

# nsd1902 py01 day01

---

我的简书：<https://www.jianshu.com/c/00c61372c46a>

## python环境准备

---

python官方站点：<http://www.python.org>

### python虚拟环境

虚拟环境相当于是把python隔离到一个目录中，安装软件包时，都安装到了虚拟环境目录。将来项目结束后，直接把虚拟环境目录删除即可。

创建虚拟环境

```
[root@room8pc16 python01]# python3 -m venv ~/nsd1902
[root@room8pc16 python01]# ls ~/nsd1902
# 激活虚拟环境
[root@room8pc16 python01]# source ~/nsd1902/bin/activate
(nsd1902) [root@room8pc16 python01]# python --version
Python 3.6.7
```

运行python

- 交互解释器

```
(nsd1902) [root@room8pc16 python01]# python
>>> print('Hello World!')
Hello World!
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>>
```

- 文件形式

```
(nsd1902) [root@room8pc16 python01]# vim /tmp/hi.py
print('Hello World!')

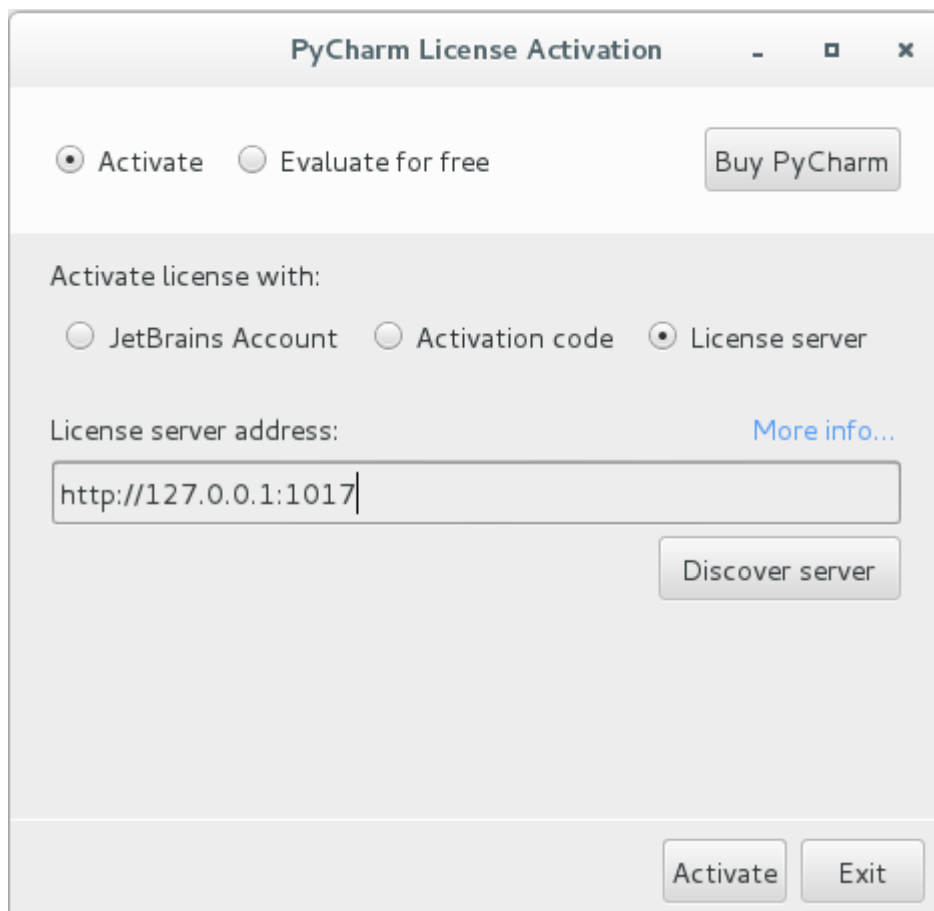
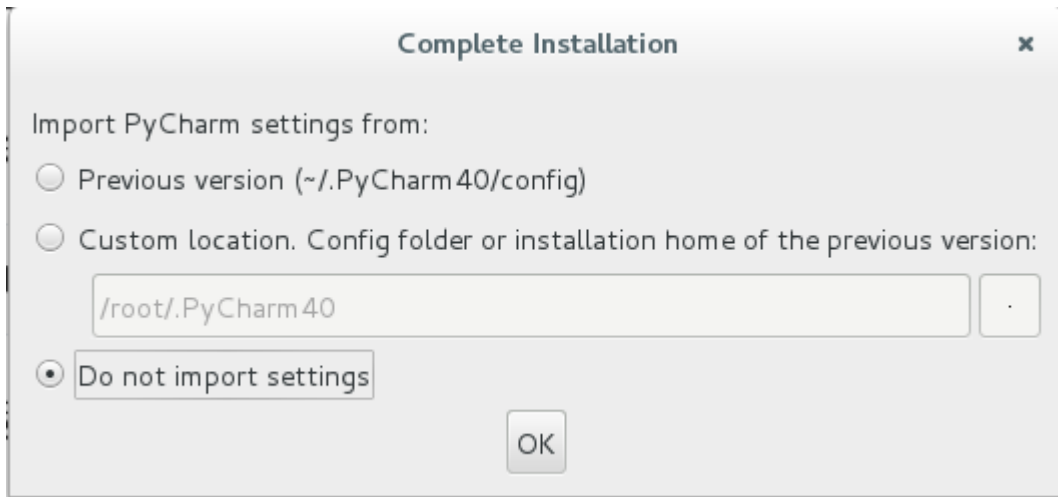
(nsd1902) [root@room8pc16 python01]# python /tmp/hi.py
Hello World!
```

