[一、字符串相关操作 1](#_Toc502933342)

[字符串相关函数 1](#_Toc502933343)

[字符串模块相关内容 string模块 6](#_Toc502933344)

[获取所有的空白字符 6](#_Toc502933345)

[获取ascii码的所有字母表（包含大写和小写） 6](#_Toc502933346)

[获取ascii码中的所有大写字母 6](#_Toc502933347)

[获取ascii码中所有的小写字母 6](#_Toc502933348)

[获取ascii码中所有10进制数字字符 6](#_Toc502933349)

[获取八进制所有数字字符 7](#_Toc502933350)

[获取十六进制的所有数字字符 7](#_Toc502933351)

[打印所有可见字符 7](#_Toc502933352)

[打印所有标点符号 7](#_Toc502933353)

[format格式化字符串 7](#_Toc502933354)

[二、 列表 8](#_Toc502933355)

[列表的操作 8](#_Toc502933356)

[创建列表 8](#_Toc502933357)

[列表的普通操作 8](#_Toc502933358)

[列表的序列操作 9](#_Toc502933359)

[列表的遍历 11](#_Toc502933360)

[序列函数 12](#_Toc502933361)

[列表专用函数 13](#_Toc502933362)

[三、元组 14](#_Toc502933363)

[元素的操作 14](#_Toc502933364)

[创建元组 14](#_Toc502933365)

[元组的特征 15](#_Toc502933366)

[序列操作 15](#_Toc502933367)

[元组的遍历 16](#_Toc502933368)

[元组的函数 16](#_Toc502933369)

[变量交换 （元组特殊用法） 17](#_Toc502933370)

[四、集合 17](#_Toc502933371)

[集合的定义 18](#_Toc502933372)

[集合的特征 18](#_Toc502933373)

[集合遍历操作 18](#_Toc502933374)

[集合内涵 19](#_Toc502933375)

[集合函数 19](#_Toc502933376)

[集合函数 20](#_Toc502933377)

[冰冻集合/固定集合 frozen 21](#_Toc502933378)

[五、 字典 21](#_Toc502933379)

[字典的定义 21](#_Toc502933380)

[字典的特征 22](#_Toc502933381)

[字典的基本操作 22](#_Toc502933382)

[遍历操作 23](#_Toc502933383)

[字典内涵 23](#_Toc502933384)

[字典函数 23](#_Toc502933385)

[字典方法 24](#_Toc502933386)

[六、内建函数 26](#_Toc502933387)

[数学相关内建函数 26](#_Toc502933388)

[七、文件操作 26](#_Toc502933389)

[打开文件 open 27](#_Toc502933390)

[关闭文件 close 28](#_Toc502933391)

[读取操作 read 28](#_Toc502933392)

[文件截取操作truncate 29](#_Toc502933393)

[文件指针操作 29](#_Toc502933394)

[文字相关内容 30](#_Toc502933395)

[八、 系统相关的操作 30](#_Toc502933396)

[OS模块 30](#_Toc502933397)

[函数部分 30](#_Toc502933398)

[值部分 32](#_Toc502933399)

[os.path模块 33](#_Toc502933400)

[绝对路径和相对路径 35](#_Toc502933401)

[归档和压缩 37](#_Toc502933402)

[zipfile zip文件操作 37](#_Toc502933403)

[九、 calendar日历模块 40](#_Toc502933404)

[日历模块 40](#_Toc502933405)

[time 模块 42](#_Toc502933406)

[时间戳 42](#_Toc502933407)

[UTC时间 42](#_Toc502933408)

[DST夏令时 43](#_Toc502933409)

[时间元组 struct\_time 43](#_Toc502933410)

[时间模块中的值 43](#_Toc502933411)

[时间模块属性 43](#_Toc502933412)

[时间模块方法 44](#_Toc502933413)

# 一、字符串相关操作

+ 字符串连接符号

\* 字符复制操作

'★' × 10 复制10个小星星

[] 通过指定位置（索引）来获取指定位置的字符

str[1] 获取第二个位置的字符

[:] 取片操作

[:] 获取所有字符

[开始位置:] 从开始位置获取到字符串的结尾

[：结束位置] 从字符串的开头截取到字符串结束位置之前（不包含结束位置）

[开始位置:结束位置] 从字符串的开始位置截取到结束位置之前（不包含结束位置）

r'字符串'

防止字符串中的转义字符串实现功能，把转义字符当作普通字符使用

## 字符串相关函数

**\*\* 注意**：以下的str不是模块名称，而是一个字符串变量！\*\*

capitalize() 首字母大写

格式： str.capitalize()

返回值：首字母大写的字符串，仅把整个字符串的第一个字符大写

upper() 将所有英文字符变为大写

格式：str.upper()

返回值：返回所有英文字符大写的字符串

lower（） 将所有英文字符变为小写

格式：str.lower()

返回值：返回所有英文字符小写的字符串

swapcase() 大小写互相转换

格式：str.swapcase()

返回值：转换之后的字符串

title() 按照标题格式进行大小写转换（每个单词首字母大写）

格式：str.title()

返回值：返回所有英文单词首字母大写的字符串

len() 计算字符串的字符个数,以后也可以用于计算元组列表等序列

格式: len(str)

返回值:整型

count() 计算一个字符串中出现指定字符串的次数

格式：str.count('查找的字符串'[,开始位置])

返回值:返回整型

find() 查找字符串中是否具有指定的字符串，查找不到返回-1

格式：str.find(查找的字符串[，开始位置])

返回值:第一次出现的位置

index() 查找字符串中是否具有指定的字符串，查找不到直接报错

格式：str.index(查找的字符串[，开始位置])

返回值:第一次出现的位置

**\*\* 注意\*\*：**find()和index()功能基本相似

startswith() 检测字符串是否以指定的字符串开头

格式： str.startswith('查找的字符串'[，开始位置])

返回值：布尔值

endswith() 检测字符串是否以指定的字符串结尾

格式： str.endswith('查找的字符串'[，开始位置])

