# Python

目录

[Python 1](#_Toc531620941)

[1.Hello Python 1](#_Toc531620942)

[2.python基础 1](#_Toc531620943)

[2.1数据类型和变量 1](#_Toc531620944)

[2.2条件判断 5](#_Toc531620945)

[2.3循环 5](#_Toc531620946)

[2.4List和tuple 7](#_Toc531620947)

[2.5dict和set 8](#_Toc531620948)

[3.函数 9](#_Toc531620949)

[3.1定义函数 9](#_Toc531620950)

[4.高级特性 9](#_Toc531620951)

## 1.Hello Python

print(**"hello,python"**)  
*#等待用户输出*name = input();  
print(name)

## 2.python基础

### 2.1数据类型和变量

#### 2.1.1整数

Python可以处理任意大小的整数，在程序上和数学上的写法一模一样

#### 2.1.2浮点数

整数和浮点数不同，整数的计算是精确地，而浮点数会有四舍五入的误差

#### 2. 1.3字符串

字符串是以单引号或双引号括起来的任意文本，如果单引号本身也是一个字符串，可以用双引号括起来，如"I'm OK"，若既包含单引号又包含双引号，则可以用\来标识，如"\'I'm OK"，或者'\"I\'m OK'，

Python的字符串类型是str,在内存以Unicode表示，一个字符对应若干字节。如果要在网络上传输，就需要把str变为以字节为单位的bytes，Python对bytes类型的数据用带b前缀的单引号或双引号表示，如

x = b'ABC'

需要注意的是: 'ABC'和b'ABC'，前者是str，后者虽然内容显示得和前者一样，但bytes的每个字符都只占用一个字节，以Unicode表示的str通过encode()方法可以编码指定为bytes，如

>>> 'ABC'.encode('ascii')

b'ABC'

>>> '中文'.encode('utf-8')

b'\xe4\xb8\xad\xe6\x96\x87'

>>> '中文'.encode('ascii')

Traceback (most recent call last):

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

UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-1: ordinal not in range(128)

反过来，如果读取了字节流，需要转换成字符，则需要调用decde(),如果bytes中只有一小部分是无效字节，可以传入errors='ignore'忽略错误的字节，要计算str含有多少个字符可以调用len()函数；字符串转int调用int(str)

#### 2. 1.4布尔值

True/False

#### 2. 1.5空值

用none表示，是一个特殊字符，代表空值

#### 2. 1.6变量

Python中的变量可以是任意类型，命名规则如驼峰、下划线；这种任意类型可以多次赋值，更加灵活

#### 2. 1.7常量

所谓的常量就是不能变的变量。比如PI就是一个常量；但python没有任机智保证PI不会被改；在python中有两种除法

>>> 10 / 3

3.3333333333333335

还有一种除法是地板除

>>> 10 *// 3*

3

#### 2. 1.8格式化

Python格式化和c语言的格式化相似，%是用来格式化字符串的，如下

| **占位符** | **替换内容** |
| --- | --- |
| %d | 整数 |
| %f | 浮点数 |
| %s | 字符串 |
| %x | 十六进制整数 |

>>> 'Hello, %s' % 'world'

'Hello, world'

>>> 'Hi, %s, you have $%d.' % ('Michael', 1000000)

'Hi, Michael, you have $1000000.'

%s永远起作用，它会把任何数据类型转换为字符串，如果字符串里的%是一个普通字符，这个时候就需要转义，用%%来表示%

>>> 'growth rate: %d %%' % 7

'growth rate: 7 %'

### 2.2条件判断

#### 2.2.1i f else

age = 3

**if** age >= 18:

**print**('your age is', age)

**print**('adult')

**else**:

**print**('your age is', age)

**print**('teenager')

注意：不要少了：

### 2.3循环

#### 2.3.1for…in

For x in …循环就是把每个元素带入变量x

sum = 0

**for** x **in** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:

sum = sum + x

print(sum)

调用range()函数可以生成一个整数序列，如range(101)会生成0-100的整数序列

#### 2.3.2while循环

只要条件满足就会不断循环

sum = 0

n = 99

**while** n > 0:

sum = sum + n

n = n - 2

**print**(sum)

#### 2.3.3break

在循环中，break可以跳出当前循环

n = 1

while n <= 100:

print(n)

n = n + 1

**if** n > 10: *# 当n = 11时，条件满足，执行break语句*

**break** *# break语句会结束当前循环*

print('END')

#### 2.3.4 continue

在循环的过程中，可以通过continue跳出当前循环，直接开始下一次循环

n = 0

**while** n < 10:

n = n + 1

**if** n % 2 == 0: *# 如果n是偶数，执行continue语句*

**continue** *# continue语句会直接继续下一轮循环，后续的print()语句不会执行*

**print**(n)

### 2.4List和tuple

#### 2.4.1 list

Python内置的一种数据类型是列表：list，list是一种有序的集合。可以随时添加和删除其中的元素

>>> classmates = ['Michael', 'Bob', 'Tracy']

