nsd1904_py01_day01

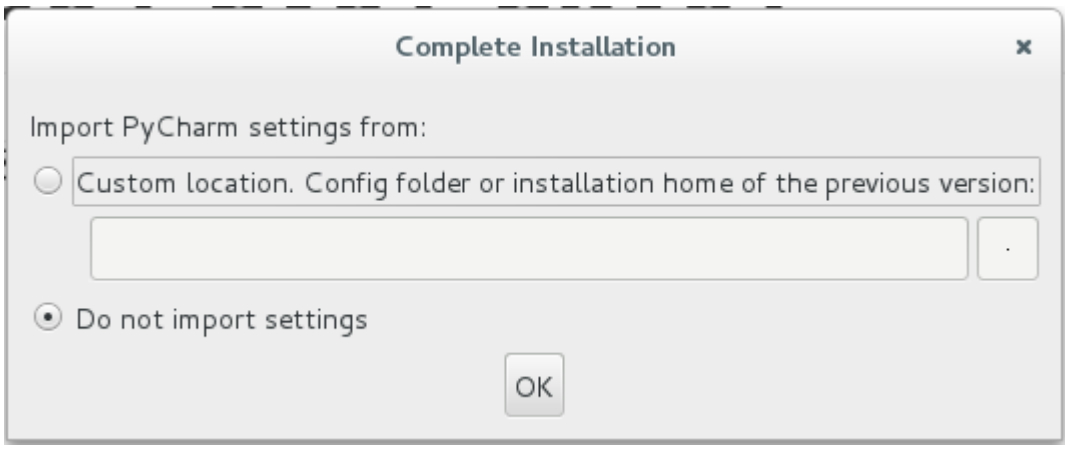
python百例：https://www.jianshu.com/c/00c61372c46a