返回值：布尔值

isupper() 检测字符串中字母是否都是大写字母

格式：str.isupper()

返回值：布尔值

islower()检测字符串中的字母是否都是小写字母

格式：str.islower()

返回值：布尔值

isalnum() 检测一个字符串是否都是有数字或者是否都是有字母组成

格式：str.isalnum()

返回值:布尔值

**\*\* 注意\*\*：**空字符串返回False

isalpha() 检测字符串是否都是有字母类型构成，汉字作为字母处理

格式：str.isalpha()

返回值：布尔值

**\*\* 注意\*\*：**空字符串返回False

isdigit() 检测字符串是否由纯数字字符组成

格式：str.isdigit()

返回值：布尔值

**\*\* 注意\*\*：**空字符串返回False

isspace() 检测字符串是否完全由空白字符组成

格式：str.isspace()

返回值：布尔值

**\*\* 注意\*\*：**回车，换行，缩进，空格都可以当作空白字符，空字符串是False

istitle() 检测字符串是否符合title()的结果，每个单词首字母大写

格式：str.istitle()

返回值：布尔值

isnumeric() 检测字符串是否有纯数字构成

格式：str.isnumeric()

返回值：布尔值

isdecimal(）检测字符串是否完全由十进制字符组成

格式：str.isdecimal()

返回值：布尔值

split() 使用指定的字符将字符串卷拆解成多个字符串

格式： str.split('用于拆解的字符串')

返回值：列表类型

splitlines() 使用\n，\r或者\r\n切割字符串

格式：str.splitlines()

返回值：列表类型

join() 使用指定的字符串将序列中的内容组成新的字符串

格式：连接字符串.join(序列)

返回值：组成的新的字符串

zfill() 0填充操作

格式： str.zfill(长度)

返回值：填充0的字符串

**\*\* 注意\*\*：** 1.填充结果原有内容靠右对齐，前面位数不足使用0来补充，常用于数字类型字符串。

center() 将字符串进行居中操作，并且在空白处进行填充

格式： str.center(未来字符宽度,空白填充字符)

返回值：新的字符串

ljust（） 将字符串进行左对齐操作，并且在空白处进行填充

格式： str.ljust(未来字符宽度,空白填充字符)

返回值：新的字符串

rjust() 将字符串进行右对齐操作，并且在空白处进行填充

格式： str.rjust(未来字符宽度,空白填充字符)

返回值：新的字符串

lstrip() 去掉字符串左侧的空白

格式： str.lstrip()

返回值：字符串

rstrip()去掉字符串右侧的空白

格式： str.rstrip()

返回值：字符串

strip()去掉字符串两侧的空白

格式： str.strip()

返回值：字符串

replace() 字符串替换操作

格式： str.replace(被替换的字符串,替换字符串)

返回值：替换之后的字符串、

maketrans() 制作一个字符串映射表，为了给translate函数使用

格式:str.maketrans（'被替换的字符串','替换字符串'）

返回值：字典类型，被替换字符串组成键，替换字符串组成值

**注意**： 两个参数的字符串个数必须一一对应，这里的str不是值字符串，而是真正的str，也可以写空字符串代替str（字符串都是str制作的）

translate() 进行字符串翻译操作，类似转换

格式：str.translate(映射表)

返回值：新的字符串

**注意**:translate用于多字符替换，replace用于长字符串替换

## 字符串模块相关内容 string模块

### 获取所有的空白字符

result = string.whitespace

print(result)

\n \r \t \x0b\x0c

### 获取ascii码的所有字母表（包含大写和小写）

print(string.ascii\_letters)

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

### 获取ascii码中的所有大写字母

print(string.ascii\_uppercase)

ABCDEFGHIJKLMNOPQRSTUVWXYZ

### 获取ascii码中所有的小写字母

print(string.ascii\_lowercase)

abcdefghijklmnopqrstuvwxyz

### 获取ascii码中所有10进制数字字符

print(string.digits)

0123456789

### 获取八进制所有数字字符

print(string.octdigits)

01234567

### 获取十六进制的所有数字字符

print(string.hexdigits)

0123456789abcdefABCDEF

### 打印所有可见字符

print(string.printable)

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"$%&'()\*+,-./:;<=>?@[\]^\_`{|}~

### 打印所有标点符号

print(string.punctuation)

!"$%&'()\*+,-./:;<=>?@[\]^\_`{|}~

### format格式化字符串

format() 格式化字符串

格式：str.format()

返回值：格式化之后的字符串

用法：

# 二、 列表

一组有序数据的组合就是列表

## 列表的操作

### 创建列表

1.创建空列表

列表变量 = []

2.创建单个数据的列表

列表变量 = [值]

3.创建多个数据的列表

列表变量 = [值,值,值....]

### 列表的普通操作

#### 列表的特征

1.列表是序列的一种，并且是由序的序列，可以使用有序序列访问方式

2.列表是可以修改的序列

3.列表的数据组成可以是任意一种数据

#### 列表的访问操作

列表变量[索引位置]

#### 添加列表操作

不可以使用索引方式添加

#### 修改列表操作

列表变量[索引位置] = 新值

#### 删除列表操作

del 列表变量[索引位置]

### 列表的序列操作

序列操作包括：索引，分片，序列相加，序列相乘，成员检测，最大值，最小值，长度计算

1.索引操作

列表 = [值,值，值....]

格式： 列表变量[索引]

作用:获取列表中指定位置的值

2.分片操作

列表 = [值,值,值...]

