目 录

[第1章 Python3 基本数据类型 1](#_Toc480360428)

[1.1 多个变量赋值 1](#_Toc480360429)

[1.2 标准数据类型 1](#_Toc480360430)

[1.3 Number（数字） 2](#_Toc480360431)

[1.3.1 数值运算 3](#_Toc480360432)

[1.3.2 数值类型实例 4](#_Toc480360433)

[1.4 布尔值 4](#_Toc480360434)

[1.5 String（字符串） 4](#_Toc480360435)

[1.5.1 字符串常用功能： 6](#_Toc480360436)

[1.6 List（列表） 6](#_Toc480360437)

[1.6.1 基本操作： 8](#_Toc480360438)

[1.7 Tuple（元组） 8](#_Toc480360439)

[1.8 Set（集合） 10](#_Toc480360440)

[1.9 Dictionary（字典） 11](#_Toc480360441)

[1.10 Python数据类型转换 12](#_Toc480360442)

# Python3 基本数据类型

Python 中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。

在 Python 中，变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。

等号（=）用来给变量赋值。

等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值。例如：

#!/usr/bin/python3

counter = 100 # 整型变量

miles = 1000.0 # 浮点型变量

name = "runoob" # 字符串

print (counter)

print (miles)

print (name)

执行以上程序会输出如下结果：

100

1000.0

runoob

## 多个变量赋值

Python允许你同时为多个变量赋值。例如：

a = b = c = 1

以上实例，创建一个整型对象，值为1，三个变量被分配到相同的内存空间上。

也可以为多个对象指定多个变量。例如：

a, b, c = 1, 2, "runoob"

以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "runoob" 分配给变量 c。

## 标准数据类型

Python3 中有六个标准的数据类型：

Number（数字）

String（字符串）

List（列表）

Tuple（元组）

Sets（集合）

Dictionary（字典）

## Number（数字）

Python3 支持 int、float、bool、complex（复数）。

在Python 3里，只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。

像大多数语言一样，数值类型的赋值和计算都是很直观的。内置的 type() 函数可以用来查询变量所指的对象类型。

>>> a, b, c, d = 20, 5.5, True, 4+3j

>>> print(type(a), type(b), type(c), type(d))

<class 'int'> <class 'float'> <class 'bool'> <class 'complex'>

此外还可以用 isinstance 来判断：

>>> a = 111

>>> isinstance(a, int)

True

>>>

isinstance 和 type 的区别在于：

class A:

pass

class B(A):

pass

isinstance(A(), A) # returns True

type(A()) == A # returns True

isinstance(B(), A) # returns True

type(B()) == A # returns False

区别就是:

type()不会认为子类是一种父类类型。

isinstance()会认为子类是一种父类类型。

**注意：**在 Python2 中是没有布尔型的，它用数字 0 表示 False，用 1 表示 True。到 Python3 中，把 True 和 False 定义成关键字了，但它们的值还是 1 和 0，它们可以和数字相加。

当指定一个值时，Number 对象就会被创建：

var1 = 1

var2 = 10

您也可以使用del语句删除一些对象引用。

del语句的语法是：

del var1[,var2[,var3[....,varN]]]]

您可以通过使用del语句删除单个或多个对象。例如：

del var

del var\_a, var\_b

### 数值运算

>>> 5 + 4 # 加法

9

>>> 4.3 - 2 # 减法

2.3

>>> 3 \* 7 # 乘法

21

>>> 2 / 4 # 除法，得到一个浮点数

0.5

>>> 2 // 4 # 除法，得到一个整数

0

>>> 17 % 3 # 取余

2

>>> 2 \*\* 5 # 乘方

32

**注意：**

1、Python可以同时为多个变量赋值，如a, b = 1, 2。

2、一个变量可以通过赋值指向不同类型的对象。

3、数值的除法（/）总是返回一个浮点数，要获取整数使用//操作符。

4、在混合计算时，Python会把整型转换成为浮点数。

### 数值类型实例



## 布尔值

　　真或假

　　1 或 0

## String（字符串）

* python中单引号和双引号使用完全相同。
* 使用三引号('''或""")可以指定一个多行字符串。
* 转义符 '\'特殊字符
* 自然字符串， 通过在字符串前加r或R。 如 r"this is a line with \n" 则\n会显示，并不是换行。
* python允许处理unicode字符串，加前缀u或U， 如 u"this is an unicode string"。
* 字符串是不可变的。
* 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。

字符串的截取的语法格式如下：

变量[头下标:尾下标]

索引值以 0 为开始值，-1 为从末尾的开始位置。

加号 (+) 是字符串的连接符， 星号 (\*) 表示复制当前字符串，紧跟的数字为复制的次数。实例如下：

#!/usr/bin/python3

str = 'Runoob'

print (str) # 输出字符串

print (str[0:-1]) # 输出第一个个到倒数第二个的所有字符

print (str[0]) # 输出字符串第一个字符

print (str[2:5]) # 输出从第三个开始到第五个的字符

print (str[2:]) # 输出从第三个开始的后的所有字符

print (str \* 2) # 输出字符串两次

print (str + "TEST") # 连接字符串

执行以上程序会输出如下结果：

Runoob

Runoo

R

noo

noob

RunoobRunoob

RunoobTEST

**Python 使用反斜杠(\)转义特殊字符，如果你不想让反斜杠发生转义，可以在字符串前面添加一个 r**，表示原始字符串：

>>> print('Ru\noob')

