目录

基本数据类型

- 一、什么是数据类型
  - 数值类型
  - 序列类型
  - 散列类型
  - 其他类型
- 二、数值类型
  - 1. 整数类型(int)
    - 1.1变量
    - 1.2赋值运算符
    - 1.3 type函数和print函数
    - 1.4 整数的常见表示形式
    - 1.5 整数的取值范围
  - 2. 浮点数类型(float)
    - 2.1 浮点数的表现形式
    - 2.2 数学运算符
    - 2.3 赋值运算符
    - 2.4 浮点数的不精确性
      - 思考：
        - 拓展：高精度浮点运算类型
      - 思考：
    - 2.4 浮点数和整数的相互转化
  - 3. 复数
- 三，序列类型
  - 1. 字符串
    - 1.1 字符串的定义
      - 定义空字符串
    - 1.2 字符串的索引
    - 1.3 字符串的切片
      - 思考

        - 1.4 字符串拼接
        - 1.5 字符串常用方法
        - 1.6 字符串和数值的相互转化
        - 1.7 转义符
        - 1.8 字符串格式化
            - % 字符串格式化
            - format函数格式化
    - 2. 列表
        - 2.1 列表的定义
        - 2.2 列表的拼接
        - 2.3 列表的索引和切片
        - 2.4 列表的常用操作
            - 2.4.1 修改元素
            - 2.4.2 增加元素
            - 2.4.3 删除元素
        - 2.5 列表的其他方法
        - 2.6 字符串和列表的转换
    - 3. 元组
        - 3.1 元组的定义
        - 3.2 元组的索引和切片
        - 3.2 元组的常用操作
        - 3.3 元组的常用方法
        - 3.4 len函数
    - 4. 可变类型与不可变类型
    - 5.赋值与深浅拷贝
        - 5.1 赋值
        - 5.2 浅拷贝
        - 5.3 深拷贝
- 四，散列类型
    - 1. 集合
        - 1.1 集合定义
        - 1.2 集合的常用操作
            - 1.2.1 添加元素
            - 1.2.2 删除元素

        - 1.2.3 集合运算
            - 交集
            - 并集
            - 补集
            - 对称差集
        - 1.3 集合去重
    - 2. 字典
        - 2.1 字典的定义
        - 2.2 字典的索引
        - 2.3 字典的常用操作
            - 2.3.1 增加元素
            - 2.3.2 修改元素
            - 2.3.3 删除元素
            - 2.3.4 查询元素
- 五，其他类型
    - 1. 布尔型
        - 1.1 比较运算符
        - 1.2 逻辑运算符
        - 1.3 成员运算符
        - 1.4 布尔型运算
        - 1.5 布尔类型转换
    - 2. None


```
1
```

基本数据类型

一、什么是数据类型

编程语言通过一些复杂的计算机物理底层机制，创造不同类型的数据，用来表示现实世界中的不同信息，以便于计算机更好的存储和计算。

每种编程语言都会有一些基本的数据类型用来表示现实世界中的常见信息。

Python中的常见数据类型如下

数值类型

名称	描述
int(整数)	数学概念中的整数
float(浮点数)	数学概念中的实数
complex(复数)	数学概念中的复数

序列类型

名称	描述
str(字符串()	字符串是字符的序列表示，用来表示文本信息
list(列表)	列表用来表示有序的可变元素集合。例如表示一个有序的数据组。
tuple(元组)	元组用来表示有序的不可变元素集合。

散列类型

名称	描述
set(集合)	数学概念中的集合，无限不重复元素的集合
dict(字典)	字典是无序键值对的集合。用来表示有关联的数据，例如表示一个人的基本信息。

其他类型

名称	描述
bool(布尔型)	bool型数据只有两个元素，True表示真，False表示假。用来表示条件判断结果。
None	None表示空。

二、数值类型

1. 整数类型(int)

与数学中的整数概念一致

```
age = 18
```

其中age是变量名，=是赋值运算符，18是值。

上面的代码表示创建一个整数18然后赋值给变量age。

1.1变量

在程序运行过程中会有一些中间值，在稍后的执行中会用到，这时可以将这些中间值赋值给变量，然后在后面的代码中通过调用这些变量名来获取这些值。可以简单的理解为给这些值取一个别名，这个别名就代表这个值。

变量的命名规则：

1. 由字符，数字和下划线_组成
2. 不能以数字开头
3. 不能是关键字
4. 变量名大小写敏感

```
age = 18
Age = 19
```

```
1age = 19
```

```
  File "<ipython-input-3-00afb59c3a30>", line 1
    1age = 19
     ^
SyntaxError: invalid syntax
```

