

程序设计

基本数据类型

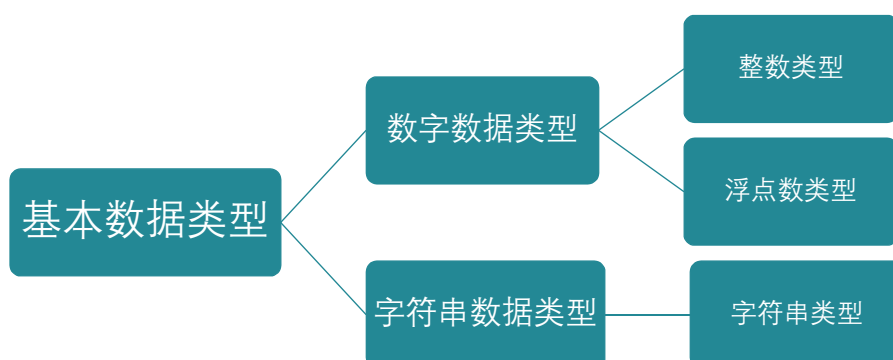
什么叫数据类型?

对数据进行分类

定义了变量所需内存的大小

变量->名称+数据类型

Python的数据类型



数字类型

数字类型概述

1. 确定性
2. 高效性



数字类型概述

数学中的整数概念一致

4种进制表示

理论上没有取值范围限制（实际上受计算机内存影响）

例子：使用pow(x, y)函数，计算 x^y

1. pow(2,10) , pow(2,15)
2. pow(2, 1000)
3. pow(2, pow(2,15))

数字类型案例

示例

1010, 99, -217

0x9a, -0X89 (0x, 0X开头表示16进制数)

0b010, -0B101 (0b, 0B开头表示2进制数)

0o123, -0O456 (0o, 0O开头表示8进制数)

浮点数类型

与数学中的实数概念一致

带有小数点及小数

两种表示方法

与整数类型相比，由不同硬件单元执行

浮点数的数值范围存在限制，小数精度也存在限制

浮点数类型

```
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
>>>
```

浮点数类型案例

示例

0.0, -77., -2.17

96e4, 4.3e-3, 9.6E5 （科学计数法）

科学计数法使用字母“e”或者“E”作为幂的符号，以10为基数。科学计数法含义如下：

$$\langle a \rangle e \langle b \rangle = a * 10^b$$

复数类型

与数学中的复数概念一致, $z = a + bj$, a 是实数部分, b 是虚数部分, a 和 b 都是浮点类型, 虚数部分用 j 或者 J 标识

示例：

12.3+4j, -5.6+7j

复数类型

$z = 1.23e-4 + 5.6e+89j$ (实部和虚部是什么?)

对于复数 z , 可以用 $z.real$ 获得实数部分, $z.imag$ 获得虚数部分

$z.real = 0.000123$ $z.imag = 5.6e+89$

数字类型操作

内置的数值运算操作符

三种类型存在一种逐渐“扩展”的关系：

整数 -> 浮点数 -> 复数

(整数是浮点数特例，浮点数是复数特例)

不同数字类型之间可以进行混合运算，运算后生成结果为 **最宽类型**

$123 + 4.0 = 127.0$ (整数 + 浮点数 = 浮点数)

内置的数值运算操作符

操作符	描述
$x + y$	x与y之和
$x - y$	x与y之差
$x * y$	x与y之积
x / y	x与y之商
$x // y$	x与y之整数商，即：不大于x与y之商的最大整数
$x \% y$	x与y之商的余数，也称为模运算
$-x$	x的负值，即： $x*(-1)$
$+x$	x本身
$x**y$	x的y次幂，即： x^y

内置的数值运算操作符

数字类型之间相互运算所生成的结果是“更宽”的类型，基本规则是：

- 整数之间运算，如果数学意义上的结果是小数，结果是浮点数；
- 整数之间运算，如果数学意义上的结果是整数，结果是整数；
- 整数和浮点数混合运算，输出结果是浮点数；
- 整数或浮点数与复数运算，输出结果是复数。