Ru

oob

>>> print(r'Ru\noob')

Ru\noob

>>>

另外，**反斜杠(\)可以作为续行符**，表示下一行是上一行的延续。也可以使用 """...""" 或者 '''...''' 跨越多行。

注意，Python 没有单独的字符类型，一个字符就是长度为1的字符串。

>>> word = 'Python'

>>> print(word[0], word[5])

P n

>>> print(word[-1], word[-6])

n P

与 C 字符串不同的是，Python 字符串不能被改变。向一个索引位置赋值，比如word[0] = 'm'会导致错误。

**注意：**

1、反斜杠可以用来转义，使用r可以让反斜杠不发生转义。

2、字符串可以用+运算符连接在一起，用\*运算符重复。

3、Python中的字符串有两种索引方式，从左往右以0开始，从右往左以-1开始。

4、Python中的字符串不能改变。

万恶的字符串拼接：

python中的字符串在C语言中体现为是一个字符数组，每次创建字符串时候需要在内存中开辟一块连续的空，并且一旦需要修改字符串的话，就需要再次开辟空间，万恶的+号每出现一次就会在内从中重新开辟一块空间。

name = "alex"

print ("i am %s " % name)

#输出: i am alex

**PS: 字符串是 %s;整数 %d;浮点数%f**

### 字符串常用功能：

移除空白

分割

长度

索引

切片

### 特性：不可修改

### capitalize()方法

（1）描述

Python capitalize()将字符串的第一个字母变成大写,其他字母变小写。

（2）语法

capitalize()方法语法：

str.capitalize()

（3）返回值

该方法返回一个首字母大写的字符串。

（4）实例

以下实例展示了capitalize()方法的实例：

#!/usr/bin/python3

str = "this is string example from runoob....wow!!!"

print ("str.capitalize() : ", str.capitalize())

以上实例输出结果如下：

str.capitalize() : This is string example from runoob....wow!!!

### count()方法

（1）描述

count() 方法用于统计字符串里某个字符出现的次数。可选参数为在字符串搜索的开始与结束位置。

（2）语法

count()方法语法：

str.count(sub, start= 0,end=len(string))

（3）参数

sub -- 搜索的子字符串

start -- 字符串开始搜索的位置。默认为第一个字符,第一个字符索引值为0。

end -- 字符串中结束搜索的位置。字符中第一个字符的索引为 0。默认为字符串的最后一个位置。

（4）返回值

该方法返回子字符串在字符串中出现的次数。

（5）实例

以下实例展示了count()方法的实例：

#!/usr/bin/python3

str="www.runoob.com"

sub='o'

print ("str.count('o') : ", str.count(sub))

sub='run'

print ("str.count('run', 0, 10) : ", str.count(sub,0,10))

以上实例输出结果如下：

str.count('o') : 3

str.count('run', 0, 10) : 1

### center()方法

（1）描述

center() 方法返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格。

（2）语法

center()方法语法：

str.center(width[, fillchar])

（3）参数

width -- 字符串的总宽度。

fillchar -- 填充字符。

（4）返回值

返回一个指定的宽度 width 居中的字符串，如果 width 小于字符串宽度直接返回字符串，否则使用 fillchar 去填充。

（5）实例

以下实例展示了center()方法的实例：

#!/usr/bin/python3

str = "[www.runoob.com]"

print ("str.center(40, '\*') : ", str.center(40, '\*'))

以上实例输出结果如下：

str.center(40, '\*') : \*\*\*\*\*\*\*\*\*\*\*\*[www.runoob.com]\*\*\*\*\*\*\*\*\*\*\*\*

### endswith()方法

（1）描述

endswith() 方法用于判断字符串是否以指定后缀结尾，如果以指定后缀结尾返回True，否则返回False。可选参数"start"与"end"为检索字符串的开始与结束位置。

（2）语法

endswith()方法语法：

str.endswith(suffix[, start[, end]])

（3）参数

suffix -- 该参数可以是一个字符串或者是一个元素。

start -- 字符串中的开始位置。

end -- 字符中结束位置。

（4）返回值

如果字符串含有指定的后缀返回True，否则返回False。

（5）实例

以下实例展示了endswith()方法的实例：

#!/usr/bin/python3

Str='Runoob example....wow!!!'

suffix='!!'

print (Str.endswith(suffix))

print (Str.endswith(suffix,20))

suffix='run'

print (Str.endswith(suffix))

print (Str.endswith(suffix, 0, 19))

以上实例输出结果如下：

True

True

False

False

### expandtabs()方法

（1）描述

expandtabs() 方法把字符串中的 tab 符号('\t')转为空格，tab 符号('\t')默认的空格数是 8。

（2）语法

expandtabs()方法语法：

str.expandtabs(tabsize=8)

（3）参数

tabsize -- 指定转换字符串中的 tab 符号('\t')转为空格的字符数。

（4）返回值

该方法返回字符串中的 tab 符号('\t')转为空格后生成的新字符串。

（5）实例

以下实例展示了expandtabs()方法的实例：

#!/usr/bin/python3

str = "this is\tstring example....wow!!!"

print ("原始字符串: " + str)

print ("替换 \\t 符号: " + str.expandtabs())

print ("使用16个空格替换 \\t 符号: " + str.expandtabs(16))

以上实例输出结果如下：

原始字符串: this is string example....wow!!!

