# 虚拟开发环境搭建

## 一、virtualenv

### 1.安装pip

pip python专用的包管理工具

```
sudo apt install python3-pip
```

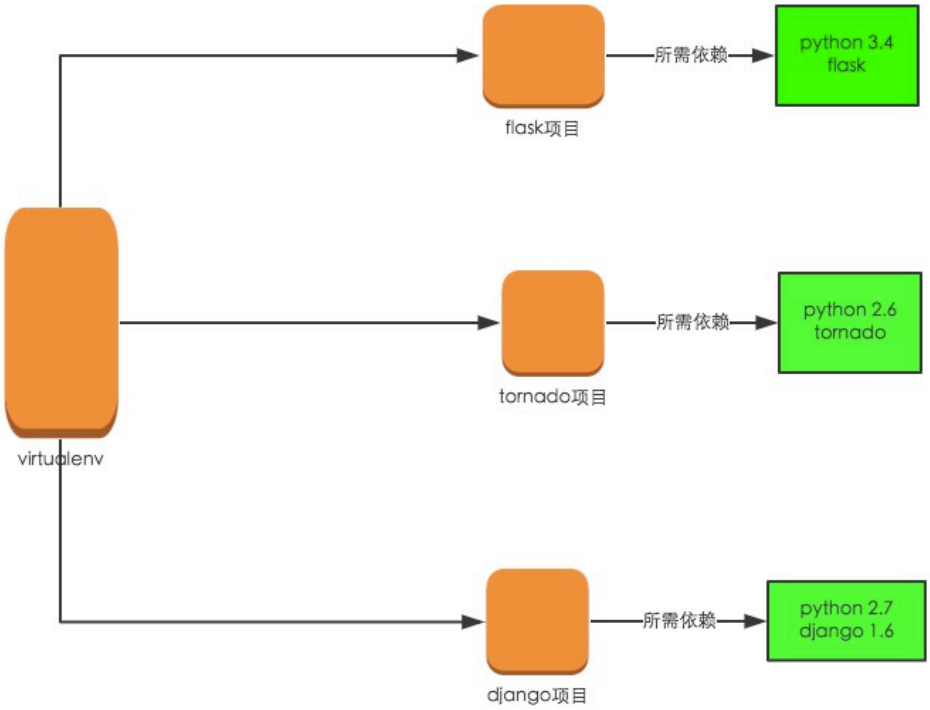
- 使用pip安装包

  ```
  pip install  包名
  pip install  包名 -i 下载源(https://pypi.douban.com/simple)
  ```

- 使用pip卸载包 pip unistall 包名
- pip freeze 列出我们自己安装的所有依赖包
- pip list 列出我们所有的依赖包

### 2.安装virtualenv

在python开发中，我们可能会遇到一种情况，就是当前的项目依赖的是某一个版本，但是另一个项目依赖的是另一个版本，这样就会造成依赖冲突，而virtualenv就是解决这种情况的，virtualenv通过创建一个虚拟化的python运行环境，将我们所需的依赖安装进去的，不同项目之间相互不干扰，如下所示。




```
sudo pip install virtualenv -y
```

virtualenv命令格式：

```
virtualenv [OPTIONS] DEST_DIR
OPTIONS常用参数：
--no-site-packages    独立的运行环境
-p 指定虚拟开发环境的python解释器
```

- 创建虚拟开发环境

```
virtualenv 虚拟环境名

# 指定python解释器
virtualenv -p python解释器路径  虚拟环境名

#实例
virtualenv --no-site-packages --
python=/usr/local/python36/bin/python  venv
```

- 激活虚拟开发环境

```
source 虚拟环境名/bin/activate
```

- 退出虚拟开发环境

```
deactivate
```

## 3.安装virtualenvwrapper

virtualenv 的一个最大的缺点就是：

每次开启虚拟环境之前要去虚拟环境所在目录下的 bin 目录下 source 一下 activate，这就需要我们记住每个虚拟环境所在的目录。

- 一种可行的解决方案是，将所有的虚拟环境目录全都集中起来，例如/opt/all_venv/，并且针对不同的目录做不同的事。
- 使用**virtualenvwrapper**管理你的虚拟环境（virtualenv），其实他就是统一管理虚拟环境的目录，并且省去了source的步骤。

