内存相关

- 计算机的本质作用：
  - 存储和运算数据
- 计算机的内存空间作用：
  - 存储数据
  - 衡量计算机内存空间大小的单位：
    - bit位：只可以存储一个二进制的数字
    - byte字节：8bit
    - kb，mb，gb
    - 计算机内存空间的大小表示什么？
      - 计算机内存空间越大存储数据的数值越大
      - 8bit：2的8次方
  - 计算中的内存空间会有两个默认的属性：
    - 内存空间的大小
      - 决定内存存储数据的大小
    - 内存空间的地址
      - 用来让cpu寻址的
- 引用

- 变量就是引用，引用就是变量

```
a = 1
#变量a表示的是什么？a表示的是数据1对应内存空间的地址，因此我们说a引用了1
```

基本数据类型

- Python数据类型
  - 在Python的世界，数据类型分两种，内置的和自定义的
- 内置数据类型
  - 内置的包括数字、字符串、布尔、列表、元组、字典、Bytes、集合这些常用的。
- 自定义数据类型
  - 自定义的，一般以类的形式，根据需要组合以上内置类型成为独特的数据类型。

数字类型

- 数字类型用于存储和表示数学意义上的数值。
  - Python 支持三种不同的数字类型，整数、浮点数和复数
- 不可变类型

    - 数字类型是不可变类型。所谓的不可变类型，指的是类型的值一旦有不同了，那么它就是一个全新的对象。数字1和2分别代表两个不同的对象，对变量重新赋值一个数字类型，会新建一个数字对象。
    - 例如：
        - a = 1 #创建数字对象1
        - a = 2 #创建数字对象2，并将2赋值给变量a，a不再指向数字对象1
        - 这里，发生了变化的是变量a的指向，而不是数字对象1变成了数字对象2。初学者可能会比较迷糊，但不要紧，可以先试着接受。
- 整数Int
    - 通常被称为整型，是正或负整数，不带小数点。例如：1，100，-8080，0，等等。
    - 不同进制的整数
        - 表示数字的时候，有时我们还会用八进制或十六进制来表示：
            - 十六进制用0x前缀和0-9，a-f表示，例如：0xff00，0xa5b4c3d2。
            - 八进制用0o前缀和0-7表示，例如0o12
- 浮点数
    - 浮点数也就是小数，如1.23，3.14，-9.01，等等。但是

对于很大或很小的浮点数，一般用科学计数法表示，把10用e替代，1.23x10^9就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

- 数字类型转换
  - 有时候，我们需要对数字的类型进行转换。Python为我们提供了方便的内置的数据类型转换函数。
  - int(x)：
    - 将x转换为一个整数。如果x是个浮点数，则截取小数部分。
  - float(x)：
    - 将x转换到一个浮点数。
- 数学计算
  - 对于数学计算，除了前面提到过的简单的加减乘除等等，更多的科学计算需要导入math这个库，它包含了绝大多数我们可能需要的科学计算函数，如下表

函数	返回值 (描述)
abs(x)	返回数字的绝对值，如abs(-10) 返回 10
ceil(x)	返回数字的上入整数，如math.ceil(4.1) 返回 5
exp(x)	返回e的x次幂(ex),如math.exp(1) 返回2.718281828459045
fabs(x)	返回数字的绝对值，如math.fabs(-10) 返回10.0
floor(x)	返回数字的下舍整数，如math.floor(4.9)返回 4
log(x)	如math.log(math.e)返回1.0,math.log(100,10)返回2.0
log10(x)	返回以10为基数的x的对数，如math.log10(100)返回 2.0
max(x1, x2,...)	返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)	返回给定参数的最小值，参数可以为序列。
modf(x)	返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示。
pow(x, y)	x**y 运算后的值。
round(x [,n])	返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
sqrt(x)	返回数字x的平方根
acos(x)	返回x的反余弦弧度值。
asin(x)	返回x的反正弦弧度值。
atan(x)	返回x的反正切弧度值。
atan2(y, x)	返回给定的 X 及 Y 坐标值的反正切值。
cos(x)	返回x的弧度的余弦值。
hypot(x, y)	返回欧几里德范数 sqrt(x*x + y*y)
sin(x)	返回的x弧度的正弦值。
tan(x)	返回x弧度的正切值。
degrees(x)	将弧度转换为角度,如degrees(math.pi/2) ， 返回90.0
radians(x)	将角度转换为弧度