1.2赋值运算符

在python中=是赋值运算符，而不是数学意义上的等于号。python解释器会先计算=右边的表达式，然后将结果复制给=左边的变量。

```
res = 1
res = res + 1
res
```

1.3 type函数和print函数

python提供了内建函数type用来查看值或者变量的类型。

只需要将变量或者值作为参数传入type函数即可。

```
type(age)
type(18)
```

print函数用来在屏幕上输出传入的数据的字符串表现形式，是代码调试最重要的函数。

```
print(age)
print(type(age))
# 注意交互式输出和print函数输出的区别
```

1.4 整数的常见表示形式

在python中整数最常见的是10进制整数，也有二进制，八进制和十六进制。

```
a = 10  # 十进制
print('a的类型为: ', type(a))
```

```
b = 0b1110  # 二进制
print('b的类型为: ', type(b))
```

```
c = 0o57    # 八进制
print('c的类型为: ', type(c))
```

```
d = 0xa5c   # 十六进制
print('d的类型为: ', type(d))
```

1.5 整数的取值范围

整数类型的理论取值范围是[-无穷，无穷]，实际取值范围受限于运行python程序的计算机内存大小。

2. 浮点数类型(float)

与数学中的实数概念一致，也可以理解为有小数。

```
a = 0.0
print('a的类型为: ', type(a))
```

2.1 浮点数的表现形式

在python中浮点数可以表示为`a.b`的格式，也可以表示为小写或大写`E`的科学计算法。例如：

```
a = 0.0
print('a的类型为: ', type(a))
```

```
b = 76.
print('b的类型为: ', type(b))
```

```
c = -3.1415926
print('c的类型为: ', type(c))
```

```
d = 9.5e-2
print('d的类型为: ', type(d))
```

注意：`76.`虽然小数部分为0，但是它的数据类型为float。

2.2 数学运算符

与数学中的常用运算符一致

运算符	描述
+	加法运算符 1+1
-	减法运算符 3-2
*	乘法运算符 9*9
/	除法运算符 9/3，除法运算后的结果一定为float类型
//	整除运算符 10/3，也称为地板除向下取整
%	取模运算符 10%3,表示10除以3取余数
**	幂次运算符 2**3，表示2的3次幂
()	括号运算符，括号内的表达式先运算 （1+2）* 3

```
1+1
```

```
3-2
```

```
9*9
```

```
9/3
```

```
10//3
```

```
10%3
```

```
2**3
```

```
(1+2)*3
```

2.3 赋值运算符

运算符	描述	实例
=	等于-简单的赋值	c = a + b print(c) # 30
+=	加等于	c += a等同于c = c + a
-=	减等于	c -= a等同于c = c - a
*=	乘等于	c *= a等同于c = c * a
/=	除等于	c /= a等同于c = c/a
%=	取余等于	c%=a等同于c = c%a
**=	幂等于	c ** =a等同于c = c ** a
//=	取整除等于	c//=a等同于c = c//a

体现了程序员的"懒惰"，这种懒惰大力提倡，使得代码简洁和高效。

```
a = 1
a += 2
a
```

2.4 浮点数的不精确性

整数和浮点数在计算机中的表示不同，python提供无限制且准确的整数计算，浮点数却是不精确的，例如：

```
0.2+0.1
```

根据sys.float_info.dig的值，**计算机只能提供15个数字的准确性**。浮点数在超过15位数字计算中产生的误差与计算机内部采用二进制运算有关。

```
import sys

print(sys.float_info.dig)
```

思考：

3.1415926535897924*1.23456789 的计算怎么准确

拓展：高精度浮点运算类型

```
import decimal
a = decimal.Decimal('3.141952653')
b = decimal.Decimal('1.23456789')
print(a * b)
```

思考：

浮点数可以表示所有的整数数值，python为何要同时提供两种数据类型?

```
相同的操作整数要比浮点数快5-20倍
```

2.4 浮点数和整数的相互转化

```
a = 1.9
# 转化为整数
# 通过调用int函数，提取浮点数的整数部分
b = int(a)
print(b, type(b))

```

```
c = 2
# 转化为浮点数
# 通过调用float函数，将整数转化为小数部分为0的浮点数
d = float(c)
print(d, type(d))
```

3. 复数

科学计算中的复数。

```
a = 12.3+4j
print('a的类型为：', type(a))
# 运行结果：a的类型为：  <class 'complex'>
print(a.real)
print(a.imag)
```

