

vnc: 172.40.50.116:6000

张志刚

配置python环境

配置python虚拟环境

虚拟环境相当于是一个文件夹，把python复制进去。激活虚拟环境后，任何操作都是基于虚拟环境的。将来项目结束，把虚拟环境文件夹删掉即可。

1. 创建虚拟环境

```
[root@room8pc16 ~]# python3 -m venv mypy
```

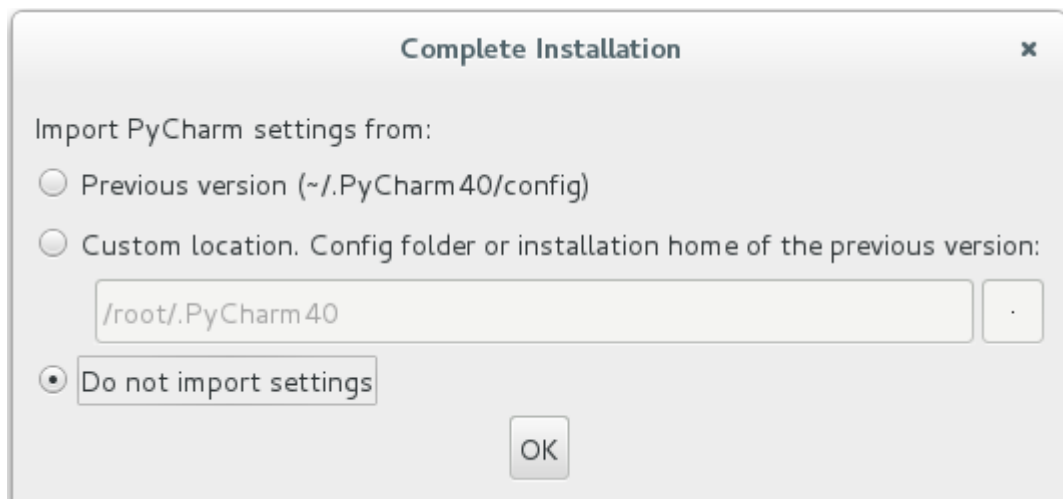
2. 激活虚拟环境

```
[root@room8pc16 ~]# source mypy/bin/activate
```

3. 退出虚拟环境

```
(mypy) [root@room8pc16 ~]# deactivate
```

4. 配置pycharm IDE



PyCharm License Activation - □ ×

☒ Activate ☐ Evaluate for free [Buy PyCharm](#)

Activate license with:

☐ JetBrains Account ☐ Activation code ☒ License server

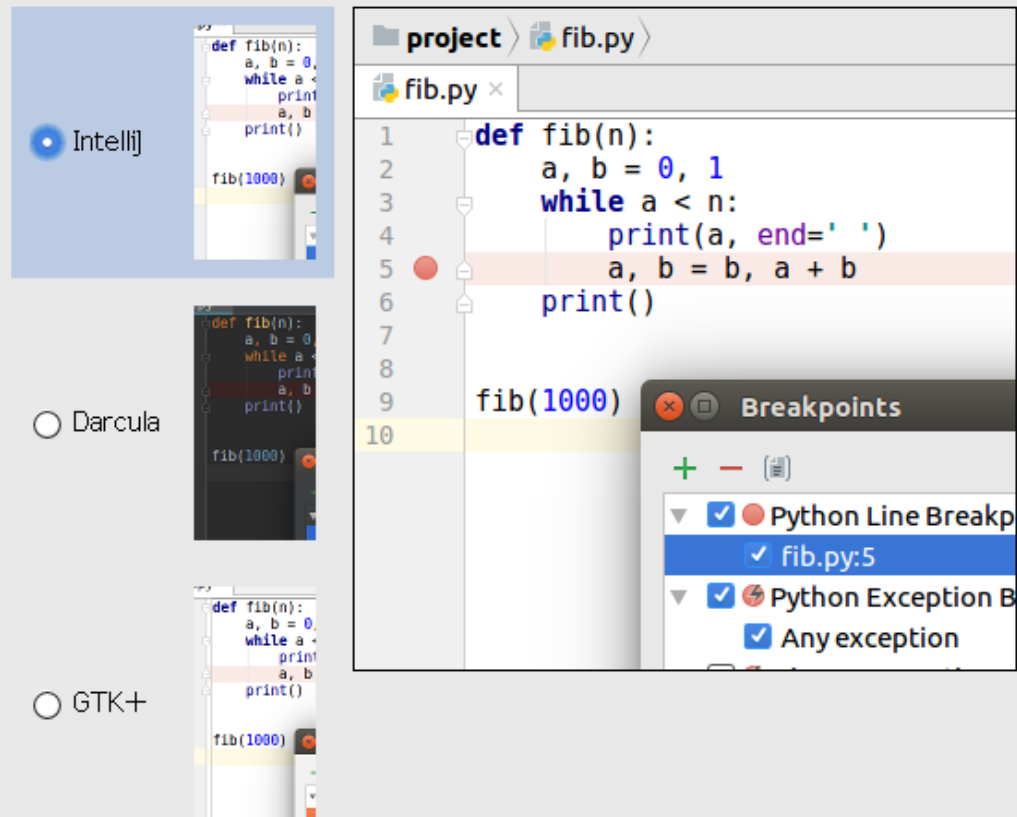
License server address: [More info...](#)

