Python 基本语法学习

List(列表)

- 列表存储有序的数据，可以包含不同类型的元素，可以用不同函数对列表做出更改

查询:用[index]查询，索引值index为元素序数减一，也可以用[2::1]查询范围元素，意为从第三项开始到最后一项，索引值差值为1，由此可得，倒序把1改为-1即可

变更函数

append()添加新元素

extend()只能添加可迭代元素

remove()填具体元素去除

pop()填元素索引值去除

insert()填要插入的位置索引值及要添加的元素

index()查询元素的索引值

clear()清除

reverse()倒序

sort()从小到大排序

count()求索同字符出现次数

copy()浅拷贝列表

copy.deepcopy(list)深拷贝列表，适用嵌套列表

列表推导式: new_list = [最终输出形式 for...in...if...]

- **例**

```
[[x,y] for x in range(10) if x%2 == 0 for y in range(10) if y%3 == 0]
#先执行for循环再条件判断筛选，最后输出
```

Dictionary(字典)

- 字典存储无序数据，每个元素由键key和值value组成，key一般要以字符串形式，value无限制

get(key，default=0)填入key,第二个参数填默认，如果无法查询到value就将查询物赋值为这个

update({key: value,})如果键已存在，即覆盖value，不存在即添加新的键值对

pop(key，default=0)去除key对应的value

keys()查询键 values()查询值 items()按顺序查询键值对

- **嵌套字典如**

```
stu1 = {"Chinese": 86, "Math": 90}
stu2 = {"Chinese": 96, "Math": 78}
stu3 = {"Chinese": 89, "Math": 95}
s = {"小明" : stu1, "小李" : stu2, "小红" : stu3}
```

Lambda 匿名函数

- 匿名函数lambda通常作为函数参数嵌套(高阶函数)，匿名函数不需要函数名但有变量，冒号后面紧跟返回值

- **例1**

```
def map_list(l,f):
    a = []
    for item in l:
        i = f(item)
        a.append(i)
    return a
l = [1,2,3,4]

l = map_list(l,lambda x: x ** 2)
print(l)
```

- **例2**

```
def fun(a):
    return len(a)
l = ['weadad','lsd','sdusa','sddwwwww']
l.sort(key=fun)
```

```
#用普通函数首先执行fun函数再排序，key=后面一定要加一个函数
print(l)

l = ['weadad','lsd','sdusa','sddwwwww']
l.sort(key=lambda a: len(a))
#用匿名函数，不用再定义 不会将函数内容直接暴露，更安全
print(l)
```

Decorate(装饰器)

- 装饰器即一个补充原有函数功能的函数，且无需修改原函数代码，而是在外部补充(在原函数上加@新函数名)

```
def decorate(func):
    def supplement(*args,**kwargs):
    #前面创建的新函数只是一个外壳，具体执行还需要设新函数
    #此处输入原函数前补充
        value = func(*args,**kwargs)
        #多加一个value变量并赋值，目的为了保持原函数特性，再次保存原函数中的返回值
100
        #此处输入原函数后补充功能
        return value  #保存原函数功能和返回值
    return supplement #保存补充函数功能和返回值
```

类（Class）与魔法方法（Magic Methods）

- 类定义对象的属性和方法，子类可以继承父类的self属性，不继承方法,用类和对象组织代码就是OOP面向对象编程思想

```
class Animal():
#创建父类，init后面跟着基本属性，接下来定义的函数即额外功能
    def __init__(self,name):
    #__init__这类双下划线的函数就是魔法方法
        self.name = name
    def greet(self):
        print(f'Hello! I am {self.name}.')

class Dog(Animal):
#子类继承父类的基本属性，额外的属性还得重新定
    def greet(self):
        print(f'wangwang...I am {self.name}.')

class Cat(Animal):
#在创建新文件时可以直接用from 文件名 import 类名称
    def greet(self):
        print(f'miaomiao...I am {self.name}.')
```

```
cat1 = Cat('Lihang')
#创建类中的单个对象，此处要给定参数以为后续使用函数提供便利
cat1.greet() #在类里面定义的函数，参数已经给定，在外部使用函数不用给额外参数
```

re 正则表达式

- 正则表达式即区别于find的精准寻找的模糊查询，通过元字符构成筛选条件。在大型字符串中筛选出目标字符,格式一般为，先import re,x = re.findall(pattern,大型字符串变量名,re.S取消.对\n的孤立)

原字符: 通配符即. 可以匹配任意字符除了\n，，，[]为字符集，相当于在a和e之间有且仅有一个a-z或者A-Z或者0-9

- **例3**

```
res = re.findall("a[a-zA-Z0-9]e",s)
```

重复符: {},+,*,?

- **例4**

```
res = re.findall("a.{1,3}e",s)
#代表左边符号重复1到3次,默认从大数开始贪婪匹配 左边符号可以替换成字符集，缩小筛选范围
```

- **例5**

```
res = re.findall("https?://www.[a-z]*?.com",a)
#*相当于是{0,} +相当于是{1,} ?相当于{0,1},有些是http有些多s所以格式里面把s重复次数调节一下,后面的问号即取消贪婪匹配，从小数开始匹配
```

边界符: ^,$

- **例6**

```
path = "https/aaa/vvv/ffd/Lisiting/blog/2021/3/fff/zz/com"
```

```
reg = "^/Lisiting/blog/[0-9]{4}/[0-9]{1,2}/$"
#^,$划清起始边界必须是L ，，结束边界必须为数字，不然就匹配不到，所以无论path前缀后缀有多少还是只筛选我们所要的那一部分
```

转义符: \d相当于[0-9],,,,,\w相当于[a-zA-Z0-9],,,,\b为单词边界符，即字母数字下划线组成的连续字符的边界符，区别于^$的作用于整条字符串

- **例7**

```
s = "The cat on the caterpillar is cute bcat"

res = re.findall(r"\bcat\b",s)#要用raw转义
#这样确保caterpillar不会被筛选
```

分组及管道符: ()为优先提取小括号也可用于分组，比如把单词括起来强调整体，但是别忘了在括号里面最前面加上?:取消强调意味,，，，|为管道符。是或者的意思。一般要与分组括号配合使用，同样要取消强调意味

Generator 生成器（yield 关键字）

- generator即通过类似for循环的next()生成一个个东西，但是并不显示出来，不占用内存，速度更快，相当于直接塞一百万个汉堡在你家和给你一个做汉堡机器按需做一百万个的区别

- **例8**

```
nums = [1,2,3,4,5]
squre_nums = (n*2 for n in nums)
#将列表推导式的方括号改为圆括号即可
```

- yield关键字可以类比return，用于定义函数中，都是保存值。但y还可以自动生成一个新列表并把保存值填入,跟普通创建列表再append的区别在于用generator的直接print的话，不会显示出来列表，而是需要next或for循环去触发

- **例9**

```
yi = 100000000

def gene_nums(nums_list):

    for n in nums_list:
```

```
        if n%3 == 0:
            yield 3*yi
        elif n%5 == 0:
            yield 5*yi
        else:
            yield n*yi
nums_list = list(range(100))
#快捷生成从0到100的列表

gene_list = gene_nums(nums_list)
#用前面定义的关于yield的函数功能，赋值gene_list为一个列表

print(next(gene_list))

#以yield形式保存的元素只能通过next求索出来，for循环其实就是next函数的外包
```

Type Hint 类型注释

- 用于指定变量、函数参数和返回值的预期类型，提高代码可读性和可维护性，一般在变量后面加个冒号并批注变量类型，有int，str，float……