## PyCharm配置

1. 删除pycharm配置

```
[root@room8pc16 python01]# rm -rf ~/.PyCharm2017.3/
```

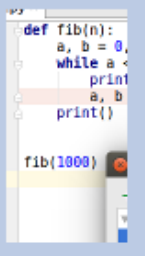
## 2. 启动pycharm



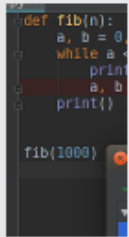
UI Themes → Launcher Script → Featured plugins

## Set UI theme

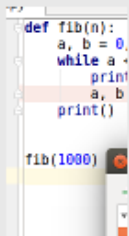
☒ IntelliJ



☐ Darcula



☐ GTK+



project > fib.py

fib.py x

```
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
9 fib(1000)
10
```

Breakpoints

+ - (📄)

Python Line Breakp

☒ fib.py:5

Python Exception B

☒ Any exception

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

Skip Remaining and Set Defaults

Next: Launcher Script



# PyCharm

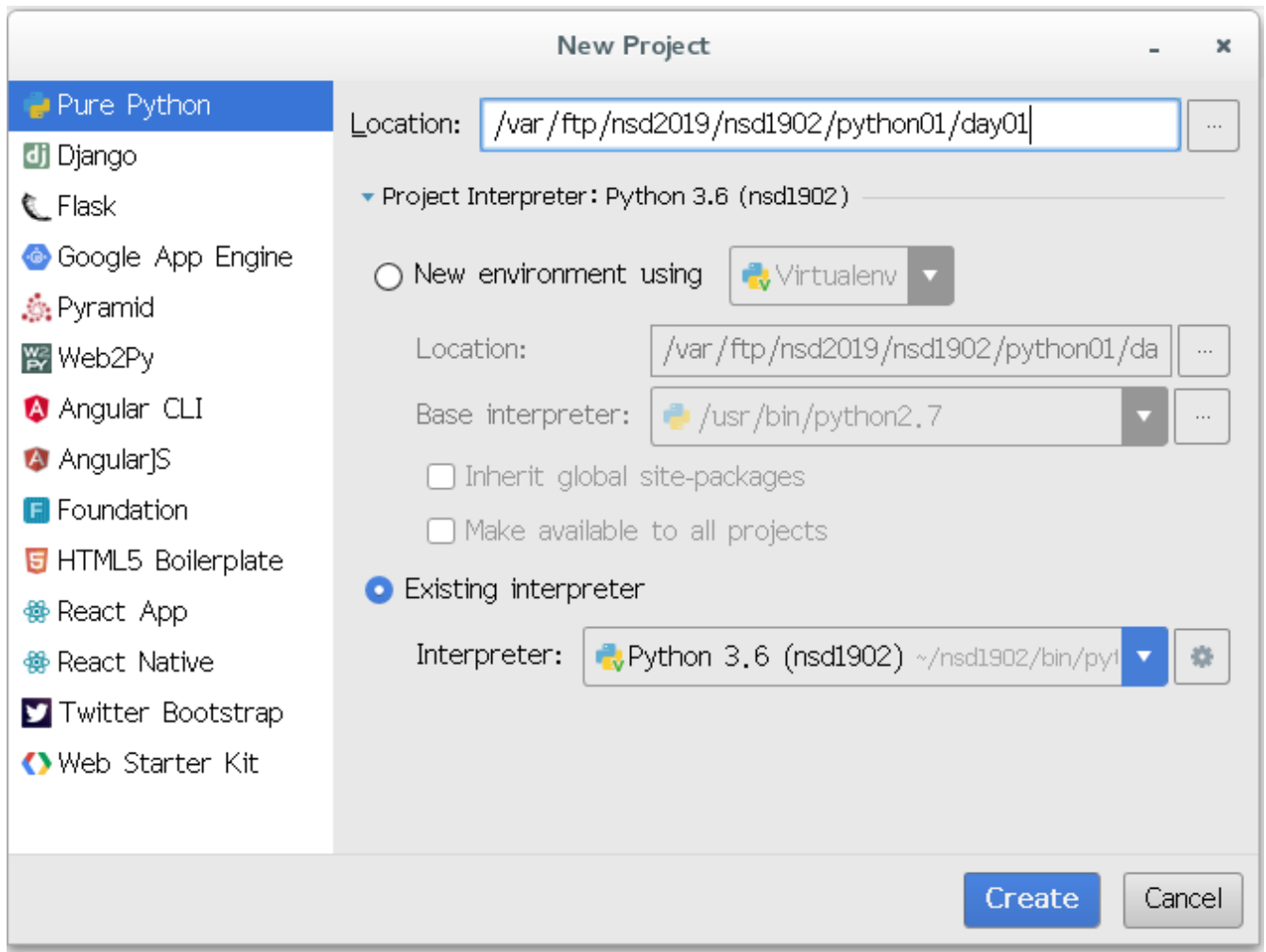
Version 2017.3

☀ Create New Project

📁 Open

⬇ Check out from Version Control ▾

🗉 3 Events ▾ ⚙ Configure ▾ 📖 Get Help ▾



3. 如果python解释器位置有误，在启动pycharm之后，选择File -> Settings -> Project: xxx -> Project Interpreter 进行修改
4. 修改编辑器文字大小：File -> Settings -> Editor -> Font

## 将vim打造成python解释器

<https://www.jianshu.com/p/29e7847f7298>

## python语法基础

- python完全靠缩进表达代码逻辑
- 注释采用#。pycharm的快捷键是ctrl+/
- 续行使用\
- 同行多个语句采用;分隔，但是不推荐，因为可读性下降

## 输出语句

```
>>> print('Hello World!')
Hello World!
>>> print('hao', 123)    # print可以打印多项
hao 123
>>> print('hao', 123, 'abc')    # 多项间默认以空格分隔
hao 123 abc
>>> print('hao', 123, 'abc', sep='***')    # 通过sep指定分隔符
hao***123***abc
>>> print('Hello' + 'World')    # 字符串可以用+拼接
HelloWorld
```

