- 环境变量
- python位置：C:\Users\Dats\AppData\Local\Programs\Python\Python39\python.exe
- 交互式编程（用于验证） cmd直接输入python，可以识别python语法，不适用于大量代码，由于不可以保存大量数据
- 编译型编程和解释型编程（python是解释型编程）
- markdown语言的语法
- 注释（用#）
    - 注释的目的是为了对代码进行说明
    - #只对单行有效，换行会报错
    - '''开始，'''结束用于多行注释，可以换行（三个单引号或者双引号）
    - CTRL+/ 可以添加或取消注释

变量

- 字符串需要用单/双引号
- ** 表示几次方，比如2**0.5表示2的开根号
- process finished with exit code**1 只要不是0程序就有问题**
- 数据类型：
    - int 整数类型
    - float 浮点数
    - python里面没有long类型
    - complex 复数型
    - 字符串类型需要用单引号或者双引号
    - 列表类型

        ```
        names = ['你好', '你好']
        ```

    - 字典类型（dict）

        ```
        names = {'nihao','hello'}
        ```

    - 元组类型(tuple)
    - 集合类型(set)
    - 布尔型；**T**rue or **F**lase eg:

        ```
        print( 4 > 5 )
        ```

        输出为Flase

查看数据类型

使用type内置类可以查看变量对应的数据类型

```
a=35
print(type(a))
34.print #在pycharm中的快捷键
print(type(3.14))
```

在python中变量是没有数据类型的，所说的数据类型是变量对应数值的数据类型

在python中变量数据类型可以改变，是动态数据类型

```
a=3.14
print(type(a)) #<class 'float'>
a = 'hello'
print(type(a)) #<class 'str'>
```

标识符和关键字

- 标识符：变量，模块名，函数名，类型
- 标识符的命名规则
    1. 由数字，字母和下划线组成，不能由数字开头 eg: a_b = 'hello'
    2. 严格区分大小写，计算机编程中有52个英文字母，因为区分大小写
    3. 不能使用关键字作为变量名；if/True/else/and/as/with/del/
- 规范：建议遵守
    1. 顾名思义，变量名有具体含义
    2. 遵守一定的命名规范
        - 小驼峰命名法：第一个单词的首字母小写，以后每个单词的首字母都大写
        - 大驼峰命名法：每个单词的首字母都大写
        - 使用下划线连接：user_name_and_password

          在python中变量，函数和模块名使用下划线连接，类名用大驼峰命名法

输出语句

python里使用print（内置函数）语句来输出内容

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
print('hello','good','yes','no',sep = '+')
#sep 表示输出时每个值之间使用哪种字符作为分隔，默认为空格
#end 当执行完一个print语句后接下来要输出的字符，默认为\n表示换行
#file 表示输出到文件，学文件操作时再说
```

输入语句

python中使用input内置函数接收用户输入

```
input('请输入密码')
password = input('请输入密码') #定义一个变量可以保存用户输入的数据，password是字符串
```

整数的表示方式

- 数据类型
  - 整型（int）：计算机可以保存二进制，八进制和十六进制；在python中都可以表示，默认为十进制的数字
    - 二进制以**0b**开头

```
a = 0b0110102
print(a) #打印默认为十进制的数
```

    - 八进制以0O开头(python3中只能以0o开头，python2中以0开头也是八进制的数字)

```
a = 0o345
print(a)
```

    - 十六进制以0x开头

```
a= 0x9834
```

- 进制转换
  - 十进制转换为二进制用余数法
  - 使用bin内置函数将数转换为二进制

```
a = 89
print(bin(a))
```

- 使用代码进行进制转换
  - 转换为二进制-----bin()
  - 转换为八进制----oct()
  - 转换为十六机制---hex()
- 转换为整数使用内置类 **int** （被转换的字符串只能是数字，转换后可以进行运算）

```
a = '1a3b'
b = int(a,16)#将a当作十六进制，然后转换为整数
print(a,b+1,sep=';')  #运行结果：1a3b;6716  #打印默认为十进制
```

- 转换为浮点数使用内置类**float**
- 转换为字符串使用内置类**str**
- 转换为布尔值使用内置类**bool**

```
print(bool(-1)) #True
print(bool('')) #False：空字符串
print(bool(None)) #False：空数据
print(bool([])) #False：空列表
print(bool(())) #False：空元组
print(bool({})) #False;空字典
```

只有数字0转换为布尔型才是False，字符串也能转换为布尔型，只有空字符串才是False，其他都是True;None转换为布尔值也是False,空字典，空集合，空列表都是False

在计算机中，True和False使用数字1和0来保存，可以当作数字来用进行加减乘除计算

```
s = set()#空集合
print(bool(s))
```

- 隐式类型转换

```
if 3:
    print('hello')#可以打印，因为3是True
if 0:
    print('hello')#不打印
```

运算符

1. 算术运算符（+-*/）
    - ** 幂运算
    - % 取余
    - // 整除

    ```
    print(6 / 2)# 3.0 #在python3中，两个整数相除得到浮点数，python2中两个整数相除是整数
    print(9 // 2)# 4
    print(1234 % 3201) #1234
    ```

    - 算术运算符在字符串里的使用
        - 加法运算符：只能用于两个字符串类型的数据，用来拼接两个字符串

          在python中数字和字符串之间不能做加法运算（其他语言可能支持）

          ```
          print('Hello' + 'World') #Helloworld
          ```

        - 乘法运算符:可以用于数字和字符串之间，用于重复多次（只有python支持该运算）

          ```
          print('Hello' * 2) #HelloHello
          ```

2. 赋值运算符（=）
    - 计算机编程中，等号作用是将等号右边的值赋值给等号的左边；等号左边一定不能是常量或者表达式
    - 复合赋值运算符 **+= / -= / *=**