vim的改造：https://www.jianshu.com/p/29e7847f7298

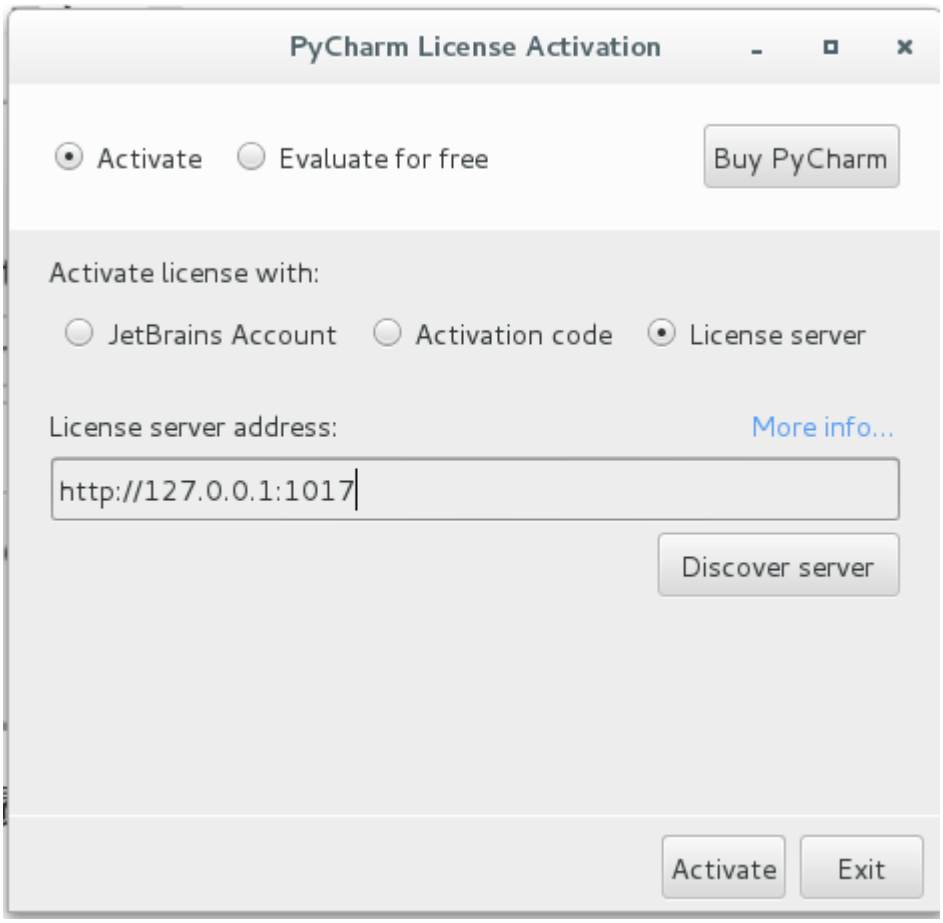
IDE：集成开发环境

PyCharm配置

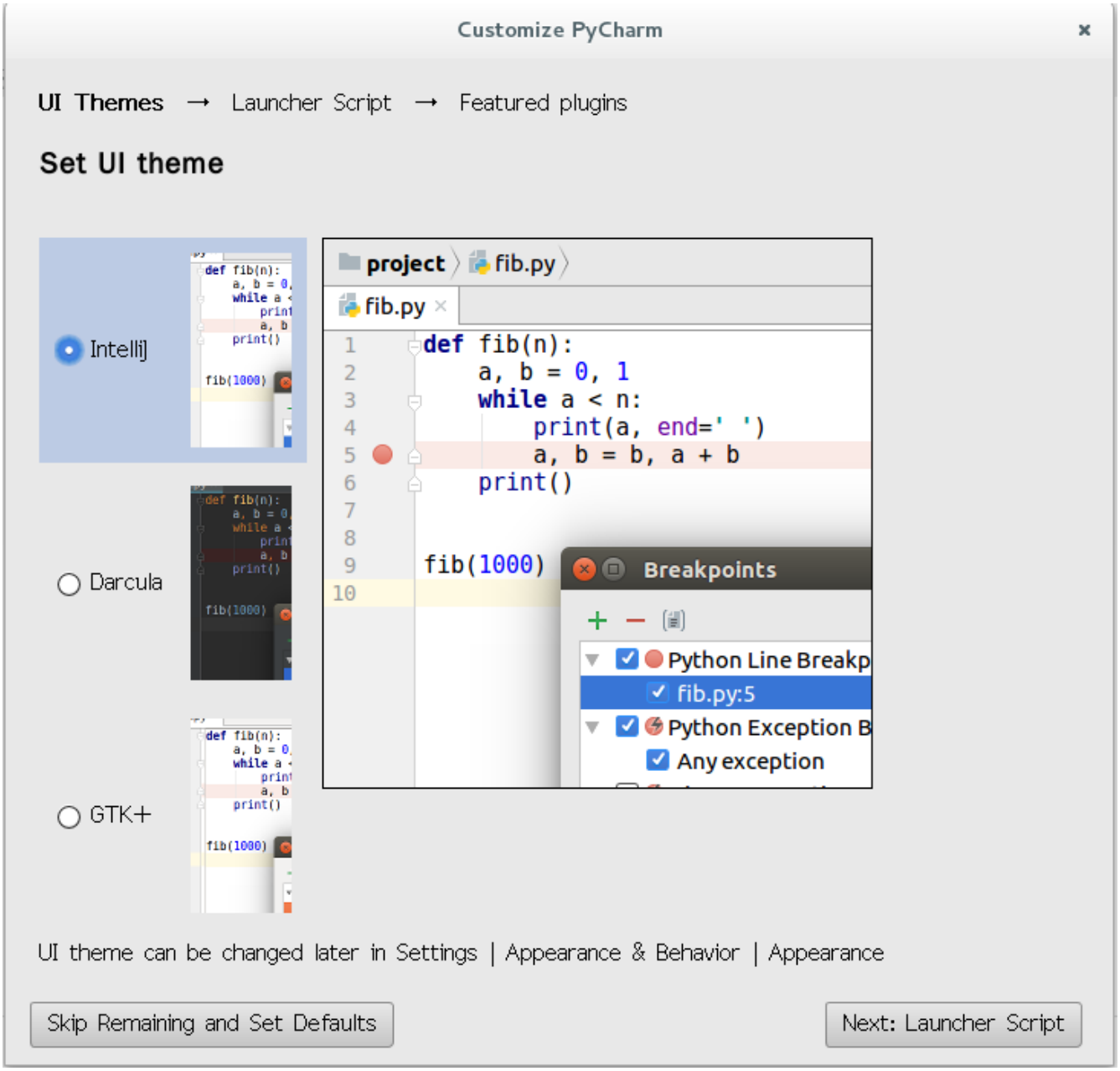
1. 打开PyCharm后，选不导入配置



2. 如果是专业版，需要付费购买



3. 选择一个界面风格，然后点击“Skip ....”