## 输入语句

```
>>> input()    # 从键读取数据
100
'100'
>>> input('number: ')    # 括号中的字符是屏幕提示语
number: 100
'100'
>>> num = input('number: ')    # 将输入保存到变量num中
number: 100
>>> num        # input读入的字符一定是字符型
'100'
>>> print(num)
100
>>> num + 5    # 字符串和数字无法直接运算
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>> int(num)    # 将字符串转成数字
100
>>> int(num) + 5    # 数字的四则运算
105
>>> str(5)        # 将数字转换成字符串
'5'
>>> num + str(5)    # 字符串拼接
'1005'
```

## 变量

字面量：如100、'name'这些数字、字符串等，它表示字面本身的含义

变量：可以变化的量，如变量a可以存100，也可以存'name'

### 变量名的约定

- 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

### 推荐的命名方法

- 变量名采用小写字母 pythonstring
- 简短、有意义 pystr
- 多个单词之间用下划线隔开 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

## 变量的应用

- 变量在使用之前必须赋值
- 变量在赋值的时候,决定了它是什么类型的
- 变量赋值顺序是自右向左进行

```
>>> a = 3 + 4
>>> a
7
>>> a = a + 1
>>> a
8
>>> a += 1      # a = a + 1的简化写法
>>> a
9
>>> a++          # python不支持此写法
File "<stdin>", line 1
    a++
    ^
SyntaxError: invalid syntax
>>> import this    # 显示python之禅
# 美胜丑、明胜暗、简胜繁
>>> ++a          # 此处不自增,只是正号而已
9
```

## 运算符

- 算术运算符

```
>>> 10 + 5
15
>>> 10 - 5
5
>>> 10 * 5
50
>>> 5 / 3
1.6666666666666667
>>> 5 // 3      # 整除,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)
>>> a, b = divmod(5, 3)  # 把商和余数分别赋值给a和b
>>> a
```

```

1
>>> b
2
>>> 2 ** 3      # 乘方，幂运算
8
>>> 3 * 2 ** 3   # 幂运算优先级高
24
>>> 3 * (2 ** 3) # 括号中的优先级高
24

```

- 比较运算符：结果是True或False

```

>>> 10 > 5
True
>>> 10 > 50
False
>>> 10 < 20
True
>>> 10 >= 10
True
>>> 3 == 3      # 判断相等关系
True
>>> 5 != 6       # 不等于
True
>>> 10 < 20 < 30   # python支持连续比较
True
>>> 10 < 20 > 15   # 容易引起歧义，不推荐
True
>>> 10 < 20 and 20 > 15 # 等同于 10 < 20 > 15
True

```

- 逻辑运算符

```

# and 两边的结果都为True，最终结果才为True
>>> 5 > 3 and 10 < 20
True
>>> 5 < 3 and 10 < 20
False
# or 两边的结果有一边为True，最终结果才为True
>>> 5 > 3 or 3 > 10
True
# not 取反
>>> not 5 > 3
False
>>> not 5 > 10
True

```

## 数据类型

### 数字分类



- 整数：没有小数点
- 浮点数：有小数点
- 布尔数：True的值是1，False的值0

```
>>> True + 1
2
>>> False * 3
0
```

- 复数：数学家为解决谁的平方等于-1，发明了复数

## 整数的表示方式

- python默认以10进制表示数字
- 8进制以0o开头
- 16进制以0x开头
- 2进制以0b开头

```
>>> 11
11
>>> 0o11
9
>>> 0o23
19
>>> 0x11
17
>>> 0x23
35
>>> 0b11
3

[root@room8pc16 nsd2019]# cp /etc/hosts /tmp/
[root@room8pc16 nsd2019]# ll /tmp/hosts
-rw-r--r--. 1 root root 9806 7月  1 15:45 /tmp/hosts
>>> import os
>>> os.chmod('/tmp/hosts', 600)
[root@room8pc16 nsd2019]# ll /tmp/hosts
---x-wx--T. 1 root root 9806 7月  1 15:45 /tmp/hosts
>>> os.chmod('/tmp/hosts', 0o600)
[root@room8pc16 nsd2019]# ll /tmp/hosts
-rw-----. 1 root root 9806 7月  1 15:45 /tmp/hosts
>>> os.chmod('/tmp/hosts', 493)
[root@room8pc16 nsd2019]# ll /tmp/hosts
-rwxr-xr-x. 1 root root 9806 7月  1 15:45 /tmp/hosts
```