```
m, n = 3, 5 #拆包（元组）
print(m, n) #3 5
```

- *表示可变长度

```
m, *n, a = 3, 5,5,3,2,6,87
print(m, n, a)
#3 [5, 5, 3, 2, 6] 87
```

```
*m, n, a = 3, 5,5,3,2,6,87
print(m, n, a)
#[3, 5, 5, 3, 2] 6 87
```

3. 比较运算符(> < >= <= != ==)

- python2中可以使用<>表示不等于运算符，python3中则不能使用
- 比较运算符在字符串中的使用

```
print('a' > 'b')#字符串之间使用比较运算符会根据字符的编码值（ASCCII）逐一进行比较
print('bc' >= 'c') #False
```

数字和字符串之间做==运算是是False，做！=结果是True，不支持其他的比较运算

4. 逻辑运算符（逻辑与and,逻辑非not,逻辑或or)
   - 逻辑与运算规则：只要有一个False结果就是False，只有所有的运算数都是True结果才是True

```
print(9 > 1 and 4 > 2 and 8 > 4) #True
```

   - 逻辑或的运算规则：只要有一个是True结果就是True，只有全部都是False结果才是False

```
print(4 > 3 or 5 < 6 or 9 < 2)#True
```

5. 逻辑运算的短路

```
#逻辑与运算只有所有的运算数都是True结果才有True
#只要有一个运算数是False结果就是False
#因此下面语句运行到第二句就停止
4 > 3 and print('hello world')
4 < 3 and print('你好世界') #结果只打印hello world
#逻辑或运算只要有一个是True结果就是True，只有全部都是False结果才是False
5 > 3 or print('hello world')#运行到 5 > 3判断为True即停止运行
4 > 9 or print('你好世界') #结果只打印你好世界
```

   - 逻辑运算结果不一定是布尔值
   - 逻辑与运算做取值时取第一个为False的值；如果所有运算数都是True，取最后一个值

```
print(3 and 5 and 0 and 'hello') #0
print('good' and 'hello' and '0' and 'ok')#ok
```

6. 位运算

   ~ 按位取反

7. 运算符的优先级：幂运算优先级最高
   - 逻辑运算的优先级：not > and > or

条件判断语句（分支语句）

- python中不支持switch语句条件语句

1. if语句

```
age = int(input("请输入你的年龄：")) #input的输出默认为字符串，记得类型的转换
if age < 18:
    print('未满18禁止入内')
```

2. if...else语句

- if 判断条件:

  条件成立时执行的条件

  else:

  if条件不成立时执行的代码

```
age = int(input("请输入你的年龄："))
if age < 18:
    print('未满18禁止入内')
else:
    print('欢迎光临')
```

- - 例子1：写出判断一个数是否同时被3和7整除的条件语句，并且打印对应的结果

```
a = int(input("请输入一个数字："))
b = a%4 and a%6
if b==0:
    print(a,'能被4和6整除',sep='')
else:
    print(a,'不能整除',sep='')
```

  - 例子2：写出判断一个数是否能被3整除或者被7整除，但是不能同时被3和7整除

```
a = int(input("请输入一个数字："))
if (a % 3 == 0 or a % 7 == 0) and (a % 21 != 0 ):
    print(a,'能被3或7整除但是不能被3和7同时整除',sep='')
else:
    print(a,'不能整除',sep='')
```

  - 例子3：输入年判断是否为闰年

```
year = int(input("请输入一个年份："))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0 ):
    print(year,'是闰年',sep='')
else:
    print(year,'不是闰年',sep='')
```

  - 例子4：定义两个变量保存一个人的身高和体重，编程实现判断这个人的身材是否正常；判断公式为：体重/身高的平方值，在18.5~24.9之间属于正常

```
weight = float(input('请输入您的体重：'))
height = float(input('请输入您的身高：'))
num = weight/(height**2)
if num > 18.5 and num < 24.9:# if 18.5 < num <24.9
    print("您的身材正常")
else:
    print('您的身材不正常')
```

3. if...elif语句
    - 用来减少程序运行的时间

```
score = float(input('请输入你的成绩'))
if 0 < score < 60:
    print('你个垃圾')
elif 60 <= score < 80:
    print('一般般')
elif 80 <= score <=100:
    print('好棒棒哦')
```

4.if语句的嵌套

- python中使用强制缩进来表示语句之间的结构

```
ticket = input('是否买票了？Y/N')
if ticket == 'Y':
    print('买票了可以进站',end='\n')
    safe = input('安检是否通过？Y/N')
    if safe == 'Y':
        print('可以进站')
    else:
        print('重新安检')
else:
    print('gun')
```

pass关键字的使用

- pass关键字在python中没有意义，只是单纯的用来占位，保证语句的完整性

```
age = int(input('年龄是多少'))
if age > 18:
    print('欢迎光临')
    print('hello')
print('yes')
  # 在任何情况下都会打印yes
```