三，序列类型

序列类型用来表示有序的元素集合。

1. 字符串

在python中字符串是使用单引号，双引号，三引号包裹起来的字符的序列，用来表示文本信息。

1.1 字符串的定义

```
a = 'a'
b = "bc"
c = """hello,world"""
d = '''hello,d'''
e = """
    hello,
    world!
    """
print('a的类型为: ', type(a))    # a的类型为:  <class 'str'>
print('b的类型为: ', type(b))    # b的类型为:  <class 'str'>
print('c的类型为: ', type(c))    # c的类型为:  <class 'str'>
print('d的类型为: ', type(d))    # d的类型为:  <class 'str'>
print('e的类型为: ', type(e))    # e的类型为:  <class 'str'>
```

使用单引号和双引号进行字符串定义没有任何区别，当要表示字符串的单引号时用双引号进行定义字符串，反之亦然。

一对单引号或双引号只能创建单行字符串，三引号可以创建多行表示的字符串。三双引号一般用来做多行注释，表示函数，类定义时的说明。

定义空字符串

```
a = ""
b = ''
```

1.2 字符串的索引

任何序列类型中的元素都有索引用来表示它在序列中的位置。

字符串是字符的序列表示，单个字符在字符串中的位置使用索引来表示，也叫下标。

索引使用整数来表示。

通过索引可以获取字符串中的单个字符

语法如下：

```
str[index]
```

```
s = 'hello'
print(s[0])
print(s[-1])
```

注意python中序列的索引从0开始。

1.3 字符串的切片

获取序列中的子序列叫切片。

字符串的切片就是获取字符串的子串。

字符串切片的语法如下：

```
str[start:end:step]
```

start表示起始索引，end表示结束索引，step表示步长。

str[m:n:t]表示从字符串索引为m到n之间不包含n每隔t个字符进行切片。

当step为1的时候可以省略。

特别的，当step为负数时，表示反向切片。

```
s = '0123456789'
print(s[1:5])  # 包头不包尾
```

```
print(s[:5])  # 从头开始切可以省略start
```

```
print(s[1:])  # 切到末尾省略end
```

```
print(s[1::2]) # 步长为2进行切片
```

```
print(s[1::-2]) # 步长为负数反向切片
```

思考

获取一个字符串的逆串，例如'abc'的逆串是'cba'。

```
a = 'abc'
a[2::-1]
```

1.4 字符串拼接

python中可以通过+拼接两个字符串

```
a = 'hello'
b = ' '
c = 'world!'
print(a+b+c)
```

字符串和整数进行乘法运算表示重复拼接这个字符串

```
print('*' * 10)
```

1.5 字符串常用方法

通过内建函数dir可以返回传入其中的对象的所有方法名列表。

```
dir(str)
```

通过内建函数help可以返回传入函数的帮助信息。

```
help(str.replace)
```

1.6 字符串和数值的相互转化

1 和 '1'不同，1.2和'1.2'也不相同,但是它们可以相互转化

```
# 整数和字符串之间的转化
int('1')
```

```
str(1)
```

```
# 浮点数和字符串之间的转化
float('1.2')
```

```
str(1.2)
```

```
# 尝试 int('1.2')看看结果会是什么
int('1.2')
```

1.7 转义符

在需要在字符中使用特殊字符时，python 用反斜杠\ 转义字符。常用转义字符如下表：

(在行尾时)	续行符
\\	反斜杠符号
\'	单引号
\"	双引号
\a	响铃
\n	换行
\t	横向制表符
\r	回车
\f	换页

```
print('窗前明月光，\n疑是地上霜。')  # 输出换行
```

```
print('对\\错')   # 输出反斜杠本身
```

```
print('\'')      # 输出单引号本身
```

