2021/6/8

笔记本:	reptile_draft		
创建时间:	2021/6/8 10:14	**更新时间:**	2021/6/9 0:44
作者:	134exetj717		
URL:	about:blank		

```
print('-'*30)            #表示打印30次'-'
```


```
one = 'I\'m cejay'      # 字符 \ 表示转义字符，意思是把 ' 在字符串中打印出来
two = ",hello,I'm lucy,youknow"    #若 ' 在“”中，那么自动表示 ' 为字符串；若 “
在''中，那么自动表示 " 为字符串。
all = '''
        nice to meet you,
        I’m so glad.
      '''
print(one+two+all)
```

字符串

1.不可变类型，可迭代对象，有序（可索引、切片）
2. 用引号引起来表示（见 引号）

3.内部 \` 表示转义，引号前面加 `r` 禁用转义

注释

1.只有`#`后面的真的是注释，不会被解释器运行 1.# 与内容之间至少一个空格
2.注释一般放在被注释代码的上面

3.如果注释用在同行代码结尾，#与代码之间至少空2个空格

2.三引号（三对单/双引号）也被用来当做多行注释（见 引号3.）

引号

1.单双引号都可以用来表示字符串，一般没区别，按需使用。

1.json模块被反序列化的字符串 内部的引号必须全是双引号

1.除了是数据内容一部分的单引号

2.shell变量的值中如果包含空格需要用双引号引起来

1.例：使用os模块时，表示windows路径的字符串中有空格时，这个字符串必须先用 双引号引起来 再用 单引号或三单引号引起来

2.三引号（三对单/双引号）表示保留原格式的字符串，或者当字符串里同时有单双引号时使用

3.三引号也被用来当做多行注释

1.有人用三引号来当普通注释，但其本质还是字符串，解释器不会真的把它当注释，会被运行，所以不建议这样使用。

2.用在模块/类/函数/方法的开头表示说明，会自动赋值给 xx.__doc__

1.如果没有的话，xx.__doc__是None

1.help(o)第一部分就是o.__doc__的内容，如果没有，就到定义句前面去找#格式的注释，如果也没有，就是None。

ps：python中一对单引号,一对双引号,三个单双引号的区别和用法

首先说明,在python中三个单双引号并不是真正的注释

1 2 3 4	>>> type("""abcde""") <class 'str'> >>> type('''abcd''') <class 'str'>

字符串的特殊使用：

```
str = 'chengdu'
print(str[1:7:2]*3)   #表示从 1 开始，到 7 结束，（其中 7 不包含在内），步进值为2， *3 表示打印三次
```

3.2 列表

Python的核心数据类型

List（列表）

- 列表可以完成大多数集合类的数据结构实现。列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）。
- 列表是写在方括号 [] 之间、用逗号分隔开的元素列表。
- 列表索引值以 0 为开始值，-1 为从末尾的开始位置。
- 列表可以使用+操作符进行拼接，使用*表示重复。

```
>>> list = ['abcd', 786 , 2.23, 'runoob', 70.2]
>>> print(list[1:3])
[786, 2.23]
>>> tinylist = [123, 'runoob']
>>> print(list + tinylist)
['abcd', 786, 2.23, 'runoob', 70.2, 123, 'runoob']
```

----------操作名称----------	操作方法	举例
访问列表中的元素	通过下标直接访问	print(list1[0])
列表的切片	使用[: :]	list1[2:5:2]
遍历列表	通过for循环	for i in list1: print(i)
【增】新增数据到列表尾部	使用append	list1.append(5)
【增】列表的追加	使用extend方法	list1.extend(list2)
【增】列表数据插入	insert方法	list1.insert(1, 3)
【删】列表的删除	del：我们通过索引删除指定位置的元素。 remove：移除列表中指定值的第一个匹配值。如果没找到的话，会抛异常。	del list1[0] list1.remove(1) 注意两种方法的区别
【删】弹出列表尾部元素	使用pop	list1.pop()
【改】更新列表中的数据	通过下标原地修改	list1[0] = 8
【查】列表成员关系	in 、not in	2 in list1
列表的加法操作	+	list3 = list1 + list2
【排】列表的排序	sort方法	list1.sort()
【排】列表的反转	reverse	list1.reverse()

```
a = ['a',123]
b = ['b',456]
a.append(b)      #表示把 b 以一个字符串的形式在末尾进行插入
print(a)
a.extend(b)      #表示把 b 以列表的形式，将其元素一个一个在末尾插入
print(a)
b.insert(1,'c') # 1 表示索引，后面的'c'表示对象或者内容---->将对象插入放在下表为 1 的地方
```

```
print(b)
```

```
['a', 123, ['b', 456]]
['a', 123, ['b', 456], 'b', 456]
['b', 'c', 456]

Process finished with exit code 0
```

```
a = ['a', 'b', 'c', 'a', 'd']
print(a.index('a', 1, 4))       #可以查找范围 1-4 中出现的第一个 'a' 的下标
#print(a.index('a', 1, 3))       范围区间左闭右开 [1,3)
                                 #找不到会报错
print(a.count('a'))         #统计出现的元素次数
```

```
3
2
```

```
import random
str = [[],[],[]]
names = ['a','b','c','d','e','f','g','h']
for name in names:
    index = random.randint(0,2)     # [0,2]
    str[index].append(name)
print(str)
```

```
[['a', 'f', 'g'], ['b', 'c', 'd'], ['e', 'h']]
```

```
#使用枚举函数，同时拿到列表中的下表和元素内容
mylist = ['a','b','c','d']
for i,x in enumerate(mylist):
    print(i+1,x)
```

```
1 a
2 b
3 c
4 d
```

