第一部分 数字类型

1.1 数字类型的组成

1.1.1 整数——不同进制的转换

- 默认输入十进制
- 二进制0b、八进制0o、十六进制0x

In [4]:

```
16 == 0b10000 == 0o20 == 0x10
```

Out[4]:

```
True
```

- 十进制与其他进制的转换

In [5]:

```
a = bin(16)    # 转二进制
b = oct(16)    # 转八进制
c = hex(16)    # 转十六进制
print(a, b, c)
```

```
0b10000 0o20 0x10
```

注意：上述转换后结果为字符串类型

In [10]:

```
a == b == c
```

Out[10]:

```
False
```

In [11]:

```
type(a)
```

Out[11]:

```
str
```

- 其他进制转十进制

In [12]:

```
d = int(a, 2)        # 二进制转十进制
e = int(b, 8)        # 八进制转十进制
f = int(c, 16)       # 十六进制转十进制
print(d, e, f)
```

16 16 16

1.1.2 浮点数——不确定性

- 不确定小数问题

In [13]:

```
(0.1+0.2) == 0.3
```

Out[13]:

False

In [14]:

```
0.1+0.2
```

Out[14]:

0.30000000000000004

计算机采用二进制小数来表示浮点数的小数部分

- 部分小数不能用二进制小数完全表示

二进制	十进制
0.0001100110011001100 1	0.09999942779541016
0.0011001100110011	0.1999969482421875
0.01001100110011001	0.29999542236328125
0.01100110011001101	0.40000152587890625
0.1	0.5

- 通常情况下不会影响计算精度

In [15]:

```
0.1 + 0.7
```

Out[15]:

0.7999999999999999

- 四舍五入获得精确解

In [16]:

```
a = 3*0.1
print(a)
```

0.30000000000000004

In [17]:

```
b = round(a, 1)
print(b)
b == 0.3
```

0.3

Out[17]:

True

1.1.3 复数——a+bj

- 大写J或小写j均可

In [31]:

```
3+4j
2+5J
```

Out[31]:

(2+5j)

- 虚部系数为1时，需要显式写出

In [ ]:

```
2+1j
```

1.2 数字运算操作符（a 操作符 b）

- 加减乘除运算 + - / *

In [7]:

```
(1+3-4*2)/5
```

Out[7]:

-0.8

- 取反 -

In [18]:

```
x = 1
-x
```

Out[18]:

-1

- 乘方运算 **

In [33]:

```
2**3
```

Out[33]:

8

- 整数商// 和 模运算%

In [41]:

```
13//5    # 整数商     x/y 向下取整数
```

Out[41]:

2

In [ ]:

```
13 % 5    # 模运算     余数 13=2*5+3
```

几点说明

- 整数与浮点数运算结果是浮点数
- 除法运算的结果是浮点数

In [19]:

```
1+1.5
```

Out[19]:

2.5

In [20]:

```
2/5
```

Out[20]:

0.4

In [21]:

```
8/4
```

Out[21]:

2.0

1.3 数字运算操作函数 function(x, ...)

- 求绝对值 abs()

In [22]:

```
abs(-5)
```

Out[22]:

5

In [23]:

```
abs(3+4j)  # 对复数a+bj 执行的是求模运算(a^2+b^2)^0.5
```

Out[23]:

5.0

- 幂次方 pow(x,n)

In [49]:

```
pow(2, 5)  # pow(x,n) x的n次方  等价于x**n
```

Out[49]:

32

In [50]:

```
pow(2, 5, 3) # 2^5 % 3  更快速
```

Out[50]:

2

- 四舍五入 round(x,n)

In [25]:

```
a = 1.618
print(round(a))       # 默认四舍五入为整数
```

2

In [26]:

```
print(round(a, 2))    # 参数2表示四舍五入后保留2位小数
```

1.62

In [27]:

```
print(round(a, 5))    # 位数不足，无需补齐
```

1.618

- 整数商和模运算 divmod(x,y)
- 等价于返回二元元组（x//y,x % y）

In [28]:

```
divmod(13, 5)    # 较（x//y,x % y）更快，只执行了一次x/y
```

Out[28]:

(2, 3)

- 序列最大/最小值 max( ) min( )

In [29]:

```
max(3, 2, 3, 6, 9, 4, 5)
```

Out[29]:

9

In [30]:

```
a = [3, 2, 3, 6, 9, 4, 5]
print("max:", max(a))
print("min:", min(a))
```

```
max: 9
min: 2
```

- 求和sum(x)

In [36]:

```
sum((1, 2, 3, 4, 5))
```

Out[36]:

15

- 借助科学计算库 math\scipy\numpy

In [38]:

```
import math   # 导入库
print(math.exp(1))   # 指数运算 e^x
print(math.log2(2))  # 对数运算
print(math.sqrt(4))  # 开平方运算  等价于4^0.5
```