```
age = int(input('年龄是多少'))
if age > 18:
    pass #使用pass来占位，没有意义，只是单纯的为了保证语句的完整性，使程序不报错
print('yes')
```

随机数random

```
import random #导入随机数模块
player = int(input('请输入 （0）剪刀（1）石头（2）布'))
computer = random.randint(0,2)#产生为0到2的随机数
print('电脑输出的是', computer)
if(player == 0 and computer ==2) or (player ==1 and computer == 0) or (player
==2 and computer == 1):
    print('恭喜你赢了')
elif player == computer:
    print('平局')
else:
    print('你输了')
```

5.*条件语句的注意事项*

- 区间判断(**可以使用连续的区间判断**)

  ```
  if 0<= score < 30
  ```

- 隐式类型转换（if后面需要的是一个bool类型的值，会自动转换成bool类型）
- 三元表达式（对if...else语句的简写）

  ```
  num1 = int(input('请输入一个数字：'))
  num2 = int(input('再输入一个数字：'))
  if num1 > num2:
      x = num1
  else:
      x = num2
      print('两个数里较大的数字是', x)
  ```

  可以简化为

  ```
  num1 = int(input('请输入一个数字：'))
  num2 = int(input('再输入一个数字：'))
  x = num1 if num1 > num2 else num2
  print('两个数中较大的数是', x)
  ```

循环语句的基本使用

python中循环有while循环和for循环，不支持do...while循环

python中没有自增自减运算符

1. while循环的基本使用

   while 判断条件：

   条件成立时的代码

```
x = 0
while x < 10:
    x = x + 1
    print('hello')8
```

- 例1(求1到100的所有数字之和)

```
x = 0
sum = 0
while x < 100:
    x = x + 1
    sum = sum + x
print(sum)
```

- 例2（求1到100所有偶数的和）

```
x = 0
sum = 0
while x < 100:
    x = x + 1
    if (x % 2 == 0):
        sum = sum + x
print(sum)
```

2. for...in循环的使用

   range 内置类用来生成指定区间的整数序列（就是一次遍历）

   ```
   for i in range(0,10):#左闭右开区间
       print(i) #打印0到9
   ```

   注意： in的后面必须是可以迭代的对象（字符串，列表，字典，元组，集合，range）

   ```
   for y in 'hello':
       print(y)#运行结果是将字符串中的每一个字符都单独拿出来
   ```

3. break和continue的关键字的使用

   - break：用来结束整个循环
   - continue:用来结束本轮循环开启下一轮循环

```
i = 0
while i < 5:
    if i == 3:
        i +=1
        continue
    print(i)
    i +=1#打印 0 1 2 4
```

```
i = 0
while i < 5:
    if i == 3:
        i +=1
        break
    print(i)
    i +=1 #只打印0 1 2
```

- 例1：不断的让用户输入用户名和密码，只要用户名不是zhangsan,密码不是123就一直问

```
username = input('请输入用户名：')
password = input('请输入密码')
while not (username == 'zhangsan' and password == '123'):
    username = input('请输入用户名：')
    password = input('请输入密码')
```

```
#使用break语句实现
while True:
    username = input('请输入用户名：')
    password = input('请输入密码')
    if username == 'zhangsan' and password == '123':
        break
```

尽量使用正面的思维而不是使用反面思维，避免陷入逻辑陷阱

1. 循环的嵌套
   - 例1

```
i = 0
while i < 5:
    print('*'*i)
    i += 1
*
**
***
****
```

   - 例2：打印乘法表

```
for i in range(1,10):#外循环用来控制行数，内循环用来控制列数
    for j in range(i, 10):
        print(i, '*', j,'=', i*j, end='  ')
        if j == 9:
            print(end='\n')
```

   - 例3：求质数
     1. for...else语句：当for语句中的break成立时执行else语句

```
count = 0 #注意for...else的格式
for i in range(101,201):
    for j in range(2,i):
        if i % j == 0:
            break
    else:
        count += 1
        print(i, '是质数')
print(count)
```

可以循环到i的开方+1即可停止

     2. 使用假设成立法求质数

```
for i in range(1,101):
    flag = True #每次都假设i是质数
    for j in range(2,int(i**0.5)+1):
        if i % j == 0:
            flag =False
            break
    if flag == True:
        print(i, '是质数')
```

3. 使用计数法求质数(欧拉线性筛选)

```
for i in range(2,101):
    count = 0 #假设这个数字可以被0个数字整除
    for j in range(2,int(i**0.5)+1):
        if i % j == 0:
            count += 1
    if count == 0:
        print(i, '是质数')
```

- 例4：求斐波那契数列(使用冒泡排序法)

```
count = int(input('斐波那契数列的第几个数啊？')) - 1
num1 = 1
num2 = 1
for i in range(1, count):
    a = num1
    num1 = num2
    num2 = a + num1
print(num2)
```

字符串

1. 字符串的表示方式

在python中可以使用一对单引号或者一对双引号表示字符串，或者一对三个单引号表示，最常用的是一对单引号

```
m = ' xiaoming said:"Hello" '#不能用双引号将内容包括
print(m)
```

如果字符串中含有双引号，外部最好用单引号

如果字符串中含有单引号，外部最好用双引号

- 字符串中的转义字符\，作用是对\后面的意义进行转义