4. 新建一个虚拟环境。与虚拟机和docker容器类似，python虚拟环境，就是创建一个目录，将Python执行环境拷贝到这个目录。以后安装Python软件包，都安装到虚拟环境，当项目完成，删除这个虚拟环境即可。物理主机环境没有修改。

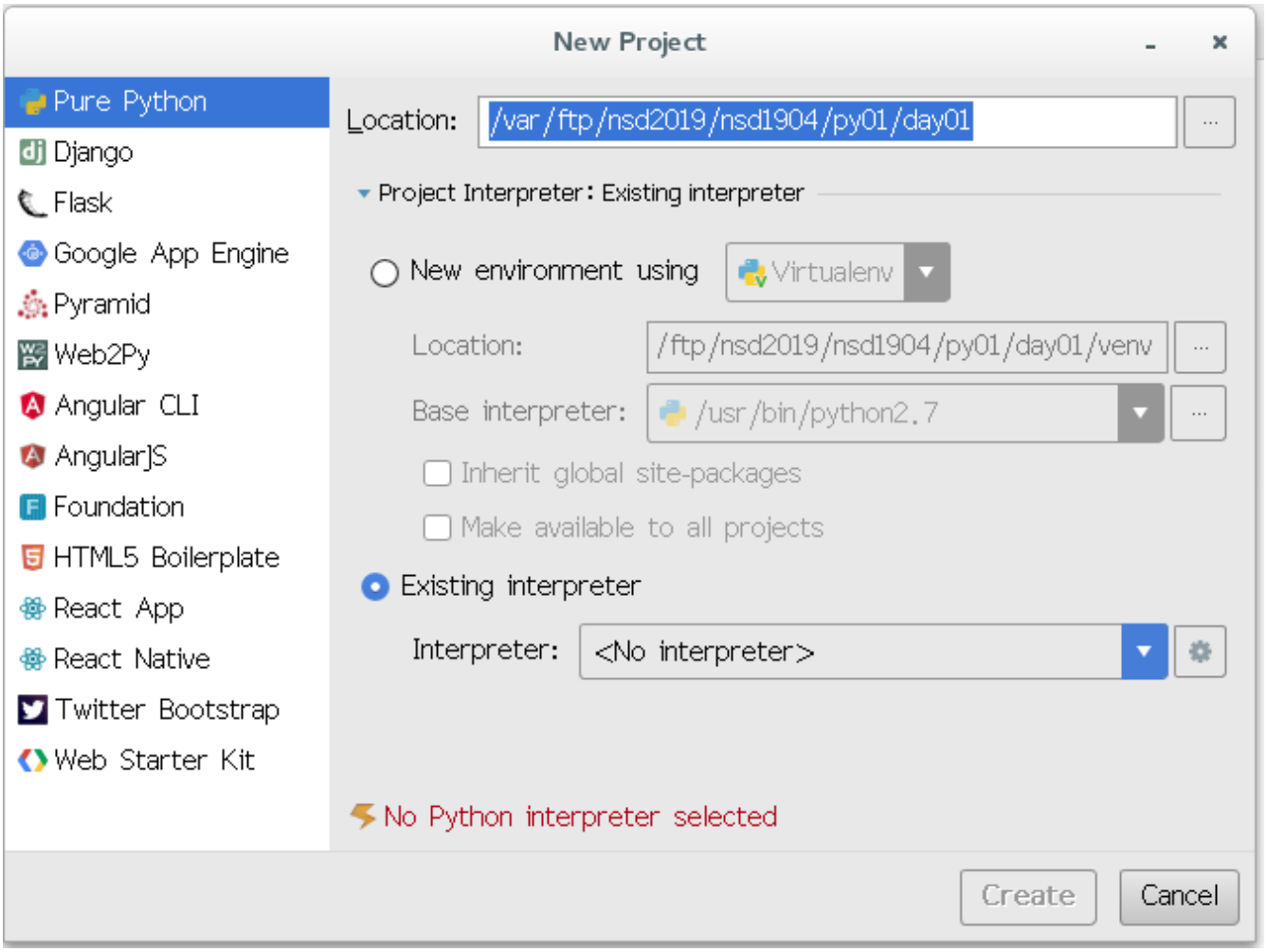
```
# 创建虚拟环境
[root@room8pc16 ~]# python3 -m venv ~/nsd1904
# 激活虚拟环境
[root@room8pc16 ~]# source ~/nsd1904/bin/activate
(nsd1904) [root@room8pc16 ~]#
(nsd1904) [root@room8pc16 ~]# python --version
Python 3.6.7

# 取消激活
(nsd1904) [root@room8pc16 ~]# deactivate
[root@room8pc16 ~]#
```