替换 \t 符号: this is string example....wow!!!

使用16个空格替换 \t 符号: this is string example....wow!!!

### find()

（1）描述

find() 方法检测字符串中是否包含子字符串 str ，如果指定 beg（开始） 和 end（结束） 范围，则检查是否包含在指定范围内，如果包含子字符串返回开始的索引值，否则返回-1。

语法

（2）find()方法语法：

str.find(str, beg=0, end=len(string))

（3）参数

str -- 指定检索的字符串

beg -- 开始索引，默认为0。

end -- 结束索引，默认为字符串的长度。

（4）返回值

如果包含子字符串返回开始的索引值，否则返回-1。

（5）实例

以下实例展示了find()方法的实例：

#!/usr/bin/python3

str1 = "Runoob example....wow!!!"

str2 = "exam";

print (str1.find(str2))

print (str1.find(str2, 5))

print (str1.find(str2, 10))

以上实例输出结果如下：

7

7

-1

### encode()

（1）描述

encode() 方法以指定的编码格式编码字符串。errors参数可以指定不同的错误处理方案。

（2）语法

encode()方法语法：

str.encode(encoding='UTF-8',errors='strict')

（3）参数

encoding -- 要使用的编码，如: UTF-8。

errors -- 设置不同错误的处理方案。默认为 'strict',意为编码错误引起一个UnicodeError。 其他可能得值有 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' 以及通过 codecs.register\_error() 注册的任何值。

（4）返回值

该方法返回编码后的字符串，它是一个 bytes 对象。

（5）实例

以下实例展示了encode()方法的实例：

#!/usr/bin/python3

str = "菜鸟教程";

str\_utf8 = str.encode("UTF-8")

str\_gbk = str.encode("GBK")

print(str)

print("UTF-8 编码：", str\_utf8)

print("GBK 编码：", str\_gbk)

print("UTF-8 解码：", str\_utf8.decode('UTF-8','strict'))

print("GBK 解码：", str\_gbk.decode('GBK','strict'))

以上实例输出结果如下：

菜鸟教程

UTF-8 编码： b'\xe8\x8f\x9c\xe9\xb8\x9f\xe6\x95\x99\xe7\xa8\x8b'

GBK 编码： b'\xb2\xcb\xc4\xf1\xbd\xcc\xb3\xcc'

UTF-8 解码： 菜鸟教程

GBK 解码： 菜鸟教程

### format()方法

>>> msg = "my name is {}, and age is {}"

>>> msg.format("alex",22)

'my name is alex, and age is 22'

>>> msg = "my name is {1}, and age is {0}"

>>> msg.format("alex",22)

'my name is 22, and age is alex'

>>> msg = "my name is {name}, and age is {age}"

>>> msg.format(age=22,name="ale")

'my name is ale, and age is 22'

（8）name.format\_map

>>> msg.format\_map({'name':'alex','age':22})

'my name is alex, and age is 22'

name.casefold() 大写全部变小写

### isalnum()方法

isalnum() 方法检测字符串是否由字母和数字组成

(1)isalnum()方法语法：

str.isalnum()

(2)返回值

如果 string 至少有一个字符并且所有字符都是字母或数字则返回 True,否则返回 False

(3)实例

#!/usr/bin/python3

str = "runoob2016" # 字符串没有空格

print (str.isalnum())

str = "www.runoob.com"

print (str.isalnum())

以上实例输出结果如下：

True

False

### isalpha()方法

（1）描述

Python isalpha() 方法检测字符串是否只由字母组成。

（2）语法

isalpha()方法语法：

str.isalpha()

（3）返回值

如果字符串至少有一个字符并且所有字符都是字母则返回 True,否则返回 False

（4）实例

以下实例展示了isalpha()方法的实例：

#!/usr/bin/python3

str = "runoob"

print (str.isalpha())

str = "Runoob example....wow!!!"

print (str.isalpha())

以上实例输出结果如下：

True

False

### str.isidentifier()

判断字符串是否是合法的标识符，字符串仅包含中文字符合法，实际上这里判断的是变量名是否合法。如：

‘\_a’.isidentifier() -->True

‘3a’.isidentifier() -->False

‘中国’.isidentifier() -->True

### isdigit()方法

（1）描述

Python isdigit() 方法检测字符串是否只由数字组成。

（2）语法

isdigit()方法语法：

str.isdigit()

（3）返回值

如果字符串只包含数字则返回 True 否则返回 False。

（4）实例

以下实例展示了isdigit()方法的实例：

#!/usr/bin/python3

str = "123456";

print (str.isdigit())

str = "Runoob example....wow!!!"

print (str.isdigit())

以上实例输出结果如下：

True

False

### join()方法

（1）描述

Python join() 方法**用于将序列中的元素以指定的字符连接生成一个新的字符串**。

（2）语法

join()方法语法：

str.join(sequence)