3.3 元组

Python的核心数据类型

- **Tuple（元组）**
  - tuple与list类似，不同之处在于tuple的元素不能修改。tuple写在小括号里，元素之间用逗号隔开。
  - 元组的元素不可变，但可以包含可变对象，如list。

⚠ 注意：定义一个只有1个元素的tuple，必须加逗号。

```
>>> t = ('abcd', 786 , 2.23, 'runoob', 70.2)
>>> t1 = (1, )
>>> t2 = ('a', 'b', ['A', 'B'])
>>> t[2][0] = 'X'
>>> t
('a', 'b', ['X', 'B'])
```

3.4 字典

Python的核心数据类型

- **dict（字典）**
  - 字典是无序的对象集合，使用键-值（key-value）存储，具有极快的查找速度。
  - 键(key)必须使用不可变类型。
  - 同一个字典中，键(key)必须是唯一的。

```
>>> d = {'Michael': 95, 'Bob': 75, 'Tracy': 85}
>>> d['Michael']
95
```

```
#字典的定义
info ={'name':'吴彦祖','age':18}
#字典的访问
print(info['name'])
print(info['age'])
#访问了不存在的键
#print(info['gender'])        #直接访问会报错
print(info.get('gender'))        #使用get方法，没找到相应的键，默认返回：none
print(info.get('gender','没有找到该数据'))      #没找到时，可以设定默认值
#删
del info['name']
print('删除后：%s'%info.get('name','没有找到name'))
#清空
info.clear()
print('清空后：',info)
#增
str = '男'
info['sex']=str
print('新增后：%s'%info['sex'])
#改
info['sex']='女'
print('修改后：',info['sex'])
#查
print(info.keys())       #得到所有的键（列表）
print(info.values())     #得到所有的值
```

```
print(info.items())      #得到所有的项（列表），每个键值对应的是一个元组
#遍历所有的键
for key in info.keys():
    print(key)
#遍历所有的键值对
for key,value in info.items():
    print('key=%s,value=%s'%(key,value))
```

```
吴彦祖
18
None
没有找到该数据
删除后：没有找到name
清空后：  {}
新增后：男
修改后：  女
dict_keys(['sex'])
dict_values(['女'])
dict_items([('sex', '女')])
sex
key=sex,value=女
```

3.5 集合

Python的核心数据类型

- **set（集合）**
  - set和dict类似，也是一组key的集合，但不存储value。由于key不能重复，所以，在set中，没有重复的key。
  - set是无序的，重复元素在set中自动被过滤。

```
>>> s = set([1, 2, 3])
>>> s
{1, 2, 3}
>>> s = set([1, 1, 2, 2, 3, 3])
>>> s
{1, 2, 3}
```

set可以看成数学意义上的无序和无重复元素的集合，因此，两个set可以做数学意义上的交集（&）、并集（|）、差集（-）等操作。

4.2 函数定义和调用

4.2.1 定义函数

定义函数的格式如下：

```
def 函数名():
    代码
```

demo:

```
# 定义一个函数，能够完成打印信息的功能
def printInfo():
    print '-----------------------------------'
    print '          人生苦短，我用Python'
    print '-----------------------------------'
```

```
#返回多个值的函数
def divid(a,b):
    shang = a//b
    yushu = a%b
    return shang,yushu
sh,yu = divid(5,2) #需要使用多个值来保存返回内容
print(sh,yu)

#结果显示： 2 1
```

3. 作用域

<1>在一个函数中定义的变量，只能在本函数中用(局部变量)

<2>在函数外定义的变量，可以在所有的函数中使用(全局变量)

5.文件操作

文件，就是把一些数据存放起来，可以让程序下一次执行的时候直接使用，而不必重新制作一份，省时省力。

5.1 文件打开与关闭

5.1.1 打开文件

在python，使用open函数，可以打开一个已经存在的文件，或者创建一个新文件

open(文件名，访问模式)

示例如下：

```
f = open('test.txt', 'w')
```

```
f = open('data.txt','w')
f.write('hello python!\n')
```

```
f.write('hello world!')
f.close()
# read 方法，读取指定的字符，每一次都往后面读
f = open('data.txt','r')
content = f.read(5)
print(content)
content = f.read(5)
print(content)
f.close()
f = open('data.txt','r')
content = f.readlines()
print(content)
f.close()
```

```
hello
 pyth
['hello python!\n', 'hello world!']
```

```
#捕获异常
try:
    print('------data---1---')
    f = open('123.txt','r')
    print('------data---2---')
except IOError:     #文件没找到，属于 IO异常（输入输出异常），符合IOError 错误
    pass    #捕获异常后执行的代码
try:
    print(num)
#except IOError:        #此时捕获不到，因为产生的错误不是IO异常
except NameError as result:
    print('产生错误了')
    print(result)
#   except(NameError,IOError):      #将可能产生的所有异常类型，都放到下面的小括号中
#   except (NameError,IOError) as result:
#       print(result)       #将对应错误信息打印
```

```
产生错误了
name 'num' is not defined
```

```
import time
try:
    f = open('data.txt','r')
    try:
        while True:
            content = f.readlines()
            if len(content)==0:
                break
            time.sleep(2)
            print(content)
    finally:    #表示在执行完毕或者捕捉到错误以后执行
        f.close()
        print('文件关闭')
except Exception as result:
    print('发生异常')
```

```
['hello python!\n', 'hello world!']
文件关闭
```