1.8 字符串格式化

在实际工作中经常需要动态输出字符。

例如，我们通过程序计算计算机的内存利用率，然后输出

10:15 计算机的内存利用率为30%

其中下划线内容会动态调整，需要根据程序执行结果进行填充，最终形成上述格式的字符串输出。

python支持两种形式的字符串格式化

%字符串格式化

语法格式如下：

```
%[(name)][flags][width][.precision]typecode
```

- (name) 可选，用于选择指定的key
- flags 可选，可供选择的值有，注意只有在和数值类型的typecode配合才起作用
  - +, 右对齐，正数前加正号，负数前加负号
  - -, 左对齐，正数前无符号，负数前加负号
  - 空格, 右对齐，正数前加空格，负数前加负号
  - 0, 右对齐，正数前无符号，复数前加负号；用0填充空白处
- width，可选字符串输出宽度
- .precision 可选，小数点后保留位数，注意只有在和数值类型的typecode配合才起作用
- typecode必选
  - s，获取传入对象的字符串形式，并将其格式化到指定位置
  - r，获取传入对象的__repr__方法的返回值，并将其格式化到指定位置
  - c，整数：将数字转换成其unicode对应的值，10进制范围为 0 <= i <= 1114111（py27则只支持0-255）；字符：将字符添加到指定位置
  - o，将整数转换成 八 进制表示，并将其格式化到指定位置
  - x，将整数转换成十六进制表示，并将其格式化到指定位置
  - d，将整数、浮点数转换成 十 进制表示，并将其格式化到指定位置
  - e，将整数、浮点数转换成科学计数法，并将其格式化到指定位置（小写e）
  - E，将整数、浮点数转换成科学计数法，并将其格式化到指定位置（大写E）
  - f， 将整数、浮点数转换成浮点数表示，并将其格式化到指定位置（默认保留小数点后6位）
  - F，同上
  - g，自动调整将整数、浮点数转换成 浮点型或科学计数法表示（超过6位数用科学计数法），并将其格式化到指定位置（如果是科学计数则是e；）`

- G，自动调整将整数、浮点数转换成 浮点型或科学计数法表示（超过6位数用科学计数法），并将其格式化到指定位置（如果是科学计数则是E；）`
- %，*当字符串中存在格式化标志时，需要用 %% 表示一个百分号*

```
res = '%s计算机的内存利用率为%s%%' % ('11:15', 75)
# '%s'作为槽位和 % 号后提供的值一一对应
res = '%(time)s计算机的内存利用率为%(percent)s%%' % {'time':'11:15', 'percent': 75}

# 输出两位数的月份，例如01，02
res = '%02d' % 8

# 保留2为小数
res = '%(time)s计算机的内存利用率为%(percent).2f%%' % {'time':'11:15', 'percent': 75.123}
```

format函数格式化

%的字符串格式化继承自C语言，python中给字符串对象提供了一个format函数进行字符串格式化，且功能更强大，并且大力推荐，所以我们要首选使用。

基本语法是<模板字符串>.format(<逗号分隔的参数>)

在模板字符串中使用{}代替以前的%作为槽位

```
'{}计算机的内存利用率为{}%'.format('11:15', 75)
```

当format中的参数使用位置参数提供时，{}中可以填写参数的整数索引和参数一一对应

```
'{0}计算机的内存利用率为{1}%'.format('11:15', 75)
```

当format中的参数使用关键字参数提供时，{}中可以填写参数名和参数一一对应

```
'{time}计算机的内存利用率为{percent}%'.format(time='11:15', percent=75)
```

{}中除了可以写参数索引外，还可以填写控制信息来实现更多的格式化功能，语法如下

```
{<参数序号>:<格式控制标记>}
其中格式控制标记格式如下
[fill][align][sign][#][0][width][,][.precision][type]
```

- fill　　【可选】空白处填充的字符

- align 【可选】对齐方式（需配合width使用）
-
  - <，内容左对齐
  - >，内容右对齐(默认)
  - =，内容右对齐，将符号放置在填充字符的左侧，且只对数字类型有效。 即使：符号+填充物+数字
  - ^，内容居中
- sign 【可选】有无符号数字
  - +，正号加正，负号加负；
  - -，正号不变，负号加负；
  - 空格 ，正号空格，负号加负；
- # 【可选】对于二进制、八进制、十六进制，如果加上#，会显示 0b/0o/0x，否则不显示
- , 【可选】为数字添加分隔符，如：1,000,000
- width 【可选】格式化位所占宽度
- .precision 【可选】小数位保留精度
- type 【可选】格式化类型
  - 传入" 字符串类型 "的参数
    - s，格式化字符串类型数据
    - 空白，未指定类型，则默认是None，同s
  - 传入" 整数类型 "的参数
    - b，将10进制整数自动转换成2进制表示然后格式化
    - c，将10进制整数自动转换为其对应的unicode字符
    - d，十进制整数
    - o，将10进制整数自动转换成8进制表示然后格式化；
    - x，将10进制整数自动转换成16进制表示然后格式化（小写x）
    - X，将10进制整数自动转换成16进制表示然后格式化（大写X）
  - 传入" 浮点型或小数类型 "的参数
    - e， 转换为科学计数法（小写e）表示，然后格式化；
    - E， 转换为科学计数法（大写E）表示，然后格式化;
    - f， 转换为浮点型（默认小数点后保留6位）表示，然后格式化；
    - F， 转换为浮点型（默认小数点后保留6位）表示，然后格式化；
    - g， 自动在e和f中切换
    - G， 自动在E和F中切换
    - %，显示百分比（默认显示小数点后6位）

```
res = "numbers: {:b},{:o},{:d},{:x},{:X}, {:%}".format(15, 15, 15, 15, 15, 15.87623, 2)

res = "numbers: {0:b},{0:o},{0:d},{0:x},{0:X}, {0:%}".format(15)

# 输出两位数的月份，例如01，02
res = '{:0>2}'.format(8)

# 保留2为小数
res = '{time}计算机的内存利用率为{percent:.2%}'.format(time='11:15', percent=0.75123)
```