内置的数值运算函数

Python解释器提供了一些内置函数，在这些内置函数之中，有6个函数与数值运算相关

函数	描述
<code>abs(x)</code>	x的绝对值
<code>divmod(x, y)</code>	$(x//y, x\%y)$ ，输出为二元组形式（也称为元组类型）
<code>pow(x, y[, z])</code>	$(x**y)\%z$ ， <code>[.]</code> 表示该参数可以省略，即： <code>pow(x,y)</code> ，它与 <code>x**y</code> 相同
<code>round(x[, ndigits])</code>	对x四舍五入，保留ndigits位小数。 <code>round(x)</code> 返回四舍五入的整数值
<code>max(x₁, x₂, ..., x_n)</code>	x ₁ , x ₂ , ..., x _n 的最大值，n没有限定
<code>min(x₁, x₂, ..., x_n)</code>	x ₁ , x ₂ , ..., x _n 的最小值，n没有限定

数字类型转换

数值运算操作符可以隐式地转换输出结果的数字类型

例如，两个整数采用运算符“/”的除法将可能输出浮点数结果。此外，通过内置的数字类型转换函数可以显式地在数字类型之间进行转换

函数	描述
<code>int(x)</code>	将x转换为整数，x可以是浮点数或字符串
<code>float(x)</code>	将x转换为浮点数，x可以是整数或字符串
<code>complex(re[, im])</code>	生成一个复数，实部为re，虚部为im，re可以是整数、浮点数或字符串，im可以是整数或浮点数但不能为字符串

数字类型转换

三种类型可以相互转换

函数：int(), float(), complex()

示例：

`int(4.5) = 4` （直接去掉小数部分）

`float(4) = 4.0` （增加小数部分）

`complex(4) = 4 + 0j`

数字类型转换

- 示例: `complex(4.5) = 4.5 + 0j`

```
>>> float(4.5+0j)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    float(4.5+0j)
TypeError: can't convert complex to float
>>>
```

数字类型的判断

- 函数: `type(x)`, 返回x的类型, 适用于所有类型的判断

- 示例:

```
>>> type(4.5)
<class 'float'>
>>> type(z)
<class 'complex'>
>>>
```

Math库的使用

什么是库？

？

Math库概述

math库是Python提供的内置数学类函数库

math库不支持复数类型

math库一共提供了4个数学常数和44个函数。

44个函数共分为4类，包括：16个数值表示函数、8个幂对数函数、16个三角对数函数和4个高等特殊函数

Math库的使用

首先使用保留字import引用该库

第一种：import math

对math库中函数采用math.()形式使用

```
>>>import math
>>>math.ceil(10.2)
11
```

第二种，from math import <函数名>

对math库中函数可以直接采用<函数名>()形式使用

```
>>>from math import floor
>>>floor(10.2)
10
```

Math库解析

math库包括4个数学常数

常数	数学表示	描述
math.pi	π	圆周率，值为3.141592653589793
math.e	e	自然对数，值为2.718281828459045
math.inf	∞	正无穷大，负无穷大为-math.inf
math.nan		非浮点数标记，NaN（Not a Number）

Math库解析

math库包括16个数值表示函数

函数	数学表示	描述
math.fabs(x)	$ x $	返回x的绝对值
math.fmod(x, y)	$x \% y$	返回x与y的模
math.fsum([x,y,...])	$x+y+\dots$	浮点数精确求和
math.ceil(x)	$\lceil x \rceil$	向上取整，返回不小于x的最小整数
math.floor(x)	$\lfloor x \rfloor$	向下取整，返回不大于x的最大整数
math.factorial(x)	$x!$	返回x的阶乘，如果x是小数或负数，返回ValueError
math.gcd(a, b)		返回a与b的最大公约数
math.frexp(x)	$x = m * 2^e$	返回(m, e)，当x=0，返回(0.0, 0)
math.ldexp(x, i)	$x * 2^i$	返回 $x * 2^i$ 运算值，math.frexp(x)函数的反运算
math.modf(x)		返回x的小数和整数部分
math.trunc(x)		返回x的整数部分
math.copysign(x, y)	$ x * y /y$	用数值y的正负号替换数值x的正负号
math.isclose(a,b)		比较a和b的相似性，返回True或False
math.isfinite(x)		当x为无穷大，返回True；否则，返回False
math.isinf(x)		当x为正数或负数无穷大，返回True；否则，返回False
math.isnan(x)		当x是NaN，返回True；否则，返回False