方法1：

列表[:]

表示获取列表的所有数据

方法2：

列表[开始索引：]

表示丛开始索引位置截取到列表的最后

方法3：

列表[:结束索引]

表示从列表的开头截取到结束索引位置之前，不包含结束位置

方法4：

列表[开始索引:结束索引]

表示从列表的开始索引位置截取到结束索引位置之前，，不包含结束位置

方法5：

列表[开始索引:结束索引:跳步值]

表示从开始索引位置到结束索引位置之间，不包含索引位置，并且获取数据时索引每次+跳步值来获取，默认跳步值为1

3.序列相加

格式：列表 = 列表1 + 列表2

结果：2个列表组成的新列表

**注意**: + 两边必须都是列表才可以运算

4.列表相乘

格式： 列表 = 列表1 \* 整数

结果：将列表重复N分组成新的列表

**注意**： 数字必须为整数

5.成员资格运算

格式：数据 in 列表

作用：检测数据是否在列表当中

返回值：布尔值

格式：数据 not in 列表

作用：检测数据是不是不再列表当中

返回值：布尔值

### 列表的遍历

1.for ... in 遍历

for i in 列表:

print(i)

2.while循环遍历

length = len(列表)

i = 0

while i < length:

print(列表[i])

i += 1

3.双层列表循环

列表 = [[值1,值2],[值1,值2]，[值1,值2]...]

格式： for 变量1,变量2 in 列表：

可以使用变量1和变量2

4.列表内涵： list content

简单的列表内涵：

格式：[变量 for 变量 in 列表]

作用：遍历列表中的数据并且组成新的列表，如果需要改变原有数据，在最开始i处修改即可

结果：新的列表

带有判断的列表内涵

格式 [变量 for 变量 in 列表 判断条件]

作用：遍历列表中的数据，根据判断条件取出符合条件的数据组成新的列表

结果:新的列表

多循环带判断的列表内涵

格式：[[变量1+变量2 for 变量1 in 列表1 for 变量2 in 列表2]

作用：将列表1中的每个数据和列表2的每个数据进行相加操作，此处可以使用别的操作（变量1+变量2 仅为参考）

结果：新的列表

格式：[[变量1+变量2 for 变量1 in 列表1 for 变量2 in 列表2 判断条件]

作用：将列表1中和列表2中每个数据进行操作，但是必须在符合判断条件的前提下进行。

结果：新的列表

作用：一行代码 99乘法表

### 序列函数

len() 获取列表的数据长度

格式：len(列表)

返回值：整型

max() 获取列表中的最大值

格式：max(列表)

返回值：列表中的最大值

min() 获取列表的最小值

格式：min(列表)

返回值：列表中的最小值

list() 将其他序列类型转化为列表

格式：list(序列)

返回值：列表

### 列表专用函数

append() 在列表的最后追加新数据

格式： 列表.append(数据)

返回值：无

**注意**：该操作直接改变原有列表

insert() 在列表指定的位置插入数据

格式：列表.insert（索引,数据）

返回值：无

**注意**：该操作直接改变原有列表

pop() 在列表中移除一个元素

格式：列表.pop()

返回值:移除掉的元素

**注意**：移除列表最后的元素

格式：列表.pop(索引)

返回值：:移除掉的元素

**注意**：移除列表中指定索引的元素

无论哪种格式都直接改变原有列表

remove() 在列表中移除指定的值的元素

格式:列表.remove(值)

返回值：无

**注意**：该操作直接改变原有列表

clear() 清空列表

格式：列表.clear()

返回值：无

**注意**：该操作直接改变原有列表

reverse() 列表反转

格式：列表.reverse（）

返回值：无

**注意**：该操作直接改变原有列表

extend() 在原有列表最后追加新的序列

格式:列表.extend(序列类型)

返回值： 无

**注意**：该操作直接改变原有列表

count() 计算指定值在列表中出现的次数

格式: 列表.count(值)

返回值：整数

copy() 复制原有列表

格式: 列表.copy()

返回值：新的列表

# 三、元组

什么是元组，tuple。元组也是一组有序的数据，元组不具备修改功能。

元组是一种特殊的列表而已。

列表就是铅笔写的有序内容

元组就是钢笔写的有序内容

## 元素的操作

### 创建元组

1.创建空元组

变量 = ()

2.创建单个数据的元组

变量 = (值,) 或 变量 = 值,

3.创建多个数据的元组

变量 = (值,值，值...) 或者 变量 = 值,值，值...

**注意**： 元组的关键符号是逗号而不是小括号。

## 元组的特征

1.元组具有序列特定，并且是有序序列

2.元组数据值可以访问不可以修改

3.元组的数据可以是任意数据类型

## 序列操作

序列操作：索引，分片，序列相加，序列相乘，成员资格

1.索引操作

元组[索引位置]

2.分片操作

元组[:] 获取元素的所有数据

元组[开始索引:] 获取开始索引位置到最后的所有数据

元组[:结束索引] 获取元组开头到结束索引位置之前的所有数据（不包含结束位置）

元组[开始索引:结束索引] 获取元素开始索引位置到结束索引位置之间的所有数据（不包含结束位置）

元组[开始索引:结束索引:跳步值] 获取元素开始位置到结束索引之间的指定间隔的数据数据（不包含结束位置）

3.相加操作

新元组 = 元组 + 元组

4.相乘操作

新元组 = 元组 \* 整数

5.成员检测

数据 in 元组

检测数据是否在元组当中，在返回True，不再返回False

数据 not in 元组

检测数据是否不再元组当中，不再返回True，在返回False

## 元组的遍历

普通元组的遍历

for 变量 in 元组：

使用变量

多层元组的遍历

for 变量1,变量2 in 元组：

使用变量

## 元组的函数

len() 获取元组中数据的长度

格式： len(元组)

返回值：整数

max() 返回元组中最大的值

格式：max(元组)

返回值：元组中的最大值

min() 返回元组中的最小值

格式：min(元组)

返回值：元组中的最小值

tuple() 将其他序列转化为元组类型

格式： tuple(任意序列)

返回值：元组

元组的方法

count() 计算指定数据出现的次数

格式:元组.count(查找的数据)

返回值：整数

index() 获取指定值在元组中的索引位置

格式：元组.index(查找的数据)