（3）参数

sequence -- 要连接的元素序列。

（4）返回值

返回通过指定字符连接序列中元素后生成的新字符串。

（5）实例

以下实例展示了join()的使用方法：

#!/usr/bin/python3

s1 = "-"

s2 = ""

seq = ("r", "u", "n", "o", "o", "b") # 字符串序列

print (s1.join( seq )) 或者是print（'-'.join(["r", "u", "n", "o", "o", "b"])）

print (s2.join( seq )) 等于print（"".join(["r", "u", "n", "o", "o", "b"])）

以上实例输出结果如下：

r-u-n-o-o-b

runoob

### ljust()方法（对应有rjust()）

（1）描述

ljust() 方法返回一个原字符串左对齐,并使用空格填充至指定长度的新字符串。如果指定的长度小于原字符串的长度则返回原字符串。

（2）语法

ljust()方法语法：

str.ljust(width[, fillchar])

（3）参数

width -- 指定字符串长度。

fillchar -- 填充字符，默认为空格。

（4）返回值

返回一个原字符串左对齐,并使用空格填充至指定长度的新字符串。如果指定的长度小于原字符串的长度则返回原字符串。

（5）实例

以下实例展示了ljust()的使用方法：

#!/usr/bin/python3

str = "Runoob example....wow!!!"

print (str.ljust(50, '\*'))

以上实例输出结果如下：

Runoob example....wow!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### lower()方法（对应有upper()）

(1)描述

Python lower() 方法转换字符串中所有大写字符为小写。

(2)语法

lower()方法语法：

str.lower()

(3)参数

无。

(4)返回值

返回将字符串中所有大写字符转换为小写后生成的字符串。

(5)实例

以下实例展示了lower()的使用方法：

#!/usr/bin/python3

str = "Runoob EXAMPLE....WOW!!!"

print( str.lower() )

以上实例输出结果如下：

runoob example....wow!!!

### strip()方法

（1）描述

Python strip() 方法用于移除字符串头尾指定的字符（默认为空格）。

（2）语法

strip()方法语法：

str.strip([chars]);

（3）参数

chars -- 移除字符串头尾指定的字符。

（4）返回值

返回移除字符串头尾指定的字符生成的新字符串。

（5）实例

以下实例展示了strip()函数的使用方法：

#!/usr/bin/python3

str = "\*\*\*\*\*this is string example....wow!!!\*\*\*\*\*"

print (str.strip( '\*' ))

以上实例输出结果如下：

this is string example....wow!!!

注释：strip()是lstrip()和rstrip()的合集

### maketrans()方法

（1）描述

maketrans() 方法用于创建字符映射的转换表，对于接受两个参数的最简单的调用方式，第一个参数是字符串，表示需要转换的字符，第二个参数也是字符串表示转换的目标。

注：两个字符串的长度必须相同，为一一对应的关系。

（2）语法

maketrans()方法语法：

str.maketrans(intab, outtab)

（3）参数

intab -- 字符串中要替代的字符组成的字符串。

outtab -- 相应的映射字符的字符串。

（4）返回值

返回字符串转换后生成的新字符串。

（5）实例

以下实例展示了使用maketrans() 方法将所有元音字母转换为指定的数字：

#!/usr/bin/python3

intab = "aeiou"

outtab = "12345"

trantab = str.maketrans(intab, outtab)

str = "this is string example....wow!!!"

print (str.translate(trantab))

以上实例输出结果如下：

th3s 3s str3ng 2x1mpl2....w4w!!!

### replace()方法

（1）描述

replace() 方法把字符串中的 old（旧字符串） 替换成 new(新字符串)，如果指定第三个参数max，则替换不超过 max 次。

（2）语法

replace()方法语法：

str.replace(old, new[, max])

（3）参数

old -- 将被替换的子字符串。

new -- 新字符串，用于替换old子字符串。

max -- 可选字符串, 替换不超过 max 次

（4）返回值

返回字符串中的 old（旧字符串） 替换成 new(新字符串)后生成的新字符串，如果指定第三个参数max，则替换不超过 max 次。

（5）实例

以下实例展示了replace()函数的使用方法：

#!/usr/bin/python3

str = "www.w3cschool.cc"

print ("菜鸟教程新地址：", str)

print ("菜鸟教程新地址：", str.replace("w3cschool.cc", "runoob.com"))

str = "this is string example....wow!!!"

print (str.replace("is", "was", 3))

以上实例输出结果如下：

菜鸟教程新地址： www.w3cschool.cc

菜鸟教程新地址： www.runoob.com

thwas was string example....wow!!!

### rfind()方法

（1）描述

Python rfind() 返回字符串最后一次出现的位置，如果没有匹配项则返回-1。

（2）语法

rfind()方法语法：

str.rfind(str, beg=0 end=len(string))

（3）参数

str -- 查找的字符串

beg -- 开始查找的位置，默认为0

end -- 结束查找位置，默认为字符串的长度。

（4）返回值

返回字符串最后一次出现的位置（下标），如果没有匹配项则返回-1。

（5）实例

以下实例展示了rfind()函数的使用方法：

#!/usr/bin/python3

str1 = "this is really a string example....wow!!!"

str2 = "is"

print (str1.rfind(str2))

print (str1.rfind(str2, 0, 10))

print (str1.rfind(str2, 10, 0))

print (str1.find(str2))

print (str1.find(str2, 0, 10))

print (str1.find(str2, 10, 0))

以上实例输出结果如下：

5

5

-1

2

2

-1

### split()方法

（1）描述

split()通过指定分隔符对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串

（2）语法

split()方法语法：

str.split(str="", num=string.count(str)).

