python 学习笔记

列表

在末尾添加元素：list.append('阿比')

插入元素：list.insert(0,'阿比)

获取正向索引：list.index('阿比')

通过索引移除元素：list.pop(1)

通过元素名称移除索引：list.remove('阿比')

统计元素出现次数：list.count('阿比')

排序：list.sort()

反转：list.reverse()

复制：list.copy()

清空：list.clear()

计算列表中所有数的和：sum(list)

分片用冒号分隔的索引位置表示，格式为 列表[起始索引：结束索引]

如获取list中第三到第五个元素，就是list[2:5]

分片支持反向索引

列表的加法用 列表+列表 组成一个新列表

列表*2 即 列表+列表

元组

元组是用()框起来的，本质上列表中的元素可改，元组中的元素不可'增删改'

一个微妙的地方：只有一个元素的元组在格式上和列表不同。

仅一个元素x的列表写成[x],但元组中则需多加一个逗号(x,)

元组和列表的相互转换

list()转换为列表，tuple()转换为元组

zip()函数将两个长度相同的列表合并起来，相同位置的元素会被一一组队，变成一个元组。

```
list1=['阿比','伊莉雅']
list2=['大老婆','二老婆']
zipped=zip(list1,list2)
print(list(zipped))
#输出：[('阿比', '大老婆'), ('伊莉雅', '二老婆')]
```

enumerate()将列表中的元素数出来，返回的是一个枚举对象(元组)。

```
a=['伊莉雅','阿比']
print(list(enumerate(a)))
#输出：[(0, '伊莉雅'), (1, '阿比')]
#可以手动输入起始数字
print(list(enumerate(a,1)))
#输出：[(1, '伊莉雅'), (2, '阿比')]
```

range()函数快速生成一个有规律的数字序列。基本格式为range(start,stop,step)。返回range对象

字符串

字符串也一样有索引，也可进行分片操作

字符串中的元素不可改变

```
name='阿比'
name[0]='伊'     #错误
new_name='伊'+name[1:]    #正确
#输出：伊比
```

小写字母转大写：upper()

大写字母转小写：lower()

字符串第一个字母大写，其余变小写 ： capitalize()

单词首字母大写：title()

大小写转换：swapcase()

split()通过指定分隔符对字符串进行分割，有两个可选参数。第一个为分隔符，默认为空格，换行的等，第二个为最大分割次数，默认为-1

join()将序列中的元素以指定的字符连接生成一个新的字符串。join()方法接收一个 序列作为参数，.前面的字符串用于连接符

```
'-'.join(['伊莉雅','老婆'])
#输出：伊莉雅-老婆
```

strip()用于移除字符串开头和结尾指定的字符。当传入一个字符串时，会将传入字符串中每个字符依次移除。

```
'_~_伊莉雅老婆~~_'.strip('~_')
#输出：伊莉雅老婆
```

count()用于统计字符串中某个字符串出现的次数，第一个参数为要统计次数的字符串，后两个可选参数为搜索的开始和结束索引。

```
'aabbccdd'.count('a')
#输出：2
```

find()用于检测字符串中是否包含子字符串，如果包含则返回第一次出现的索引值，第一个参数为要查找的子字符串，后两个为开始和结束索引

```
'伊莉雅老婆'.find('老婆')
#输出：3
```

replace()用于将字符串中的指定字符串替换成别的字符串，第一个参数为被替换字符串，第二个参数为替换字符串，第三个参数为替换的最大次数，默认为无限次

```
'伊莉雅老婆'.replace('老婆','亲亲')
#输出：伊莉雅亲亲
```

字典

```
harem={'伊莉雅':'大老婆','阿比':'二老婆'}
#伊莉雅是键，大老婆是值
#键只能唯一，值可以重复
```

```
#修改
harem['伊莉雅']='最可爱的宝贝'
#输出：{'伊莉雅':'最可爱的宝贝','阿比':'二老婆'}

#添加
harem['智乃']='三老婆'
#输出：{'伊莉雅':'大老婆','阿比':'二老婆','智乃':'三老婆'}

#删除
del harem['阿比']
#输出：{'伊莉雅':'大老婆'}
```

