学习python常用的快捷键：

1. PyCharm常用快捷键

Ctrl + / # 注释

Ctrl + A # 全选

Ctrl + C # 不需要选中一行，直接复制整行内容

Ctrl + X #不需要选中一行，直接剪切整行内容

Ctrl + V # 粘贴

Ctrl + D # 复制并粘贴，直接在下一行粘贴该行整行（或选中）的内容

Ctrl + Z # 回退到上一步操作

Ctrl + Shift + N # 通过文件名快速查找工程内的文件

Tab # 选中一段代码然后按 Tab 可以进行缩进

Shift + Tab # 选中一段代码然后按 Shift + Tab 可以进行反向缩进

Ctrl + Alt + L # 选中一段代码自动调整缩进格式（有的是Ctrl + Alt + I）

Alt + Enter # 导入系统模块

Shift + Enter # 回车到下一行（跟 vim 中的 o 功能一样，无需把光标定位到行末再回车）

Shift + F10 # 运行程序

2. 如何运行Python代码

如运行test.py脚本

1.1 在Linux中

a. 直接在命令行中执行python test.py命令

b. 先给脚本加上执行权限 chmod +x test.py，然后再执行该脚本 ./test/py

1.2 在Windows中

a. 进入到cmd，切换到脚本存放目录，执行python test.py命令

b. 使用PyCharm工具，快捷键Shift + F10执行程序，或者点击Run进行运行

c. 使用Python自带的IDE打开脚本，运行程序

3. PyCharm的调试模式

F9：进入调试模式选择框，可以选择进行调试的脚本（）

F8：一直往下一步走，每次跳一步，遇到调用方法（函数）时，直接返回函数结果，仍然继续往下

F7：一直往下一步走，当遇到调用方法（函数）时，跳到该方法（函数）位置，执行完函数后再回到原来位置继续往下走

Shift + F9：直接进入Debugger模式，在Debugger中的Variables中可以查看步骤执行之后的变量值，想要查看没有显示的变量值，可以通过“+”号添加，或者“-”号移除

序列是python中最基本的数据结构，序列中的每一个元素都分配一个数字，它的位置或索引，位置或索引下标从0开始。

一，列表（list）：是python中一种数据结构，它可以存储不同类型的数据，创建列表的时候，只需要把逗号分隔的不同的数据项使用方括号括起来即可，列表的索引从0开始

1，列表的循环遍历：

① for i in val:

print(i)

② i = 0（len(val)获取列表的长度，val[i]访问第i个元素）

while i < len(val):

print(val[i])

i += 1

2，常见列表的操作：

1）在列表中添加元素：如：

```
val = ['a',23,23.2,"dfjskf","安启力","dfj",342]
val1 = [2,34,534,"rejfd","eire"]
val.extend(val1)
```

①append方法向列表中添加的元素位于列表的末尾，当添加的元素是一个列表的时候，将添加进去的列表作为一个元素添加到

末尾

②extend方法可以将一个列表中的元素全部添加到另外一个列表，添加到末尾，添加进去的时候，是将该列表中的元素一次

添加到另一个列表的末尾

③insert（index，object）方法将指定位置index前插入元素object，object只能是一个元素，不能是列表，index以及之后的元素均向后移一位

2）在列表中查找元素：如：

```
val = ['a',23,23.2,"dfjskf","安启力","dfj",342]
print("lingmingjum" in val)
```

```
print("hello" not in val)
```

①in（存在）：如果存在那么结果为true，否则为false

②not in （不存在）：如果不存在返回true，如果存在返回false

3）在列表中修改元素：如：

```
val = ['a',23,23.2,"dfjskf","安启力","dfj",342]
val[4] ="林明俊"
print(val[4])
```

方法：通过访问下标然后对该下标位置的元素重新赋值实现对列表中元素的修改

4）在列表中删除元素①del：根据下标进行删除，可以删除整个列表的元素如 `del val1[i]`

②pop：删除最后一个元素如val1.pop()

③remove：根据元素的值进行删除如 val.remove(i)

注意：想要删除列表中多个位置的元素的时候，注意列表在删除第一个位置之后，长度发生了改变，剩下的元素进行重新排列，下标发生了变化

5）在列表中进行排序：①sort将列表中的元素按照特定的顺序进行排列，默认为由小到大，可以将sort方法中的reverse参数的值设置为true，使得列表中的元素由大到小排序，如