返回值：索引位置或者错误信息

## 变量交换 （元组特殊用法）

a = 值1

b = 值2

a,b = b,a

这是python中常用的变量互换方式，不要用列表

# 四、集合

集合是高中数据中的一个概念。

确定的一堆无序数据。集合中的每个数据称为一个元素

## 集合的定义

1.创建空集合

变量 = set()

2.创建多个数据的集合

变量 = {值,值,值....}

**注意**：集合是内部元素绝对唯一的数据，所以集合中出现多个相同数据时只会保留一份

## 集合的特征

1.集合是一种无序的数据类型，无法通过索引和分片进行操作

2.集合是内部元素绝对唯一的数据，常常用于去掉重复数据

3.集合的数据，str，int，float,bool,tuple，冰冻集合。

集合序列操作

成员检测操作

数据 in 集合

检测数据是否在集合当中

数据 not in 集合

检测数据是否不在集合当中

## 集合遍历操作

普通集合遍历

集合 = {值,值，值....}

for 变量 in 集合：

使用变量

带有元组的集合遍历

集合 = {（值1,值2..）,(值3,值4..)...}

for 变量1,变量2.. in 集合:

使用变量1和变量等

## 集合内涵

普通集合内涵

新集合 = {变量 for 变量 in 集合}

带条件的集合内涵

新集合 = {变量 for 变量 in 集合 if 条件}

多循环的集合内涵

新集合 = {变量1+变量2 for 变量1 in 集合1 for 变量2 in 集合2 if 条件}

## 集合函数

len() 获取集合中元素的个数

格式:len(集合)

返回值：整数

max() 获取集合中最大的元素

格式:max(集合)

返回值：集合中最大的值

min() 获取集合中最小的元组

格式：min(集合)

返回值：集合中的最小值

set() 将其他序列转化为集合类型

格式：set(序列)

返回值：集合类型

## 集合函数

add() 向集合中添加数据

格式： 集合.add(数据)

返回值：无

**注意**：直接改变原有集合，已存在的数据无法添加

clear() 清空集合

格式：集合.clear()

返回值：无

**注意**：直接改变原有集合

copy() 集合复制

格式:集合.copy()

返回值:原集合所有数据组成的新的集合

remove() 移除指定的值

格式:集合.remove(值)

返回值：无

**注意**：直接改变原有集合，如果要删除的值不存在，就报错

discard()移除集合中指定的值

格式:集合.discard(值)

返回值：无

**注意**：直接改变原有集合，如果要删除的值不存在，就不作任何操作

pop() 随机移除一个元素

格式：集合.pop（）

返回值：被移除的元素

**注意**：随机移除元素，没啥规律

## 冰冻集合/固定集合 frozen

冰冻集合就是不可以进行任何修改相关操作的集合

##### 创建冰冻集合

创建空的冰冻集合

变量 = frozenset()

创建具有数据的冰冻集合

变量 = fronzenset(一个序列)

##### 操作方法和函数

冰冻集合的操作相关函数和方法同集合类型一样。

只是不可以使用修改集合相关的方法和函数

**注意**

frozenset 和普通的集合一样没有什么本质区别，能够使用的函数数量不一样，仅此而已哦

# 五、 字典

字典也是一种组合数据，没有顺序的组合数据。数据以键值对的方式存在。

## 字典的定义

1.创建空字典

变量 = {} 或者 变量 = dict()

2.创建有数据的字典

1.变量 = {键:值,键:值,键:值...}

2.变量 = dict({键:值,键:值,键:值...})

3.变量 = dict(键=值,键=值,键=值...)

**注意**：该方法键的名称要符合变量的基本名规则

4.变量 = dict([(键,值),(键,值),（键,值）...])

5.变量 = dict(zip((键，键，键....),(值,值，值...)))

## 字典的特征

1.字典也是序列类型但是是无序序列，所以分片操作无法使用

2.字典中每个数据都由键和值两部分组成：

值：任意数据都可以当作值来使用

键：可以使用 int，float,str,tuple，冰冻集合，但是list，set，dict等不可以使用

## 字典的基本操作

访问数据：

字典变量[键]

添加数据

字典变量[新键] = 值

修改数据

字典变量[键] = 新值

删除数据

del 字典变量[键]

序列操作

数据 in 字典

检测数据是否在字典的键中

数据 not in 字典

检测数据是否不再字典的键中

## 遍历操作

1.键遍历操作

for 变量 in 字典:

使用变量(变量就是字典中的键)

2.键值遍历

for 键变量,值变量 in 字典.items()：

使用键变量和值变量

## 字典内涵

基本的字典内涵

变量= {key:value for key,value in 字典.items()}

带有条件的字典内涵

变量 = {key:value for key,value in 字典.items() if 条件}

## 字典函数

len() 计算字典的长度

格式： len(字典)

返回值:字典的长度

max() 获取字典中的最大值 （默认比较字典的键）

格式:max(字典)

返回值：最大的键

min() 获取字典中的最小值（默认比较字典的键）

格式：min(字典)

返回值:最小值

dict() 将其他序列转化为字典

详情请看字典的定义

\*str(字典) 返回字典的字符串格式\*

## 字典方法

clear() 清空字典

格式：字典.clear()

返回值：无 None

**注意**：直接改变原有字典

items() 获取字典的键值对组成的元组格式

格式：字典.items()

返回值：将键值对变成元组的列表形式。

**注意**:返回的哦数据是内建的dict\_items类型，专用于字典的转换中间类型

keys() 获取字典中所有键组成的序列，常用语字典遍历

格式： 字典.keys()

返回值:返回所有键的组合

**注意**：类型为 dict\_keys类型，也是一种字典转换的中间类型

values() 获取字典中所有值组成的序列，常用语字典遍历

格式： 字典.values()

返回值：所有值的组合

**注意**：类型为dict\_values类型，也是一种字典转换的中间类型

get（） 根据指定的键获取值

格式：字典.get(键,默认值)