[Discover server](#)

[Activate](#) [Exit](#)

UI Themes → Launcher Script → Featured plugins

Set UI theme



The screenshot displays the 'Customize PyCharm' window. The 'Set UI theme' section is active. Three themes are available: IntelliJ (selected), Darcula, and GTK+. The main preview area shows the 'fib.py' file with the following code:

```
1 def fib(n):
2     a, b = 0, 1
3     while a < n:
4         print(a, end=' ')
5         a, b = b, a + b
6     print()
7
8 fib(1000)
```

A 'Breakpoints' window is open, showing the following settings:

- Python Line Breakpoint: ☒ fib.py:5
- Python Exception Breakpoint: ☒ Any exception

UI theme can be changed later in Settings | Appearance & Behavior | Appearance

[Skip Remaining and Set Defaults](#)

Next: Launcher Script



PyCharm

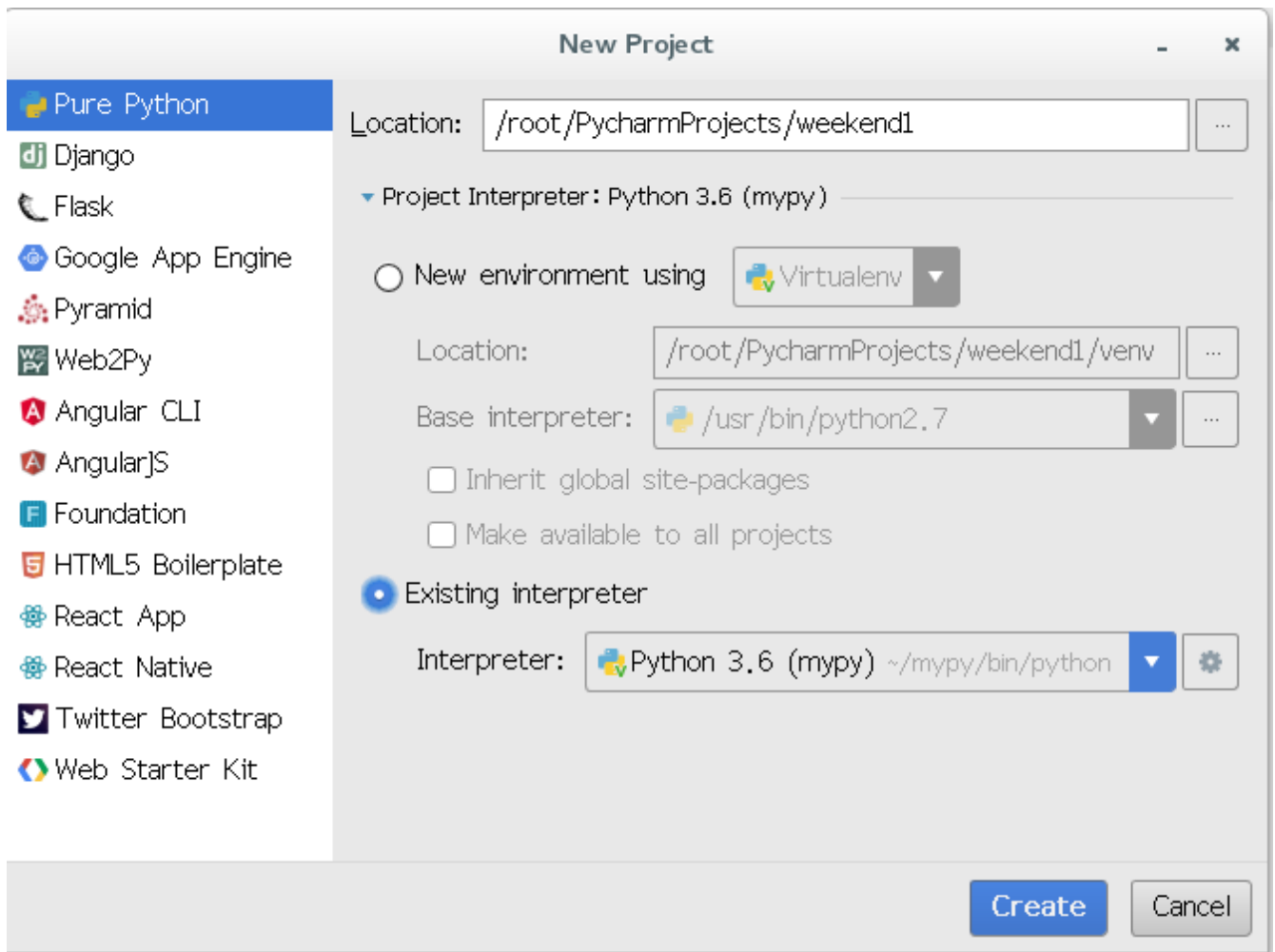
Version 2017.3

☀ Create New Project

📁 Open

⬇ Check out from Version Control ▾

🗨 3 Events ▾ ⚙ Configure ▾ 📖 Get Help ▾



python应用

python运行方式

- CLI

```
(mypy) [root@room8pc16 ~]# python
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
```

- 文件形式

```
(mypy) [root@room8pc16 py0101]# vim hello.py
print('Hello World!')

(mypy) [root@room8pc16 py0101]# python hello.py
Hello World!
```

pycharm注释：ctrl + /

print语句

```
>>> print('Hello World!')
Hello World!
>>> print('Hao', 123)
Hao 123
>>> print('Hello', 'World')    # 空格是默认的分隔符