```
m = 'I\'m xiaoming said:"Hello" '
print(m)
```

```
m = 'I\'m xiaoming \n said:"Hello" '
print(m)   #\n表示换行，不会进行转义
```

```
m = 'I\'m xiaoming \t said:"Hello" '
print(m) #\t 表示增加一个制表符（tab）
```

\\表示普通的反斜杠

在字符串的前面添加r，在python中表示的是原生字符串

2. 字符串的下标和切片
   - 下标又称为索引，表示第几个数据，可迭代对象都可以遍历，都可以进行遍历，通过下标来获取或者操作数据，在计算机中下标都是从0开始的

```
word = 'zhangsan'
print(word[4])
```

字符串是不可变数据类型，不能改变原有的字符串，只能进行查看；只能通过新的字符串

   - 切片就是从字符串中复制一段指定的内容，生成一个新的字符串

```
word = 'zhangsan'
#切片语法 m[start:end:step]#左闭右开区间
print(word[4:6:1])#包含start，不包含end
print(word[2:]) #如果只设置了start，会截取到最后
print(word[:6]) #如果只设置了end,会从头开始截取
print(word[::]) #从头到尾复制一遍
print(word[::-1])#从尾到头复制一遍
print(word[-5:-1])#从右边的第五个开始的，但是是从左往右数的 #ngsa
```

step表示步长，默认为1，不能为0，可以为负数，负数表示从右往左找

3. 字符串常见的操作（使用内置函数）

```
#is开头判断是否是什么
word = 'abcdefghijk'
print(len(word)) #获取字符串的长度
print(word.find('f')) #获取指定字符的下标，若字符串中没有则输出-1，输出最小下标，rfind输出最大的下标
print(word.index('d')) #获取指定字符的下标，如果内容在字符串中没有则报错
print('Hello'.startswith('H'))#判断是否以H开头
print('hello'.endwith('o'))#判断是否以o结尾
print('hello'.isalpha())#判断是否是字母
print('hello'.isdigit())#判断是否为数字
print('hello'.isalnum())#判断是否是数字或者字母，不能有其他符号
#........and so on
```

4.字符串的替换（replace的方法）

```
word = 'hello'
m = word.replace('l','x')
print(word)
print(m)#字符串不可改变，只能生成新的字符串
```

5.字符串的内容分割*(split rsplit splitlines partition rpartition)*

```
x = 'zhangsan lisi wanwu zhoaliu'
y = x.split(' ')
z = x.split(' ',2)#2表示切割次数
print(x)
print(y)
print(z)
#zhangsan lisi wanwu zhoaliu
#['zhangsan', 'lisi', 'wanwu', 'zhoaliu'] #切割后是一个列表
#['zhangsan', 'lisi', 'wanwu zhoaliu']
```

rsplit于split没有区别

```
x = 'zhangsanlisiwanwuzhoaliu' #指定一个字符作为分隔符，分为三部分 前面 分隔符 后面
print(x.partition('g'))
#('zhan', 'g', 'sanlisiwanwuzhoaliu')  #partitution用于获取文件名和后缀名；
rpartitution取最后
```

快捷键的使用

- 双击shift会弹出全局搜索功能；jetbrains很多开发工具都有此功能
- ALt+shift+enter或者Crtl+Alt+L快速格式化代码
- Ctrl+D快速复制粘贴代码
- ALt/Ctrl+shift+上下箭头快速移动一行代码
- 在file--setting--keymap中修改快捷键
- ctrl+Y快速删除一行代码
- HOME/ENG将光标移到最前面或者最后面

字符集

- 计算机只能处理0和1，一个字节8位
- ASCII码使用一个字节来表示一个字符。最多只能表示128个，不使用最高位
- Unicode编码--绝大部分国家的文字都有一个对应的编码
- GBK（）统一编码，汉字占三个字节）,UTF-5,BIG5编码方式
- 使用内置函数chr和ord能够查看数字和字符的对应关系

```
print(ord('a')) #97 根据字符获取对应的编码
print(chr(65)) #A  根据编码获取对应的字符
print(ord('你'))
```

- 字符串转化为指定编码集的结果

```
print('你'.encode('gbk'))#b'\xc4\xe3'  #‘你’在gbk编码集时的编码结果
y = '你好'.encode('utf-8')
print(y) #b'\xe4\xbd\xa0\xe5\xa5\xbd'
print(y.decode('utf-8')) #你好
```

成员运算符

- 用来判断一个内容在可迭代对象里是否存在

```
word = 'hello'
x = input('请输入一个字符：')
for c in word:
    if x == c:
        print('你输入的字符存在')
        break
else:
    print('你输入的字符不存在')
```

```
word = 'hello'
x = input('请输入一个字符：')
if word.find(x) == -1:
    print('你输入的内容不存在')
else:
    print('你输入的字符存在')
```

```
word = 'hello'
x = input('请输入一个字符：')
if x in word:  ######in语句
    print('你输入的字符存在')
else:
    print('你输入的字符不存在')
```

```
word = 'hello'
x = input('请输入一个字符：')
if x  not in word:
    print('你输入的字符不存在')
else:
    print('你输入的字符存在')
```

格式化打印字符串

- 可以使用%占位符来表示格式化一个字符串