返回值：字典键对应的数据

**注意**：如果键不存在与字典中，则返回默认值而不会报错

pop() 根据键从字典中删除数据

格式:字典.pop(键)

返回值：删除的键对应的值

**注意**：pop在字典中使用必须有参数，因为字典无序序列，直接改变原有字典

copy() 赋值一个字典

格式：字典.copy()

返回值：复制出来的新的字典

setdefault() 设置字典中不存在的键的值

格式：字典.setdefault(键,默认值)

返回值：设置时的默认值

**注意**：如果设置时键已经存在于字典中，则不进行任何操作，直接改变原有字典

fromkeys() 使用指定的序列作为键,使用一个值作为字典的所有值

格式:字典.fromkeys(序列,值)

返回值：由序列作为字典的键，由值作为所有键对应的值组成的字典

update() 使用字典更新另外一个字典

格式：字典1.update(字典2)

返回值：无

**注意**:直接改变原有字典

# 六、内建函数

## 数学相关内建函数

round() 对一个数值进行四舍五入操作

格式：round(数值)

返回值：整型

max() 获取一个数值序列中的最大值

格式:max(数值相关的序列)

返回值：序列中的最大值

min() 获取一个数值序列中的最小值

格式:min(数值相关的序列)

返回值：序列中的最小值

abs() 计算一个数值的绝对值

格式:abs(数值)

返回值： 数值的非负数数表现

# 七、文件操作

本章内容主要对文件的读写操作进行详细介绍。

在python中处理系统文件和目录相关的函数，存在于三大模块当中

1.内建函数可以操作文件的读写

2.os模块，操作目录相关，环境信息相关及路径处理相关的函数

3.shutil模块，高级文件处理模块，复制移动删除及文件归档等操作

赠送文件压缩模块

## 打开文件 open

open()

格式：open(文件的完整路径,打开模式)

返回值：文件io对象

打开模式：

w模式 写模式(write) 文件不存在则新建文件，文件存在则清空文件，并且打开文件

w模式打开文件，指针在最开始位置

r模式 读模式(read) 文件存在则打开文件，文件不存报错

r模式打开文件，指针在最开始位置

a模式 追加模式(append) 文件不存在则新建文件，文件存在则打开文件

a模式打开文件，指针在文件内容的末尾

x模式 抑或模式（xor） 文件存在就报错，文件不存在则新建文件

x模式打开文件，指针在文件最开始位置

b模式 二进制模式(binary) 使用二进制方式打开文件，辅助模式，不能单独使用

+模式 增强模式（plus） 为了让文件读写同时操作的模式

所有模式太多了。。自己慢慢组合

w，w+，wb，wb+，r，r+，rb，rb+，a，a+，ab,ab+,x,x+,xb,xb+

## 关闭文件 close

close()

文件io对象.close（）

写入操作 write

write() 将字符串写入文件当中

格式:文件io对象.write（字符串）

返回值：写入字符的长度

writelines() 将序列写入文件当中

格式:文件io对象.writelines(序列)

返回值：无

**注意**：writelines写入序列内容时不会在每一个数据后有任何额外的内容添加

## 读取操作 read

read() 读取指定的长度的内容或者全部内容

格式：文件io对象.read() 读取全部内容

返回值：读取的字符富川

格式：文件io对象.read(字符长度) 读取指定长度的内容

返回值：读取的字符串

readline() 一次读取一行信息

格式：文件io对象.readline（） 读取一行的所有信息

返回值：读取的字符串

格式：文件io对象.readline（长度） 读取一行的指定长度的信息

返回值：读取的字符串

**注意**：如果readline读取指定长度后，当前行没有读取完毕，下次readline还会在当前行没有读取的内容之后继续读取

**注意**：无论是read 还是readline函数 读取内容之后 指针都会移动到读取内容的最后位置

readlines() 将文件内容按照行读取到列表当中

格式：文件io对象.readlines()

返回值：由每一行内容组成的列表

**注意**：换行\n也会也当作字符读取到列表当中

格式：文件io对象.readlines(读取总长度)

返回值：读取的组成的列表

**注意**：如果指定了长度，则按照内容长度来进行行数的读取，不足一行当一行读取

## 文件截取操作truncate

格式：文件io对象.truncate(字节长度)

返回值:文件剩余内容的字节个数

**注意**:截取内容是需要根据文件的编码类型来决定截取字节的长度

## 文件指针操作

tell() 获取当前指针的位置

格式：文件io对象.tell()

返回值：整型 表示指针的位置

seek() 调整指针的位置

格式：文件io对象.seek(N) 将文件指针调整到开头之后的第N位

格式：文件io对象.seek(偏移位置，参照方式)

偏移位置使用整数即可

参照方式:0 表示以文件最开头作为参照 1表示以指针当前的位置作为参照

返回值：指针移动之后的位置 整数

**注意**：指针移动以字节作为单位，而不是字符！

## 文字相关内容

字符长度

文字的个数，一个汉字是一个字符，一个英文字母也是一个字符

字节：

字节是计算机存储数据的单位，一个英文确实是一个字节

其他文字具体是多少个字节村一个字符取决于文件的编码类型

UTF-8 3个字节表示一个汉字

GBK/GB2312/BIG5/ANSI 2个字节存储一个汉字

# 八、 系统相关的操作

与系统相关的操作主要存在于三个大的模块当中

1.os模块 操作系统目录相关等操作

2.os.path模块 系统路径相关的操作模块

3.shutil模块 高级文件操作模块 目录树的操作，文件的复制，删除，移动....

