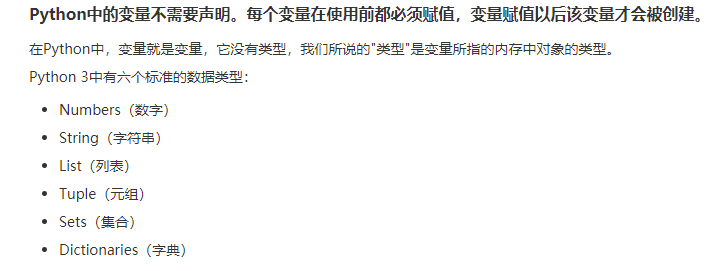
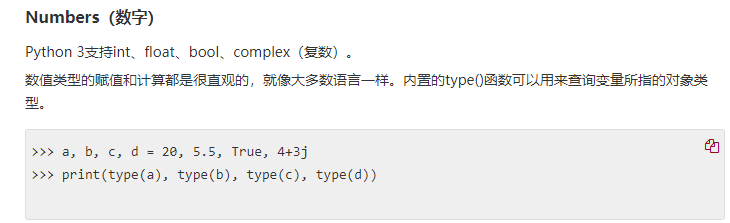
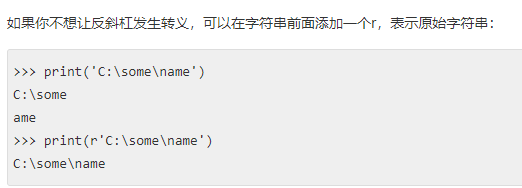
## 1 Python的变量和类型