2. 列表

python中列表用来表示有序可变元素的集合，元素可以是任意数据类型。

2.1 列表的定义

列表由一对中括号进行定义，元素与元素直接使用逗号隔开。

```
a = []                          # 空列表
b = ["a", "b", "cde"]           # 字符串列表项
c = [1, "b", "c"]               # 数字列表项
d = [1, "b", []]                # 列表列表项
e = [1, "b", [2, "c"]]          # 列表作为列表的元素叫做列表的嵌套
L = ['a', 'b', 'c']     # 功能集案例

print('a的类型为: ', type(a))    # a的类型为:  <class 'list'>
print('b的类型为: ', type(b))    # b的类型为:  <class 'list'>
print('c的类型为: ', type(c))    # c的类型为:  <class 'list'>
print('d的类型为: ', type(d))    # d的类型为:  <class 'list'>
print('e的类型为: ', type(e))    # e的类型为:  <class 'list'>
```

2.2 列表的拼接

像字符串一样，列表之间可以进行加法运算实现列表的拼接，列表可以和整数进行乘法运算实现列表的重复。

```
[1,2,3] +  [4,5,6]
```

```
[1,2,3] * 3
```

注意：上述两种操作都是生成新的列表对象。

2.3 列表的索引和切片

序列的切片操作完全一致，参见字符串

注意嵌套列表的元素获取

```
ls = [1,2,['a','b']]
ls[2][0]
```

2.4 列表的常用操作

python中的列表操作非常灵活，是非常重要和经常使用的数据类型。

2.4.1 修改元素

列表的中的元素可以进行修改，只需使用索引赋值即可。

```
l = [1,2,3]
l[1] = 'a'
print(l)
```

2.4.2 增加元素

给列表添加元素需要使用到列表的方法

`.append(el)`，在列表的末尾添加一个元素

```
l = [1,2,3]
l.append(4)
print(l)
```

`.insert(index, el)`，在列表的指定索引出插入一个元素

```
l = [1,2,3]
l.insert(0,0)
print(l)
```

`.extend(iterable)`，扩展列表，元素为传入的可迭代对象中的元素

```
l = [1,2,3]
l.extend([4,5,6])
print(l)
```

2.4.3 删除元素

`.pop(index=-1)`，删除指定索引的元素,并返回该元素，没有指定索引默认删除最后一个元素

```
l = [1,2,3]
l.pop()

```

```
print(l)
```

```
l.pop(0)
```

```
print(l)
```

`.remove(value)`，从列表中删除第一个指定的值value，如不不存在value则报错。

```
l = [1,2,3,1]
l.remove(1)
print(l)
```

`.clear()`，清空列表，原列表变成空列表

```
l = [1,2,3]
l.clear()
print(l)
```

2.5 列表的其他方法

`.copy()` 返回一个列表的浅拷贝。在讲可变与不可变类型的时候再详细讨论。

`.count(value)`，统计列表中value的出现次数

```
l = [1,2,3,1]
l.count(1)
```

`.index(self, value, start=0, stop=9223372036854775807)`，返回列表中指定值value的第一个索引，不存在则报错

```
l = [1,2,3,1]
l.index(1)
```

```
l.index(1,1)
```

.reverse()，翻转列表元素顺序

```
l = [1,2,3]
l.reverse()
print(l)
```

.sort(key=None, reverse=False)，对列表进行排序，默认按照从小到大的顺序，当参数reverse=True时，从大到小。注意列表中的元素类型需要相同，否则抛出异常。

```
l = [2,1,3]
l.sort()
print(l)
```

```
l.sort(reverse=True)
print(l)
```

2.6 字符串和列表的转换

字符串是字符组成的序列，可以通过list函数将字符串转换成单个字符的列表。

```
s = 'hello world!'
s_list = list(s)
print(s_list)
```