## OS模块

### 函数部分

OS模块是扩展模块，使用之前必须导入模块

import os

getcwd() 获取当前的工作目录

格式：os.getcwd()

返回值：当前工作目录的字符串

当前工作目录就是程序在进行文件相关操作，默认查找文件的目录

chdir() 改变当前的工作目录

格式：os.chdir（路径）

返回值：无

listdir() 获取一个目录中所有子目录和文件的名称列表

格式:os.listdir(路径)

返回值：所有子目录和文件名称的列表

mkdir() 创建一个目录

格式：os.mkdir（路径）

返回值:无

makedirs（） 递归创建文件夹

格式：os.makedirs(递归路径)

返回值：无

递归路径：多个文件夹层层包含的路径就是递归路径 例如 a/b/c...

rmdir() 移除一个空目录

格式：os.rmdir(路径)

返回值：无

**注意**：只能删除没有任何内容的空目录，删除之后内容不会进入回收站

removedirs（） 递归删除目录

格式:os.removedirs(递归目录)

返回值：无

**注意**：只能删除没有任何内容的空目录，删除之后内容不会进入回收站

rename() 修改文件夹名称

格式：os.rename(原文件夹路径，新文件夹路径)

返回值：无

stat() 获取文件相关信息

格式：os.stat(路径)

返回值：文件信息组成的元组

system() 运行系统shell命令

格式：os.system(系统命令)

返回值：打开一个shell或者终端界面

getenv() 获取指定的系统环境变量值

格式：os.getenv('环境变量名')

返回值：指定环境变量名对应的值

putenv() 设置或者添加系统环境变量值

格式：os.putenv(环境变量名,环境变量值)

返回值：无

exit() 推出当前程序

格式：exit()

返回值:无

### 值部分

os.curdir 获取当前路径

值：.

os.pardir 获取当前文件夹的上一级文件夹路径

值：..

os.sep 获取当前系统的路径分隔符

值： window \ linux /

os.linesep 获取当前系统的换行符号

值： window \r\n line \n

os.name 获取当前系统的名称

值： window nt mac posix linux 暂无

## os.path模块

abspath() 将路径转化为绝对路径

格式:os.path.abspath('路径')

返回值：路径的绝对路径形式

basename() 获取路径中的文件名部分

格式:os.path.basename(路径)

返回值：文件名字符串

dirname() 获取路径中的文件夹部分

格式:os.path.dirname(路径)

返回值：文件夹部分字符串

join() 将多个路径拼合成一个路径

格式：os.path.join(路径1，路径2....)

返回值：组合之后的新路径字符串

split() 将路径切割为文件夹部分和当前文件部分

格式:os.path.split（路径）

返回值：路径和文件名组成的元组

splitext() 将文件名切割成名称和后缀2部分

格式：os.path.splitext(文件名)

返回值：名称和后缀组成的元组

getsize() 获取文件的大小

格式:os.path.getsize(文件路径)

返回值：文件的字节大小

getctime() 文件创建时间 createtime

格式：os.path.getctime()

返回值：时间戳 浮点数

getmtime() 文件修改时间 modifytime

格式：os.path.getmtime()

返回值：时间戳 浮点数

getatime() 文件访问时间 activetime

格式：os.path.getatime()

返回值：时间戳 浮点数

isfile（） 检测是否是文件

格式：os.path.isfile（路径）

返回值：布尔值

isdir（） 检测是否是目录

格式：os.path.isdir(路径)

返回值：布尔值

exists() 检测文件或者目录是否存在

格式：os.path.exists(路径)

返回值:布尔值

isabs() 检测一个路径是否是绝对路径

格式：os.path.isabs（路径）

返回值：布尔值

islink() 检测是否是一个链接

格式：os.path.islink()

返回值：布尔值

samefile（） 检测两个路径是否指向同一个文件

格式：os.path.samefile(路径1,路径2)

返回值：布尔值

## 绝对路径和相对路径

绝对路径就是路径中带有参照位置的路径格式

相对路径以当前工作文件夹作为参照位置的路径格式

相对路径

./1.txt 表示在当前文件夹下查找使用1.txt文件

1.txt 表示在当前文件夹下查找使用1.txt文件

../1.txt 表示在当前文件夹的外层文件夹中查找使用1.txt文件

绝对路径

C:/abc/1.txt /user/work/。。。。

**shutil模块**

import shutil

copy() 复制文件

格式：shutil.copy(来源路径，目标路径)

返回值：返回目标路径

copy2() 复制文件，保留元数据（文件信息）

格式：shutil.copy2(来源路径，目标路径)

返回值：返回目标路径

**注意**：copy和copy2的唯一区别在于copy2复制文件时尽量保留元数据

copyfileobj() 将一个文件中的内容复制到另外一个文件当中

格式：shutil.copyfileobj(open('源路径','r'),open('目标路径','w'))

返回值：无

copyfile()将一个文件中的内容复制到另外一个文件当中

格式：shutil.copyfile（'源路径','目标路径')

返回值：无

**注意**：如果源路径和目标路径相同，会报错误！

copytree（） 直接复制整个目录树中的所有内容

格式：shutil.copytree（源文件夹，目标文件夹）

返回值：返回目标路径

copymode() 复制权限 仅供参考

copystat() 复制状态 仅供参考

rmtree() 删除整个目录树，即使非空目录也可以删除

格式：shutil.rmtree(路径)

返回值：无

move() 移动文件/文件夹

格式：shutil.move(源路径，目标路径)

返回值：目标路径！

which() 获取可执行程序的位置

格式：shutil.which('命令字符串')

返回值:命令程序位置的路径字符串

disk\_usage() 获取磁盘使用情况

格式：shutil.usage('盘符')

返回值：元组（总计,使用，剩余）

## 归档和压缩

归档：将多个文件或者文件夹合并到一个文件当中

减少文件数目，方便发送邮件等操作

压缩：使用算法将多个文件或者文件夹无损或者有损的合并到同一个文件当中

减少磁盘占用空间，也方便发送邮件等操作。

make\_archive() 归档操作

格式:shutil.make\_archive('归档之后的目录和文件名','后缀','需要归档的文件夹')

返回值：归档之后的地址

unpack\_archive() 解包操作

格式：shutil.unpack\_archive('归档文件地址','解包之后的地址')

返回值：解包之后的地址

get\_archive\_formats() 获取系统已注册的归档格式和信息

格式：shutil.archive\_formats()

