2. Python 数据类型之：数字、字符串、布尔值

> 注意：本教程建立在 Python3 的基础上

2.1 Python 用作简单的计算器

Python 可以用来做数学计算器，以下是一些最简单的运算操作：

- `+`，`-`，`*`，`/` 分别代表加、减、乘、除；
- `**`，`//`，`%` 分别代表乘方、整除（即商取整数）、取模（即求余数）。

In [ ]:
```
print('加法', 1 + 1 ,'\n'
      '减法', 1 - 1,'\n'
      '乘法', 2 * 2,'\n'
      '除法', 8/6)

# 注释

# ctrl + ? 注释
# 你好
# 你好
# 你好

# ctrl + ] 缩进

# ctrl + s 保存
```

```
加法 2
减法 0
乘法 4
除法 1.3333333333333333
```

In [ ]:
```
print('乘方', 2**3,'\n'
      '整除', 8/6, '\n'
      '取模', 8%6)
```

```
乘方 8
整除 1.3333333333333333
取模 2
```

这些运算操作的优先级和数学中类似，可以通过 `()` 来改变运算操作的优先级。

In [ ]:
```
print(2 + 3 * 6)
print((2 + 3) * 6)
```

```
20
30
```

还有一些相对复杂的二目运算符，比如 `+=`，`-=`，`*=`，`/=`，`**=`，`//=`，这些实际使用中并不算太多，如果可以的话还是尽可能使用单目运算符更为简洁清晰。

比如 `x += y`，代表的含义是 `x = x + y`。其余是类似的。

In [ ]:
```
x = 2
y = 3
```

```
x **= y
print(x)
```

```
8
```

2.2 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。我们所说的"类型"就是变量所指的内存中对象的类型。

Python3 中**常见**的数据类型有：

- Number （数字）
- String （字符串）
- Bool （布尔类型）
- List （列表）
- Tuple （元组）
- Set （集合）
- Dictionary （字典）

这里我们先介绍**数字**，**字符串**,**布尔类型**和**列表**四种最常见的数据类型。

2.2.1 Number 数字

Python3 支持以下几种数字类型：

- int，整数类型；
- float，浮点数类型（就是带有小数点的数）；
- bool，布尔型类型（这里需要注意一点，Python3 中 bool 是 int 的子类，`True` 和 `False` 可以和数字相加，它们分别代表 `1` 和 `0`）；
- complex，复数类型（复数单位用 `j` 表示，它满足 $j^2 = -1$）。

可以用内置的 `type()` 查看类型，也可以用 `isinstance` 来判断。

对于一些较大的数或者较小的数，也可以利用科学计数法表示，比如 $1.23 * 10^8$ 可以用 `1.23e8` 表示，$6.7 * 10^{-10}$ 可以用 `6.7e-10` 表示。

In [ ]:
```
print(type(-2),
      type(0.0), # 注意区分 0.0 和 0，前者是浮点数，后者是整数
      type(1.25),
      type(3 + 1j), # 需要注意 + j是不合法的，需要使用 + 1j
      type(3 + 3j))
```

```
<class 'int'> <class 'float'> <class 'float'> <class 'complex'> <class 'complex'>
```

复数相乘

$$(a + bj) \times (c + dj) = ac + adj + bcj + bdj^2 = (ac - bd) + (ad + bc)j$$

复数相除

$$\frac{a+bj}{c+dj}=\frac{(a+bj)(c-dj)}{(c+dj)(c-dj)}=\frac{(ac+bd)+(-ad+bc)j}{c^2+d^2}=\frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}$$

In [ ]:
```
print('复数相乘',(1+1j) * (1+2j),'\n'
      '复数相除',(3+3j) / (6+7j))
```

```
复数相乘 (-1+3j)
复数相除 (0.4588235294117647-0.035294117647058844j)
```

In [ ]:
```
print(True)
print(True + 2)
print(False * 5)
```

```
True
3
0
```

In [ ]:
```
print(isinstance(True,(int)))
print(isinstance(False,(int)))
```

```
True
True
```

In [ ]:
```
print(1.23e8,'\n',0.67e-5,'\t',123)
```

```
123000000.0
 6.7e-06         123
```

2.2.2 String 字符串

Python 也可以有文本值，称为字符串，用单引号和双引号括起来均可，对于一些特殊字符需要用反斜杠 `\` 来转义。

多个字符串可以通过 `+` 进行连接，也可以对一个字符串通过 `*` 进行复制。

注意字符串和数字是不能相加的。

In [ ]:
```
a = 'Hello world'
b = '666'
c = '你好'

a,b,c
```

Out[ ]:
```
('Hello world', '666', '你好')
```

In [ ]:
```
print(a + b)
print(a + ' ' + b)
print(a,b)
print(b*5)
```

```
Hello world666
Hello world 666
Hello world 666
666666666666666
```

In [ ]:
```
d = 'I'm "OK"!'
```

```
  Cell In[25], line 1
    d = 'I'm "OK"!'
              ^
SyntaxError: unterminated string literal (detected at line 1)
```