```
name = 'zhangsan'
age = 18
print('我的名字是%s,年龄是%d'%(name, age))
print('我的年龄是%3d'%(age)) #我的年龄是 18
print('我的年龄是%03d'%(age)) #我的年龄是018
print('我的年龄是%.5f'%(age))#我的年龄是18.00000
print('我的年龄是%x'%(age)) #我的年龄是12(输出为十六进制)
print('我的年龄是%%x')#我的年龄是%%x
```

- 字符串的format使用方法（使用{}进行占位）

```
x = '大家好，我是{},今年{}岁了'.format('zhangsan', 19)
print(x)  #大家好，我是zhangsan,今年19岁了
```

    - **{数字}--根据数字进行占位，从数字0开始**(或使用变量名)

```
x = '大家好，我是{1},今年{0}岁了'.format(19, 'zhangsan')
print(x)#大家好，我是zhangsan,今年19岁了
print('大家好，我叫{name},今年{age}岁'.format(name = 'zhangsna', age =
'15'))#使用变量名
#数字和变量名可以混合使用
```

```
d = ['zhangsan', 18, '上海', 180]#（列表）顺序不能乱
b = '大家好，我是{},今年{}岁，来自{},身高是{}'.format(*d)
print(b)
#大家好，我是zhangsan,今年18岁，来自上海,身高是180
```

```
info = {'name': 'zhangsan', 'age': 13, 'addr': '北京', 'height':
180}#（字典）按照变量名
c = '大家好，我是{name},今年{age}岁，来自{addr},身高是
{height}'.format(**info)
print(c)
```

列表

- 当我们有多个数据需要按照一定顺序保存时我们可以考虑列表，列表是有序可变的
- 和字符串一样可以利用下标来获取元素和对元素进行切片，并且可以使用下标来修改列表中的元素
- 使用[]来表示一个列表，列表的每一个数据称为元素，元素之间使用逗号分割

```
names = ['张三','李四', '王五', '张飞', '关羽']
```

- list(可迭代对象)--将可迭代对象变为一个列表

```
names = list(('张三','李四', '王五', '张飞', '关羽'))
print(names)
#['张三', '李四', '王五', '张飞', '关羽']
```

- ```
  names = list(('张三','李四', '王五', '张飞', '关羽'))
  names[3] = '马超'
  print(names)
  #['张三', '李四', '王五', '马超', '关羽']
  ```

- **列表的操作--增加数据，删除数据，修改数据，查询数据**
  - 添加元素的方法 append insert extend
    1. append--在列表最后插入元素

       ```
       names = list(('张三','李四', '王五', '张飞', '关羽'))
       names.append('诸葛亮')
       print(names)
       #['张三', '李四', '王五', '张飞', '关羽', '诸葛亮']
       ```

    2. insert（index，object）需要两个参数，index是下标，表示在哪个位置插入数据，object是对象，表示具体插入哪个数据
       ```

```
names = list(('张三','李四', '王五', '张飞', '关羽'))
names.insert(3,'诸葛亮')
print(names)
#['张三', '李四', '王五', '诸葛亮', '张飞', '关羽']
```

3.extent(可迭代对象)--将可迭代对象增加到原来的列表中

```
names = list(('张三','李四', '王五', '张飞', '关羽'))
x = ['李白', '王维', '苏轼']
names.extend(x)
print(names)
#['张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼']
```

- 列表的删除--pop,remove,clear
  1. pop---默认删除最后一个数据

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
x = names.pop()
print(x)
print(names)
#苏轼
#['张三', '李四', '王五', '张飞', '关羽', '李白', '王维']
```

还可以根据下标删除指定位置的数据

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
x = names.pop(3)
print(x)
print(names)
#张飞
#['张三', '李四', '王五', '关羽', '李白', '王维', '苏轼']
```

2. remove用来删除指定位置上的元素

```
    names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
    names.remove('李四')
    print(names)
```

3. clear用来清空一个列表

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
names.clear()
print(names)
```

4. del运算符将变量直接删除，最好不要用

- 列表的查询相关方法
  1. index--查询所在的位置

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
print(names.index('王五'))
# 2
```

2. count--查询数量

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
print(names.count('王五'))
# 1
```

3. in--查询是否存在

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
print('王五' in names)
# True
```

```
for name in names:
   print('该用户已经存在')
```

- 修改元素--使用下标直接修改元素

```
names = list(('张三', '李四', '王五', '张飞', '关羽', '李白', '王维', '苏轼'))
names[5] = '王勃'
print(names)
#['张三', '李四', '王五', '张飞', '关羽', '王勃', '王维', '苏轼']
```

- 列表的遍历
  - while循环遍历

```
killer = ['李白', '赵云', '孙悟空', '韩信']
i = 0
while i < len(killer):
    print(killer[i])
    i += 1
```

  - for...in循环：不断调用迭代器next方法查找下一个数据

```
killer = ['李白', '赵云', '孙悟空', '韩信']
for k in killer:
    print(k)
```

- 交换两个变量的值

```
a = 2
b = 3
c = a
a = b
b = c
print(a,b,sep='\n')
```

```

```

使用异或方法(并不常用)

a = 3
b = 9
a = a ^ b
b = a ^ b
a = a ^ b
print(a,b)

```
  ```python
# 使用python特有的
  a = 3
  b = 9
  a, b = b, a#########
  print(a, b)
```