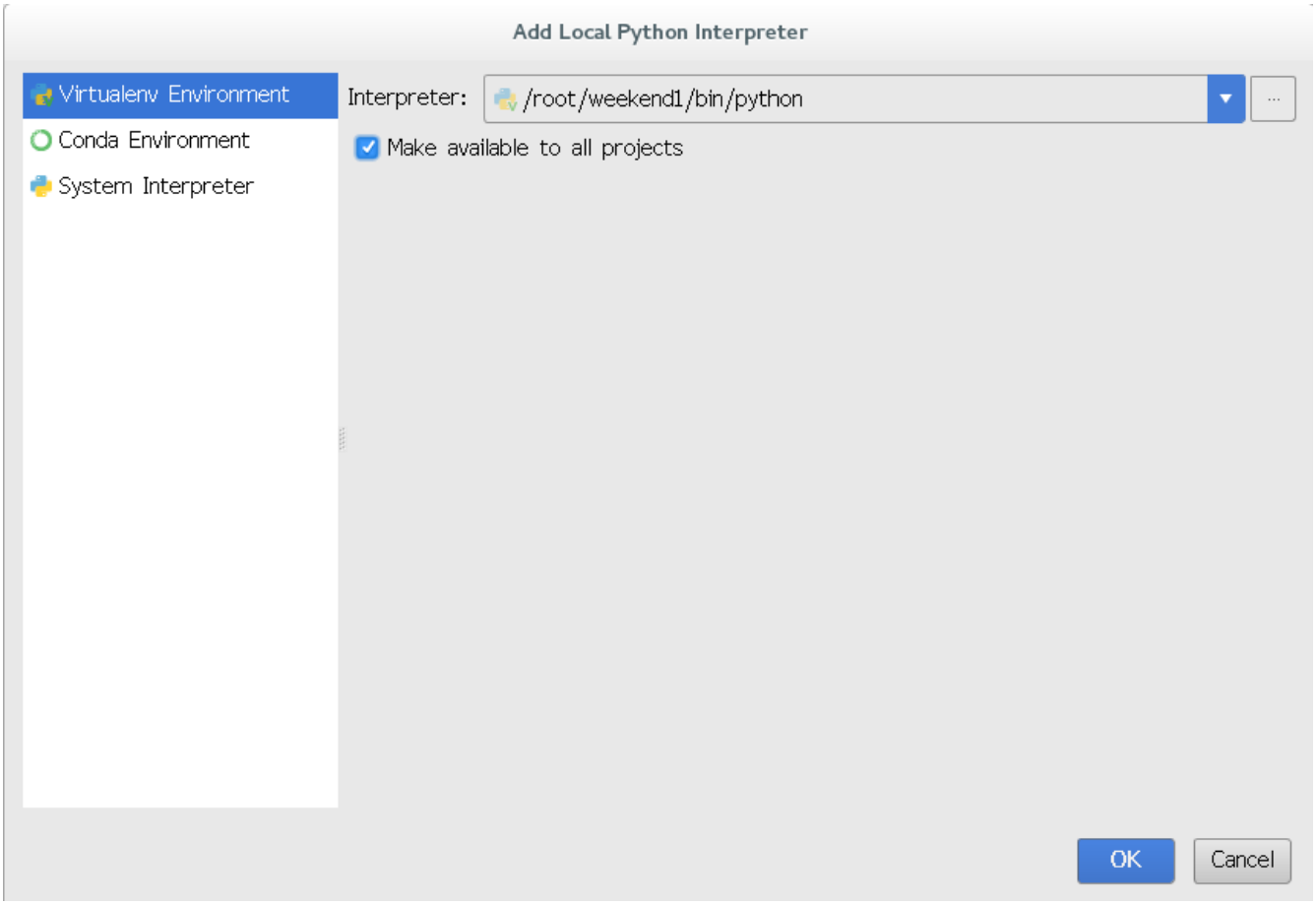
5. 新建项目


Welcome to PyCharm
PyCharm
Version 2017.3
Create New Project
Open
Check out from Version Control
Events
Configure
Get Help

