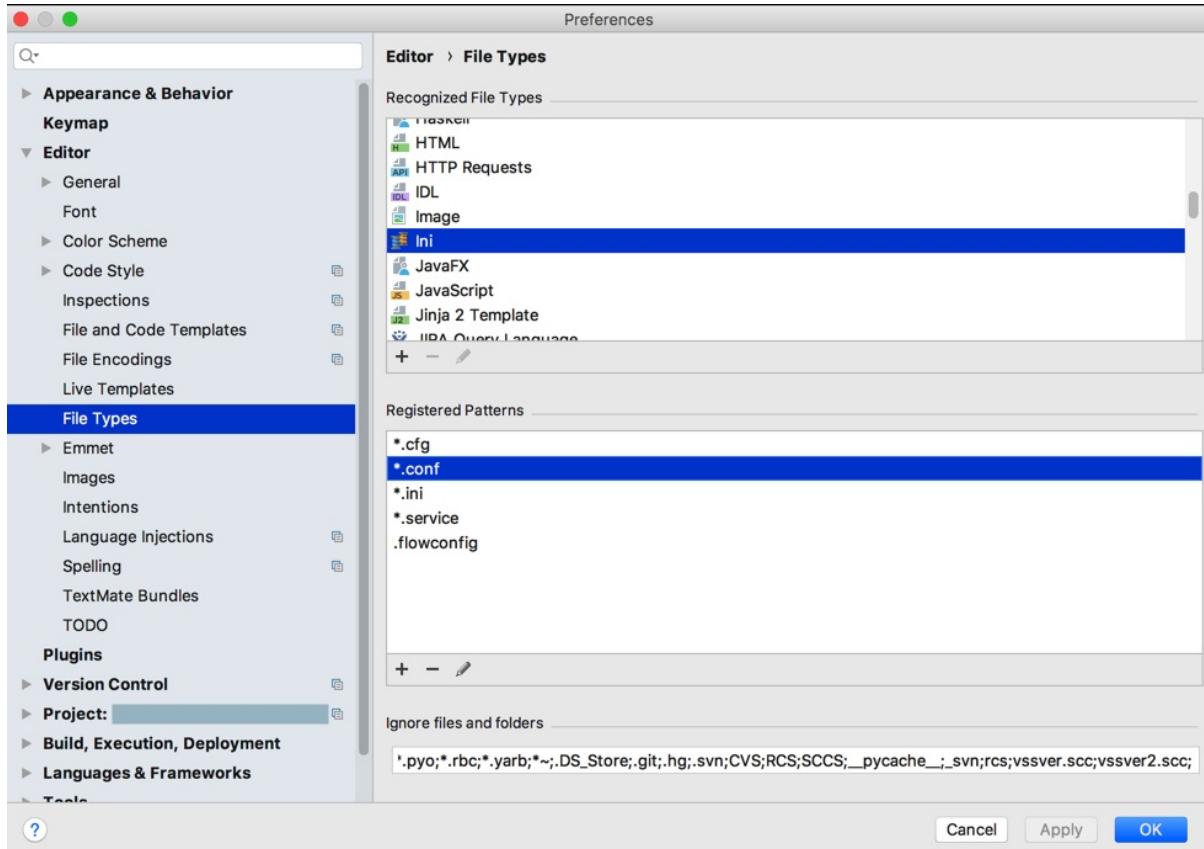
想要改为 ini 格式，以便于文件内容可以被语法高亮。

具体步骤：

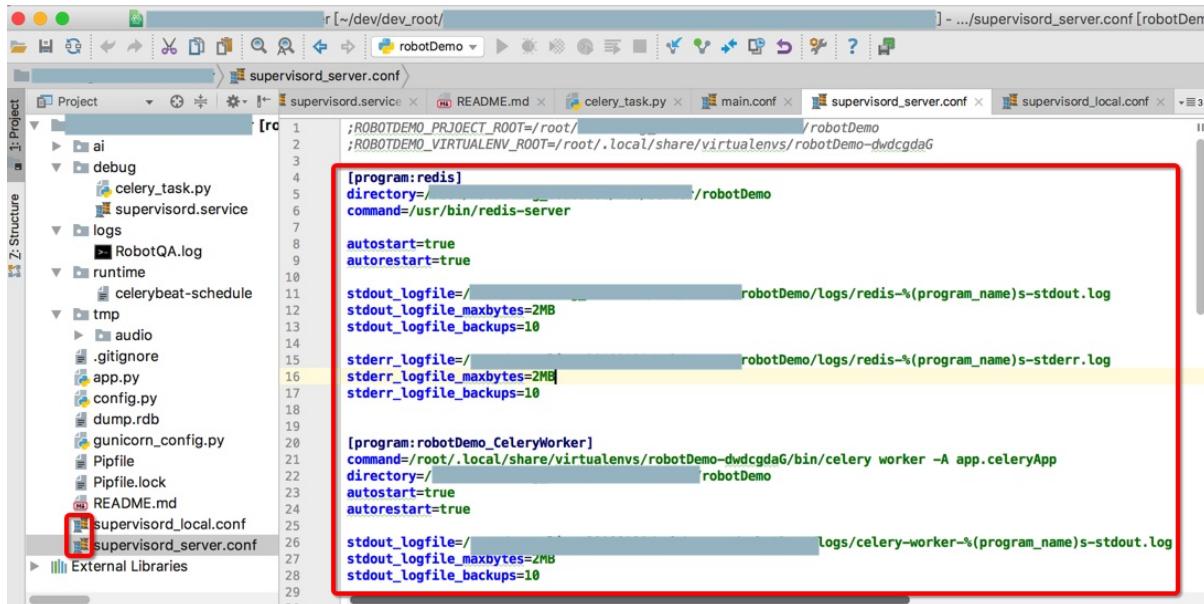
PyCharm -> Preferences -> Editor -> File Types for macOS

然后删除 Text 中的 *.conf

再给 ini 格式加上 *.conf



即可使得此处的 conf 文件，语法高亮显示了：



crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新：2020-02-15

18:07:04

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-15
11:59:44

参考资料

- 【已解决】 Mac中的PyCharm如何移动到文件最开始和最末尾 – 在路上
- 【已解决】 Mac中PyCharm中选用Python3结果：ModuleNotFoundError: No module named selenium
- 【已解决】 Mac中PyCharm中找不到已安装的Scrapy库: No module named scrapy
- 【已解决】 Python的正则re.search查找不到值
- 【未解决】 PyCharm中添加log文件支持语法高亮显示
- 【心得】 PyCharm正确设置Python解析器当前项目才能检测识别导入的库
- 【已解决】 PyCharm中如何调试pipenv的虚拟环境中的python3的文件
- 【已解决】 PyCharm中调试pipenv虚拟环境中文件出错：FileNotFoundException Errno 2 No such file or directory
pipenv run python py
- 【已解决】 PyCharm中创建logger的RotatingFileHandler出错：FileNotFoundException Errno 2 No such file or directory
- 【已解决】 Mac中PyCharm中去加断点实时调试scrapy的项目
- PyCharm 如何设置SciView工具窗口 - CSDN博客
- pycharm中显示额外的“figure”窗口 - CSDN博客
- 新版pycharm中，取消matplotlib默认输出到sciview - CSDN博客
- PyCharm: 2017.3版即将新增科学计算模式，预览版现在可以下载使用 - lemonbit - 博客园
- In pycharm, the chart displayed by matplotlib.pyplot.show () is exported to sciview and how to play it? - Code Blog