>>> classmates

['Michael', 'Bob', 'Tracy']

如上，classmates就是一个list，当下表为-1默认去取最后一个元素，左后一个元素的索引是len(classmates) – 1

#### 2.4.2 tuple

tuple是另一种有序列表叫元组，和list相似，但是tuple一旦初始化就不能修改，可使代码更具安全性

>>> t = (1, 2)

>>> t

(1, 2)

定义固定长度的tuple

>>> t = (1,)

>>> t

(1,)

定义空的tuple

>>> t = ()

>>> t

()

### 2.5dict和set

#### 2.5.1dict

使用键-值得存储，具有极快的查找速度相当于java里的map

#### 2.5.2 set

set和dict相似，是一组key的组合，但不存储value，由于key不能重复，所以在set中没有重复的key，要创建一个set需要提供一个list作为输入集合

>>> s = set([1, 2, 3])

>>> s

{1, 2, 3}

Set可以看做数学意义上的无序和无重复元素的集合，因此两个set可以做数学意义上的交集、并集等操作

>>> s1 = set([1, 2, 3])

>>> s2 = set([2, 3, 4])

>>> s1 & s2

{2, 3}

>>> s1 | s2

{1, 2, 3, 4}

## 3.函数

### 3.1定义函数

def my\_abs(x):

if x >= 0:

return x

else:

return –x

## 4.高级特性

### 4.1切片

如取list前三个元素

>>> L = ['Michael', 'Sarah', 'Tracy', 'Bob', 'Jack']

>>> L[0:3]

['Michael', 'Sarah', 'Tracy']

由于list支持倒数第一个元素-1，痛死也支持倒数切片

L【-1，1】

tuple切片结果仍然是tuple

### 4.2迭代

#### 4.2.1迭代list

for (i=0; i<list.length; i++) {

n = list[i];

}

#### 4.2.2迭代dist

>>> d = {'a': 1, 'b': 2, 'c': 3}

>>> **for** key **in** d:

... print(key)

...

a

c

b

单独迭代值

for value in d.values()

可同时迭代键和值

for k, v in d.items()

#### 4.3判断对象是否可迭代

通过collections模块的Iterable类型判断

>>> **from** collections **import** Iterable

>>> isinstance('abc', Iterable) *# str是否可迭代*

True

>>> isinstance([1,2,3], Iterable) *# list是否可迭代*

True

>>> isinstance(123, Iterable) *# 整数是否可迭代*

False

若要对list实现下标循环，可用内置的enumerate函数把对象变成索引-元素对，这样可以在for循环中同时迭代索引和元素

>>> **for** i, value **in** enumerate(['A', 'B', 'C']):

... print(i, value)

...

0 A

1 B

2 C

### 4.3列表生成式

>>> L = []

>>> **for** x **in** range(1, 11):

... L.append(x \* x)

...

>>> L

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

>>> [x \* x **for** x **in** range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

>>> [x \* x **for** x **in** range(1, 11) **if** x % 2 == 0]

[4, 16, 36, 64, 100]

>>> [m + n **for** m **in** 'ABC' **for** n **in** 'XYZ']

['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']

### 4.4生成器

通过列表生成式，我们可以直接创建一个列表。但是，受到内存限制，列表容量肯定是有限的。而且，创建一个包含100万个元素的列表，不仅占用很大的存储空间，如果我们仅仅需要访问前面几个元素，那后面绝大多数元素占用的空间都白白浪费了。

所以，如果列表元素可以按照某种算法推算出来，那我们是否可以在循环的过程中不断推算出后续的元素呢？这样就不必创建完整的list，从而节省大量的空间。在Python中，这种一边循环一边计算的机制，称为生成器：generator。

将[]换成（）即成为一个生成器

>>> L = [x \* x **for** x **in** range(10)]

>>> L

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> g = (x \* x **for** x **in** range(10))

>>> g

<generator object <genexpr> at 0x1022ef630>

>>> g = (x \* x **for** x **in** range(10))

>>> **for** n **in** g:

... print(n)

...

### 4.5迭代器

可直接用for循环的对象统称为可迭代对象一类是generator,一类是集合数据类型，如list,tuple,dict,set,str等，这些可以直接作用于for循环的对象成为迭代对象：Interable

可以使用ininstance()判断一个对象是否是Interable对象

>>> **from** collections **import** Iterable

>>> isinstance([], Iterable)

True

>>> isinstance({}, Iterable)

True

>>> isinstance('abc', Iterable)

True

>>> isinstance((x **for** x **in** range(10)), Iterable)

True

>>> isinstance(100, Iterable)

False