In [ ]:
```
d = 'I\'m \"OK\"!'

# 注意区分以下三者
print(d)
print(r'I\'m \"OK\"!') # 加 r 表示默认不转义
d
```

```
I'm "OK"!
I\'m \"OK\"!
```

Out[ ]:
```
'I\'m "OK"!'
```

允许用 `'''...'''` 的格式表示多行内容

In [ ]:
```
print('''line1
... line2
... line3''')
```

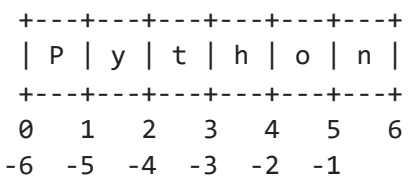
```
line1
line2
line3
```

字符串的索引和切片

当我们命名了一个字符串后，如何截取字符串的其中部分片段呢?

最基本的截取方式语法如下：`变量[开始索引:结束索引]`，注意索引值以 0 为开始值，以 -1 为末尾的开始值。省略开始索引，默认值为 0，省略结束索引时，默认到字符串的结尾。

还可以这样理解切片，索引指向的是字符**之间**，第一个字符的左侧标为 0，最后一个字符的右侧标为 n，n 是字符串长度。例如：



但实际上这还能玩出花来，具体请见以下的实例。

In [ ]:
```
a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

print(a)
print(a[0:-1]) # 打印字符串第一个到倒数第二个字符（不包含倒数第一个）
print(a[0]) # 打印字符串第一个字符
print(a[3:5]) # 打印字符串第四个到第五个字符（包含第五个，不包含第六个）
print(a[3:]) # 打印字符串从第四个直到最后一个
print(a[:-3]) # 打印字符串从第一个到倒数第四个字符（不包含倒数第三个）
print(a[0:10:2]) # 打印字符串从第一个到第十个字符（包含第十个），但每间隔两个取
print(a[0::2]) # 打印字符串每间隔两个取，从第一个字符直到最后一个
print(a[-1::-1]) # 打印字符串每间隔一个取，但从最后一个开始打印（相当于倒序打印）
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXY
A
DE
DEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVW
ACEGI
ACEGIKMOQSUWY
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

注意：Python 的字符串不能修改，因此为字符串中某个索引位置赋值会报错。如果想生成不同的字符串，应该新建一个字符串。

In [ ]:
```
print(a[0:3] +'d' +a[4:])
```

```
ABCdEFGHIJKLMNOPQRSTUVWXYZ
```

In [ ]:
```
len(a) # len() 函数返回字符串的长度
```

Out[ ]:
```
26
```

2.2.3 Bool 布尔类型

布尔类型即 `True` 和 `False` （注意大小写），在前面介绍数字时涉及到了一点点。

布尔类型可以用来控制程序流程，比如判断某个条件是否成立，或者在某个条件满足时执行某段代码。在 Python 中可以直接用 `True` 和 `False` 来表示布尔值，也可以通过布尔运算计算出来。

In [ ]:
```
print(1==2,3>2, 6!=5)
```

```
False True True
```

布尔类型可以通过 `and`，`or` 和 `not` 进行逻辑运算，它们分别代表与、或、非。

In [ ]:
```
print(True and False)
print(True or False)
print(not 2>3)
```

```
False
True
True
```

2.2.4 数据类型转换

数据类型转换分为隐式和显式两种形式，前者自动完成，后者需要使用类型函数进行转换。

比如一个整数和一个浮点数相加时，会自动将前者转换为浮点数类型，这是隐式的。

在显示类型转换中，我们可以使用 `int()`，`float()`，`str()` 等内置函数进行转换。更多地，如 `complex()` 可以创建一个复数，`list()` 将序列转为列表，`hex()` 将一个整数转换为十六进制字符串.....

注意 `bool()` 对于所有非零的数返回的都是 True，只有 0 才会返回 False。

In [ ]:
```
num_int = 123
num_flo = 1.23

print(type(num_int))

num_new = num_int + num_flo
print(type(num_new))
```

```
<class 'int'>
<class 'float'>
```

In [ ]:
```
print(type(int(1.0)),
      type(float('3')),
      type(str(2)))
```

```
<class 'int'> <class 'float'> <class 'str'>
```

2.3 变量与赋值

变量的概念和小学初中学习过得代数的方程变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。

2.3.1 变量

变量命名需要遵守以下 3 条规则：

1. 只能是一个词。
2. 只能包含字母、数字和下划线。
3. 不能以数字开头。

比如 `current-balance`，`4account`，`total_&um`，`'hello'`，`hello world` 都是不合法的变量名。另外变量名是区分大小写的，`spam`，`SPAM`，`Spam`，`sPaM` 是4个不同的变量。变量用小写字母开头是 Python 的惯例。

这里额外强调一点，在你的程序代码里使用描述性语言对变量进行命名，会提高代码的可读性，尽量不要使用 `a`，`b` 等比较空洞、无实际指向的命名（除非它们无关紧要）。比如我需要统计一个箱子里红色球、黑色球的个数，我可以将它们命名为 `red_ball`，`black_ball`。

2.3.2 赋值语句

赋值语句就是将值保存在变量中。赋值语句的语法通常是 `变量名 = 存储的值`，Python 中也支持同时对多个变量进行赋值。

In [ ]:
```
# 注意体会以下的例子
x = 10
x = x+2

