相关内容：https://www.cnblogs.com/wupeiqi/articles/5433893.html

一.字符串

(1)字符串的函数(一)

```
tex = "asddff"
v = tex.count("as",2,4)
n = tex.find("dd")
n = tex.find("dd",起始位置，结束位置)
```

count()是返回子序列在字符串中的个数，后面可以接起始位置。

```
def capitalize(self):
    """ 首字母变大写 """
        return ""
def center(self, width, fillchar=None):
    """ 内容居中，width：总长度；fillchar：空白处填充内容，默认无 """
def count(self, sub, start=None, end=None):
def endswith(self, suffix, start=None, end=None):
    """ 是否以 xxx 结束 ，返回True或者False"""
    return False
def find(self, sub, start=None, end=None):
    """ 寻找子序列位置，如果没找到，返回 -1 """
def index(self, sub, start=None, end=None):
     """跟find()差不多 """
def swapcase(self):
     """ 大写变小写，小写变大写 """
```

(2)字符串的函数(二)

```
def isalnum(self):
    """ 是否是字母和数字 """
def isalpha(self):
    """ 是否是字母 """
def isdigit(self):
    """ 是否是数字 """
def isdecimal(self):
    """ 是否是数字 """
def islower(self):
    """ 是否小写 """
 def isspace(self):
```

```
    """ 是否空格 """
def isupper(self):
    """ 是否大写 """
def isisidentifier(self)
    """ 是否标志符 """
def join(self, iterable):
    """ 按照字符连接 """
def lstrip(self, chars=None):
    """ 移除左侧空白   rstrip()移除右侧空白 strip()移除两侧空白 也可以指定清除内容"""
```

(3)字符串的函数(三)-分割

```
def rpartition(self, sep):
    """分割为三份 partition() 包含分割元素"""
```

```
text = "asdsddfdddsssasfd"
print(text.rpartition("s"))
print(text.partition("s"))
```

```
('asdsddfdddsssa', 's', 'fd')
('a', 's', 'dsddfdddsssasfd')
```

```
def split(self, sep=None, maxsplit=None):
    """ 分割， maxsplit最多分割几次 不包含分割元素 """
```

```
text = "asdsddfdddsssasfd"
print(text.split("s"))
print(text.split("s",1))
```

```
['a', 'd', 'ddfddd', '', '', 'a', 'fd']
['a', 'dsddfdddsssasfd']
```

```
def replace(self, old, new, count=None):
```

```
    """ 替换 old：旧字符串 new：新字符串 count：替换几次"""
```

(4)字符串格式化

Python的字符串格式化有两种方式：百分号方式、format方式。

百分号方式比如：

```
tpl = "i am %s age %d" % ("alex", 18)
tpl = "i am %(name)s age %(age)d" % {"name": "alex", "age": 18}
tpl = "i am %(pp).2f" % {"pp": 123.425556, }
```

format方式比如：

```
pl = "i am {}, age {}, {}".format("seven", 18, 'alex')
tpl = "i am {name}, age {age}, really {name}".format(name="seven", age=18)
#字典的形式 带两个**
tpl = "i am {name}, age {age}, really {name}".format(**{"name": "seven", "age": 18})
#带格式化
tpl = "i am {:s}, age {:d}, money {:f}".format("seven", 18, 88888.1)
tpl = "numbers: {:b},{:o},{:d},{:x},{:X}, {:%}".format(15, 15, 15, 15, 15, 15.87623, 2)
```

二.列表

(1)含义：用中括号括起来，中间用逗号隔开。列表中可以放数字、字符串、列表、布尔值等等。列表底层是用链表实现的。列表是有序的，而且列表的元素可以被修改。

```
li = [1,2,3,4,5]
# 3.
# 索引取值
print(li[3])
# 4.切片 切片的结果也还是列表
print(li[2:-1])
# 5.for循环和while循环也可以用于列表
for item in li:
    print(item)
```

```
# 6.索引
# print(li)
# 7.删除的一种方式
# del li
# print(li)
# 8.in 操作
if 3 in li:
    print("列表中存在该元素！")
```

(2).基本方法

```
 def append(self, p_object):
     """ 向列表中最后进行追加元素 """
 def clear(self)
     """清空列表"""
 def copy(self)
     """拷贝列表  浅拷贝"""
 def count(self, value):
     """ 对value元素在列表中出现的次数进行计数 """
 def extend(self, iterable):
     """ 将一个列表追加到另一个列表 """
def index(self, value, start=None, stop=None):
   """根据值获取索引位置"""
def pop(self, index=None):
   """删除某一个值 根据索引删除 默认删除最后一个"""
def remove(self):
   """ 根据值删除列表中的元素 """
def reverse(self):
   """ 将列表进行翻转 """
def sort(self, cmp=None, key=None, reverse=False):
    """对列表中的元素进行排序 默认是从小到大 reverse = True 是从大到小 """
```