- 冒泡排序法（难点但不是重点）

```
a = [12, 54, 3, 35, 74, 84, 5]
i=0
for j in range(0, len(a) - i-1):
    for i in range(0, len(a) - 1):
        if a[i] < a[i + 1]:
            a[i], a[i + 1] = a[i + 1], a[i]
    print(a)
```

- 用sort方法排序，sort直接对原有的列表排序

```
a = [12, 54, 3, 35, 74, 84, 5]
a.sort()
print(a)
#[3, 5, 12, 35, 54, 74, 84]
```

```
a = [12, 54, 3, 35, 74, 84, 5]
a.sort(reverse=True)
print(a)
#倒序排列
#[84, 74, 54, 35, 12, 5, 3]
```

- 内置函数sorted--不改变原有列表，生成一个新的列表
```

```
a = [12, 54, 3, 35, 74, 84, 5]
x = sorted(a)
print(x)
```

- 列表的反转

```
a = [12, 54, 3, 35, 74, 84, 5]
a.reverse()
print(a)
#[5, 84, 74, 35, 3, 54, 12]
```

```
a = ['张三', '李四', '王五', 35, 74, 84, 5]
a.reverse()
print(a)
#[5, 84, 74, 35, '王五', '李四', '张三']
```

- 列表的复制
  - 可变数据类型和不可变数据

```
nums1 = [1,34,12]
nums2 = nums1
nums1[0] = 89
print(nums2)
#[89, 34, 12]
```

python中的数据保存在内存中，数据可以分为可变类型和不可变类型

不可变类型：字符串；数字；元组

可变类型：列表，字典，集合

不可变数据类型如果修改值，内存地址会发生改变

可变数据类型如果修改值内存地址不会改变

```
nums1 = [1, 34, 12]#使用copy方法
nums2 = nums1 #指针指向同一内存空间，会相互影响，等号是内存地址的赋值
x = nums1.copy()
print(x)
```

```
#使用copy模块
import copy
nums1 = [1, 34, 12]
nums2 = nums1 #指针指向同一内存空间，会相互影响，等号是内存地址的赋值
x = copy.copy(nums1)
print(x)
```

- 求列表中的最大的元素

```
nums = [1, 2, 5, 66, 3, 65, 37, 54]
x = nums[0]
for num in nums:
    if num > x:
        x = num
print('最大数就是%d,他的下标是%d' % (x, nums.index(x)))
```

- 删除列表中的空元素

```
nums = [1, 2, '', 5, 66, 3, 65, 37, 54]
for words in nums:
    if words == '':
        nums.remove('')
print(nums)
#有bug,不能删除连续的空字符串
#注意：在使用for循环时尽量不要对原字符串进行增删操作
```

```
nums = [1, 2, '', '', 5, 66, 3, 65, 37, 54]
i = 0
while i < len(nums):
    if nums[i] == '':
        nums.remove(nums[i])
        i -= 1 ###############将光标往前移动一位
    i += 1
print(nums)
```

```
#利用空列表将数字加入空列表
nums = [1, 2, '', '', 5, 66, 3, 65, 37, 54]
new_num = []
for num in nums:
    if num != '':
        new_num.append(num)
nums = new_num
print(new_num)
```

- 列表的嵌套（多维数组）
  - 练习：一个学校有8个老师，有3个教室，随机为老师分配工位

```
import random
teachers = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
rooms = [[],[],[]]
for teacher in teachers:
    room = random.choice(rooms)
    room.append(teacher)
print(rooms)
```

```
import random
teachers = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
rooms = [[],[],[]]
for teacher in teachers:
    room = random.choice(rooms)
    room.append(teacher)
print(rooms)
```

```
for i, room in enumerate(rooms): #下标的for循环
    print('房间%d里一共有%d个老师'%(i,len(room)),',分别是', end='')
    for teacher in room:
      print(teacher,end=' ')
    print(end='\n')
#  [['C', 'H', 'I', 'J'], ['A', 'B', 'D', 'E', 'F', 'G'], []]
#房间0里一共有4个老师 ,分别是C H I J
#房间1里一共有6个老师 ,分别是A B D E F G
#房间2里一共有0个老师 ,分别是
```

- 列表的推导式--使用简单的语法创建一个列表

```
nums = [i for i in range(10)]
print(nums)
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
nums = [i for i in range(10) if i % 2 == 0] #取偶数
print(nums)
#[0, 2, 4, 6, 8]
nums = [i for i in range(10) if i % 2] #取奇数
print(nums)
```

```
```

points是一个列表，这个列表里的元素都是元组

points = [(x, y) for x in range(5, 9) for y in range(3, 8)]
print(points)
#[(5, 3), (5, 4), (5, 5), (5, 6), (5, 7), (6, 3), (6, 4), (6, 5), (6, 6), (6, 7), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7)]

```
- 练习：将列表list中的元素实现分组，比如[1,2,3,4,5,6]变成[[1,2,3],[4,5,6]]

  *（了解即可）*

  ```python
list = [i for i in range(1,101)]
  n = [list[j:j+3] for j in range(0,100,3)]
  print(n)
#[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18], [19, 20, 21], [22, 23, 24], [25, 26, 27], [28, 29, 30], [31, 32, 33], [34, 35, 36], [37, 38, 39], [40, 41, 42], [43, 44, 45], [46, 47, 48], [49, 50, 51], [52, 53, 54], [55, 56, 57], [58, 59, 60], [61, 62, 63], [64, 65, 66], [67, 68, 69], [70, 71, 72], [73, 74, 75], [76, 77, 78], [79, 80, 81], [82, 83, 84], [85, 86, 87], [88, 89, 90], [91, 92, 93], [94, 95, 96], [97, 98, 99], [100]]
```