### 3.1 安装

```
pip install virtualenvwrapper
```

### 3.2 配置虚拟环境

- 创建虚拟环境管理目录

```
mkdir ~/.virtualenvs
```

- 修改虚拟环境变量

```
#打开文件
vi ~/.bashrc
#在文件内容末尾添加如下内容：
export WORKON_HOME=$HOME/.virtualenvs  # 所有虚拟环境存储的目录
export VIRTUALENVWRAPPER_PYTHON=/usr/local/python36/bin/python
  #指定python解释器
source /usr/local/bin/virtualenvwrapper.sh  #执行virtualenvwrapper
安装脚本
#让配置文件其生效
source ~/.bashrc
```

### 3.3 基本使用virtualenvwrapper

```
#创建一个虚拟环境：
$ mkvirtualenv my_django115
这会在 ~/.virtualenvs 中创建 my_django115 文件夹。

#激活虚拟环境my_django115
$ workon my_django115


再创建一个新的虚拟环境
$ mkvirtualenv my_django2

virtualenvwrapper 提供环境名字的tab补全功能。


#退出虚拟环境
deactivate

#删除虚拟环境，需要先退出虚拟环境
rmvirtualenv my_django115
```

# 二、pyenv

pyenv是一个Python版本管理工具，它能够进行全局的Python版本切换，也可以为单个项目提供对应的Python版本。使用pyenv以后，可以在服务器上安装多个不同的Python版本，也可以安装不同的Python实现。不同Python版本之间的切换也非常简单。pyenv官方地址[https://github.com/pyenv/pyenv-installer](https://github.com/pyenv/pyenv-installer)

- 安装pyenv

```
#1 安装pyenv，在命令行下键入：
$ curl -L https://github.com/pyenv/pyenv-
installer/raw/master/bin/pyenv-installer | bash

#默认安装到当前用户的工作目录下的.pyenv，我们可以查看一下
$ cd ~
$ ls -la

#如果安装过git，也可以使用以下安装方式安装
```

```
 $ git clone https://github.com/pyenv/pyenv.git ~/.pyenv

#2 将安装路径写入~/.bashrc
将以下三条语句写入 ~/.bashrc
export PATH="/home/zhu/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"

$ source  ~/.bashrc  #配置立刻生效

$ echo $PATH
#如果路径第一个结尾是shims则表示成功
/home/python/.pyenv/plugins/pyenv-
virtualenv/shims:/home/python/.pyenv/shims:/home/python/.pyenv/bin:/h
ome/python/bin:/home/python/.local/bin:/usr/local/sbin:/usr/local/bin
:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin


#也可以将上面三条命令写入到~/.barsh_profile，然后重启一下shell


#3 更新一下
$ pyenv update
```

- 使用pyenv

```
#1.查看pyenv当前支持哪些Python版本
python@ubuntu:~$ pyenv install --list
Available versions:
  2.1.3
  2.2.3
  2.3.7
  ...

#2.列出pyenv中所有可用的python版本
python@ubuntu:~$ pyenv versions
  system
  3.5.4
* 3.6.4 (set by /home/python/.pyenv/version)  # *表示当前使用的
3.6.4版本

#3.选择指定的python版本
```

```
python@ubuntu:~$ pyenv global 3.5.4  #设置指定的版本
python@ubuntu:~$ python
Python 3.5.4 (default, Mar 29 2018, 11:02:03)  #已经切换到了3.5.4
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

切换Python版本以后，与版本相关的依赖也会一起切换。因此，我们不用担心不同的版本在系统中是否会相互干扰。

```
#4. 删除指定python版本
python@ubuntu:~$ pyenv uninstall 3.5.4
pyenv: remove /home/python/.pyenv/versions/3.5.4? y
python@ubuntu:~$ pyenv versions
  system
* 3.6.4 (set by /home/python/.pyenv/version)
```

- 安装python

```
#1.在安装python之前，我们必须要安装python所需要的依赖包，这个必须要安装，安装会失败的：
$ sudo apt install libc6-dev gcc
$ sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm

#2. 安装python
$ pyenv install 3.6.4 -v  #-v 以日志模式显示安装过程
#因为pyenv会自动到github上下载，速度超慢，所以一般会选择使用curl或者wget下载到~/.pyenv/cache下，然后再用pyenv安装，下面是可选的安装模式
$ cd ~/.pyenv
$ sudo mkdir cache
$ wget -c http://mirrors.sohu.com/python/3.6.4/Python-3.6.4.tar.xz -P  ~/.pyenv/cache/
$ pyenv install 3.6.4 -v