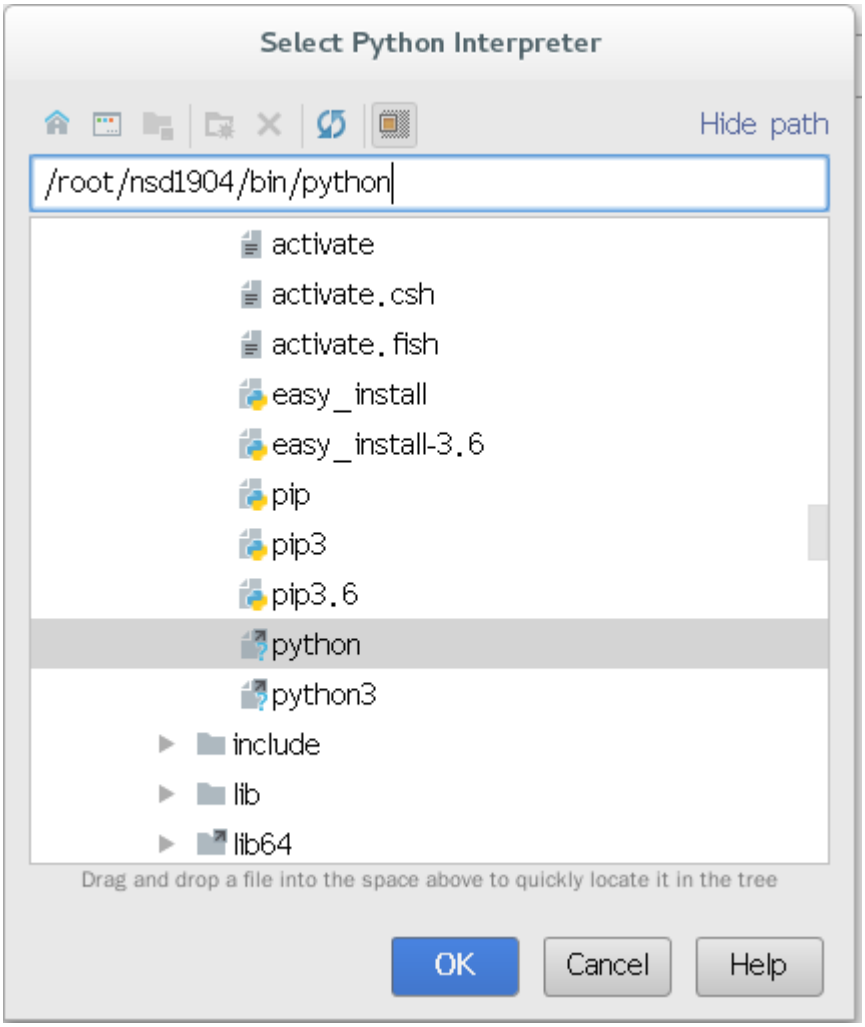
6. 为新项目选择解释器



Existing interpreter -> 右侧齿轮 -> add local，出现以下界面：



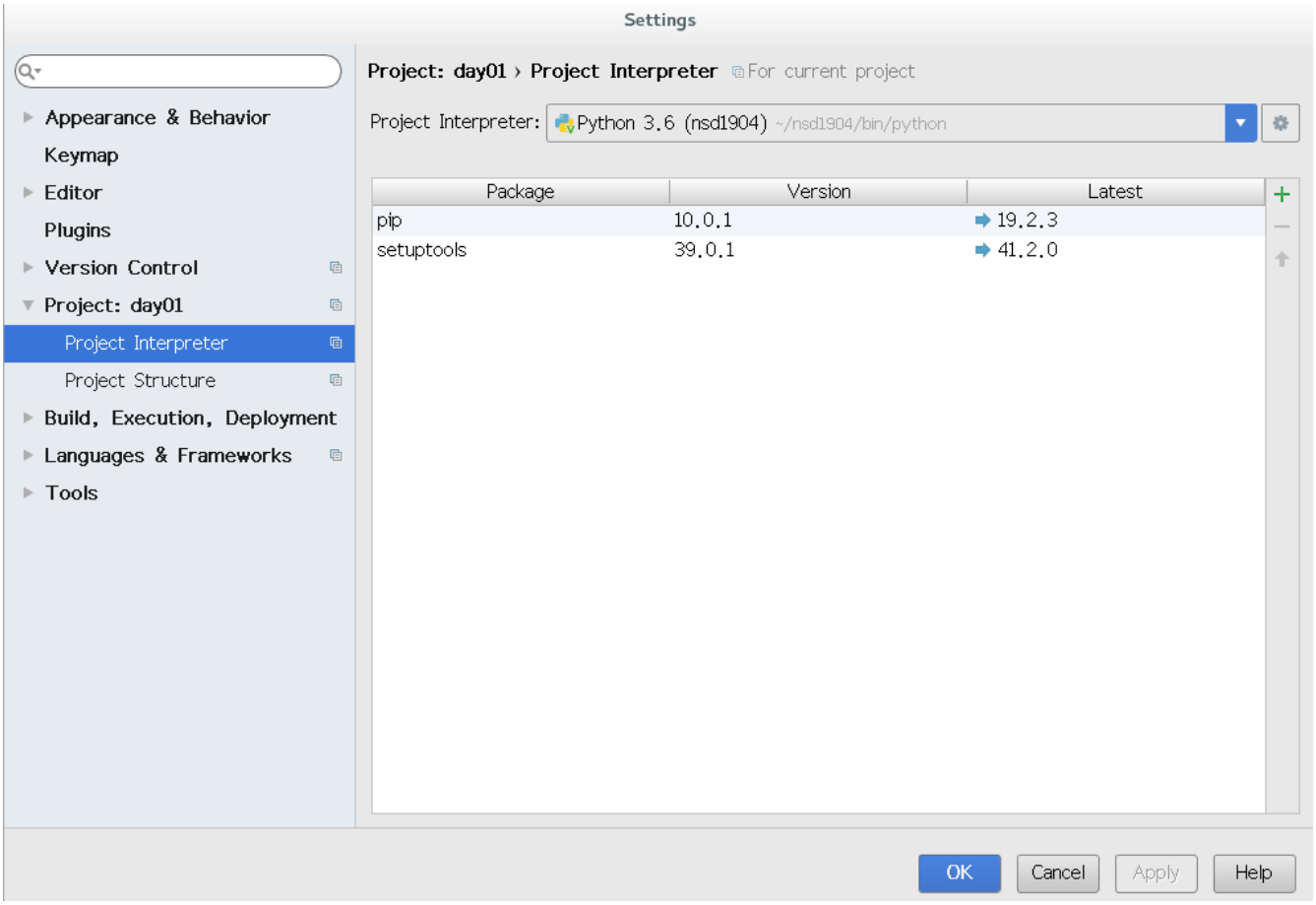
勾选Make available to all projects后，点三个点...，选取自己新建的虚拟环境，如下图所示：