Bt
- Scientific Mode Tutorial - Help | PyCharm
- Scientific Mode - Help | PyCharm
- SciView - Help | PyCharm
- Python环境搭建及IDE选择 - 简书
- Download PyCharm: Python IDE for Professional Developers by JetBrains
- Professional vs. Community - Compare Editions | PyCharm
- PyCharm Community Edition and Professional Edition Explained: Licenses and More | PyCharm Blog
- Differences between Pycharm community and professional edition?: Python
- Pycharm的教育版和社区版有什么区别？ - 知乎
- 使用PyCharm进行Python远程调试
- Pycharm的远程代码编辑 - kiwik's blog
- 【已解决】 PyCharm一直在重新indexing建立索引
- 【已解决】 PyCharm中重新设置文件解析语法格式
- File Types – Help | PyCharm
- Register New File Type Association Dialog – Help | PyCharm
- 【记录】 PyCharm中安装Cython去加速调试
- 【已解决】 用PyCharm同步代码到服务器以实现代码部署
- 【未解决】 PyCharm中markdown编辑器支持粘贴图片
- 【已解决】 PyCharm中Python代码提示：Shadows name from outer scope
- 【已解决】 用PyCharm写Python的MongoDB代码并调试
- 【已解决】 PyCharm中安装MongoDB的插件：mongo4idea
- 【已解决】 用PyCharm的MongoDB插件连接远程MongoDB数据库
- 【已解决】 PyCharm连接远程添加security的authorization的MongoDB出错：
com.mongodb.MongoCommandException: Command failed with error 13
- 【已解决】 Mac中PyCharm中Python代码调试报错：SyntaxError Non-ASCII character \xe7 in file on line but no encoding
-

22:57:07