- 深拷贝和浅拷贝
```

```
import copy
nums = [1, 2, 3, 4]
nums1 = nums #不是拷贝，而是赋值
nums2 = nums.copy() #浅拷贝，两个内容一样，但是不是同一个对象
nums3 = copy.copy(nums) #和上面是一样的，都是浅拷贝
#深拷贝只能用copy模块实现
words = ['Hello',[1, 20, 123, 32],'hi','my dear', 'How']
word = words.copy() #浅拷贝，两个是不一样的地址
words[1][1] = 5
print(words) #['Hello', [1, 5, 123, 32], 'hi', 'my dear', 'How']
#浅拷贝只拷贝一层，里面的一层修改后源内容也改变
word1 = copy.deepcopy(words) #只能用这一种方式实现深拷贝
print(word1)
```

元组的使用

元组和列表很像，都是用来保存多个数据的，使用一个小括号来表示一个元组，和列表的区别在于列表是可变的，而元组是不可变的数据类型，不能进行修改

```
nums = (1, 2, 3, 4, 6)
print(nums[3]) #4
print(nums.index(4)) #打印4的下标  #3
print(nums.count(4)) #计算某一值的出现的次数 #1
```

- 如何表示只有一个元素的元组

```
num = (12,) #在元素后面加一个,号
print(type(num)) #<class 'tuple'>
```

- tuple是一个内置类

```
print(tuple('Hello'))
#('H', 'e', 'l', 'l', 'o')
```

- 如何将列表转化为元组/元组转化为列表

```
num = [1,2,4,6,67]
print(tuple(num))#将列表转化为元组
```

```
num = (1,2,4,6,67)
print(list(num))#将元组转化为列表
```

```
num = ('q', 'w', 'r', 't')
print('*'.join(num))
#q*w*r*t
```

```
num = ('q', 'w', 'r', 't')
print(''.join(num))
#qwrt      #变为字符串
```

- 元组的遍历

```
num = ('q', 'w', 'r', 't')
for i in num:
    print(i)
#也可使用while语句
```

字典的使用

- 列表可以储存任意类型数据，但是一般情况下我们都储存单一数据类型
- 列表只能存储一个值，但是无法对值进行描述
- 字典不仅能够保存值，还可以对值进行描述
- 使用大括号来表示一个字典，不仅有值value，还有值的描述key
- 字典里的数据都是以键值对key-value的形式来保留的，字典是可变数据类型
- key-value之间使用冒号来连接

```
person = {'name': 'zhangsan',
          'age': 34,
          'English': 23,
          'Chinese': 76,
          'isPass': True,
          'Hobbies': ['唱', '跳', 'rap'],
          4: 'good',
          (100, 20): 'yes'
          }
```

  - 注意：
    - **字典中的key不可以重复，如果重复则后面的值会覆盖前面一个的值**
    - **字典中的value可以是任意数据类型，但是key只能使用不可变数据类型，一般使用字符串**
- 字典的查找数据--字典在保存时是无序的，不能够通过下标来获取数据，使用key来获取数据

```
person = {'name': 'zhangsan',
          'age': 34,
          'English': 23,
          'Chinese': 76,
          'isPass': True,
          'Hobbies': ['唱', '跳', 'rap'],
          4: 'good',
          (100, 20): 'yes'
          }
print(person['age']) #34
x = 'name'
print(person[x])
```

- **如果要查找的key不存在就会直接报错**
- **使用字典的get方法实现获取一个不存在的key时不报错，如果这个key不存在，使用默认值**

```
person = {'name': 'zhangsan',
          'age': 34,
          'English': 23,
          'Chinese': 76,
          'isPass': True,
          'Hobbies': ['唱', '跳', 'rap'],
          4: 'good',
          (100, 20): 'yes'
          }
print(person.get('height')) #返回none
print(person.get('gender', 'man')) #man   #如果没有获取key则使用给定的默认值
print(person.get('name', 'lisi'))  #zhangsan
```

- 字典的增删改

```
person = {'name': 'zhangsan',
          'age': 34,
          'English': 23,
          'Chinese': 76,
          'isPass': True,
          'Hobbies': ['唱', '跳', 'rap'],
          4: 'good',
          (100, 20): 'yes'
          }
person['name'] = 'lisi' #直接使用key可以修改对应的value
person['femal'] = 'female' #如果key不存在直接在字典里添加一个新的key-value
person.pop('age')  #直接将键值对删除
result = person.popitem()  #随机删除一个键值对，可以返回被删除的键值对
print(person)
print(result)
person.clear()#清空一个字典
del person['Hobbies']
```

- update的使用
  - 列表可以使用extent方法将两个列表合并成一个列表，在字典中用updata,字典不能用加法运算

```
person1 = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
person2 = {'name': 'lisi', 'addr':'jiangsu', 'age':30}
person1.update(person2)
print(person1)
```

- 字典的遍历
  - 特殊在列表和元组是单一的数据，但是字典是键值对的形式
  - 第一种遍历的方式:for...in循环获取的是key，常用