#3.更新pyenv数据库
 $ pyenv rehash

#4.列出所安装的python版本
  $ pyenv versions

#5.切换python版本
```

```
$ pyenv  global 3.6.4

#6.验证版本
 $ python
```

- 注意：
  - 使用pyenv管理python，必须是用pyenv安装的python才行，系统以前有的，需要重新用pyenv安装
  - 使用pip安装第三方模块时会安装到~/.pyenv/versions/xxx下，不会和系统模块发生冲突；
  - 使用pip安装模块后，可能需要执行pyenv rehash更新数据库。

## virtualenv

virtualenv本身是一个独立的项目，用以隔离不同项目的工作环境。例如，项目A和项目B都是使用Python 2.7.13，但是，项目A需要使用Flask 0.8版本，项目B需要使用Flask 0.9版本。我们只要组合pyenv和virtualenv这两个工具，就能够构造Python和第三方库的任意版本组合，拥有了很好的灵活性，也避免了项目之间的相互干扰。

virtualenv本身是一个独立的工具，用户可以不使用pyenv单独使用virtualenv。但是，如果你使用了pyenv，就需要安装pyenv-virtualenv插件而不是virtualenv软件来使用virtualenv的功能。

项目主页：https://github.com/yyuu/pyenv-virtualenv

- 安装virtualenv（可选）

  如果是python3以上，安装完python就已经安装了virtualenv，就不用安装了

```
#安装
git clone https://github.com/yyuu/pyenv-virtualenv.git
~/.pyenv/plugins/pyenv-virtualenv
```

- 使用virtualenv创建项目的虚拟环境

**一个项目创建一个virtualenv的虚拟环境**，在这个环境中，可以用pip安装项目所需的库，不会影响其他项目。切记一个项目一个虚拟环境，否则可能会发生莫名的错误。

```
#1.首先创建项目目录
```

```
$ sudo mkdir -p myproject/blog
cd myproject/blog

#2.创建项目的虚拟环境
#用法：pyenv virtualenv python版本号 虚拟环境名
$ pyenv virtualenv 3.6.4 env36   #注意版本号必须是已经安装的，否则会报错

#3.切换到虚拟环境
python@ubuntu:/myproject/blog$ pyenv activate env36

pyenv-virtualenv: prompt changing will be removed from future
release. configure `export PYENV_VIRTUALENV_DISABLE_PROMPT=1' to
simulate the behavior.

(env36) python@ubuntu:/myproject/blog$  # (env36)表示该项目处于虚拟环境
中

#验证
(env36) python@ubuntu:/myproject/blog$ python
Python 3.6.4 (default, Mar 29 2018, 10:33:37)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more
information.