由字符组成的列表可以通过字符串的join方法进行拼接

```
# 接上面的案例
''.join(s_list)
```

3. 元组

元组表示有序不可变元组的集合，元素可以是任意数据类型，可以说元组就是不可变的列表。

3.1 元组的定义

元组通过一对小括号进行定义，元组之间使用逗号隔开。

```
a = ()                      # 空元祖
b = ("a", "b", "cde")       # 字符串
```

```
c = (1, "b", "c")            # 数字
d = (1, "b", [])             # 列表
e = (1, "b", (2, "c"))       # 元祖
f = 1,2
L = ('a', 'b', 'c')      # 功能集案例

print('a的类型为: ', type(a))    # a的类型为:  <class 'tuple'>
print('b的类型为: ', type(b))    # b的类型为:  <class 'tuple'>
print('c的类型为: ', type(c))    # c的类型为:  <class 'tuple'>
print('d的类型为: ', type(d))    # d的类型为:  <class 'tuple'>
print('e的类型为: ', type(e))    # e的类型为:  <class 'tuple'>
print('f的类型为: ', type(f))    # f的类型为:  <class 'tuple'>
```

注意单元素元组的定义

```
g = ('hello')
h = ('hello',)
print('g的类型为: ', type(g))    # g的类型为:  <class 'str'>
print('h的类型为: ', type(h))    # h的类型为:  <class 'tuple'>
```

3.2 元组的索引和切片

序列的索引和切片完全一致，参加字符串。

3.2 元组的常用操作

元组的元素不能修改，增加，删除其他和列表的操作一致。

元组利用不可修改的特性，应用在多变量赋值和函数多返回值上。

```
a, b = (1, 2)
# 经常简写为a, b= 1, 2
```

当然多变量赋值时可以使用可迭代对象，但是元组最安全，它是不可变的。

关于函数多返回值的问题我们后面再讲

3.3 元组的常用方法

元组只有两个公有方法count,index用法与列表相同。

3.4 len函数

python内建函数len可以获取对象中包含的元素个数

```
s = 'hello'
l = [1,2,3]
t = (1,2,3)
len(s)
```

```
len(l)
```

```

len(t)
```

4. 可变类型与不可变类型

python中的数据类型根据底层内存机制分为可变和不可变两种。基本数据类型中可变类型有列表，集合和字典，其他为不可变类型。直观的体现为，不可变类型数据创建后不能修改，只能重新创建。

通过内建函数hash可以对数据进行运算，凡是不可hash的都是可变类型，可以hash的是不可变类型

```
hash([1,2])
```

5.赋值与深浅拷贝

5.1 赋值

python是解释型编程语言，当解释器在碰到赋值语句时它首先会计算赋值符号右边的表达式的值，然后再创建左边的变量。变量中实际存储的是值在内存中的地址，引用变量时通过地址指向内存中的值。通过内建函数id可以查看解释器中变量的虚拟内存地址整数值。

```
a = 1
id(a)
```

将一个变量赋值给另外一个变量时，并不会创建新的值，只是新变量会指向值的内存地址

```
a = 1
b = a
id(a) == id(b)
```

对于字符串和数字这样的简单数据类型，当上例中的变量a自加1时，会创建一个新值重新，它不会改变原来的值。因此对变量b没有影响。

```
a += 1
print(a)
print(b)
```

但是看下面的案例

```
ls = [1,2,3]
ln = ls
ls[0] = 2
print(ls)
```

会发现变量ls在修改列表的值后，变量ln的值也发生了同样的改变，这是因为ls，ln指向相同的列表。对可变数据类型进行变量赋值时要考虑这个特性。

5.2 浅拷贝

导入copy模块中的copy函数就是浅拷贝操作

```
import copy
a = 123
s = 'hello'
b = copy.copy(a)
d = copy.copy(s)
a += 1
print(b)
```

对于字符串数字简单数据类型来说，浅拷贝会创建新的数据。修改原变量不会影响新变量。

拓展：注意为了进行内存优化，对于常用整数-5-256使用内存驻留机制，即会开辟一块内存空间专门存放这些整数。字符串也有相应的优化。了解即可。

对于复杂数据类型，列表，元组，字典，集合等浅拷贝会有不一样的结果。

当对复杂数据类型进行浅拷贝时，会创建一个新的数据，会拷贝源数据中类型为字符串和数字的元素，但是不会拷贝复杂类型数据，而是会直接引用。

```
ls = [1,'2',['a','b']]
ln = copy.copy(ls)
print(ln)
```

```
# 修改ls
ls[0] = 2
print(ls)
```

```
print(ln)
# 没有影响ln
```

```
ls[2][0] = 'b'
print(ls)
print(ln)
# ln中的列表元素也改变了
```

5.3 深拷贝

字符串和数字的深浅拷贝一致。

复杂数据类型进行深拷贝会对数据中的所有元素完全重新复制一份，不管有多少层嵌套，互不影响。

```
import copy
ls = [1,2,3,['a', 'b']]
# 深拷贝使用deepcopy
ln = copy.deepcopy(ls)
ls[3][0]='b'
print(ls)
print(ln)
```