三.元组tuple

(1).含义：用中括号括起来，中间用逗号隔开，元组中的元素是有序的。元素不可以修改，不能被增加或者删除。一般写元组的时候，在最后加一个逗号，为了和函数进行区分。可以进行索引、切片取值。

(2).基本方法(特有)

```
 def count(self, value):
     """ 对value元素在元组中出现的次数进行计数 """
 def index(self, value, start=None, stop=None):
     """根据值获取索引位置"""
```

四.字典

(1).含义：字典中保存的是键值对。用大括号括起来，键值对之间用逗号隔开，键和值之间用：表示。原理是hash表，字典的value可以是任意值，字典、列表不可以作为字典的key。key重复的时候，只会保留一个key。

```
# 打印的是dict字典的key
for i in dict:
```

```
    print(i)
#打印的是dict字典里的value
for i in dict.values():
    print(i)
#打印的是diet字典中的键值对
for key, value in dict.items():
    print(key, value)
```

(2).基本方法(特有)

```
 @staticmethod
 def fromkeys(S, v=None):
     """静态方法 序列来创建字典  s是key的参数 v是value的参数  """
def get(self, k, d=None):
     """ 根据key获取值，d是默认值 """
def has_key(self, k):
     """ 是否有key """
 def pop(self, k, d=None):
     """ 获取并在字典中移除 """
def popitem(self):
     """ 随机获取并在字典中移除 """
def setdefault(self, k, d=None):
    """ 如果key不存在，则创建，如果存在，则返回已存在的值且不修改 """
ef update(self, E=None, **F):
        """ 更新       """
```

五.集合set

(1).含义：由不同的元素组成的，集合中是一组无序排列可哈希的值(字符串、数组、元组)。用大括号括起来，元素间用逗号隔开。也可以用set()函数，括号内是集合的元素，也可以用frozenset()来生成集合，此时是不可变集合。

(2).基本方法(特有)

```
def add(self, *args, **kwargs):
  """  Add an element to a set    """
def difference(self, *args, **kwargs): # real signature unknown
  """ 相当于s1-s2 Return the difference of two or more sets as a new set. """
def pop(self):
  """  随机删除集合中的元素    """
def remove(参数):
  """  删除集合中的参数元素 不存在会报错   """
```

```
def discard(参数):
  """  删除集合中的参数元素 不存在的时候不会报错    """
```

(3).集合的运算

```
def intersection(self, *args, **kwargs):
   """相当于s1&s2 s1.intersection(s2)是求s1与s2的交集  """
def union(self, *args, **kwargs):
   """相当于s1|s2 s1.union(s2) s1与s2的并集 """
def difference(self, *args, **kwargs):
   """ 相当于s1-s2 s1.difference(s2) 是求s1与s2的差集   """
def isdisjoint(self, *args, **kwargs):
   """ 两个集合交集为空 返回true """
 def issubset(self, *args, **kwargs):
  """ 相当于s1<=s2 s1.issubset(s2)判断s1是否为s2的子集"""
def issubset(self, *args, **kwargs):
  """ 相当于s1<=s2 s1.issubset(s2)判断s1是否为s2的父集"""
```

五.函数

(1).定义:

```
 #例子
 def test(x):
      "The function definitions"
      x+=1
      return x
```

函数和过程的区别:

函数有返回值，过程没有返回值(只在python中进行区分)。当没有使用return显示的定义返回值时，python解释器会隐式的返回None，所以在python中即便是过程也可以算作函数。

```
#过程
def test01():
      msg='hello The little green frog'
      print msg
#函数
def test02():
      msg='hello WuDaLang'
      return msg
```

(2).参数组

**与字典有关系， *与元组有关系。

```
def tes(x, *args):
    print(x)
    print(args)
tes(1,2,3,4,5,6)
tes(1,*(2,3,4,5,6))//结果同上
```

结果：

```
1
(2, 3, 4, 5, 6)
```

当一个函数参数中的形参都有*args和**kwargs，*args必须在**kwargs的前面。

关于*args和**kwargs的博客：

https://blog.csdn.net/lllxxq141592654/article/details/81288741

(3).全局变量和局部变量：

在函数内利用

```
global 变量名
```

此时是利用的全局变量，而不是局部变量。如果没有global关键字，程序优先读取局部变量，再读取全局变量，但是无法对全局变量进行赋值操作。

```
name = "产品"
def test():
    print(name) #在函数内部使用全局变量，正确。
    name = "李正" #错误，在没有global关键字的情况下在函数内部对全局变量赋值
test()
def test1(): #test1()#正确 在global关键字的情况下，既可以对全局变量使用，也可以进行修改。
    global name
    print(name)
    name = "李正"
test1()
```