#4. 切出虚拟环境
(env36) python@ubuntu:/myproject/blog$ pyenv deactivate env36
```

- 使用pip下载库

使用pip下载，会从国外的网站下载，速度超慢，所以要切换pip到国内的镜像源，一般会用psm切换pip的源

- 1.安装一个软件psm

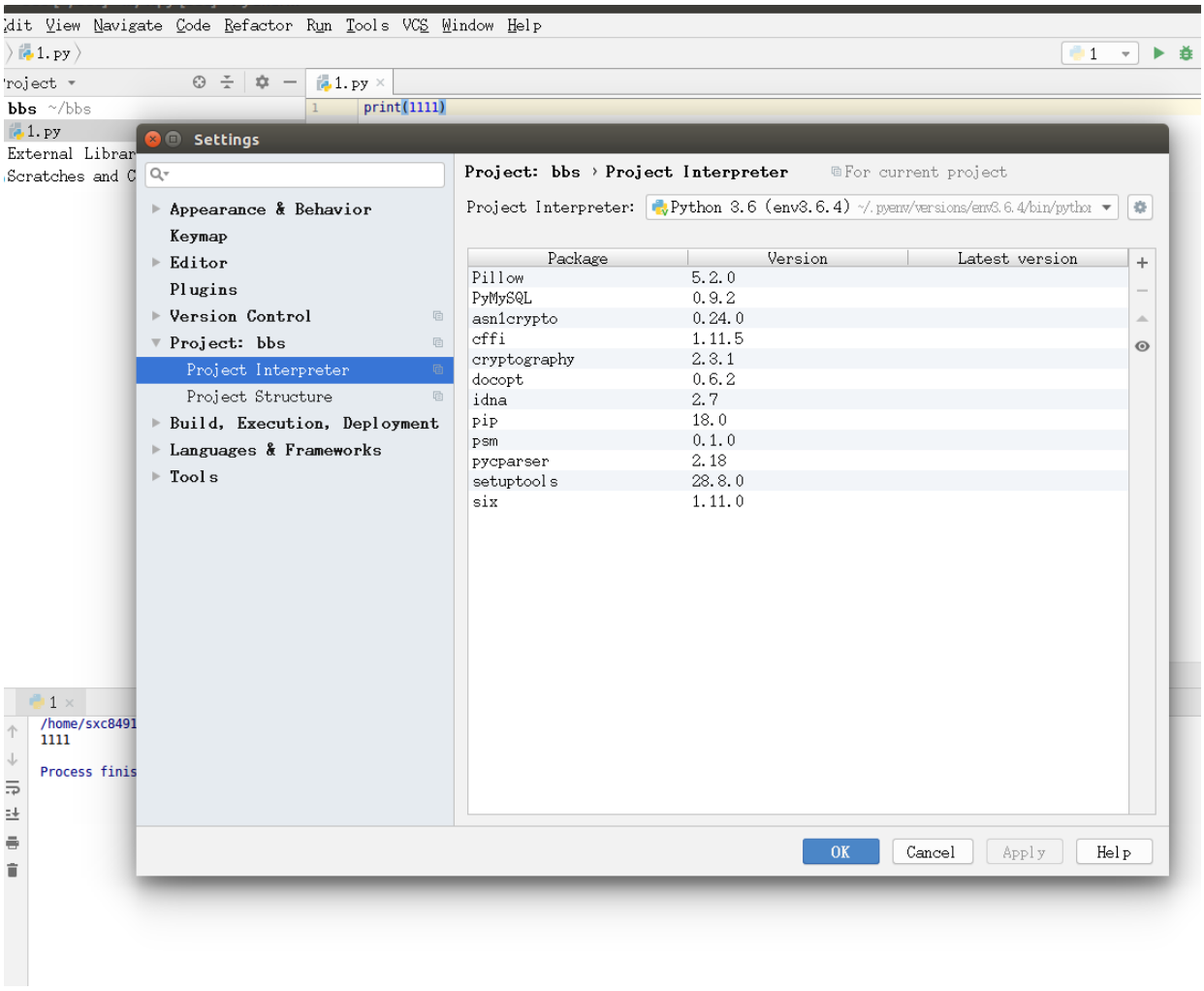
```