Math库解析

■ math库中包括8个幂对数函数

函数	数学表示	描述
math.pow(x,y)	x^y	返回x的y次幂
math.exp(x)	e^x	返回e的x次幂，e是自然对数
math.expm1(x)	$e^x - 1$	返回e的x次幂减1
math.sqrt(x)	\sqrt{x}	返回x的平方根
math.log(x[,base])	$\log_{base} x$	返回x的对数值，只输入x时，返回自然对数，即 $\ln x$
math.log1p(x)	$\ln(1+x)$	返回1+x的自然对数值
math.log2(x)	$\log_2 x$	返回x的2对数值
math.log10(x)	$\log_{10} x$	返回x的10对数值

Math库解析

■ math库包括六个“三角双曲函数”

函数	数学表示	描述
math.degree(x)		角度x的弧度值转角度值
math.radians(x)		角度x的角度值转弧度值
math.hypot(x,y)	$\sqrt{x^2 + y^2}$	返回(x,y)坐标到原点(0,0)的距离
math.sin(x)	$\sin x$	返回x的正弦函数值，x是弧度值
math.cos(x)	$\cos x$	返回x的余弦函数值，x是弧度值
math.tan(x)	$\tan x$	返回x的正切函数值，x是弧度值
math.asin(x)	$\arcsin x$	返回x的反正弦函数值，x是弧度值
math.acos(x)	$\arccos x$	返回x的反余弦函数值，x是弧度值
math.atan(x)	$\arctan x$	返回x的反正切函数值，x是弧度值
math.atan2(y,x)	$\arctan y/x$	返回y/x的反正切函数值，x是弧度值
math.sinh(x)	$\sinh x$	返回x的双曲正弦函数值
math.cosh(x)	$\cosh x$	返回x的双曲余弦函数值
math.tanh(x)	$\tanh x$	返回x的双曲正切函数值
math.asinh(x)	$\operatorname{arcsinh} x$	返回x的反双曲正弦函数值
math.acosh(x)	$\operatorname{arccosh} x$	返回x的反双曲余弦函数值
math.atanh(x)	$\operatorname{arctanh} x$	返回x的反双曲正切函数值

Math库解析

■ math库包括4个高等特殊函数

函数	数学表示	描述
<code>math.erf(x)</code>	$\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$	高斯误差函数，应用于概率论、统计学等领域
<code>math.erfc(x)</code>	$\frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$	余补高斯误差函数， <code>math.erfc(x)=1 - math.erf(x)</code>
<code>math.gamma(x)</code>	$\int_0^\infty x^{t-1} e^{-x} dx$	伽玛（Gamma）函数，也叫欧拉第二积分函数
<code>math.lgamma(x)</code>	<code>ln(gamma(x))</code>	伽玛函数的自然对数

Math库概述

$$0.1+0.2+0.3=?$$

函数使用入门

函数的定义

函数（function）是指一个有命名的、执行某个计算的语句序列。在定义一个函数的时候，你需要指定函数名和语句序列。之后，你可以通过定义的函数名字调用（call）该函数。例如：print, input

内置函数

Python内部定义的函数，例如：print和input


自定义函数

当我们想实现一些特殊的功能而Python的自带函数中没有我们所需要的功能时，我们就需要自己编写函数来实现我们需要的特定功能。编写函数的语法形式如下：

```
def <函数名>(<参数列表>):  
    <函数体>  
    return <返回值列表>
```

函数使用

- 函数的调用和执行的一般形式：<函数名>(<参数列表>)
- 形参与实参概念
- 函数传递实参的方式



实例3: 天天向上的力量

实例代码3.1: 天天向上

一年365天，以第1天的能力值为基数，记为1.0，当好好学习时能力值相比前一天提高1‰，当没有学习时由于遗忘等原因能力值相比前一天下降1‰。每天努力和每天放任，一年下来的能力值相差多少呢？

实例代码3.1: 天天向上

实例代码3.1 e3.1DayDayUp365.py

```
1 #e3.1DayDayUp365.py
2 import math
3 dayup = math.pow(1.0 + 0.001, 365) # 提高0.001
4 daydown = math.pow(1.0 - 0.001, 365) # 放任0.001
5 print("向上: {:.2f}, 向下: {:.2f}.".format(dayup, daydown))
```