（3）参数

str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。

（4）返回值

返回分割后的字符串列表。

（5）实例

以下实例展示了split()函数的使用方法：

#!/usr/bin/python3

str = "this is string example....wow!!!"

print (str.split( ))

print (str.split('i',1))

print (str.split('w'))

以上实例输出结果如下：

['this', 'is', 'string', 'example....wow!!!']

['th', 's is string example....wow!!!']

['this is string example....', 'o', '!!!']

### splitlines()方法

（1）描述

Python splitlines() 按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。

（2）语法

splitlines()方法语法：

str.splitlines([keepends])

（3）参数

keepends -- 在输出结果里是否去掉换行符('\r', '\r\n', \n')，默认为 False，不包含换行符，如果为 True，则保留换行符。

（4）返回值

返回一个包含各行作为元素的列表。

（5）实例

以下实例展示了splitlines()函数的使用方法：

>>> 'ab c\n\nde fg\rkl\r\n'.splitlines()

['ab c', '', 'de fg', 'kl']

>>> 'ab c\n\nde fg\rkl\r\n'.splitlines(True)

['ab c\n', '\n', 'de fg\r', 'kl\r\n']

>>>

### swapcase()方法

（1）描述

swapcase() 方法用于对字符串的大小写字母进行转换。

（2）语法

swapcase()方法语法：

str.swapcase();

（3）参数

NA。

（4）返回值

返回大小写字母转换后生成的新字符串。

（5）实例

以下实例展示了swapcase()函数的使用方法：

#!/usr/bin/python3

str = "this is string example....wow!!!"

print (str.swapcase())

str = "This Is String Example....WOW!!!"

print (str.swapcase())

以上实例输出结果如下：

THIS IS STRING EXAMPLE....WOW!!!

tHIS iS sTRING eXAMPLE....wow!!!

### translate()方法

（1）描述

translate() 方法根据参数table给出的表(包含 256 个字符)转换字符串的字符, 要过滤掉的字符放到 del 参数中。

（2）语法

translate()方法语法：

str.translate(table[, deletechars]);

（3）参数

table -- 翻译表，翻译表是通过maketrans方法转换而来。

deletechars -- 字符串中要过滤的字符列表。

（4）返回值

返回翻译后的字符串。

（5）实例

以下实例展示了 translate()函数的使用方法：

#!/usr/bin/python3

intab = "aeiou"

outtab = "12345"

trantab = str.maketrans(intab, outtab)

str = "this is string example....wow!!!"

print (str.translate(trantab))

以上实例输出结果如下：

th3s 3s str3ng 2x1mpl2....w4w!!!

以上实例去除字符串中的 'x' 和 'm' 字符：

#!/usr/bin/python

from string import maketrans # Required to call maketrans function.

intab = "aeiou"

outtab = "12345"

trantab = maketrans(intab, outtab)

str = "this is string example....wow!!!";

print str.translate(trantab, 'xm');

以上实例输出结果：

th3s 3s str3ng 21pl2....w4w!!!

### title()方法

（1）描述

Python title() 方法返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle())。

（2）语法

title()方法语法：

str.title();

（3）参数

NA。

（4）返回值

返回"标题化"的字符串,就是说所有单词都是以大写开始。

（5）实例

以下实例展示了 title()函数的使用方法：

#!/usr/bin/python

str = "this is string example from runoob....wow!!!"

print (str.title())

以上实例输出结果如下：

This Is String Example From Runoob....Wow!!!

### zfill()方法

（1）描述

Python zfill() 方法返回指定长度的字符串，原字符串右对齐，前面填充0。

（2）语法

zfill()方法语法：

str.zfill(width)

（3）参数

width -- 指定字符串的长度。原字符串右对齐，前面填充0。

（4）返回值

返回指定长度的字符串。

（5）实例

以下实例展示了 zfill()函数的使用方法：

#!/usr/bin/python3

str = "this is string example from runoob....wow!!!"

print ("str.zfill : ",str.zfill(40))

print ("str.zfill : ",str.zfill(50))

以上实例输出结果如下：

str.zfill : this is string example from runoob....wow!!!

str.zfill : 000000this is string example from runoob....wow!!!