```
person = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
for x in person:
    print(x, '=', person[x]) #x不能加引号因为x是一个变量
```

  - 第二种方式：获取到所有的key，然后遍历key，根据key获取value

```
person = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
for k in person.keys():
    print(k, '=', person[k])
```

- 第三种方式：只获取所有的value，不常用

```
person = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
for k in person.values():
    print(k)
```

- 第四种方式：将一对键值对一起取出,常用

```
person = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
for item in person.items():
    print(item)
```

```
#将字典进行拆包
person = {'name': 'zhangsan', 'addr': 'jiangxi', 'age': 18}
for k, v in person.items():
    print(k, '=', v)
```

- 求出现次数最多的元素

```
chars = ['a', 'b', 'd', 'a', 'e', 's', 'a','b']
#将列表的元素和出现的次数组成字典
char_chount = {}
for char in chars:
    if char in char_chount:
        char_chount[char] +=1
    else:
        char_chount[char] = 1#{'a' : 1 }
print(char_chount)
# {'a': 3, 'b': 2, 'd': 1, 'e': 1, 's': 1}
```

```
chars = ['a', 'b', 'd', 'a', 'e', 's', 'a','b']
#将列表的元素和出现的次数组成字典
char_chount = {}
for char in chars:
    if char not in char_chount:
        char_chount[char] = chars.count(char)
print(char_chount)
```

```
chars = ['a', 'b', 'd', 'a', 'e', 's', 'a','b']
#将列表的元素和出现的次数组成字典
char_chount = {}
for char in chars:
    if char not in char_chount:
        char_chount[char] = chars.count(char)
print(char_chount)
vs = (char_chount.values())
#可以使用内置函数max取最大数
max_count = max(vs)
for k, v in char_chount.items():
    if v == max_count:
        print(k)
```

- 练习：让用户输入姓名，如果姓名存在，提示用户；如果姓名不存在，继续输入年龄，并存入列表中

  in运算符如果直接用在字典上是判断key是否存在，而不是value

```
persons = [
    {'name': 'zhansan', 'age': '21'},
    {'name': 'lisi', 'age': '32'},
    {'name': 'jerry', 'age': '45'}
]
username = input('请输入姓名')
for person in persons:
    if person['name'] == username:
        print('宁输入的姓名已经存在')
        break
else:
    print('您输入的姓名不存在')
    new_person = {'name' : username}
    y = int(input('请输入您的年龄：'))
    new_person['age'] = y
    persons.append(new_person)
    print(persons)
```

- 练习：将字典dict1 = {'a' : 100, 'b' : 200, 'c' : 300}变成dict2 = {100, 'a', 200, 'b', 300, 'c'}

```
dict1 = {'a': 100, 'b': 200, 'c': 300}
dict2 = {}
for k,v in dict1.items():
    dict2[v] = k
print(dict2)
```

  - 用字典推导式实现

```
dict1 = {'a': 100, 'b': 200, 'c': 300}
dict1 = {v: k for k, v in dict1.items()} #字典推导式
```

集合的使用

- 集合是一个不重复的无序的数据集合，可以使用{}或者set 来表示
- {}有两种意思：如果里面是键值对就是字典，如果是单个的值就是一个集合

- 会自动去除里面重复的数据，而且每次打印出来的顺序都不一样

```
names = {'zhangsan', 'lisi', 'zhangsan', 'jack'}
print(names)
#{'lisi', 'zhangsan', 'jack'}
```

- set可以进行增删改查

```
names.add('mary')
print(names)
#{'jack', 'mary', 'lisi', 'zhangsan'}
```

```
names.clear()
print(names)
#清空一个集合
```

```
names.pop()
print(names)
#随机删除一个元素
```

```
names.remove('zhangsan')
print(names)
#删除指定的元素，如果元素不存在就会报错
```

```
print(names.union({"赵四", 'heilongjiang'}))
print(names)
#将两个集合连在一起，不改变原来的集合
#{'zhangsan', 'jack', 'lisi', 'heilongjiang', '赵四'}
#{'zhangsan', 'jack', 'lisi'}
```

```
names = {'zhangsan', 'lisi', 'zhangsan', 'jack'}
names.update({'lijun', 'nihao'})
print(names)
#改变原来的集合
```

- 集合的高级使用
  - 去重

    ```
    first = {'江苏', '北京', '上海', '天津', '广州'}
    second = {'赣州', '广州', '北京', '哈尔滨', '成都'}
    print(first - second)
    #{'江苏', '上海', '天津'}
    ```

  - 求交集

    ```
    first = {'江苏', '北京', '上海', '天津', '广州'}
    second = {'赣州', '广州', '北京', '哈尔滨', '成都'}
    print(first & second)
    #{'广州', '北京'}
    ```

  - 求并集

```
first = {'江苏', '北京', '上海', '天津', '广州'}
second = {'赣州', '广州', '北京', '哈尔滨', '成都'}
print(first | second)
#{'广州', '天津', '北京', '上海', '哈尔滨', '成都', '江苏', '赣州'}
```

- 求差集的并集(将不同的元素组合在一起)

```
first = {'江苏', '北京', '上海', '天津', '广州'}
second = {'赣州', '广州', '北京', '哈尔滨', '成都'}
print(first ^ second)
#{'成都', '江苏', '上海', '哈尔滨', '天津', '赣州'}
```