■ 运行结果如下，每天努力1‰，一年下来将提高44%，好像不多？请继续分析。

实例代码3.2: 天天向上

一年365天，如果好好学习时能力值相比前一天提高5‰，当放任时相比前一天下降5‰。效果相差多少呢？

实例代码3.2: 天天向上

实例代码 e3.2DayDayUp365.py
3.2

```
1 #e3.2DayDayUp365.py
2 import math
3 dayup = math.pow((1.0 + 0.005), 365) # 提高0.005
4 daydown = math.pow((1.0 - 0.005), 365) # 放任0.005
5 print("向上: {:.2f}, 向下: {:.2f}.".format(dayup, daydown))
```

■ 运行结果如下，每天努力5‰，一年下来将提高6倍！这个，不容小觑了吧？

实例代码3.3: 天天向上

一年365天，如果好好学习时能力值相比前一天提高1%，当放任时相比前一天下降1%。效果相差多少呢？

实例代码3.3: 天天向上

实例代 码3.3 e3.3DayDayUp365.py

```
#e3.3DayDayUp365.py
import math
1 dayfactor = 0.01
2 dayup = math.pow((1.0 + dayfactor), 365) # 提高
3 dayfactor
4 daydown = math.pow((1.0 - dayfactor), 365) # 放任
5 dayfactor
6 print(" 向上: {:.2f}, 向下: {:.2f}.".format(dayup,
daydown))
```

■ 运行结果如下，每天努力1%，一年下来将提高37倍。这个相当惊人吧！

实例代码3.4: 天天向上

一年365天，一周5个工作日，如果每个工作日都很努力，可以提高1%，仅在周末放任一下，能力值每天下降1%，效果如何呢？

实例代码3.4: 天天向上

实例代 码3.4 e3.4DayDayUp365.py

```
1 #e3.4DayDayUp365.py
2 dayup, dayfactor = 1.0, 0.01
3 for i in range(365):
4     if i % 7 in [6, 0]:#周六周日
5         dayup = dayup * (1 - dayfactor)
6     else:
7         dayup = dayup * (1 + dayfactor)
8     print("向上5天向下2天的力量: {:.2f}.".format(dayup))
```

■ 猜猜运行结果？每周努力5天，而不是每天，一年下来，水平仅是初始的4.63倍！与每天坚持所提高的237倍相去甚远

字符串类型及操作

字符串类型

- 字符串是用双引号""或者单引号'括起来的一个或多个字符。
- 字符串可以保存在变量中，也可以单独存在。
- 可以用type()函数测试一个字符串的类型

字符串类型

- Python语言转义符： \
- 输出带有引号的字符串，可以使用转义符
- 使用 \\ 输出带有转移符的字符串

```
>>> print("\\"大家好\\")
"大家好"
>>>
```

字符串类型

- 字符串是一个字符序列：字符串最左端位置标记为0，依次增加。字符串中的编号叫做“索引”

H	e	l	l	o		J	o	h	n
0	1	2	3	4	5	6	7	8	9

字符串类型

- 单个索引辅助访问字符串中的特定位置，格式为<string>[<索引>]

```
>>> greet="Hello John"
>>> print(greet[2])
l
>>> x=8
>>> print(greet[x-2])
J
>>>
```

字符串类型

- Python中字符串索引从0开始，一个长度为L的字符串最后一个字符的位置是L-1
- Python同时允许使用负数从字符串右边末尾向左边进行反向索引，最右侧索引值是-1

```
>>> greet[-4]
'J'
>>>
```

字符串类型

- 可以通过两个索引值确定一个位置范围，返回这个范围的子串，格式：
`<string>[<start>:<end>]`
- start和end都是整数型数值，这个子序列从索引start开始直到索引end结束，但不包括end位置。

```
>>> greet[0:3]
'Hel'
>>>
```

字符串类型

- 字符串之间可以通过+或*进行连接
- 加法操作(+)将两个字符串连接成为一个新的字符串
- 乘法操作(*)生成一个由其本身字符串重复连接而成的字符串

```
>>> "pine" + "apple"
'pineapple'
>>> 3 * "pine"
'pinepinepine'
>>>
```

字符串类型

- len()函数能否返回一个字符串的长度

```
>>> len("pine")
4
>>> len("祖国，您好！")
6
```

字符串类型

- 大多数数据类型都可以通过str()函数转换为字符串

```
>>> str(123)
'123'
>>> str(123.456)
'123.456'
>>> str(123e+10)
'1230000000000.0'
```