## List（列表）

List（列表） 是 Python 中使用最频繁的数据类型。

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。

和字符串一样，列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表。

列表截取的语法格式如下：

变量[头下标:尾下标]

创建列表：

name\_list = ['alex', 'seven', 'eric']

或

name\_list ＝ list(['alex', 'seven', 'eric'])

索引值以 0 为开始值，-1 为从末尾的开始位置。

加号（+）是列表连接运算符，星号（\*）是重复操作。如下实例：

#!/usr/bin/python3

list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]

tinylist = [123, 'runoob']

print (list) # 输出完整列表

print (list[0]) # 输出列表第一个元素

print (list[1:3]) # 从第二个开始输出到第三个元素 ，想到于切片处理

print (list[2:]) # 输出从第三个元素开始的所有元素

print (tinylist \* 2) # 输出两次列表

print (list + tinylist) # 连接列表

print （list[-1]） #输出最后一个

print (list[-3:-1]) #输出倒数第三个到倒数第一个

print (list[-5:]) #输出全部

以上实例输出结果：

['abcd', 786, 2.23, 'runoob', 70.2]

abcd

[786, 2.23]

[2.23, 'runoob', 70.2]

[123, 'runoob', 123, 'runoob']

['abcd', 786, 2.23, 'runoob', 70.2, 123, 'runoob']

[70.2]

[2.23, 'runoob']

['abcd', 786, 2.23, 'runoob', 70.2]

与Python字符串不一样的是，列表中的元素是可以改变的：

>>> a = [1, 2, 3, 4, 5, 6]

>>> a[0] = 9

>>> a[2:5] = [13, 14, 15]

>>> a

[9, 2, 13, 14, 15, 6]

>>> a[2:5] = [] # 删除

>>> a

[9, 2, 6]

List内置了有很多方法，例如append()、pop()等等，这在后面会讲到。

**注意：**

1、List写在方括号之间，元素用逗号隔开。

2、和字符串一样，list可以被索引和切片。

3、List可以使用“+”操作符进行拼接。

4、List中的元素是可以改变的。

### 基本操作：

索引

切片

追加

删除

长度

切片

循环

包含

#### 追加append()函数

list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]

list.append("110")

print(list)

输出结果：

['abcd', 'higk', 786, 2.23, 'runoob', 70.2, '110']

#### 插入insert()函数

list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]

list.insert(1,"higk")

输出结果：

['abcd', 'higk', 786, 2.23, 'runoob', 70.2 ]

#### 切片：取多个元素

print (list[1:3]) # 从第二个开始输出到第三个元素 ，想到于切片处理

print (list[2:]) # 输出从第三个元素开始的所有元素

print （list[-1]） #输出最后一个

print (list[-3:-1]) #输出倒数第三个到倒数第一个

print (list[-5:]) #输出全部

（1）按指定步长切片

names = ['Alex','Tenglan',["jack","sunny"],'Eric','Rain','Tom','Amy']

print(names[0:-1:2])

print(names[::2])

输出结果：

['Alex', ['jack', 'sunny'], 'Rain']

['Alex', ['jack', 'sunny'], 'Rain', 'Amy']

#### 修改

>>> names

['Alex', 'Tenglan', '强行从Eric前面插入', 'Eric', 'Rain', '从eric后面插入试试新姿势', 'Tom', 'Amy', '我是新来的']

>>> names[2] = "该换人了"

>>> names

['Alex', 'Tenglan', '该换人了', 'Eric', 'Rain', '从eric后面插入试试新姿势', 'Tom', 'Amy', '我是新来的']

修改子列表内容：

names = ['Alex', 'Tenglan',["jack","sunny"], 'Eric', 'Rain', '从eric后面插入试试新姿势', 'Tom', 'Amy', '我是新来的']

names[2][0] = "JACK"

print(names)

['Alex', 'Tenglan', ['JACK', 'sunny'], 'Eric', 'Rain', '从eric后面插入试试新姿势', 'Tom', 'Amy', '我是新来的']

#### 删除（del names、names.remove、names.pop）

>>> del names[2]

>>> names

['Alex', 'Tenglan', 'Eric', 'Rain', '从eric后面插入试试新姿势', 'Tom', 'Amy', '我是新来的']

>>> del names[4]

>>> names

['Alex', 'Tenglan', 'Eric', 'Rain', 'Tom', 'Amy', '我是新来的']

>>>

>>> names.remove("Eric") #删除指定元素

>>> names

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy', '我是新来的']

>>> names.pop() #如果不指定下标，默认删除列表最后一个值

'我是新来的'

>>> names

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy']

>>>del names #删除整个列表内容

#### 扩展（names.extend）

>>> names

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy']

>>> b = [1,2,3]

>>> names.extend(b)

>>> names

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy', 1, 2, 3]

#### 拷贝（names.copy）

>>> names

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy', 1, 2, 3]

>>> name\_copy = names.copy()

>>> name\_copy

['Alex', 'Tenglan', 'Rain', 'Tom', 'Amy', 1, 2, 3]

#### 深拷贝deepcopy与浅拷贝copy

（1）浅拷贝copy

names = ['Alex', 'Tenglan',["jack","sunny"], 'Eric', 'Rain','Tom', 'Amy']

names2 = names.copy() ##对象拷贝，浅拷贝，等于copy.copy(names)

print(names)

print(names2)

names[1] = "自由人"

names[2][0] = "JACK"

print(names)

print(names2)

输出结果：

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', '自由人', ['JACK', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', 'Tenglan', ['JACK', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

**结论：copy.copy 浅拷贝 只拷贝父对象，不会拷贝对象的内部的子对象。即只把子对象的内存地址copy一份。（浅拷贝产生的对象本身是新的，但是它的内容不是新的，只是对原子对象的一个引用）**

**浅拷贝三种方式：**

P1= copy.copy(names)

P2=names[:]

P3=list(names)