②reverse方法是将列表重置

注意：在python2.x中，使用sort（）进行的排序是永久性的，列表被排序后不能恢复带排序前的状态，所以只能先对列表进行sort之后的下一行，才能使用print函数对列表进行输出,同时还可以`sort(reverse=True)`使得按照从大到小进行排序；sorted（）进行的排序是临时的，可以在print函数中直接输出sorted临时排序的列表，如：

print(sorted(val ,reverse=True)), , sorted不会影响列表本身，但是sort会影响列

表本身；reverse（）只能把列表元素顺序导致，不会根据字母排序，也是永久性的，

要想恢复，再次使用reverse()就可以了，所以只能先对列表进行reverse之后的下一

行，才能使用print函数对列表进行输出；但是Python3.x和Python2.x的sorted函数有点

不太一样，少了cmp参数。

6）列表的嵌套：一个列表元素又是一个列表元素，如：

```
vals = [[],[],[],[]]
val1 = [90,95,98,93,99]
vals[0].append(val1)
```

二，元组（tuple）：python的元组（tuple）与列表（list）类似，不同之处在于元组的元素不能被修改，元组使用圆括号包含元素，而列表使用方括号包含元素，创建元组的时候，只需要在圆括号中添加元素，并使用逗号分隔即可，元组的索引是从0开始的

1）元组的操作：

①访问元组：可以使用下标索引来访问元组中的值，访问的时候使用中括号表示索引

```
如：while j <k:
    print(tuple1[j])
```

②修改元组：元组中的元素是不允许修改的，但是我们可以对元组进行连接组合，形成

一个新的元组

使用+进行元组之间的连接 如：

```
tuple2 =(21,32,23)
tuple3 = ("dhfj","dfhjs","yeeu")
tuple4 = tuple2 + tuple3
```

注意：在python中，不允许修改元组的数据，包含不能删除元组中的元素，否则会报错

③元组的遍历：方法1：使用for循环：

```
tuple1 = ("hello","world","kangkang","mical")
```

```
for i in tuple1:
    print(i)
```

方法2：使用while循环：

```
j = 0
k = len(tuple1)
while j <k:
    print(tuple1[j])
    j += 1
```

④元祖的内置函数：

注意：如需输出固定的字符串和变量的话，需要使用转义字符，将变量进行转译

一下在进行输出

内置函数1：len（tuple）计算元组元素的个数

内置函数2：max（tuple）返回元组中元素的最大值

内置函数3：min（tuple）返回元组中元素的最小值

内置函数4：tuple（list1）将列表list1转化为元组

三，字典：（dict）既能存储多个书数据，又可以快速准确定位到某个元素，字典的每个元素由俩部分组成，分别是键和值，键必须是唯一的，但是值可以是任意类型的

1，字典的常见操作：

①根据键访问值：

```
info = {"name":"anqili","id":1611010201,"address":"guizhou"}
print(info["name"])----结果是：anqili
```

如果我们想获取的某个键对应的值，但是又不是很确定字典中是否含有这个键，我们可以通过get方法进行，获取，get方法返回指定键的值，如果访问的键不在字典中，则会返回默认值None

```
info = {"name":"anqili","id":1611010201,"address":"guizhou"}
print (info.get("id"))
```

②修改字典中的元素：

```
info = {"name":"anqili","id":1611010201,"address":"guizhou"}
inp = input("请输入：")
info["id"] = int(inp)
```

如果要添加的键已经存在，那么字典中的该键对应的值会被新的值所代替

如果要添加的键不存在，则会创建新的键，该键的值就是新的值

③添加字典元素：如果字典中不存在这个键，那么字典中就会新增一个元素

```
info = {"name":"anqili","id":1611010201,"address":"guizhou"}
inp = input("请输入：")
info["age"] = int(inp)
```

④删除字典元素：可以使用del或者clear俩个命令实现，其中del用于删除字典或者元素，而clear只是单纯的清除字典中的所有数据，字典依然存在

如：

```
info = {"name":"anqili","id":1611010201,"address":"guizhou"}
del info["address"]
print(info)
info.clear()
print(info)
```

⑤计算字典中键值对的个数：*len（字典名称）*

⑥获取字典的键视图：*字典名称.keys（）*，实时反映字典中键的变化

⑦获取字典的值视图：*字典名称.values（）*，实时反映字典中值的变化

⑧获取字典的元素视图：*字典名称.items（）*实时反映字典中键和值的变化

2，字典的遍历：

①遍历字典的键：

```
for key in info.keys():
print(key)
```

②遍历字典的值：

```
for values in info.values():
print(values)
```

③遍历字典中的元素：

```
for ke in info.items():
print(ke)
```

④遍历字典中的键值对：

```
for key,value in info.items():
print("key=%s,value=%s"%(key,value))
```

注意：1，**set会自动去掉重复的元素**

2，如果需要将数据输出在一行，则需要在文件的开头写

from __future__ import print_function，然后print(输出内容,end="")

3，集合（set）不能使用下标运算

4，pop删除最后一个元素

5，python序列类型包含字符串、列表和元组三种，字典是python中唯一的映射类型

6，在python中，列表（list）和字典（dict）是可变数据类型；整型（int）、浮点

型float）、字符串类型和元组（tuple）类型是不可变数据类型