字符串五大操作总结

操作符	描述
x+y	连接字符串x与y
x*n	复制n次字符串x
str[i]	索引，返回第i个字符
str[N:M]	切片，返回索引第N到第M的子串，其中不包含M
x in s	如果x是s的字串，返回True

字符串使用实例

输入一个月份数字，返回对应月份名称缩写，这个问题的IPO模式是：

输入：输入一个表示月份的数字(1-12)

处理：利用字符串基本操作实现该功能

输出：输入数字对应月份名称的缩写

字符串使用实例

	月份	字符串中位置
Jan	1	0
Feb	2	3
Mar	3	6
Apr	4	9

字符串使用实例

```
# month.py
months="JanFebMarAprMayJunJulAugSepOctNovDec"
n=input("请输入月份数(1-12):")
pos=(int(n)-1) * 3
monthAbbrev=months[pos:pos+3]
print("月份简写是"+monthAbbrev+".")
```

```
>>>
请输入月份数(1-12):7
月份简写是Jul.
>>>
```

转义操作符

用转义符可以在字符串中表达一些不可直接打印的信息

例如：用\n表示换行

字符串"Hello\nWorld\n\nGoodbye 32\n"

用print()函数打印后的输出效果如下：

Hello

World

Goodbye 32

转义操作符

控制字符	作用
\a	蜂鸣，响铃
\'	单引号
\n	换行，光标移动到下行首行
\r	换行，光标移动到本行首行
\t	水平制表
\v	垂直制表

内置的字符串处理函数

操作	含义
+	连接
*	重复
<string>[]	索引
<string>[:]	剪切
len(<string>)	长度
<string>.upper()	字符串中字母大写
<string>.lower()	字符串中字母小写
<string>.strip()	去两边空格及去指定字符
<string>.split()	按指定字符分割字符串为数组
<string>.join()	连接两个字符串序列
<string>.find()	搜索指定字符串
<string>.replace()	字符串替换
for <var> in <string>	字符串迭代

内置的字符串处理函数

微实例3.2：恺撒密码。

恺撒密码是古罗马恺撒大帝用来对军事情报进行加密的算法，它采用了替换方法对信息中的每一个英文字符循环替换为字母表序列该字符后面第三个字符，对应关系如下：

原文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

原文字符P，其密文字符C满足如下条件：

$$C = (P + 3) \bmod 26$$

解密方法反之，满足：

$$P = (C - 3) \bmod 26$$

内置的字符串处理函数

微实例

m3.2 CaesarCode.py

3.2

```
1 plaincode = input("请输入明文: ")
2 for p in plaincode:
3     if ord("a") <= ord(p) <= ord("z"):
4         print(chr(ord("a")+(ord(p)-
5 ord("a")+3)%26), end='')
6     else:
7         print(p, end='')
```

微实例运行结果如下：

```
>>>
请输入明文: python is an excellent language.
sbwkrq lv dq hafhoohqw odqjxdjh.
```

内置的字符串处理方法

方法	描述
str.lower()	返回字符串str的副本，全部字符小写
str.upper()	返回字符串str的副本，全部字符大写
str.islower()	当str所有字符都是小写时，返回True，否则False
str.isprintable()	当str所有字符都是可打印的，返回True，否则False
str.isnumeric()	当str所有字符都是字符时，返回True，否则False
str.isspace()	当str所有字符都是空格，返回True，否则False
str.endswith(suffix[, start[, end]])	str[start: end] 以suffix结尾返回True，否则返回False
str.startswith(prefix[, start[, end]])	str[start: end] 以prefix开始返回True，否则返回False
str.split(sep=None, maxsplit=-1)	返回一个列表，由str根据sep被分割的部分构成
str.count(sub[, start[, end]])	返回str[start: end]中sub子串出现的次数
str.replace(old, new[, count])	返回字符串str的副本，所有old子串被替换为new，如果count给出，则前count次old出现被替换
str.center(width[, fillchar])	字符串居中函数，详见函数定义
str.strip([chars])	返回字符串str的副本，在其左侧和右侧去掉chars中列出的字符
str.zfill(width)	返回字符串str的副本，长度为width，不足部分在左侧添0
str.format()	返回字符串str的一种排版格式，3.6节将详细介绍
str.join(iterable)	返回一个新字符串，由组合数据类型（见第6章）iterable变量的每个元素组成，元素间用str分割