函数	说明
int(x [,base])	将x转换为一个整数
float(x)	将x转换为一个浮点数
complex(real [,imag])	创建一个复数，real为实部，imag为虚部
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表

Python int() 函数

Python 内置函数

描述

int() 函数用于将一个字符串或数字转换为整型。

语法

以下是 int() 方法的语法:

```
class int(x, base=10)
```

参数

- x -- 字符串或数字。
- base -- 进制数，默认十进制。

返回值

返回整型数据。

实例

以下展示了使用 int() 方法的实例:

```
>>>int()               # 不传入参数时，得到结果0
0
>>> int(3)
3
>>> int(3.6)
3
>>> int('12',16)        # 如果是带参数base的话，12要以字符串的形式进行输入，12 为 16进制
18
>>> int('0xa',16)
10
>>> int('10',8)
8
```

795

怒写一波：

```
int(x,base)
```

x 有两种：**str / int**

1、若 x 为纯数字，则不能有 base 参数，否则报错；其作用为对入参 x 取整

```
>>> int(3.1415926)
3

>>> int(-11.123)
-11

>>> int(2.5,10)
#报错
>>> int(2.5)
2
```

2、若 x 为 str，则 base 可略可有。

base 存在时，视 x 为 base 类型数字，并将其转换为 10 进制数字。

若 x 不符合 base 规则，则报错。如:

```
>>>int("9",2)  #报错，因为2进制无9
>>> int("9")
9
#默认10进制

>>> int("3.14",8)
>>> int("1.2")
#均报错，str须为整数
>>>int("1001",2)
9
# "1001"才是2进制格式，并转化为十进制数字9
>>> int("0xa",16)
10
# ≥16进制才会允许入参为a,b,c...
>>> int("b",8) #报错
>>> int("123",8)
83
#视123为8进制数字，对应的10进制为83
```

deepblue 5年前(2020-01-12)

int("...",2或8或16),2或8或16是指将双引号里面的数据看成二进制或者八进制或者16进制数，然后将其转化为十进制数输出。

Python float() 函数

Python 内置函数

描述

float() 函数用于将整数和字符串转换成浮点数。

语法

float()方法语法：

```
class float([x])
```

参数

- x -- 整数或字符串

返回值

返回浮点数。

实例

以下实例展示了 float() 的使用方法：

```
>>>float(1)
1.0
>>> float(112)
112.0
>>> float(-123.6)
-123.6
>>> float('123')     # 字符串
123.0
```

float函数就是单纯将其转为为浮点型

Python str() 函数

Python 内置函数

描述

str() 函数将对象转化为适于人阅读的形式。

语法

以下是 str() 方法的语法:

```
class str(object='')
```

参数

- object -- 对象。

返回值

返回一个对象的string格式。

实例

以下展示了使用 str() 方法的实例:

```
>>>s = 'RUNOOB'
>>> str(s)
'RUNOOB'
>>> dict = {'runoob': 'runoob.com', 'google': 'google.com'};
>>> str(dict)
"{'google': 'google.com', 'runoob': 'runoob.com'}"
>>>
```

Python 内置函数

dict字典

Python repr() 函数

Python 内置函数

描述

repr() 函数将对象转化为供解释器读取的形式。

语法

以下是 repr() 方法的语法:

```
repr(object)
```

参数

- object -- 对象。

返回值

返回一个对象的 string 格式。

实例

以下展示了使用 repr() 方法的实例:

```
>>> s = 'RUNOOB'
>>> repr(s)
"'RUNOOB'"
>>> dict = {'runoob': 'runoob.com', 'google': 'google.com'};
>>> repr(dict)
"{'google': 'google.com', 'runoob': 'runoob.com'}"
>>>
```

以下是str()函数和repr()函数的区别

python中转换成字符有两种方法：str()和repr()，这两种又有什么区别？什么时候用str？什么时候用repr?

str()函数：将值转化为适于人阅读的字符串的形式

repr()函数：将值转化为供解释器读取的字符串形式

代码示例

下面我们用例子来说明两个函数是差异点，还有就是print输出字符串时需要注意的点

将整型转换为字符串

```
>>> a = 123  #int类型
>>> type(a)
<class 'int'>
>>> str(a)
'123'
>>> type(str(a))
<class 'str'>
>>> print(str(a))  #print输出时会去掉引号，但是仍然是str类型
123
>>> repr(a)
'123'
>>> type(repr(a))
<class 'str'>
>>> print(repr(a))
123
>>> len(repr(a))  #转换后的数据都是'123'，所以长度是3
3
>>> len(str(a))   #转换后的数据都是'123'，所以长度是3
3
```