四，散列类型

散列类型用来表示无序集合。

1. 集合

python中集合类型与数学中的集合类型一致，用来表示无序不重复元素的集合。

1.1 集合定义

集合使用一对大括号{}进行定义，元素直接使用逗号隔开。集合中的元素必须是不可变类型。

```
b = {1, 2, 3, 4, 5, 6}
c = {1,2,'a',('a',),1.5}      # 集合中元素必须是不可变类型

print('a的类型为：', type(a))    # a的类型为：  <class 'set'>
print('b的类型为：', type(b))    # b的类型为：  <class 'set'>
```

注意空集合的定义方式是set()

```
a = set()                          # 空集合
# 注a = {} 是空字典
```

1.2 集合的常用操作

1.2.1 添加元素

集合添加元素常用函数有两个：add和update

set.add(obj),向集合中添加元素obj，如果集合中不存在则添加

```
s = {1,2}
s.add(1)
print(s)
```

```
s.add(3)
print(s)
{1,2,3}
```

set.update(iterable)，向集合中添加多个元素，如何集合中不存在则添加

s = {1,2}
s.update({2,3})
print(s)

```
s.update([3,4])
print(s)
```

1.2.2 删除元素

set.pop()随机删除并返回集合中的一个元素，如何集合中元素为空则抛出异常。

```
s = {'a','b','c'}
s.pop()
```

```
s.pop('d')
```

set.remove(ele),从集合中删除元素ele，如果不存在则抛出异常。

```
s = {'a','b','c'}
s.remove('a')
print(s)
```

set.discard(ele),从集合中删除元素ele，如果不存在不做任何操作

```
s = {'a','b','c'}
s.discard('d')
print(s)
```

set.clear(),清空集合

```
s = {1,2,3}
s.clear()
print(s)
```

1.2.3 集合运算

数学符号	python运算符	含义	定义
∩	&	交集	一般地,由所有属于A且属于B的元素所组成的集合叫做AB的交集.
∪	|	并集	一般地,由所有属于集合A或属于集合B的元素所组成的集合,叫做AB的并集
-或\	-	相对补集/差集	A-B，取在A集合但不在B集合的项
	^	对称差集/反交集	A^B，取只在A集合和只在B集合的项，去掉两者交集项

交集

取既属于集合A和又属于集合B的项组成的集合叫做AB的交集

```
s1 = {1,2,3}
s2 = {2,3,4}
s = s1 & s2
print(s)
```

并集

集合A和集合B的所有元素组成的集合称为集合A与集合B 的并集

```
s1 = {1,2,3}
s2 = {2,3,4}
s = s1|s2
print(s)
```

补集

取在集合A中不在集合B中的项组成的集合称为A相对B的补集

```
s1 = {1,2,3}
s2 = {2,3,4}
s = s1-s2
print(s)
```

对称差集

取不在集合AB交集里的元素组成的集合称为对称差集，也叫反交集

```
s1 = {1,2,3}
s2 = {2,3,4}
s = s1^s2
print(s)
```

1.3 集合去重

集合具有天生去重的性质，因此可以利用它来去除序列中的重复元素

```
ls = [1,1,2,3,4,4,3,2,5]
ls = list(set(ls))
print(ls)
```

```
set('aabbcc')

```

2. 字典

因为集合无序，因此不能很便捷的获取特定元素。利用集合元素不重复的特性，使集合中的元素映射值组成键值对，再通过键来获取对应的值。

2.1 字典的定义

python中的字典数据类型就是键值对的集合，使用一对大括号进行定义，键值对之间使用逗号隔开，键和值使用冒号分割。

字典中的键必须是不可变数据类型，且不会重复，值可以使任意数据类型。

```
a = {}                      # 空字典
b = {
    1: 2,                   # key:数字; value: 数字
    2: 'hello',             # key:数字; value: 字符串
    ('k1',): 'v1',          # key:元祖; value: 字符串

```

```
    'k2': [1, 2, 3],          # key:字符串; value: 列表
    'k3': ('a', 'b', 'c'),    # key:字符串; value: 元祖
    'k4': {                   # key:字符串; value: 字典
        'name': 'feifei',
        'age': '18'
    }
}

print('a的类型为: ', type(a))    # a的类型为:  <class 'dict'>
print('b的类型为: ', type(b))    # b的类型为:  <class 'dict'>
```

2.2 字典的索引

字典通过键值对中的键作为索引来获取对应的值。字典中的键是无序的。

```
d = {1:2, 'key': 'value'}
print(d[1])
```

```
print(d['key'])
```

这种方式很好的将键和值联系起来，就像查字典一样。

2.3 字典的常用操作

2.3.1 增加元素

字典可以直接利用key索引赋值的方式进行添加元素,如果key存在则修改字典

```
d = {'name': 'Felix'}
d['age'] = 18
print(d)
```

