



Python基础语法

目录

1	Python语法元素
2	Python数据类型
3	Python程序流控制
4	函数、类与参数
5	文件与数据组织

‘Hello,world!’

Python 的输出与输入

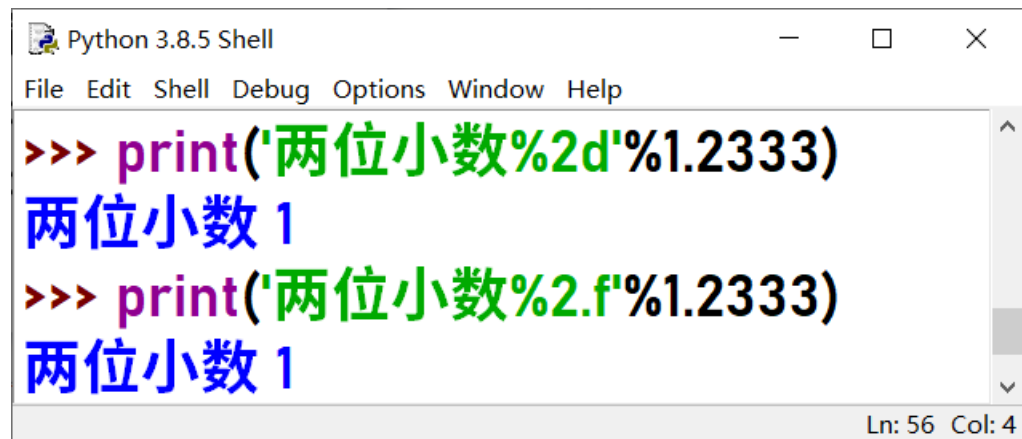
- 输出函数: `print()`
- 输入函数: `input()`



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> print('Hello, world!')
Hello, world!
>>> input('请输入: ')
请输入: Hello, World!
'Hello, World!'
Ln: 40 Col: 4
```

Python 格式化输出

- 使用%符号
- 使用 `format()` 方法



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> print('两位小数%2d'%1.2333)
两位小数 1
>>> print('两位小数%2.f'%1.2333)
两位小数 1
Ln: 56 Col: 4
```

Python赋值语句

直接赋值

```
Python 3.8.5 S...  
File Edit Shell Debug Options  
Window Help  
>>> a=3  
>>> a  
3  
Ln: 6 Col: 4
```

序列赋值

```
Python 3.8.5 S...  
File Edit Shell Debug Options  
Window Help  
>>> a,b=2,3  
>>> a  
2  
>>> b  
3  
Ln: 11 Col: 4
```

链式赋值

```
Python 3.8.5 S...  
File Edit Shell Debug Options  
Window Help  
>>> a=b=3  
>>> a  
3  
>>> b  
3  
Ln: 20 Col: 4
```

增强赋值

```
Python 3.8.5 S...  
File Edit Shell Debug Options  
Window Help  
>>> a=3  
>>> a+=6  
>>> a  
9  
Ln: 15 Col: 4
```

单行注释: #

多行注释: ''' 注释

可换行'''