(bbs36)python@ubuntu:/myproject/blog$ pip install psm
[sudo] python 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
E: 无法定位软件包 psm
```

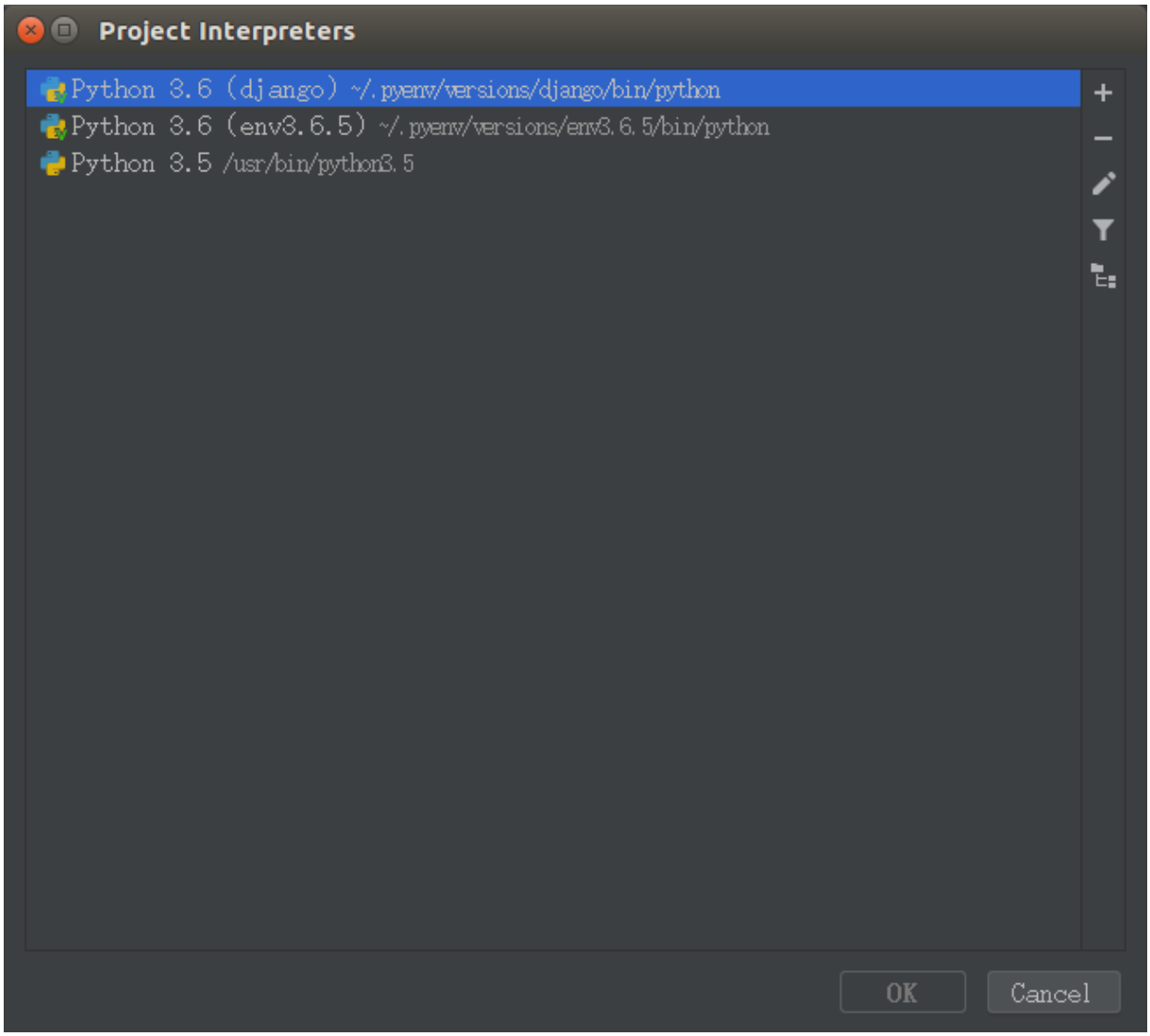
- 2.psm的使用 以下操作需要在虚拟环境外进行，选择好镜像源后在进入虚拟环境