Hello World
>>> print('Hello', 'World', sep='***')    # 各项用***分隔
Hello***World
>>> print('Hello' + 'World')    # 字符串用+拼接
HelloWorld
```

input语句

```
>>> input()
100
'100'
>>> input('number: ')    # 括号中的字符是屏幕提示语
number: 100
'100'
>>> num = input('number: ')    # 用户输入保存到变量num中
number: 100
>>> print(num)    # num直接使用，不用像shell那样加$
100
>>> num + 5    # input得到的一定是字符，字符不能与数字运算
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>> int(num)
100
>>> int(num) + 5    # int将字符100转成整数100
105
>>> str(5)
'5'
>>> num + str(5)    # str将数字5转成字符5
'1005'
```

变量

变量定义

- 首字符必须是字母或下划线
- 其他字符可以是字母、数字或下划线
- 区分大小写

变量在使用之前必须先赋值，否则出现NameError。

推荐采用的命名方法

- 变量名全部采用小写字母 pythonstring
- 简短、有意义 pystr

- 多个单词间用下划线分隔 py_str
- 变量名用名词,函数名用谓词(动词+名词) phone update_phone
- 类名采用驼峰形式 MyClass

变量赋值

赋值操作自右向左进行，把 = 右边表达式的计算结果，赋值给 = 左边的变量

```
>>> a = 10 + 5
>>> a = a + 1
>>> a += 1    # a = a + 1的简化写法
>>> a++
File "<stdin>", line 1
    a++
    ^
SyntaxError: invalid syntax
>>> import this    # 打印python之禅
# 美胜丑、明胜暗、简胜繁
```

运算

```
>>> 5 / 3
1.6666666666666667
>>> 5 // 3    # 只保留商
1
>>> 5 % 3     # 模运算，求余
2
>>> divmod(5, 3)    # 同时得到商和余数
(1, 2)
>>> 2 ** 3     # 乘方，幂运算
8
>>> 10 < 20 < 30
True
>>> 10 < 20 > 15
True
>>> 10 < 20 and 20 > 15
True
```

各种各样的数据类型也可以作为判断条件，任何值为0的数字都是False，非0是True；其他非空对象是True，空对象是False。

```
>>> not 10
False
>>> 'a'
'a'
>>> not 'a'
False
>>> not 0
True
```