返回值：列表

get\_unpack\_foramts() 获取系统已经注册的解包格式和信息

格式：shutil.unpack\_formats()

返回值：列表

## zipfile zip文件操作

import zipfile

zip文件格式是通用的文档压缩标准，在ziplib模块中，使用ZipFile类来操作zip文件，下面具体介绍一下：

zipfile.ZipFile(file[, mode[, compression[, allowZip64]]])

创建一个ZipFile对象，表示一个zip文件。参数file表示文件的路径或类文件对象(file-like object)；参数mode指示打开zip文件的模式，默认值为’r’，表示读已经存在的zip文件，也可以为’w’或’a’，’w’表示新建一个zip文档或覆盖一个已经存在的zip文档，’a’表示将数据附加到一个现存的zip文档中。参数compression表示在写zip文档时使用的压缩方法，它的值可以是zipfile. ZIP\_STORED 或zipfile. ZIP\_DEFLATED。如果要操作的zip文件大小超过2G，应该将allowZip64设置为True。

ZipFile还提供了如下常用的方法和属性：

ZipFile.getinfo(name):

获取zip文档内指定文件的信息。返回一个zipfile.ZipInfo对象，它包括文件的详细信息。将在下面 具体介绍该对象。

ZipFile.infolist()

获取zip文档内所有文件的信息，返回一个zipfile.ZipInfo的列表。

ZipFile.namelist()

获取zip文档内所有文件的名称列表。

ZipFile.extract(member[, path[, pwd]])

将zip文档内的指定文件解压到当前目录。参数member指定要解压的文件名称或对应的ZipInfo对象；参数path指定了解析文件保存的文件夹；参数pwd为解压密码。下面一个例子将保存在程序根目录下的txt.zip内的所有文件解压到D:/Work目录：

Python

import zipfile, os

zipFile = zipfile.ZipFile(os.path.join(os.getcwd(), 'txt.zip'))

for file in zipFile.namelist():

zipFile.extract(file, r'd:/Work')

zipFile.close()

import zipfile, os

zipFile = zipfile.ZipFile(os.path.join(os.getcwd(), 'txt.zip'))

for file in zipFile.namelist():

zipFile.extract(file, r'd:/Work')

zipFile.close()

ZipFile.extractall([path[, members[, pwd]]])

解压zip文档中的所有文件到当前目录。参数members的默认值为zip文档内的所有文件名称列表，也可以自己设置，选择要解压的文件名称。

ZipFile.printdir()

将zip文档内的信息打印到控制台上。

ZipFile.setpassword(pwd)

设置zip文档的密码。

ZipFile.read(name[, pwd])

获取zip文档内指定文件的二进制数据。下面的例子演示了read()的使用，zip文档内包括一个txt.txt的文本文件，使用read()方法读取其二进制数据，然后保存到D:/txt.txt。

Python

import zipfile, os

zipFile = zipfile.ZipFile(os.path.join(os.getcwd(), 'txt.zip'))

data = zipFile.read('txt.txt')

(lambda f, d: (f.write(d), f.close()))(open(r'd:/txt.txt', 'wb'), data) 一行语句就完成了写文件操作。仔细琢磨哦~\_~

zipFile.close()

import zipfile, os

zipFile = zipfile.ZipFile(os.path.join(os.getcwd(), 'txt.zip'))

data = zipFile.read('txt.txt')

(lambda f, d: (f.write(d), f.close()))(open(r'd:/txt.txt', 'wb'), data) 一行语句就完成了写文件操作。仔细琢磨哦~\_~

zipFile.close()

ZipFile.write(filename[, arcname[, compress\_type]])

将指定文件添加到zip文档中。filename为文件路径，arcname为添加到zip文档之后保存的名称, 参数compress\_type表示压缩方法，它的值可以是zipfile. ZIP\_STORED 或zipfile. ZIP\_DEFLATED。下面的例子演示了如何创建一个zip文档，并将文件D:/test.doc添加到压缩文档中。

ZipFile.writestr(zinfo\_or\_arcname, bytes)

writestr()支持将二进制数据直接写入到压缩文档。

ZipFile.getinfo(name) 方法返回的是一个ZipInfo对象，表示zip文档中相应文件的信息。它支持如下属性：

ZipInfo.filename： 获取文件名称。

ZipInfo.date\_time： 获取文件最后修改时间。返回一个包含6个元素的元组：(年, 月, 日, 时, 分, 秒)

ZipInfo.compress\_type： 压缩类型。

ZipInfo.comment： 文档说明。

ZipInfo.extr： 扩展项数据。

ZipInfo.create\_system： 获取创建该zip文档的系统。

ZipInfo.create\_version： 获取 创建zip文档的PKZIP版本。

ZipInfo.extract\_version： 获取 解压zip文档所需的PKZIP版本。

ZipInfo.reserved： 预留字段，当前实现总是返回0。

ZipInfo.flag\_bits： zip标志位。

ZipInfo.volume： 文件头的卷标。

ZipInfo.internal\_attr： 内部属性。

ZipInfo.external\_attr： 外部属性。

ZipInfo.header\_offset： 文件头偏移位。

ZipInfo.CRC： 未压缩文件的CRC-32。

ZipInfo.compress\_size： 获取压缩后的大小。

ZipInfo.file\_size： 获取未压缩的文件大小。

# 九、 calendar日历模块

在python中与日期时间相关的模块主要有4个：

calendar 日历模块

time 时间模块

datetime 日期时间模块（综合模块）

timeit 时间检测模块

## 日历模块

calendar() 获取一年的日历字符串

格式：calendar.calendar(年份,w=2,l=1,c=6)

返回值：整年的日历字符串

参数： 年份:整数

w = 每个日期之间的间隔字符数

l = 每周所占用的函数

c = 每个月之间的间隔字符数

isleap() 检测是否是闰年

格式：calendar.isleap(年份)

返回值：布尔值

leapdays() 获取指定年份之间闰年的个数

格式:calendar.leapdays(开始年份，结束年份)