```
#1.查看列出pip的镜像源
(bbs36)python@ubuntu:/myproject/blog$ psm ls

pypi     https://pypi.python.org/simple/
douban   http://pypi.douban.com/simple/
aliyun   http://mirrors.aliyun.com/pypi/simple/

#查看当前的镜像源
(bbs36)python@ubuntu:/myproject/blog$ psm show

Current source is douban

#2.选择指定的镜像源

(bbs36)python@ubuntu:/myproject/blog$ psm use douban

Source is changed to douban.

#3 使用pip下载库
(bbs36) python@ubuntu:/myproject/blog$ pip install pymysql
```

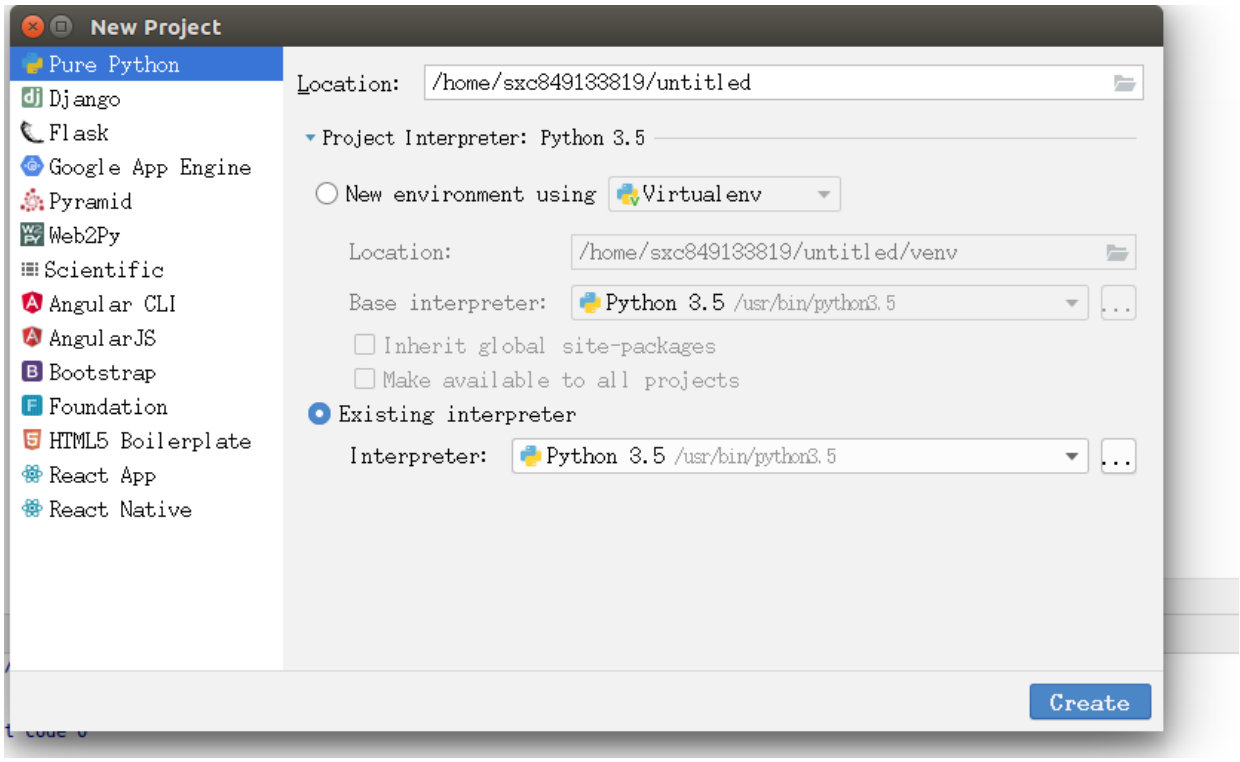
# 三、在pycharm中使用虚拟开发环境

1.创建一个新的工程，file-setting->project interpreter ， 点击右边的...按钮-->show all

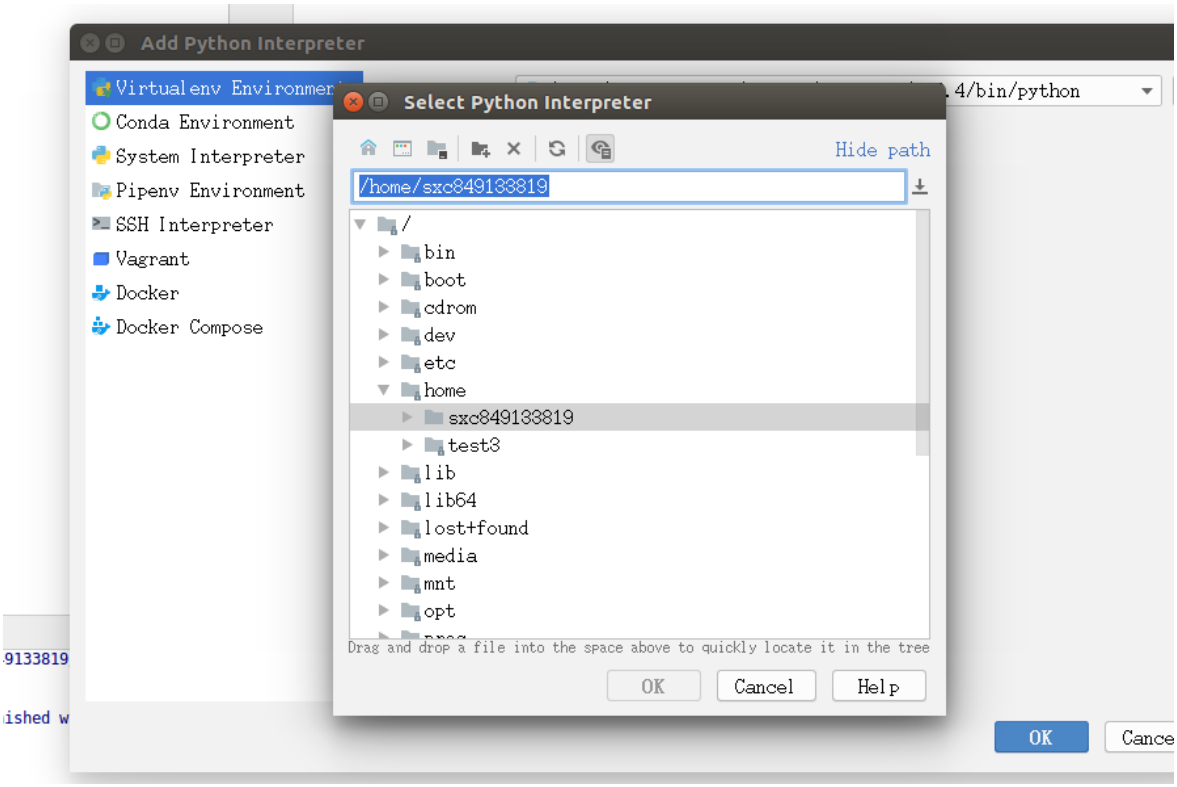