Python命名规则

1) 命名的规范性

变量名可以包括字母、数字、下划线
变量名区分大小写



变量名不可以使用数字开头



系统关键字（保留字）不能做变量名使用

除了下划线之外，其它符号不能做为变量名使用

2) 驼峰命名法

大驼峰:每一个单词的首字母都大写 FirstName LastName

小驼峰:第一个单词以小写字母开始，后续单词的首字母大写 firstName lastName

3) 下划线分隔

命名举例 first_name last_name

系统关键字（保留字）

系统关键字（保留字） 是程序语言设计中作为命令或常量等的单词

保留字和关键字**不允许**作为变量或其他标识符使用

Python保留字和关键字列表

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', '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']
```

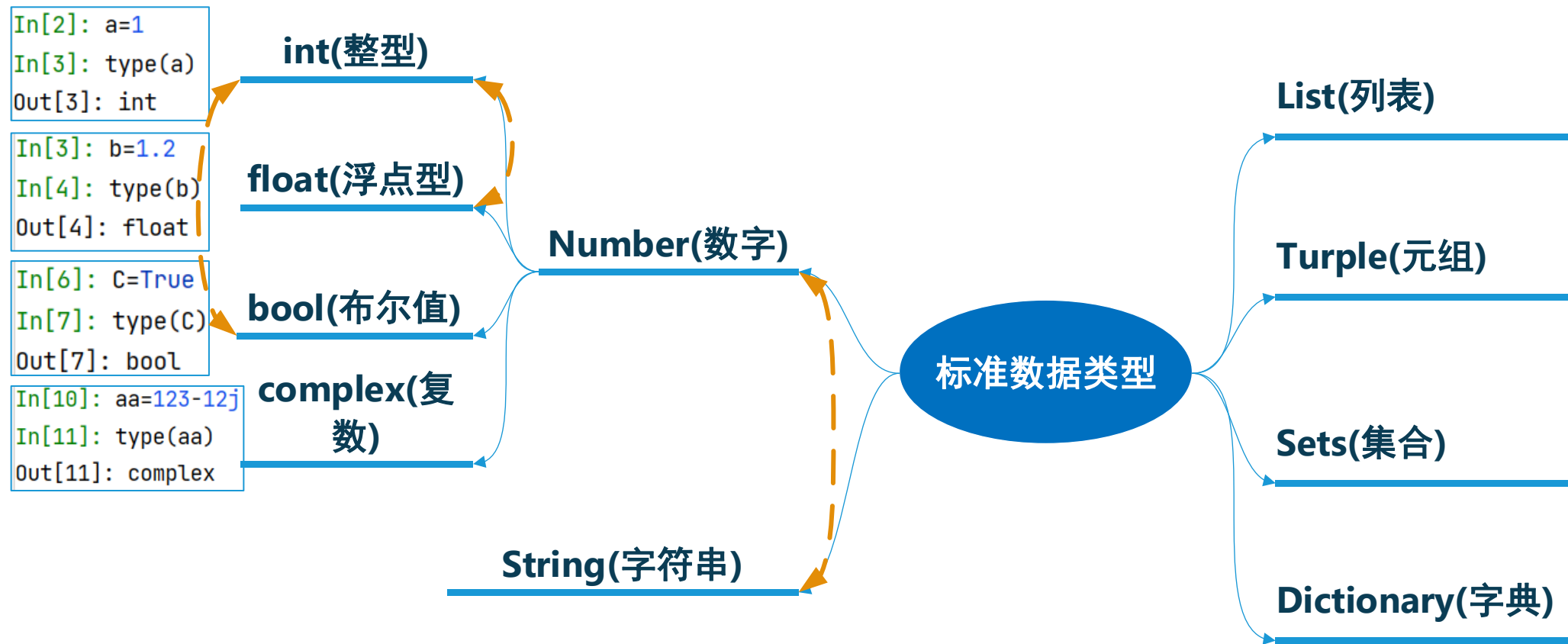
查看保留字和关键字

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', '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']
```

目录

1	Python语法元素
2	Python数据类型
3	Python程序流控制
4	函数、类与参数
5	文件与数据组织

Python数据类型



通过type()函数可以查看数据类型

Python计算操作

操作符	描述	实例
+	加法-返回两操作数相加的结果	3+2返回5
-	减法-返回左操作数减去右操作数的结果	3-2返回1
*	乘法-返回两操作数相乘的结果	3*2返回6
/	除法-返回右操作数除左操作数的结果	3/2返回1但3.0/2返回1.5
%	模-返回右操作数对左操作数取模的结果	5%3返回2
**	指数-执行对操作指数的计算	3**2返回9
//	取商-返回右操作数对左操作数取商的结果	3.0//2返回1.0

Python增强赋值操作符

操作符	描述	例子
=	简单的赋值运算符，赋值从右侧操作数左侧操作数	c=a+b 将 a 和 b 相加的值赋值给 c
+=	加法AND赋值操作符，它增加了右操作数左操作数和结果赋给左操作数	c+=a 相 当 于 c=c+a
-=	减法AND赋值操作符，它减去右边的操作数从左边操作数，并将结果赋给左操作数	c-=a相当于c=c-a
=	乘法AND赋值操作符，它乘以右边的操作数与左操作数，并将结果赋给左操作数	c=a 相 当 于 c=c*a
/=	除法AND赋值操作符，它把左操作数与正确的操作数，并将结果赋给左操作数	c/=a相当于c=c/a
%=	模量AND赋值操作符，它需要使用两个操作数的模量和分配结果左操作数	c%=a 相 当 于 c=c%a
=	指数AND赋值运算符，执行指数（功率）计算操作符和赋值给左操作数	c=a 相 当 于 c=c**a
//=	取商，并分配一个值，执行取商并将结果赋值给左操作数	c//=a 相 当 于 c=c//a