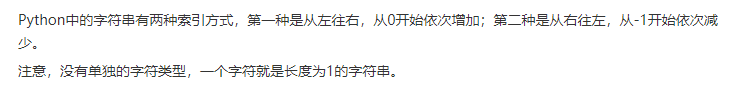


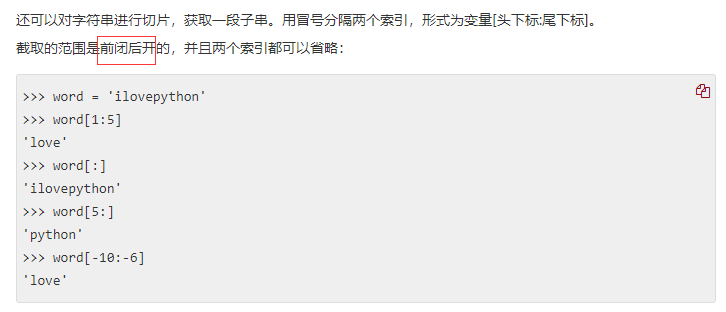
### Numbers类型

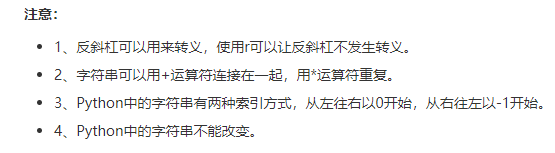


### String类型

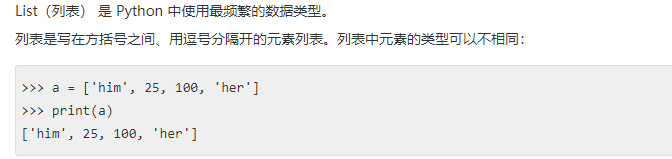


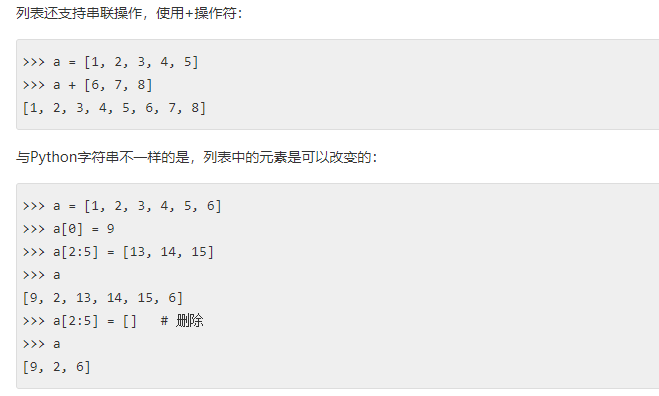




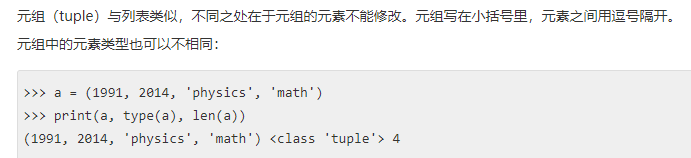


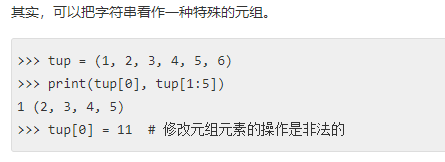
### List类型

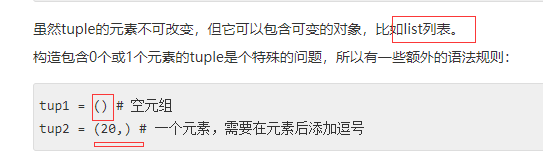




### Tuple类型

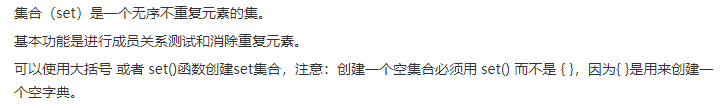


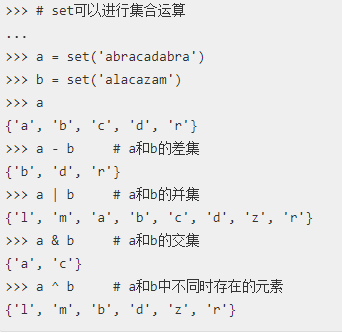




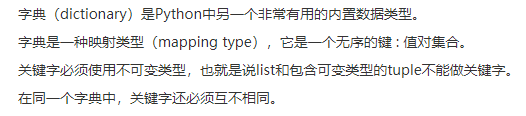


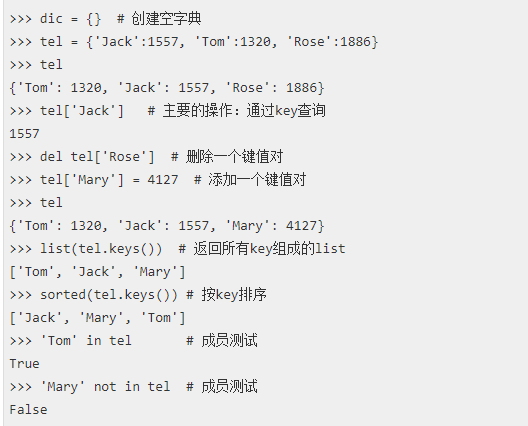
### Set类型

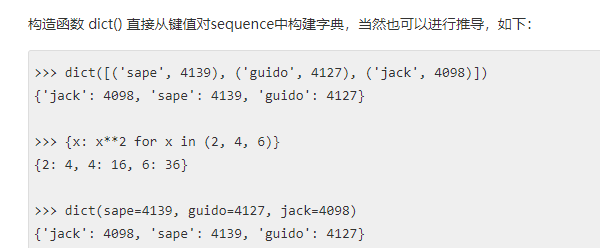




### Dictionaries类型







## 2 注释

单行：#

多行：’’’或者”””

## 3 运算符

主要注意\*\*(幂)和//(取整除)，两者还有赋值功能：\*\*=和//=；





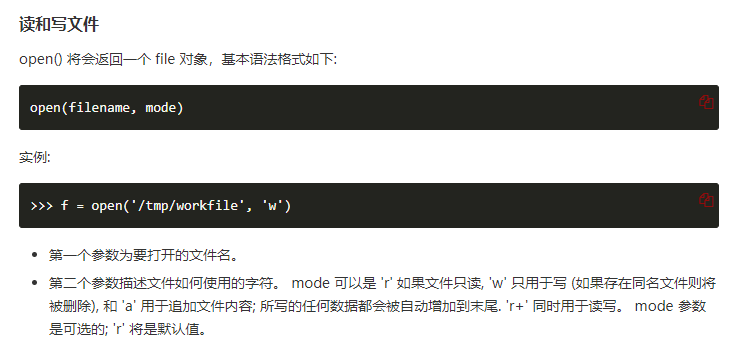


## 4 报错：IndexError: tuple index out of range

原因：参数数目不对，如：print('We are the {} who say {} who say "{}!"'.format('knights','Ni'))

