屏幕截图： shift+print(打印键)

**Pycharm 配置桌面快捷方式：**

执行：sudo gedit /usr/share/applications/Pycharm.desktop，创建.desktop配置文件

在该配置文件中输入以下内容：

[Desktop Entry]

Type = Application

Name = Pycharm

GenericName = Pycharm3

Comment = Pycharm3:The Python IDE

Exec = sh /usr/local/pycharm/pycharm-2017.3.4/bin/pycharm.sh

Icon = /usr/local/pycharm/pycharm-2017.3.4/bin/pycharm.png

Terminal = pycharm

StartupNotify = true

Categories = Development;Pycharm

保存后退出即可。

**Pycharm**退出Setting等相关子界面，按ESC健；

Anaconda中Python版本切换命令：

创建3.5版本：conda create -n py35 python=3.5 anaconda

激活3.5版本：source activate py35

注销3.5版本：source deactivate

指定版本创建虚拟环境：

创建3.5版本：virtualenv -p /usr/local/anaconda3/envs/py35/bin/python3.5 venv

激活3.5版本：source venv/bin/activate

注销3.5版本:deactivate

github远程配置:

1、本地/home/260137/PycharmProjects/flasky创建flasky仓库；

2、github远程创建空仓库flask,不带README文件；

3、终端执行

ssh-keygen -t rsa -C "[sishuiyibai@163.com](mailto:sishuiyibai@163.com)"

创建ssh密钥文件：id\_rsa. 和id\_rsa.pub.

4、github远程setting/SSH keys&GPG keys中配置id\_rsa.pub密钥文件

5、终端执行

git remote add origin [git@github.com](mailto:git@github.com):sishuiyibai/flasky.git

git push -u origin master

将本地仓库推送至github远程flasky仓库中

Anaconda虚拟环境管理：

### **列出所有的环境：**conda info --envs

删除环境（不要乱删啊啊啊）:conda remove -n py35 –all

pycharm安装：

tar -xvzf pycharm-professional-2017.1.2.tar.gz -C /usr/local/pycharm/

pycharm service license:https://idea.ouyanglol.com/

版本控制：

创建新项目文件夹myflaskapp,添加项目文件

执行以下等命令：

git init；

git config –global user.name “sishuiyibai”

git config –global user.email “[sishuiyibai@163.com](mailto:sishuiyibai@163.com)”

Alt+9打开版本控制窗口，切换到local change栏，蓝色：与远程仓库同步后本地修改的文件，绿色：新增但还没有commit的文件，

Unversioned Files ：添加入该项目文件夹中的文件，但还没有被git跟踪；

当被跟踪的文件里面有不想跟踪的文件时，使用命令git rm删除文件。如：

git rm --cached readme1.txt 删除readme1.txt的跟踪，并保留在本地。

git rm --f readme1.txt 删除readme1.txt的跟踪，并且删除本地文件。

然后git commit即可。但是git status查看状态时还是会列出来。

创建并切换至新分支branch1：git checkout -b branch1;

创建分支：git branch branch1;切换分支：git checkout branch1

查看分支情况：git branch

合并分支：git merge branch1

合并完后删除branch1分支：git branch -d branch1

### 1. 修改错误的提交信息（commit message）

提交信息很长时间内会一直保留在你的代码库（code base）中，所以你肯定希望通过这个信息正确地了解代码修改情况。 下面这个命令可以让你编辑最近一次的提交信息，但是你必须确保没有对当前的代码库（working copy）做修改，否则这些修改也会随之一起提交。

$ git commit --amend -m ”YOUR-NEW-COMMIT-MESSAGE”

假如你已经将代码提交（git commit）推送（git push）到了远程分支，那么你需要通过下面的命令强制推送这次的代码提交。

$ git push <remote> <branch> --force

### 2. 提交之前撤销git add

如果你往暂存区（staging area）中加入了一些错误的文件，但是还没有提交代码。你可以使用一条简单的命令就可以撤销。如果只需要移除一个文件，那么请输入：

$ git reset <文件名>

或者如果你想从暂存区移除所有没有提交的修改：

$ git reset

### 3. 撤销最近一次代码提交

有时候你可能会不小心提交了错误的文件或一开始就遗漏了某些东西。下面这三步操作可以帮助你解决这个问题。

$ git reset --soft HEAD~1

# 对工作文件进行必要的更改

$ git add -A .

$ git commit -c ORIG\_HEAD

你执行第一个命令时，Git会将HEAD指针（pointer）后移到此前的一次提交，之后你才能移动文件或作必要的修改。

然后你就可以添加所有的修改，而且当你执行最后的命令时，Git会打开你的默认文本编辑器，其中会包含上一次提交时的信息。如果愿意的话，你可以修改提交信息，或者你也可以在最后的命令中使用-C而不是-c，来跳过这一步。

### 4. Git仓库撤销至前一次提交时的状态

“撤销”（revert）在许多情况下是非常有必要的——尤其是你把代码搞的一团糟的情况下。最常见的情况是，你想回到之前代码版本，检查下那个时候的代码库，然后再回到现在状态。这可以通过下面的命令实现：

$ git checkout <SHA>

“”是你想查看的提交拥有的哈希值（Hash Code）中前8至10个字符。 这个命令会使<HEAD>指针脱离（detach），可以让你在不检出（check out）任何分支的情况下查看代码——脱离HEAD并不像听上去那么可怕。如果你想在这种情况下提交修改，你可以通过创建新的分支来实现：

$ git checkout -b <SHA>

要想回到当前的工作进度，只需要检出（check out）你之前所在的分支即可。

5. 删除本地和远程Git分支

删除本地分支：

$ git branch --delete --force <branchName>

#### 或者使用选项-D作为简写：

$ git branch -D

删除远程分支：

$ git push origin --delete <branchName>

.gitignore文件生效：

先清除本地缓存，然后在PUSH，操作如下：

git rm -r --cached . //删除本地缓存

git add . // 添加要提交的文件

git commit -m 'update .gitignore' // 更新本地缓存