## 字符串

字符串指的是在引号中间的字符，单双引号没有区别。

```
>>> "tom's pet is a cat"
```

```

"tom's pet is a cat"
>>> name = 'tom'
>>> age = 20
>>> "%s is %s years old." % (name, age)
'tom is 20 years old.'
>>> words = "hello\nwelcome\ngreet"
>>> print(words)
hello
welcome
greet
# 三引号可以保留用户的输入格式
>>> wds = '''hello
... welcome
... greet'''
>>> print(wds)
hello
welcome
greet
>>> wds
'hello\nwelcome\ngreet'

>>> py_str = 'python'
>>> len(py_str)
6
>>> py_str[0]
'p'
>>> py_str[5]
'n'
>>> py_str[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> py_str[-1]
'n'
>>> py_str[-6]
'p'
>>> py_str[2:3]    # 切片操作，起始下标包含，结束下标不包含
't'
>>> py_str[2:4]
'th'
>>> py_str[2:6]    # 没有下标6，但切片不会报错
'thon'
>>> py_str[2:6000]
'thon'
>>> py_str[2:]    # 结束下标不写，表示到结尾
'thon'
>>> py_str[:2]    # 开头不写，表示从开头取
'py'
>>> py_str[:2]    # 步长值为2
'pto'
>>> py_str[1::2]
'yhn'
>>> py_str[::-1]  # 负数表示自右向左取

```

```
'nohtyp'

>>> 'python' + ' good'
'python good'
>>> py_str + ' good'
'python good'
>>> py_str * 3
>>> '*' * 50
!*****!

#####
# yum install -y boxes cowsay
# cowsay 'Hello World'
# cowsay -l      # 列出所有可用形象
# cowsay -f dragon Hello World
# boxes -l
# echo 'hello world' | boxes -d ian_jones
#####
```

## 列表和元组

列表和元组与字符串有很大的相似之处

```
>>> alist = [10, 20, 30, 'tom', 'jerry', [1, 2, 3]]
>>> len(alist)
6
>>> alist[0]
10
>>> alist[-1]
[1, 2, 3]
>>> alist[3:5]
['tom', 'jerry']
>>> alist + [100]    # 列表拼接
[10, 20, 30, 'tom', 'jerry', [1, 2, 3], 100]
>>> alist * 2        # 列表重复
>>> alist[-1] = 100
>>> alist
[10, 20, 30, 'tom', 'jerry', 100]

# 元组相当于静态的列表
>>> atuple = (10, 20, 30, 'tom', 'jerry', 100)
>>> atuple[0]
10
>>> atuple[-1]
100
>>> atuple[2:4]
(30, 'tom')
>>> len(atuple)
6
>>> atuple[-1] = 200    # 报错，元组不能改变
```

# 字典

字典由无序的键值对构成

```
>>> adict = {'name': 'bob', 'age': 20}
>>> 'bob' in adict    # bob是字典的key吗?
False
>>> 'name' in adict   # name是字典的key吗?
True
>>> adict['name']     # 通过key取出value
'bob'
>>> adict['age'] = 22
>>> adict
{'name': 'bob', 'age': 22}
```

## 数据类型比较：重要

### 按存储模型分为：

- 标量：数字、字符串
- 容器：列表、元组、字典

### 按更新模型分为：

- 不可变：数字、字符串、元组
- 可变：列表、字典

```
>>> s1 = 'python'
>>> alist = [1, 2, 3]
>>> s1[0]
'p'
>>> alist[0]
1
>>> alist[0] = 10
>>> s1[0] = 'P'      # 报错，因为字符串不可变，不能原位修改
>>> s1 = 'Python'    # 整体重新赋值
```

### 按访问模型分为

- 直接：数字
- 顺序：字符串、列表、元组
- 映射：字典