2.718281828459045
1.0
2.0

In [39]:

```
import numpy as np
a = [1, 2, 3, 4, 5]
print(np.mean(a))    # 求均值
print(np.median(a))  # 求中位数
print(np.std(a))     # 求标准差
```

3.0
3.0
1.4142135623730951

第二部分 字符串类型

2.1 字符串的表达

- 用""或''括起来的任意字符

In [40]:

```
print("Python")
print('Python')
```

Python
Python

- 字符串中有双引号或单引号的情况

双中有单

In [41]:

```
print("I'm 18 years old")
```

```
I'm 18 years old
```

单中有双

In [42]:

```
print('"Python" is good')
```

```
"Python" is good
```

**双中有双，单中有单——转义符 **

In [44]:

```
# print(""Python" is good")
print("\"Python\" is good")  # \ 我是个字符呀
```

```
"Python" is good
```

转义符可以用来换行继续输入

In [45]:

```
# 等等，我还没完事！
s = "py\
thon"
print(s)
```

```
python
```

2.2 字符串的性质

2.2.1 字符串的索引

In [47]:

```
s = "My name is Peppa Pig"
```

变量名[位置编号]

- 正向索引——从零开始递增
- 位置编号不能超过字符串的长度

In [48]:

```
print(s[0])
print(s[2])
print(s[5])
```

```
M

m
```

In [49]:

```
s = "My name is Peppa Pig"
```

- 反向索引——从-1开始递减

In [50]:

```
print(s[-1])
print(s[-3])
print(s[-5])
```

```
g
P
a
```

索引只能获得一个字符，如何获得多个字符?

2.2.2 字符串的切片

变量名[开始位置：结束位置：切片间隔]

- 切片间隔如不设置默认为1，可省略
- 切片范围不包含结束位置

In [52]:

```
s = "Python"
print(s[0:3:1])
```

```
Pyt
```

In [53]:

```
print(s[0:3])
```

```
Pyt
```

In [54]:

```
print(s[0:3:2])
```

Pt

- 起始位置是0 可以省略
- 结束位置省略，代表可以取到最后一个字符
- 可以使用反向索引

In [55]:

```
s = "Python"
print(s[0:6])
```

Python

In [56]:

```
print(s[:6])
```

Python

In [57]:

```
print(s[:])
```

Python

In [59]:

```
print(s[-6:])
```

Python

反向切片

- 起始位置是-1也可以省略
- 结束位置省略，代表可以取到第一个字符

In [62]:

```
s = "123456789"
print(s[-1:-10:-1])
```

987654321

In [71]:

```
print(s[:-10:-1])
```

987654321

In [72]:

```
print(s[::-1])
```

```
987654321
```

2.3 字符串操作符

2.3.1 字符串的拼接

- 字符串1+字符串2

In [73]:

```
a = "I love "
b = "my wife "
a+b
```

Out[73]:

```
'I love my wife '
```

2.3.2 字符串的成倍复制

- 字符串 * n　n * 字符串

In [74]:

```
c = a+b
print(c*3)
print(3*c)
```

```
I love my wife I love my wife I love my wife
I love my wife I love my wife I love my wife
```

2.2.3 成员运算

- **子集in全集**　任何一个连续的切片都是原字符串的子集

In [75]:

```
folk_singers = "Peter, Paul and Mary"
"Peter" in folk_singers
```

Out[75]:

```
True
```

In [76]:

```
"PPM" in folk_singers
```

Out[76]:

False

- **遍历字符串字符**　for 字符 in 字符串

In [77]:

```
for s in "Python":
    print(s)
```

P
y
t
h
o
n

2.4 字符串处理函数

2.4.1 字符串的长度

- 所含字符的个数

In [78]:

```
s = "python"
len(s)
```

Out[78]:

6

2.4.2 字符编码

将中文字库，英文字母、数字、特殊字符等转化成计算机可识别的二进制数

- 每个单一字符对应一个唯一的互不重复的二进制编码
- Python 中使用的是Unicode编码

将字符转化为Unicode码——ord(字符)

In [79]:

```
print(ord("1"))
print(ord("a"))
print(ord("*"))
print(ord("中"))
print(ord("国"))
```

```
49
97
42
20013
22269
```

将Unicode码转化为字符——chr(Unicode码)

In [80]:

```
print(chr(1010))
print(chr(10000))
print(chr(12345))
print(chr(23456))
```

```
ϲ
✐
卄
宠
```

2.5 字符串的处理方法

2.5.1 字符串的分割——字符串.split(分割字符)

- 返回一个列表
- 原字符串不变

上述特性适合以下所有字符串处理方法

In [81]:

```
languages = "Python C C++ Java PHP R"
languages_list = languages.split(" ")
print(languages_list)
print(languages)
```

```
['Python', 'C', 'C++', 'Java', 'PHP', 'R']
Python C C++ Java PHP R
```