数据类型

数字

```
>>> 11
11
>>> 0o11    # 8进制
9
>>> 0o23
19
>>> 0x11    # 16进制
17
>>> 0x23
35
>>> 0b11    # 2进制
3
```

字符串

引号中的字符是字符串类型，单双引号没有区别

```
>>> hi = 'hello tom'
>>> hello = "hello tom"
>>> name = 'alice'
>>> 'hello %s' % name    # 引号中变化的部分用%s占位
'hello alice'
>>> '%s is %s years old' % ('alice', 20)
'alice is 20 years old'
>>> s = '%s is %s years old' % ('alice', 20)
>>> s
'alice is 20 years old'
>>> words = "hello\nhi\ngreet\nwelcome"
>>> print(words)
hello
hi
greet
welcome
>>> words = "hello\nhi\ngreet\nwelcome"
# 3引号可以保留格式
>>> wds = '''hello
... nihao
... greet
... welcome'''
>>> print(wds)
hello
nihao
greet
welcome
>>> wds
'hello\nnihao\ngreet\nwelcome'
```


字符串切片

```
>>> py_str = 'python'
>>> len(py_str)    # 求长度
6
>>> py_str[1]
'y'
>>> py_str[0]      # 下标从0开始
'p'
>>> py_str[6]      # 下标超出范围将报错
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> py_str[5]
'n'
>>> py_str[-1]     # 负数表示从右向左取
'n'
>>> py_str[2:3]    # 切片操作，起始下标包含，结束不包含
't'
>>> py_str[2:4]
'th'
>>> py_str[2:6]    # 切片不会出现下标越界报错
'thon'
>>> py_str[2:6000]
'thon'
>>> py_str[2:]     # 结束下标不写表示到结尾
'thon'
>>> py_str[0:2]
'py'
>>> py_str[:2]     # 开头不写，表示从开头取
'py'
>>> py_str[:]      # 从头到尾
'python'
>>> py_str
'python'
>>> py_str[:2]     # 步长值为2
'pto'
>>> py_str[1:2]
'yhn'
>>> py_str[::-1]   # 负数表示从右向左取
'nohtyp'
```

字符串连接

```
>>> 'hao' + ' 123'
'hao 123'
>>> '123' * 3
'123123123'
>>> '*' * 50
'*****'
```

列表和元组

```
>>> alist = [10, 20, 30, 'bob', 'alice', 'tom', [1, 2, 3]]
>>> len(alist)
7
>>> alist[0]
10
>>> alist[3:6]
['bob', 'alice', 'tom']
>>> alist[-1]
[1, 2, 3]
>>> alist[-1] = 100
>>> alist
[10, 20, 30, 'bob', 'alice', 'tom', 100]
>>> alist + [200]
[10, 20, 30, 'bob', 'alice', 'tom', 100, 200]
>>> alist * 2
[10, 20, 30, 'bob', 'alice', 'tom', 100, 10, 20, 30, 'bob', 'alice', 'tom', 100]
>>> atuple = (10, 20, 30, 'bob', 'alice', 'tom', 100)
>>> atuple[-1]
100
>>> atuple[:3]
(10, 20, 30)
>>> len(atuple)
7
>>> atuple[-1] = 1000 # 元组相当于静态的列表，不能修改
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

字典

```
>>> userdb = {'name': 'bob', 'age': 20}
>>> len(userdb)
2
>>> 'bob' in userdb # bob是字典的key吗?
False
>>> 'name' in userdb
True
>>> userdb['name'] # 通过key取value
'bob'
```

数据类型分类

按存储模型

- 标量：数字、字符串
- 容器：列表、元组、字典

按更新模型

- 不可变：数字、字符串、元组
- 可变：列表、字典

按访问模型

- 直接：数字
- 顺序：字符串、列表、元组
- 映射：字典

判断

```
>>> if 3 > 0:
...     print('yes')
...     print('ok')
...
yes
ok
>>> if 3 > 10:
...     print('yes')
... else:
...     print('no')
...
no
>>> if -0.0:
...     print('ok')
...
>>> if ' ':    # 空格也是字符，字符串非空，表示True
...     print('ok')
...
ok
```

条件表达式（三元运算符）