(4).匿名函数：

```
# 匿名函数为lambda开头 变量1是匿名函数的返回值， 表达式是计算过程
lambda 变量1 : 表达式
```

(5).Map函数、filter函数、reduce函数及其他内置函数：

https://blog.csdn.net/qq_32618817/article/details/80633848

```python
#map()函数
n = [1,2,3,4,5,6]
print("内置map函数")
ret1 = map(lambda x:x+1, n) #匿名函数
print(ret1)#打印的ret1的地址
print(list(ret1))#打印的是ret1的内容
```

其他内置函数：

网址(可以是中文)：https://docs.python.org/zh-cn/3/library/functions.html?highlight=built#ascii

六.文件处理

```python
#1. 打开文件，得到文件句柄并赋值给一个变量
f=open('a.txt','r',encoding='utf-8') #默认打开模式就为r
#2. 通过句柄对文件进行操作
data=f.read()
#3. 关闭文件
f.close()
f.read() #读取所有内容,光标移动到文件末尾
f.readline() #读取一行内容,光标移动到第二行首部
f.readlines() #读取每一行内容,存放于列表中
f.flush() #立刻将文件内容从内存刷到硬盘
```

七.迭代器和生成器

```python
#1、为何要有迭代器？
对于序列类型：字符串、列表、元组，我们可以使用索引的方式迭代取出其包含的元素。
但对于字典、集合、文件等类型是没有索引的，若还想取出其内部包含的元素，
则必须找出一种不依赖于索引的迭代方式，这就是迭代器
#2、什么是可迭代对象？
可迭代对象指的是内置有__iter__方法的对象，即obj.__iter__，如下
'hello'.__iter__
(1,2,3).__iter__
[1,2,3].__iter__
{'a':1}.__iter__

#3、什么是迭代器对象？
可迭代对象执行obj.__iter__()得到的结果就是迭代器对象
而迭代器对象指的是即内置有__iter__又内置有__next__方法的对象
```

```
文件类型是迭代器对象
open('a.txt').__iter__()
open('a.txt').__next__()

dic={'a':1,'b':2,'c':3}
 #得到迭代器对象，迭代器对象即有__iter__又有__next__，
 #但是：迭代器.__iter__()得到的仍然是迭代器本身
iter_dic=dic.__iter__()
print(iter_dic.__next__()) #等同于next(iter_dic)
```

只要函数内部包含有yield关键字，那么函数名()的到的结果就是生成器，并且不会执行函数内部代码。

```
def func():
    yield 1
    yield 2
    yield 3
    print('====>end')
#有了生成器就可以用迭代器的方式取值。
g=func()
g.__iter__
g.__next__
```

生成器表达式和生成器函数：https://www.cnblogs.com/wj-1314/p/8490822.html

八.装饰器

1.装饰器定义：本质就是函数，功能是为其他函数添加新功能

装饰器原则：

(1).不修改被装饰函数的源代码（开放封闭原则）。

(2).为被装饰函数添加新功能后，不修改被修饰函数的调用方式。

装饰器=高阶函数+函数嵌套+闭包

2.高阶函数

(1).函数接收的参数是一个函数名

(2).函数的返回值是一个函数名

(3).满足上述条件任意一个,都可称之为高阶函数

```
import time
def foo():
    print('from the foo')
def timmer(func):#参数是个函数
    start_time=time.time()
```

```
    func()
    stop_time=time.time()
    print('函数%s 运行时间是%s' %(func,stop_time-start_time))
timmer(foo)
```

3.函数的嵌套和闭包

函数内定义另一个函数

闭包:在一个作用域里放入定义变量,相当于打了一个包

装饰器的例子:

```
import time
def test():
    time.sleep(1)
    print('test函数运行完毕')
def timer(fun):
    def getTime():
        start_time = time.time()
        res = fun()
        end_time = time.time()
        print("函数运行时间是%s"%(end_time - start_time))
        return res
    return getTime
test = timer(test)
test()
#也可以用语法糖 直接test()函数上加上@timer
```

九.模块

而对于一个复杂的功能来，可能需要多个函数才能完成（函数又可以在不同的.py文件中），n个.py 文件组成的代码集合就称为模块。

模块分为三种:

- 自定义模块
- 第三方模块
- 内置模块

模块调用方法:

```
import module
from module.xx.xx import xx
from module.xx.xx import xx as rename
from module.xx.xx import *
```

模块介绍：https://www.cnblogs.com/wupeiqi/articles/5501365.html

https://www.cnblogs.com/yuanchenqi/articles/5732581.html

十.面对对象

https://www.cnblogs.com/linhaifeng/articles/6182264.html