然后在各个页面点击OK，创建项目。

PyCharm调整

File -> Settings



点击右上角的齿轮，设置解释器。点左侧Editor->Font可以改变文字大小。

python基础语法

- Python纯粹靠缩进表达代码逻辑。习惯缩进4个空格
- Python使用#注释，使用\续行
- 同行多语句用;分隔，但不推荐

input注意事项

input函数读入的数据一定是字符类型。

```
>>> n = input('number: ')
number: 10
>>> n + 2   # 字符串不能和数字直接相加
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>> int(n) + 2   # 通过int函数，将字符串转为数字
12
>>> n + str(2)   # 字符串拼接
'102'
```

变量

- 可以变化的量，称为变量。a=10
- 字面量，即字面本身含义的量，如'hello'，100。
- 变量的命名规范：
    - 首字符必须是字母或下划线
    - 其他字符可以是字母、数字、下划线
    - 区分大小写
- 变量命名推荐做法
    - 名字应该有意义
    - 变量名全部采用小写字母, pythonstring
    - 可以使用单词简写, pystr
    - 多个单词间用下划线分隔，py_str
    - 变量名使用名词, phone，函数名用谓词(动词＋名词)update_phone
    - class类名建议使用驼峰形式，MyClass
- 变量赋值，自右向左进行，用=实现赋值
- 变量在使用之前，必须初始化（赋值）

Python之禅

```
>>> import this
美胜丑、明胜暗、简胜繁
```

运算符

- 算术运算符

```
>>> 5 / 3
1.6666666666666667
>>> 5 // 3   # 只保留商
1
>>> 5 % 3    # 求余，也叫模运算
2
>>> divmod(5, 3)    # 同时得到商和余数
(1, 2)
>>> a, b = divmod(5, 3)
>>> a
1
>>> b
2
>>> round(5 / 3)   # 四舍五入
2
>>> round(5 / 3, 2)   # 四舍五入，保留两位小数
1.67
>>> 2 ** 3   # 幂运算，2的3次方
8
```

- 比较运算符：判断的结果是True或False

```
>>> 10 < 20 < 30   # python支持连续比较
True
>>> 10 < 20 > 15   # 相当于以下写法
True
>>> 10 < 20 and 20 > 15
True
```

- 逻辑运算符

```
>>> 20 > 10 and 10 < 30   # and两边全为True，结果才为True
True
>>> 20 > 10 and 10 < 3
False
>>> 20 > 10 or 10 < 3   # or两边只要有一个True，结果就为True
True
>>> 10 > 3
True
>>> not 10 > 3   # not颠倒真假
False
>>> not (10 < 5 and 20 > 10)
True
```

数据类型

1. 数字
    1. 无小数点的整数
    2. 有小数点的浮点数
    3. 布尔数也是整数：True为1，False为0

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

    4. 整数不同的前缀表示不同的进制，数字默认以10进制输出

```
>>> 11       # 没有任何前缀，表示10进制数
11
>>> 0o11     # 0o开头，表示8进制数，大小写均可
9
>>> 0O11
9
>>> 0b11     # 0b开头，表示2进制数
3
>>> 0B11
3
>>> 0x11     # 0x开头，表示16进制数
17
>>> 0X11
17
```

- 8进制数取值范围：0 - 7
- 2进制数取值范围：0 - 1
- 16进制数取值范围：0 - 9 a - f

5. 将10进制转成其他进制

```
>>> hex(100)    # 转成16进制
'0x64'
>>> oct(100)    # 转成8进制
'0o144'
>>> bin(100)    # 转成2进制
'0b1100100'
>>> 0x64
100
>>> 0o144
100
>>> 0b1100100
100
```

2. 字符串

1. 字符串必须使用引号引起来，否则会当成关键字或其他名字
2. 字符串不区分双引号还是单引号，表示完全一样的含义。如果引号中有需要替换的内容，可以使用%s占位。`

```
>>> uname = 'tom'
>>> myname = 'jerry'
>>> "Hi, %s. I am %s" % (uname, myname)
'Hi, tom. I am jerry'
>>> "Hi, %s" % uname   # 如果字符串中只的一个%s，后面数据不需要()
'Hi, tom'
>>> 'Hi, %s. I am %s' % (uname, myname)
'Hi, tom. I am jerry'
```