```
>>> a = 100
>>> b = 150
>>> if a <= b:
...     smaller = a
... else:
...     smaller = b
...
>>> smaller
100
>>> s = a if a <= b else b
>>> s
100
```

循环

一般来说，如果循环次数不确定，使用while循环；如果循环次数可以预知，使用for循环。

git

基础配置

```
[root@node3 ~]# yum install -y git
[root@node3 ~]# git config --global user.name "Mr.Zhang"
[root@node3 ~]# git config --global user.email "zzg@tedu.cn"
[root@node3 ~]# git config --global core.editor vim
[root@node3 ~]# git config --list
user.name=Mr.Zhang
user.email=zzg@tedu.cn
core.editor=vim
[root@node3 ~]# cat ~/.gitconfig
[user]
    name = Mr.Zhang
    email = zzg@tedu.cn
[core]
    editor = vim
```

git重要的工作区域

- 工作区：编写程序的目录
- 暂存区：工作区和版本库之间的缓冲地带
- 版本库：在工作区中有一个.git目录，它是版本库

git应用

创建版本库

- 没有项目时

```
[root@node3 ~]# git init mypro1
初始化空的 Git 版本库于 /root/mypro1/.git/
[root@node3 ~]# ls -ld mypro1
drwxr-xr-x. 3 root root 18 6月 30 17:06 mypro1
[root@node3 ~]# ls mypro1
[root@node3 ~]# ls -A mypro1
.git
```

- 已有项目目录

```
[root@node3 ~]# mkdir mypro2
[root@node3 ~]# echo '<h1>Hello World</h1>' > mypro2/index.html
[root@node3 ~]# cd mypro2/
[root@node3 mypro2]# git init .
初始化空的 Git 版本库于 /root/mypro2/.git/
[root@node3 mypro2]# ls -A
.git  index.html
```

查看状态

```
[root@node3 mypro2]# git status
# 位于分支 master
#
# 初始提交
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#   index.html
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
[root@node3 mypro2]# git status -s
?? index.html
```

添加文件到跟踪文件（暂存区）

```
[root@node3 mypro2]# git add .      # 整个目录全部添加到暂存区
[root@node3 mypro2]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更:
#   (使用 "git rm --cached <file>..." 撤出暂存区)
#
#   新文件:       index.html
#
[root@node3 mypro2]# git status -s
A  index.html
```

撤出暂存区

```
[root@node3 mypro2]# git rm --cached index.html
rm 'index.html'
[root@node3 mypro2]# git status
# 位于分支 master
#
# 初始提交
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#   index.html
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

重新加入到暂存区，并commit到版本库

```
[root@node3 mypro2]# git add .
[root@node3 mypro2]# git commit
# git commit将打开vim，写提交说明，如果不存盘退出，表示放弃提交
[root@node3 mypro2]# git commit -m "project init" # 直接提交
```

查看相关信息

```
[root@node3 mypro2]# git status
# 位于分支 master
无文件要提交，干净的工作区
[root@node3 mypro2]# git log
commit f332018bee3336ded987fecfa980045e0c779a71
Author: Mr.Zhang <zzg@tedu.cn>
Date:   Sun Jun 30 17:18:19 2019 +0800

    project init
```

继续编写代码

```
[root@node3 mypro2]# echo '<h2>how are you?</h2>' >> index.html
[root@node3 mypro2]# cp /etc/hosts .
[root@node3 mypro2]# ls
hosts  index.html
[root@node3 mypro2]# git status -s
M index.html
?? hosts
[root@node3 mypro2]# git add .
[root@node3 mypro2]# git status -s
A hosts
M index.html
[root@node3 mypro2]# git commit -m "modify index.html and add hosts"
[root@node3 mypro2]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

恢复误删除的文件

```
[root@node3 mypro2]# rm -rf *
[root@node3 mypro2]# git checkout -- *
[root@node3 mypro2]# ls
hosts  index.html
```

删除、改名等

```
[root@node3 mypro2]# git rm hosts
[root@node3 mypro2]# git status
[root@node3 mypro2]# git reset HEAD hosts # 恢复删除的文件
[root@node3 mypro2]# git checkout -- hosts

[root@node3 mypro2]# git mv hosts zhuji
[root@node3 mypro2]# git status
[root@node3 mypro2]# git commit -m "rename hosts => zhuji"
```

git参考书籍：<https://down.51cto.com/data/285697> 《pro git》