如果列表是直接赋值，则结果一样：

names2 = names

输出结果：

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', '自由人', ['JACK', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', '自由人', ['JACK', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

这区别于简单的数字和字符串的赋值！！！

（2）深拷贝deepcopy

import copy

names = ['Alex', 'Tenglan',["jack","sunny"], 'Eric', 'Rain','Tom', 'Amy']

names2 = copy.deepcopy(names) #深拷贝

print(names)

print(names2)

names[1] = "自由人"

names[2][0] = "JACK"

print(names)

print(names2)

输出结果：

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', '自由人', ['JACK', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

['Alex', 'Tenglan', ['jack', 'sunny'], 'Eric', 'Rain', 'Tom', 'Amy']

**结论：copy.deepcopy 深拷贝 拷贝对象及其子对象。**

#### 统计（names.count）

>>> names

['Alex', 'Tenglan', 'Amy', 'Tom', 'Amy', 1, 2, 3]

>>> names.count("Amy")

#### 排序&翻转（names.reverse）

>>> names

['Alex', 'Tenglan', 'Amy', 'Tom', 'Amy', 1, 2, 3]

>>> names.sort() #排序

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unorderable types: int() < str() #3.0里不同数据类型不能放在一起排序了，擦

>>> names[-3] = '1'

>>> names[-2] = '2'

>>> names[-1] = '3'

>>> names

['Alex', 'Amy', 'Amy', 'Tenglan', 'Tom', '1', '2', '3']

>>> names.sort() #排序按ASCII码进行排序的

>>> names

['1', '2', '3', 'Alex', 'Amy', 'Amy', 'Tenglan', 'Tom']

>>> names.reverse() #反转

>>> names

['Tom', 'Tenglan', 'Amy', 'Amy', 'Alex', '3', '2', '1']

#### 获取下标（names.index）

>>> names

['Tom', 'Tenglan', 'Amy', 'Amy', 'Alex', '3', '2', '1']

>>> names.index("Amy")

2 #只返回找到的第一个下标

#### 清除列表（names.clear）

names.clear()

### for循环打印列表

for i in names:

print(i)

输出结果:

Alex

Tenglan

['jack', 'sunny']

Eric

Rain

Tom

Amy

## Tuple（元组）

元组（tuple）与列表类似，不同之处在于元组的元素不能修改。元组写在小括号(())里，元素之间用逗号隔开。它只有2个方法，一个是count,一个是index，完毕。

创建元组：

ages = (11, 22, 33, 44, 55)

或

ages = tuple((11, 22, 33, 44, 55))

元组中的元素类型也可以不相同：

#!/usr/bin/python3

tuple = ( 'abcd', 786 , 2.23, 'runoob', 70.2 )

tinytuple = (123, 'runoob')

print (tuple) # 输出完整元组

print (tuple[0]) # 输出元组的第一个元素

print (tuple[1:3]) # 输出从第二个元素开始到第三个元素

print (tuple[2:]) # 输出从第三个元素开始的所有元素

print (tinytuple \* 2) # 输出两次元组

print (tuple + tinytuple) # 连接元组

以上实例输出结果：

('abcd', 786, 2.23, 'runoob', 70.2)

abcd

(786, 2.23)

(2.23, 'runoob', 70.2)

(123, 'runoob', 123, 'runoob')

('abcd', 786, 2.23, 'runoob', 70.2, 123, 'runoob')

元组与字符串类似，可以被索引且下标索引从0开始，-1 为从末尾开始的位置。也可以进行截取（看上面，这里不再赘述）。

其实，可以把字符串看作一种特殊的元组。

>>> tup = (1, 2, 3, 4, 5, 6)

>>> print(tup[0])

1

>>> print(tup[1:5])

(2, 3, 4, 5)

>>> tup[0] = 11 # 修改元组元素的操作是非法的

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

>>>

虽然tuple的元素不可改变，但它可以包含可变的对象，比如list列表。

构造包含 0 个或 1 个元素的元组比较特殊，所以有一些额外的语法规则：

tup1 = () # 空元组

tup2 = (20,) # 一个元素，需要在元素后添加逗号

**string、list和tuple都属于sequence（序列）。**

**注意：**

1、与字符串一样，元组的元素不能修改。

2、元组也可以被索引和切片，方法一样。

3、注意构造包含0或1个元素的元组的特殊语法规则。

4、元组也可以使用+操作符进行拼接。

## Set（集合）

集合（set）是一个无序不重复元素的序列。

基本功能是进行成员关系测试和删除重复元素。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

#!/usr/bin/python3

student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

print(student) # 输出集合，重复的元素被自动去掉

# 成员测试

if('Rose' in student) :

print('Rose 在集合中')

else :

print('Rose 不在集合中')