我们可以通过 字典[键] 的方式获取键对应的值，而当键不存在的时候会报错

python中有一种更安全的查询方法 get()

```
#接续上面的harem
harem.get('伊莉雅')
#输出：大老婆
```

key()方法可以获取字典中所有的键

```
names=harem.keys()
#输出：dict_keys(['伊莉雅', '阿比'])
```

keys() 方法返回的是一个特殊类型 dict_keys。

当我们将 students.keys() 的结果保存到变量 names 后，可以通过成员运算符 in 判断某个元素在不在里面，也可以使用 for 循环遍历它

但它是不能被索引的。如果想通过索引访问元素，需要先用 list() 函数将它转换成列表类型，再进行访问。

与keys()相对应，我们可以通过values()方法获取字典中所有的值

```
harem.values()
#输出：dict_values(['大老婆', '二老婆'])
```

除了获取所有的键，值，我们也可以通过items()方法，一次性获取字典中所有的键值对，每一个键值对都是形如(dict_values(键，值)的元组。

```
harem.items()
#输出：dict_items([('伊莉雅', '大老婆'), ('阿比', '二老婆')])
```

键值组成的元组遍历方法

```
for name,status in harem.items():
    print(name,status)
#输出：伊莉雅 大老婆
#      阿比 二老婆
```

类与面向对象

基本概念

```
class Harem:        #类的创建
    wife_1='阿比'
    wife_2='智乃'    #类的属性

    #进行类的实例化时，初始化方法会自动被执行
    def __init__(self):
        print('初始化方法')
    #直接打印实例的结果是__str__的返回值
    def __str__(self):
        return '老婆是{}'.format(wife_1)
    def daily_work(self):    #实例方法的创建
        print('亲亲')
    def kiss(self):
        print('亲亲{}'.format(self.wife_1))    #类的方法里访问类的变量


#类的实例化
harem=Harem()

print(harem)
#输出：老婆是阿比
print(harem.wife_1)
#输出：阿比
```

类的继承

1. **单一继承**

```
class Harem_part_one(Harem):
    pass

harem = Harem_part_one()
print(harem)
#输出：老婆是阿比
```

我们可以看到，Harem_part_one 类中什么都没有，实例化出来的 harem 却拥有 Harem 类的属性和方法。我们通过 class 子类名(父类名) 的方式实现了类的继承，让子类拥有了父类的所有属性和方法

2. **多重继承**

```

```

我们可以看到，ABC 类继承了 Chinese 类和 American 类的所有属性和方法，但 hair 和 skin 这两个类都有的属性，ABC 类只继承了 Chinese 类的。

这是因为 class ABC(Chinese, American): 中 Chinese 靠前，调用实例的属性和方法时会优先在子类中查找，找不到再从最左侧的父类依次往右查找，直到找到为止，如果都没有找到则会报错。

进阶

defaultdict

当我使用普通的字典时，用法一般是dict={},添加元素的只需要dict[element] =value即，调用的时候也是如此，dict[element] = xxx,但前提是element字典里，如果不在字典里就会报错这时defaultdict就能排上用场了，defaultdict的作用是在于，当字典里的key不存在但被查找时，返回的不是keyError而是一个默认值

```
from collections import defaultdict
dict =defaultdict( factory_function)
```

这个factory_function可以是list、set、str等等，作用是当key不存在时，返回的是工厂函数的默认值，比如list对应[ ]，str对应的是空字符串，set对应set( )，int对应0

lambda函数

是指一类无需定义标识符（函数名）的函数或子程序。
lambda 函数可以接收任意多个参数 (包括可选参数) 并且返回单个表达式的值。

```
#第一种用法  将lambda函数赋值给一个变量，通过这个变量间接调用该lambda函数。
#不使用lambda函数
def sum(x,y):
    return x+y
print(sum(1,2))

#使用lambda函数
sum = lambda x,y : x+y
print(sum(1,2))

#第二种用法  将lambda函数作为参数传递给其他函数。部分Python内置函数接收函数作为参数。
#不使用lambda函数
def odd(x):
    return x%2
temp = range(10)
show = filter(odd,temp)
print(list(show))   #[1, 3, 5, 7, 9]

#使用lambda函数
print(list(filter(lambda x: x%2,range(10))))    #[1, 3, 5, 7, 9]
```