3. python还支持三引号，三个连续的单引号或双引号，用于保存输入格式

```
>>> words = """hello
... ni hao
... abc"""
>>> print(words)
hello
ni hao
abc
>>> words
'hello\nni hao\nabc'
>>> w = "hello\nnihao\ngreet"
>>> print(w)
hello
nihao
greet
```

4. 字符串切片等

```
>>> py_str = 'python'
>>> len(py_str)
6
>>> py_str[0]   # 取出第一个字符，下标为0
'p'
>>> py_str[6]   # 报错
>>> py_str[5]
'n'
>>> py_str[-1]   # 下标为负数，表示从右向左取
'n'
>>> py_str[-6]
'p'
>>> py_str[2:3]   # 切片，起始下标包含，结束下标不包含
't'
>>> py_str[2:4]
'th'
>>> py_str[2:6]   # 切片时，如果有不存在的下标也不会报错
'thon'
>>> py_str[2:]   # 结束下标不写，表示取到结尾
'thon'
>>> py_str[0:2]
'py'
>>> py_str[:2]   # 起始下标不写，表示从开头取
'py'
>>> py_str[::2]   # 第2个冒号后面的数字是步长值
'pto'
>>> py_str[1::2]
'yhn'
>>> py_str[::-1]   # 步长值为负，表示自右向左取
'nohtyp'

>>> 'hao' + '123'  # 字符串拼接
'hao123'
>>> '*' * 30   # 生成30个星号
'******************************'
>>> '#' * 30
'##############################'
>>> 'ab' * 30
'abababababababababababababababababababababababababababababababab'

>>> 't' in py_str   # 成员关系判定，t在python中吗？
True
>>> 'th' in py_str
True
>>> 'to' in py_str
False
>>> 'to' not in py_str
True
```

3. 列表

1. 类似于shell的数组，可以将任意对象保存

```
>>> alist = [10, 20, 30, 'tom', 'jerry']
>>> len(alist)
5
>>> alist[2]
30
>>> alist[3:]
['tom', 'jerry']
>>> 30 in alist
True
>>> alist + [40]   # 列表拼接
>>> alist * 2
[10, 20, 30, 'tom', 'jerry', 10, 20, 30, 'tom', 'jerry']
>>> alist.append(40)   # 把40追加到列表
>>> alist[-1] = 400    # 将列表最后一项的值改为400
```

4. 元组

1. 可以认为元组是静态的列表，元组一旦定义，不可改变

```
>>> atup = (10, 20, 30, 'tom', 'jerry')
>>> len(atup)
5
>>> atup[0]
10
>>> atup[3:]
('tom', 'jerry')
>>> 'tom' in atup
True
>>> atup * 2
(10, 20, 30, 'tom', 'jerry', 10, 20, 30, 'tom', 'jerry')
>>> atup[-1] = 'bob'   # 报错
```

5. 字典

1. 字典是无序的
2. 采用key: val的形式
3. 字典的key是唯一的，不会重现重复的key

```
>>> adict = {'name': 'tom', 'age': 20}
>>> len(adict)
2
>>> 'tom' in adict   # tom是字典的key吗？
False
>>> 'name' in adict
True
>>> adict['name']    # 字典通过key取出val
'tom'
>>> adict['age'] = 22   # 字典已有key，则更新val
>>> adict
{'name': 'tom', 'age': 22}
>>> adict['email'] = 'tom@tedu.cn'   # 向字典加入新项
```

```
>>> adict
{'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn'}
```

数据类型分类

- 按存储方式
  - 标量：只有一种对象。数字、字符串
  - 容器：可以保存各种对象。列表、元组、字典
- 按存储方式：
  - 可变：列表、字典
  - 不可变：数字、字符串、元组
- 按访问方式：
  - 直接：数字
  - 顺序：字符串、列表、元组
  - 映射：字典