# set可以进行集合运算

a = set('abracadabra')

b = set('alacazam')

print(a)

print(a - b) # a和b的差集

print(a | b) # a和b的并集

print(a & b) # a和b的交集

print(a ^ b) # a和b中不同时存在的元素

以上实例输出结果：

{'Mary', 'Jim', 'Rose', 'Jack', 'Tom'}

Rose 在集合中

{'b', 'a', 'c', 'r', 'd'}

{'b', 'd', 'r'}

{'l', 'r', 'a', 'c', 'z', 'm', 'b', 'd'}

{'a', 'c'}

{'l', 'r', 'z', 'm', 'b', 'd'}

## Dictionary（字典）

字典（dictionary）是Python中另一个非常有用的内置数据类型。

**列表是有序的对象结合，字典是无序的对象集合**。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

字典是一种映射类型，**字典用"{ }"标识，它是一个无序的键(key) : 值(value)对集合**。

键(key)必须使用不可变类型。

在同一个字典中，键(key)必须是唯一的。

#!/usr/bin/python3

dict = {}

dict['one'] = "1 - 菜鸟教程"

dict[2] = "2 - 菜鸟工具"

tinydict = {'name': 'runoob','code':1, 'site': 'www.runoob.com'}

print (dict['one']) # 输出键为 'one' 的值

print (dict[2]) # 输出键为 2 的值

print (tinydict) # 输出完整的字典

print (tinydict.keys()) # 输出所有键

print (tinydict.values()) # 输出所有值

以上实例输出结果：

1 - 菜鸟教程

2 - 菜鸟工具

{'name': 'runoob', 'site': 'www.runoob.com', 'code': 1}

dict\_keys(['name', 'site', 'code'])

dict\_values(['runoob', 'www.runoob.com', 1])

构造函数 dict() 可以直接从键值对序列中构建字典如下：

>>> dict([('Runoob', 1), ('Google', 2), ('Taobao', 3)])

{'Taobao': 3, 'Runoob': 1, 'Google': 2}

>>> {x: x\*\*2 for x in (2, 4, 6)}

{2: 4, 4: 16, 6: 36}

>>> dict(Runoob=1, Google=2, Taobao=3)

{'Taobao': 3, 'Runoob': 1, 'Google': 2}

另外，字典类型也有一些内置的函数，例如clear()、keys()、values()等。

**注意：**

1、字典是一种映射类型，它的元素是键值对。

2、字典的关键字必须为不可变类型，且不能重复。

3、创建空字典使用 { }。

## Python数据类型转换

有时候，我们需要对数据内置的类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。

以下几个内置的函数可以执行数据类型之间的转换。这些函数返回一个新的对象，表示转换的值。



# 数据运算

## 算数运算：



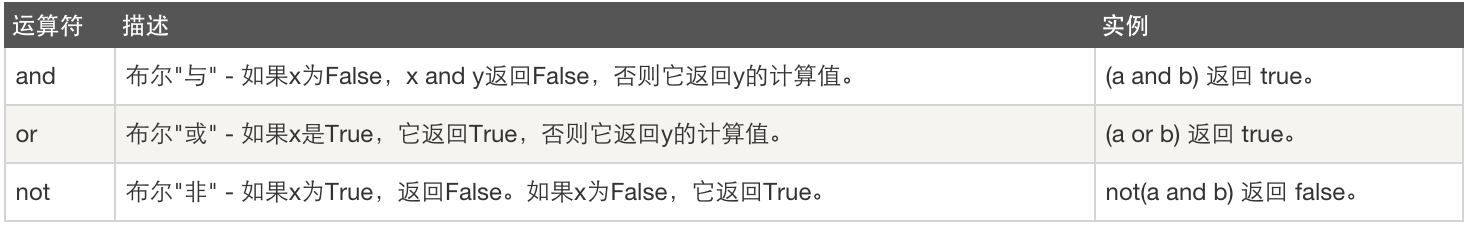
## 比较运算：



## 赋值运算：



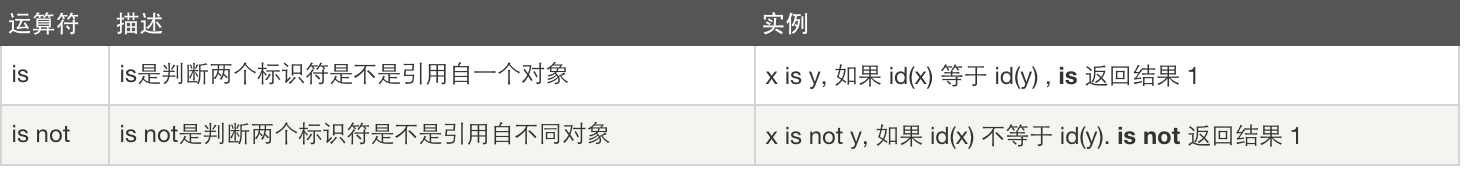
## 逻辑运算：



## 成员运算：



## 身份运算：



## 位运算：



如下实例：

#!/usr/bin/python

a = 60 # 60 = 0011 1100

b = 13 # 13 = 0000 1101

c = 0

c = a & b; # 12 = 0000 1100

print "Line 1 - Value of c is ", c

c = a | b; # 61 = 0011 1101

print "Line 2 - Value of c is ", c

c = a ^ b; # 49 = 0011 0001 #相同为0，不同为1

print "Line 3 - Value of c is ", c

c = ~a; # -61 = 1100 0011

print "Line 4 - Value of c is ", c

c = a << 2; # 240 = 1111 0000

print "Line 5 - Value of c is ", c

c = a >> 2; # 15 = 0000 1111

print "Line 6 - Value of c is ", c

## 运算符优先级：