将字符串再转换为字符串

```
>>> repr('abd')  #repr转换后是在'abd'的外层又加了一层引号
"'abd'"
>>> str('abd')   #str转换后还是原来的值
'abd'
>>> str('abd') == 'abd'
True
>>> repr('abd') == 'abd'
False
>>> len(repr('abd'))  #repr转换后的字符串和str转换后的字符串个数都是不一样的
5
>>> len(str('abd'))
3
```

repr的使用场景

根据以上代码示例，可以得出只有当repr再次作用在字符串上时会多一层引号，那么这一特性在拼接完字符串用eval执行时是特别有用的，如果不用repr而是采用str会报错，举例，将字符串s = 'abdcf'转换成列表，如果用eval自己实现的话可以这样写：

```
>>> s = 'abdcf'
>>> eval('['+','.join([repr(i) for i in s])+']')
['a', 'b', 'd', 'c', 'f']
>>> eval('['+','.join([str(i) for i in s])+']')    #str报错
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'b' is not defined
```

为什么会报错呢？当','.join([str(i) for i in s])拼接后的结果'a,b,d,c,f'只有一层引号，eval执行时会去掉这层引号，就成了a,b,d,c,f，解释器就会当做变量对待，但是并没有定义这样的变量，所以报NameError错误

```
>>> ','.join([repr(i) for i in s])
"'a','b','d','c','f'"
>>> ','.join([str(i) for i in s])
'a,b,d,c,f'
>>>
```

总结

1.除了字符串类型外，使用str还是repr转换没有什么区别，字符串类型的话，外层会多一对引号，这一特性有时候在eval操作时特别有用；

2.命令行下直接输出对象调用的是对象的repr方法，print输出调用的是str方法

python函数汇总：eval(str)

eval(str)函数很强大，官方解释为：将字符串str当成有效的表达式来求值并返回计算结果。所以，结合math当成一个计算器很好用。

描述

eval() 函数用来执行一个字符串表达式，并返回表达式的值。

语法

以下是 eval() 方法的语法:

```
eval(expression[, globals[, locals]])
```

参数

- expression -- 表达式。
- globals -- 变量作用域，全局命名空间，如果被提供，则必须是一个字典对象。
- locals -- 变量作用域，局部命名空间，如果被提供，可以是任何映射对象。

eval()函数常见作用有：
1、计算字符串中有效的表达式，并返回结果

```
>>> eval('pow(2,2)')
4
>>> eval('2 + 2')
4
>>> eval("n + 4")
85
```

2、将字符串转成相应的对象（如list、tuple、dict和string之间的转换）

```
>>> a = "[[1,2], [3,4], [5,6], [7,8], [9,0]]"
>>> b = eval(a)   字符串-列表
>>> b
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 0]]
>>> a = "{1:'xx',2:'yy'}"
>>> c = eval(a)   字符串-字典
>>> c
{1: 'xx', 2: 'yy'}
>>> a = "(1,2,3,4)"
>>> d = eval(a)   字符串-元组
>>> d
(1, 2, 3, 4)
```

3、将利用repr函数转换的字符串再反转回对象

```
>>> list1 = [1,2,3,4,5]
>>> repr(list1)
'[1, 2, 3, 4, 5]'
>>> type(repr(list1))
<type 'str'>
>>> type(eval(repr(list)))
<type 'list'>
>>> a = eval(repr(list1))
>>> a
[1, 2, 3, 4, 5]
```

Python中列表和元组的区别

Python中列表和元组的区别

数据结构	定义符号	是否可变	存储空间	能否作为字典的键
列表（list）	[ ]	可变，动态	内存较大	不能
元组（tuple）	( )	不可变，静态	内存较小	能

在Python中，列表（list）和元组（tuple）的区别主要有以下几点：

1.定义符号：

列表用方号[ ]定义；元组用圆括号( )定义。

2.是否可变：

列表是可变的动态序列，可以访问，并增加、删除和修改列表中的元素，支持切片操作；元组是不可变的静态序列，可以访问、不可以增删改元组中的元素，同样支持切片操作。

3.存储空间：

列表一经创建，长度仍可变化，所占的内存较大；元组一经创建，长度不可变化，直接被写死，所占的内存较小。

4.能否作为字典的键：

由于元组是不可变、静态的，则其可以作为字典的键（key），具有可靠性。