布尔类型

- 真于假、0和1，都是传统意义上的布尔类型。
- 但在Python语言中，布尔类型只有两个值，True与False。请注意，是英文单词的对与错，并且首字母要大写，不能其

它花式变型。

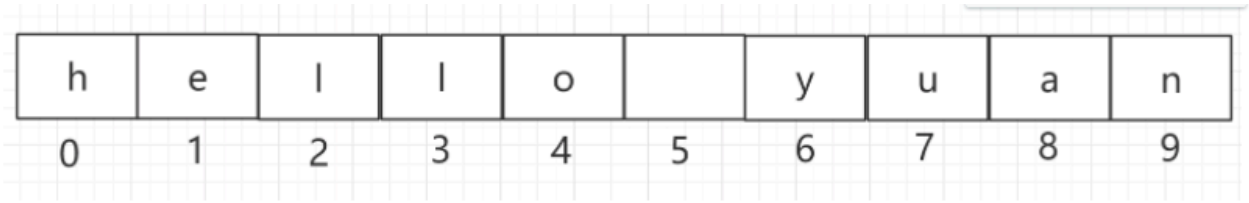
- 所有计算结果返回的结果是True或者False的过程都可以称为布尔运算，例如比较运算。

```
#布尔类型的数据可以进行运算吗?
ret = True + True + False
print(ret)
#True就表示1False表示0
```

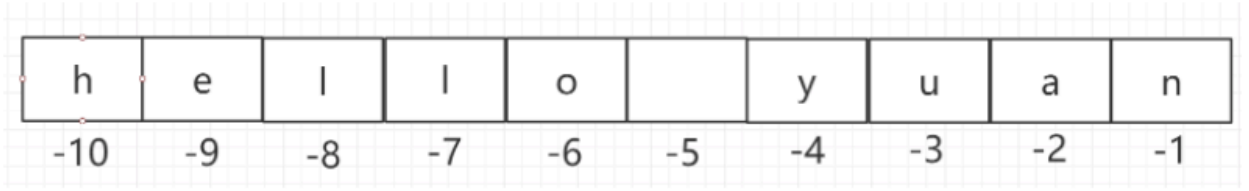
字符串类型

- 字符串是由零个或多个字符组成的有限序列。字符串的内容可以包含字母、标点、特殊符号、中文、日文等全世界的所有字符。
- 在python中字符串是通过单引号`''`或者双引号`""`标识的。
- 字符串特性
  - 字符串是不可变的序列数据类型，不能直接修改字符串本身，和数字类型一样！Python3全面支持Unicode编码，所有的字符串都是Unicode字符串，所以传统Python2存在的编码问题不再困扰我们，可以放心大胆的使用中文。
  - 字符串属于序列类型，所谓序列，指的是一块可存放多个值的连续内存空间，这些值按一定顺序排列，可通过每个值所在位置的编号（称为索引）访问它们。

```
s= "hello yuan"
```