2.5.2 字符串的聚合——“聚合字符”.join(可迭代数据类型)

- 可迭代类型 如：字符串、列表

In [82]:

```
s = "12345"
s_join = ",".join(s)
s_join
```

Out[82]:

'1,2,3,4,5'

- 序列类型的元素必须是字符类型

In [168]:

```
# s = [1, 2, 3, 4, 5]
s = ["1", "2", "3", "4", "5"]
"*".join(s)
```

Out[168]:

'1*2*3*4*5'

3.5.3 删除两端特定字符——字符串.strip(删除字符)

- strip从两侧开始搜索，遇到指定字符执行删除，遇到非指定字符，搜索停止
- 类似的还有左删除lstrip和右删除rstrip

In [85]:

```
s = "      I have many blanks      "
print(s.strip(" "))                          # 还有吗？hahaha
print(s.lstrip(" "))
print(s.rstrip(" "))
print(s)
```

```
I have many blanks
I have many blanks
      I have many blanks
      I have many blanks
```

3.5.4 字符串的替换——字符串.replace("被替换"，"替换成")

In [84]:

```
s = "Python is coming"
s1 = s.replace("Python","Py")
print(s1)
```

```
Py is coming
```

3.5.5 字符串统计——字符串.count("待统计字符串")

In [179]:

```
s = "Python is an excellent language"
print("an:", s.count("an"))
print("e:", s.count("e"))
```

```
an: 2
e: 4
```

3.3.6 字符串字母大小写

- 字符串.upper() 字母全部大写

In [86]:

```
s = "Python"
s.upper()
```

Out[86]:

'PYTHON'

- 字符串.lower() 字母全部小写

In [92]:

```
print(s.lower())
print(s)
```

```
python
Python
```

- 字符串.title()首字母大写

In [89]:

```
s.title()
```

Out[89]:

'Python'

第三部分 布尔类型 TRUE or False

3.1 逻辑运算的结果

In [93]:

```
a = 10
print(a > 8)
print(a == 12)
print(a < 5)
```

```
True
False
False
```

- any() all()

In [94]:

```
print(any([False,1,0,None]))    # 0 False None 都是无
print(all([False,1,0,None]))
```

```
True
False
```

3.2 指示条件

In [186]:

```
n = 2800
while True:
    m = eval(input("请输入一个正整数："))
    if m == n:
        print("你猜对啦")
        break
    elif m > n:
        print("太大了")
    else:
        print("太小了")
```

```
请输入一个正整数：28
太小了
请输入一个正整数：2800
你猜对啦
```

3.3 作为掩码

In [194]:

```
import numpy as np
x = np.array([[1, 3, 2, 5, 7]])   # 定义 numpy数组
print(x > 3)
x[x > 3]
```

```
[[False False False  True  True]]
```

Out[194]:

```
array([5, 7])
```

第四部分 类型判别及类型转换

4.1 类型判别

- type(变量)

In [96]:

```
age = 20
name = "Ada"
print(type(age))
print(type(name))
```

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

- isinstance(变量,预判类型) **承认继承**
- 变量类型是预判类型的子类型，则为真，否则为假

In [97]:

```
print(isinstance(age, int))        # 承认继承
```

```
True
```

In [98]:

```
print(isinstance(age, object))
print(isinstance(name, object))    # object 是老祖宗
```

```
True
True
```

- 字符串检查方法

字符串.isdigit()字符是否只有数字组成

In [99]:

```
age = "20"
name = "Ada"
```

In [100]:

```
age.isdigit()
```

Out[100]:

True

In [101]:

```
name.isdigit()
```

Out[101]:

False

字符串.isalpha()字符是否只有字母组成

In [206]:

```
name.isalpha()
```

Out[206]:

True

In [207]:

```
age.isalpha()
```

Out[207]:

False

字符串.isalnum()字符是否只有数字和字母组成

In [102]:

```
"Ada20".isalnum()     # 比如可用于判断用户名是否合法
```

Out[102]:

True

4.2 类型转换

- 数字类型转字符串　str(数字类型)

In [210]:

```
age = 20
print("My age is "+str(age))
```

```
My age is 20
```

- 仅有数字组成的字符串转数字 int() float() eval()

In [105]:

```
s1 = "20"
s2 = "10.1"
```

In [108]:

```
int(s1)      # 仅整型
# int(s2)
```

Out[108]:

```
20
```

In [109]:

```
float(s1)
```

Out[109]:

```
20.0
```

In [110]:

```
float(s2)
```

Out[110]:

```
10.1
```

In [111]:

```
eval(s1)
```

Out[111]:

```
20
```

In [112]:

```
eval(s2)
```

Out[112]:

```
10.1
```