x
```

Out[ ]: 12

In [ ]:
```
a = 'ABC'
b = a
a = 'XYZ'
```

```
b
```

Out[ ]: 'ABC'

In [ ]:
```
a , b, c = 12 , 'Hello' ,True

a,b,c
```

Out[ ]: (12, 'Hello', True)

Python 标准库提供了 keyword 模块，可以输出当前版本的所有关键字。它们不能作为变量进行赋值。

另外请不要对 Python 内置的函数进行赋值，比如 `print` 函数。（你也不想 print 打印时候出来一个 print 的赋值结果吧？）

In [ ]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'clas
s', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

2.4 本节相关的常用内置函数

2.4.1 数字相关函数

这里介绍一些常用的内置的函数，比如数学运算并不止加、减、乘、除、乘方、求模等操作，还有一些数学函数等内置函数。注意别忘了导入 `math` 库。

具体地有：

- `abs(x)`，返回 x 的绝对值
- `ceil(x)`，返回 x 的向上取整
- `floor(x)`，返回 x 的向下取整
- `round(x [,n])`，返回 x 的四舍五入值，n 若给出则代表舍入到小数点后的位数。
- `exp(x)`，返回 e^x
- `log(x)`，返回 $\ln(x)$
- `log10(x)`，返回 $\log_{10}(x)$
- `max(x1,x2,...)`，返回其中的最大值
- `min(x1,x2,...)`，返回其中的最小值
- `sqrt(x)`，返回 $\sqrt{x}$

还有三角函数相关的：

- `acos(x)`，`asin(x)`，`atan(x)`，分别返回 $\arccos x, \arcsin x, \arctan x$
- `sin(x)`，`cos(x)`，`tan(x)`，分别返回 $\sin x, \cos x, \tan x$
- `degrees(x)`，将弧度转化为角度
- `radians(x)`，将角度转化为弧度

In [ ]:
```
import math
```

In [ ]:
```
print(abs(-3.2),'\n',
      math.ceil(3.5),'\n',
      math.floor(3.5),'\n',
      round(3.1415926,3),'\n',
      math.exp(1),'\n',
      math.log(math.e),'\n',
      math.log10(100),'\n',
      max(1,2,3,5),'\n',
      min(1,2,3,5),'\n',
      math.sqrt(4)
      )
```

```
3.2
 4
 3
 3.142
 2.718281828459045
 1.0
 2.0
 5
 1
 2.0
```

In [ ]:
```
math.sin(math.pi/6), math.cos(math.pi), math.tan(math.pi/2)
```

Out[ ]:
```
(0.49999999999999994, -1.0, 1.633123935319537e+16)
```

2.4.2 字符串相关函数

可以用 `in` 和 `not in` 来判断字符串中是否含有指定字符

Python 支持格式化字符串的输出，常见的有：

- `%s`，格式化字符串
- `%d`，格式化整数
- `%f`，格式化浮点数
- `%e`，用科学计数法格式化浮点数

格式化输出通常需要用在，输出内容依据变量变化的时候，请见以下的例子

In [ ]:
```
'py' not in 'python'
```

Out[ ]:
```
False
```

In [ ]:
```
user_name = input()
user_age = int(input())

print('尊敬的 %s 用户，您今年已经 %d 岁了。'% (user_name,user_age))

a = user_name
b = user_age
print('尊敬的',a,'用户，您今年',b,'岁了。')
```

```
a
20
尊敬的 a 用户，您今年已经 20 岁了。
尊敬的 a 用户，您今年 20 岁了。
```

还有其他的函数，如

- `capitalize()`，将字符串第一个字符转换为大写
- `find(str,beg=0,end=len(string))`，检测 str 是否包含在字符串中
- `join(str)`，以指定字符串为分隔符，将 str 中所有的元素合并为一个新的字符串
- `len(str)`，返回 str 的长度
- `lower(str)`，将 str 中的所有大写转换为小写
- `lstrip()`，截掉 str 左边的空格或者指定字符
- `replace(old,new[,max])`，将字符串中的 old 替换成 new，如果 max 指定，则替换不超过 max 次
- `rstrip()`，删除字符串末尾的空格或者指定字符
- `upper()`，转换字符串中的小写字母为大写

In [ ]:
```
print('python'.replace('p','P'))

print('python'.capitalize())
```

```
Python
Python
```

In [ ]:
```
s1 = "*"
s2 = ""
seq = ("r", "u", "n", "o", "o", "b") # 字符串序列
print (s1.join( seq ))
print (s2.join( seq ))

type(seq)
```

```
r*u*n*o*o*b
runoob
```

Out[ ]:
```
tuple
```