返回值：整数

month（） 获取某个月的日历字符串

格式:calendar.month(年，月)

返回值：月日历的字符串

monthrange（） 获取一个月的周几开始即总天数

格式：calendar.monthrange(年,月)

返回值：元组(周几开始,总天数)

注意：周默认 0 -6 表示周一到周天

monthcalendar() 返回一个月每天的矩阵列表

格式：calendar.monthcalendar(年，月)

返回值：二级列表

注意：矩阵中没有天数用0表示

prcal() 直接打印整年的日历

格式：calendar.prcal(年份)

返回值：无

prmonth() 直接打印整个月的日历

格式：calendar.prmonth(年，月)

返回值：无

firstweekday() 获取每周开始的天数

格式：calendar.firstweekday()

返回值： 0 - 6 默认数值，在不改变的情况

setfirstweekday() 设置每周开始的天数

格式：calendar.setfirstweekday(开始值)

返回值：无

weekday() 获取周几

格式:calendar.weekday(年，月，日)

返回值:周几对应的数字

timegm（）获取一个指定时间的时间戳

格式：calendar.timegm(时间元组)

返回值：时间戳

## time 模块

## 时间戳

时间戳是值使用数值来表示一个时间的方式，数值可能是整型可能是浮点型，根据不同的语言不太一样。

1.时间戳是指从1970年1月1日0时0分0秒到指定时间的秒数(毫秒数)。

2.1970年以前的时间或者太遥远的未来使用时间戳可能出现异常。

3.32位操作系统能够支持到2038年的某一天

## UTC时间

UTC又称为世界协调时间，以英国的格林尼治天文所在地区的时间作为参考的时间，也叫做世界标准时间。

中国时间是 UTC+8 东八区（地理老师）

## DST夏令时

夏令时就是在夏天的时候将时间调快一小时，本意是督促大家早睡早起节省蜡烛！ 每天变成25个小时，本质没变还是24小时

## 时间元组 struct\_time

时间元组只是一个包含时间的内容的普通元组而已，别害怕。

（年,月，日，时，分，秒，周几，一年中的第几天,是否是夏令时）

索引 内容 属性 值

0 年 tm\_year 2015

1 月 tm\_mon 1～12

2 日 tm\_mday 1～31

3 时 tm\_hour 0～23

4 分 tm\_min 0～59

5 秒 tm\_sec 0～61 60表示闰秒 61保留值

6 周几 tm\_wday 0～6

7 第几天 tm\_yday 1～366

8 夏令时 tm\_isdst 0，1，-1（表示夏令时）

## 时间模块中的值

timezone 获取当前时区与协调时间之间的间隔秒数

东边是复数,东八区 -28800

## 时间模块属性

timezone 获取当前时区与UTC时间相差的秒数，在没有夏令时的情况下

altzone 获取当前时区与UTC时间相差的秒数，在有夏令时的情况下，与timezone相差3600秒

daylight 检测当前是否是夏令时时间状态

0 表示是 其他值 不是

## 时间模块方法

asctime() 返回元组的正常字符串化之后的时间格式

格式：time.asctime（时间元组）

返回值:字符串 Tue Jun 6 11:11:00 2017

localtime() 获取指定时间的时间元组格式

格式：time.localtime()

返回值：时间元组

格式：time.localtime(时间戳)

返回值：时间元组

gmtime() 获取指定时间的时间元组格式（UTC时间）

格式：time.gmtime()

返回值：时间元组

格式：time.gmtime(时间戳)

返回值：时间元组

ctime() 相当于 asctime（localtime(时间戳)）

格式：time.ctime([时间戳])

返回值：字符串 Tue Jun 6 11:11:00 2017

mktime() 使用时间元组获取对应的时间戳

格式：time.mktime（时间元组）

返回值：浮点数时间戳

clock() 获取cpu时间，用于计时操作，3.0 - 3.3版本之前使用

格式：time.clock()

返回值:浮点数时间点

使用时通常程序开始获取一次，结束获取一次，然后求差值即可

perf\_counter() 获取cpu时间，用于计时操作，3.3之后版本之前使用

格式：time.perf\_counter()

返回值:浮点数时间点

使用时通常程序开始获取一次，结束获取一次，然后求差值即可

sleep（）程序睡眠，程序等待

格式:time.sleep(秒数)

返回值：无

time（） 获取当前的时间戳

格式：time.time（）

返回值：时间戳

strftime() 将时间元组转化为自定义的字符串格式

格式：time.strftime(格式化字符串[,时间元组])

返回值：根据格式化字符串得到的时间字符串

格式 含义 备注

%a 本地（locale）简化星期名称

%A 本地完整星期名称

%b 本地简化月份名称

%B 本地完整月份名称

%c 本地相应的日期和时间表示

%d 一个月中的第几天（01 - 31）

%H 一天中的第几个小时（24 小时制，00 - 23）

%I 一天中的第几个小时（12 小时制，01 - 12）

%j 一年中的第几天（001 - 366）

%m 月份（01 - 12）

%M 分钟数（00 - 59）

%p 本地 am 或者 pm 的相应符 注1

%S 秒（01 - 61） 注2

%U 一年中的星期数（00 - 53 星期天是一个星期的开始）第一个星期天之前的所有天数都放在第 0 周 注3

%w 一个星期中的第几天（0 - 6，0 是星期天） 注3

%W 和 %U 基本相同，不同的是 %W 以星期一为一个星期的开始

%x 本地相应日期

%X 本地相应时间

%y 去掉世纪的年份（00 - 99）

%Y 完整的年份

%z 用 +HHMM 或 -HHMM 表示距离格林威治的时区偏移（H 代表十进制的小时数，M 代表十进制的分钟数）

%% %号本身

strptime() 将自定义的字符串格式转化为时间元组 strftime的逆向操作

格式：time.strptime('时间字符串','格式化字符串')

返回值：时间元组