只有两个参数传进去，但是我写错了，写了三个{}

## 5 文件对象的方法



### f.read()

为了读取一个文件的内容，调用 f.read(size), 这将读取一定数目的数据, 然后作为字符串或字节对象返回。

size 是一个可选的数字类型的参数。 当 size 被忽略了或者为负, 那么该文件的所有内容都将被读取并且返回。

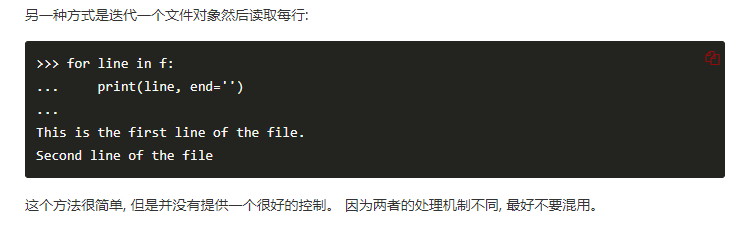
### f.readline()

f.readline() 会从文件中读取单独的一行。换行符为 '\n'。f.readline() 如果返回一个空字符串, 说明已经已经读取到最后一行。

### f.readlines()

f.readlines() 将返回该文件中包含的所有行。

如果设置可选参数 sizehint, 则读取指定长度的字节, 并且将这些字节按行分割。



### f.seek()

如果要改变文件当前的位置, 可以使用 f.seek(offset, from\_what) 函数。

from\_what 的值, 如果是 0 表示开头, 如果是 1 表示当前位置, 2 表示文件的结尾，例如：

* seek(x,0) ： 从起始位置即文件首行首字符开始移动 x 个字符
* seek(x,1) ： 表示从当前位置往后移动x个字符
* seek(-x,2)：表示从文件的结尾往前移动x个字符

### f.close()

在文本文件中 (那些打开文件的模式下没有 b 的), 只会相对于文件起始位置进行定位。

当你处理完一个文件后, 调用 f.close() 来关闭文件并释放系统的资源，如果尝试再调用该文件，则会抛出异常。

>>> f.close()

>>> f.read()

Traceback (most recent call last):

File "<stdin>", line 1, in ?

ValueError: I/O operation on closed file

<pre>

<p>

当处理一个文件对象时, 使用 with 关键字是非常好的方式。在结束后, 它会帮你正确的关闭文件。 而且写起来也比 try - finally 语句块要简短:</p>

<pre>

>>> with open('/tmp/workfile', 'r') as f:

... read\_data = f.read()

>>> f.closed

True

## 6 报错：io.UnsupportedOperation: can't do nonzero end-relative seeks

**原因：因为Pyhon3在文本文件中，没有使用b模式选项打开的文件，只允许从文件头开始计算相对位置，从文件尾计算时就会引发异常**

解决：使用’rb’打开文件就好了

## 7 序列化和反序列化

递归数据结构有指向原数据源的引用来表示，形式为<Recursion on typename with id=number>

## pickle 模块

python的pickle模块实现了基本的数据序列和反序列化。

通过pickle模块的序列化操作我们能够将程序中运行的对象信息保存到文件中去，永久存储。

通过pickle模块的反序列化操作，我们能够从文件中创建上一次程序保存的对象。

import pickle

# u表示unicode编码

data1 = {'a':[1,2.0,3,4+6j],'b':('string',u'Unicode string'),'c':None}

selfref\_list = [1,2,3]

selfref\_list.append(selfref\_list)

output = open('data.pkl','wb')

# Pickle dictionary using protocol 0.

pickle.dump(data1,output)

# Pickle the list using the highest protocol available.

pickle.dump(selfref\_list,output,-1)

output.close()

#使用pickle模块从文件中重构python对象

import pprint,pickle

with open("data.pkl",'rb') as pkl\_file:

data1 = pickle.load(pkl\_file)

pprint.pprint(data1)

data2 = pickle.load(pkl\_file)

pprint.pprint(data2)

# 递归数据结构有指向原数据源的引用来表示，形式为<Recursion on typename with id=number>

pkl\_file.closed