Python 还支持索引值是负数，此类索引是从右向左计数，换句话说，从最后一个元素开始计数，从索引值 -1 开始，如图 所示。



- 序列类型支持的操作：

```
#  (1) 索引取值
s = "hello yuan"
print(s[6])
print(s[-10])

#  (2) 切片取值: 序列类型对象[start : end : step]
s = "hello yuan"
print(s[1:4]) # ell  : 取索引1到索引3 (左闭又开)
print(s[:4])  # hell : start缺省, 默认从0取
print(s[1:])  # ello yuan :  end缺省, 默认取到最后
print(s[1:-1]) # ello yua

print(s[6:9]) # yua
print(s[-4:-1]) # yua
```

```
print(s[-1:-4]) # 空
print(s[-1:-4:-1]) #nau  step为1：从左向右一个一个取。为-1 ，从右向左一个取

#  （3）判断存在：Python 中，可以使用 in 关键字检查某元素是否为序列的成员。
s = "hello yuan"
print("yuan" in s) # True

#  （4）支持两种类型相同的序列使用“+”运算符做相加操作，它会将两个序列进行连接，但不会去除重复的元素。
#       使用数字 n 乘以一个序列会生成新的序列，其内容为原来序列被重复 n 次的结果
s = "hello"+" yuan"
print(s) # hello yuan
s= "*"*10
print(s) # **********
```

- 多行字符串
  - 在字符串中，可以使用三引号（三单或三双引号都可以）编写跨行字符串，在其中可以包含换行符、制表符以及其他特殊字符。
- 字符串内置方法
  - 内置方法有很多，但是我们主要记住如下几个即可：

        * encode() # 编码成bytes类型
        * find() # 查找子串
        * index() # 获取下标
        * replace() # 替换子串
        * len(string) # 返回字符串长度，Python内置方法，非字符串方法。
        * lower() # 小写字符
        * upper() # 大写字符
        * split() # 分割字符串
        * strip() # 去除两端的指定符号
        * startswith() # 字符串是否以xxx开头
        * endswith() # 字符串是否以xxx结尾

```
s1 = 'hello nihao how are you'
ret = s1.find('how')
print(ret)

s2 = 'hello nihao how are you'
ret = s2.find('how123')
print(ret) #find返回值为-1表示子串在大字符串中没有被包含，否则表示被包含

s3 = 'hello-nihao-how-are-you'
ret = s3.split('-') #字符串切分
print(ret)
```

```
s4 = ' \t\n hello 你好 \t how are you \t\n'
ret = s4.strip() #可以将一个字符串收尾的\n \t空格进行剔除
print(ret)

s5 = 'www.baidu.com'
ret = s5.startswith('www') #判断字符串是以什么开头
print(ret)
```

- str.format()格式化方法
  - format的参数和用法很多，全部记下来显然是没必要的，浪费脑细胞。去除复杂的参数，简单的format格式化方法基本有两类：
    - 1.{0}、{1}、{2}:这一类是位置参数，引用必须按顺序，不能随意调整，否则就乱了。例如：
      - tpl = "i am {0}, age {1}".format("seven", 18)
    - 2.{name}、{age}、{gender}：这一类是关键字参数，引用时必须以键值对的方式，可以随意调整顺序。例如：
      - tpl = "i am {name}, age {age}".format(name="seven", age=18)
- %格式化方法

```
name = 'bobo'
age = 20
score = 99.5123456

msg = '该同学的名字是%s，年级是%d,成绩是%f'%
(name,age,score)
print(msg)
```

- 字符编码
  - 计算机只能处理数字01，如果要处理文本，就必须先把文本转换为数字01二进制的形式，这种转换方式就称为字符编码。
  - 对于我们而言，你只需要简单记住下面几种编码就好：
    - ASCII编码：早期专门为英语语系编码，只有255个字符，每个字符需要8位也就是1个字节。不兼容汉字。
    - Unicode编码：又称万国码，国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。用2个字节来表示汉字。
    - UTF-8编码：为了节省字节数，在Unicode的基础上进行优化的编码。用1个字节表示英文字符，3个字符表示汉字。天生兼容ASCII编码，所以最为流行。
    - GB2312：我国早期自己制定的中文编码，世界范围内不通用。

- GBK： 全称《汉字内码扩展规范》，向下与GB2312兼容，向上支持ISO10646.1国际标准，是前者向后者过渡过程中的一个承上启下的产物。windows中文版的汉字编码用的就是GBK。也非世界范围通用的编码
- 其它编码：非以上类型者的统称。属于能不用就不要碰的编码。