`dict.update(new_dict)`，将new_dict合并进dict中。

```
d = {'name': 'Felix'}
n_d = {'age':18, 'sex':'男'}
d.update(n_d)
print(d)
```

```
d.update({'sex': '女','height': 170})  # 当有重复key的时候会覆盖原值
print(d)
```

2.3.2 修改元素

直接通过key索引赋值的方式可以对字典进行修改，如果key不存在则添加

d = {'name': 'Felix'}
d['name'] = 'felix'
print(d)

2.3.3 删除元素

`dict.pop(key[,d])`,删除指定的key对应的值并返回该值，如果key不存在则返回d，如果没有给定d，则抛出异常

```
d = {'name': 'Felix','age': 18}
d.pop('ag',)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)

<ipython-input-8-a9f7504ca045> in <module>
      1 d = {'name': 'Felix','age': 18}
----> 2 d.pop('ag',)
```

```
KeyError: 'ag'
```

```
print(d)
```

`dict.popitem()`，任意删除字典dict中的一个键值对，并以二元元组`(key,value)`的方式返回

```
d = {'name': 'Felix','age': 18}
d.popitem()
```

```
('age', 18)
```

2.3.4 查询元素

通过key索引可以直接获取key对应的值，如果key不存在则抛出异常。

```
d = {1:2, 'key': 'value'}
print(d[1])
```

```
2
```

`dict.get(key,default=None)`,获取key对应的value如果不存在返回default

```
d = {1:2, 'key': 'value'}
d.get(1)
```

```
2
```

```
d.get('name','a')
```

```
'a'
```

五，其他类型

1. 布尔型

条件表达式的运算结果返回布尔型，布尔型数据只有两个，`True`和`False`表示真和假。

1.1 比较运算符

运算符	描述	实例
==	等于-比较对象是否相等	print(a==b) # False
is	等于-比较对象的内存地址是否相同	
!=	不等于	print(a!=b) # True
>	大于	print(a>b) # False
<	小于	print(a<b) # True
>=	大于等于	print(a>=b) # False
<=	小于等于	print(a<=b) # True

```
1 1 == 2
```

```
1 1 != 2
```

```
1 1 > 2
```

```
1 1 < 2
```

```
1 1 >= 2
```

```
1 1 <= 2
```

```
1 a = [1,2]
2 b = [1,2]
```

```
1 a == b
```

```
1 a is b
```

1.2 逻辑运算符

运算符	描述	实例
and	与，如果x为False,x and y返回False，否则返回y的值	print(a and b) # True
or	或，如果x为True,x and y返回True，否则返回y的值	print(a or b) # True
not	非，如果x为True，返回False，反之，返回True	print(not a) # False

```
True and True
```

```
False and True
```

```
0 and 1  # 短路，and运算符如果第一个表示式的值为False则返回第一个表达式的值，否则返回第二个表达式的值
```

```
1 and False
```

```
True or False
```

```
False or False
```

```
1 or 2  # 短路，or运算符入股第一个表达式的结果为True则返回第一个表达式的值，否则返回第二个表达式的值
```

```
0 or ''
```

```
not False
```

```
not 0
```

1.3 成员运算符

运算符	描述	实例
in	如果在指定的序列中找到值返回True，否则False	`L = [1, 2, 3, 4, 5] a = 3 print(a in L) # True`
not in	如果在指定的序列中没有找到值返回True，否则False	`print(a not in L) # False`

```
ls = [1,2,3,4,5]
1 in ls
```

```
s = ['abcdefg']
'a' in s
```

```
t = (1,2,3)
4 in t
```

```
False
```

```
d = {'name': 'Felix','age':18}
'name' in d
```

```
st = {1,2,3}
1 in st
```

1.4 布尔型运算

布尔型数据可以和数值类型数据进行数学计算，这时`True`表示整数`1`,`False`表示整数`0`

```
True * 1.2
```

```
1.2
```

```
False * 1
```

```
0
```

1.5 布尔类型转换

任意数据都可以通过函数`bool`转换成布尔型。

在python中，None, 0（整数），0.0（浮点数），0.0+0.0j（复数），""(空字符串)，空列表，空元组，空字典，空集合的布尔值都为`False`，其他数值为`True`

```
bool(0)
```

```
bool(0.0)
```

```
bool(0.0+0.0j)
```

```
bool("")
```

```
bool([])
```

```
bool(())
```

```
bool({})
```

```
bool(set())
```

2. None

`None`是python中的特殊数据类型，它的值就是它本身`None`，表示空，表示不存在。