Edit View Navigate Code Refactor Run Tools VCS Window Help
1.py
1
Project
bbs ~/bbs
1.py
External Libra
Scratches and C
1.py
1 print(1111)
Settings
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: bbs
Project Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Project: bbs › Project Interpreter
For current project
Project Interpreter: Python 3.6 (env3.6.4) ~/.pyenv/versions/env3.6.4/bin/pytho
Package Version Latest version
Pillow 5.2.0
PyMySQL 0.9.2
asn1crypto 0.24.0
cffi 1.11.5
cryptography 2.3.1
docopt 0.6.2
idna 2.7
pip 18.0
psm 0.1.0
pycparser 2.18
setuptools 28.8.0
six 1.11.0
OK
Cancel
Apply
Help
1
/home/sxc8491
1111
Process finis

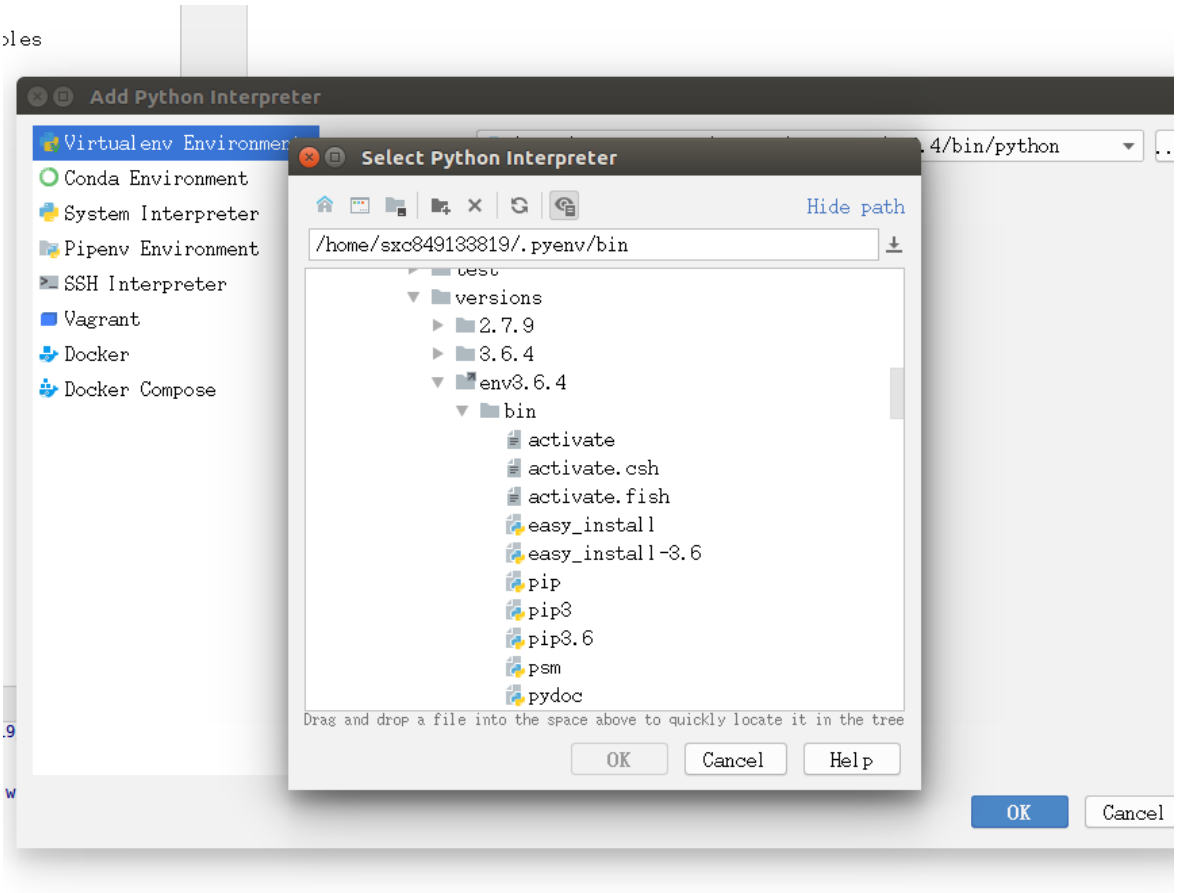


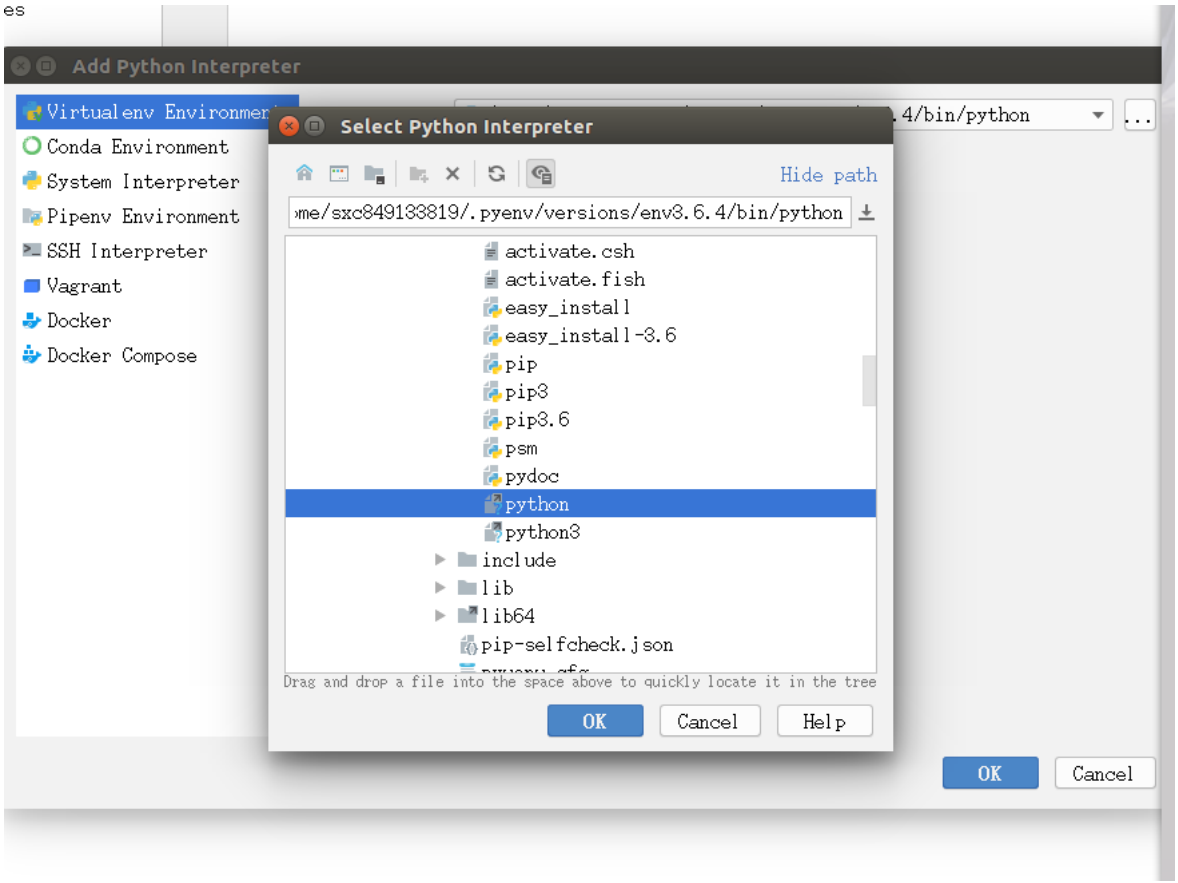

点击“+”，然后，在下图中选择Existing interperter后面的....



1. 打开virtual enviroment 选择已经存在的虚拟开发环境. 在你的家目录下-->.pyenv->versions --> 你的虚拟开发环境名称-->bin 选python






对已经存在的工程来说，file->settingd-> project xxx -->project interpreter 选择已经存在的虚拟开发环境

# 四、windows中创建虚拟开发环境

windows中安装完python之后默认会安装virtualenv，我们可以直接安装virtualenvwrapper

## 1.打开cmd，输入：

```
pip install virtualenvwrapper-win
```

## 2.新建虚拟环境

```
mkvirtualenv 虚拟环境名
```

## 3.启用虚拟环境

```
workon 虚拟环境名
```

## 4.退出当前虚拟环境

```
deactivate
```

## 5.删除某个虚拟环境

```
rmvritualenv 虚拟环境名
```