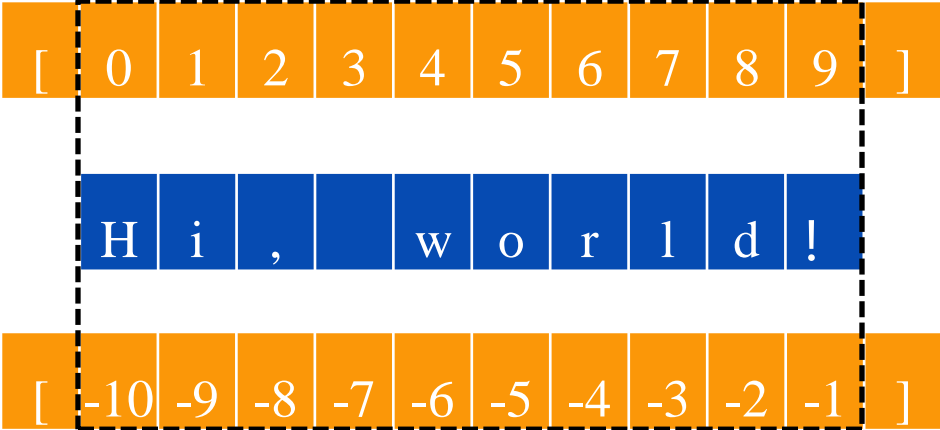
Python比较操作符

操作符	描述	实例
==	如果两个操作数的值相等则返回True， 否则返回False	3==2返回False
!=	如果两个操作数的值不等则返回True， 否则返回False	3!=2返回True
>	如果左操作数大于右操作数则返回True， 否则返回False	3>2返回True
<	如果左操作数小于右操作数则返回True， 否则返回False	3<2返回False
>=	如果左操作数大于或等于右操作数则返回True， 否则返回False	3>=3返回True
<=	如果左操作数小于或等于右操作数则返回True， 否则返回False	2<=2返回True

Python字符串相关概念

字符串是str类型的对象，由引号(成对的单引号、双引号、三引号)括起来的序列字符组成，也可以使用str()函数规定。

操作	实现过程
合并字符串	str1+str2
复制字符串	str*n
修改字符串	str1.replace(old, New[, count])
查找特定字符	str.find(char)



切片操作

切片操作基本表达式： `object[start_index:end_index:step]`

*len()函数为Python内置函数，用来返回对象的长度或项目个数。

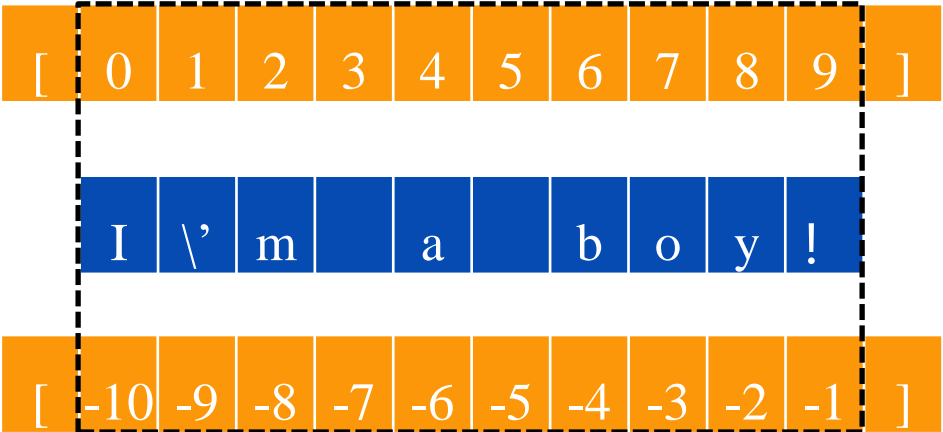
字符串相关函数

名称	说明
S.split(sep="", num=string.count(str))	num=string.count(str)) 以 sep 为分隔符截取字符串，如果 num 有指定值，则仅截取 num 个子字符串
S.strip([chars])	返回字符串的一个副本，删除前导和尾随字符。chars参数是一个字符串，指定要移除的字符集。如果省略或为None，则chars参数默认为删除空白字符。
S.isalnum()	如果字符串至少有一个字符，并且所有字符都是数字或者字母，则返回true，否则返回false。
S.count(sub[,start[,end]])	返回在[start, end]范围内的子串sub非重叠出现的次数。可选参数start和end都以切片表示法解释。
S.lstrip()	去掉字符串的左边空格。
S.rstrip()	去掉字符串的右边空格。
S.upper()	将小写字母完全变成大写字母。
S.lower()	将大写字母完全变成小写字母。
S.capitalize()	把字符串的第一个字母变成大写。
S.title()	把所有单词的第一个字母变成大写。

转义字符

所有的ASCII码都可以用 “\” 加数字（一般是8进制数字）来表示。而C中定义了一些字母前加 “\” 来表示常见的那些不能显示的ASCII字符，如\0,\t,\n等，就称为**转义字符**。

转义字符	意义
\n	换行(LF)，将当前位置移到下一行开头
\r	回车(CR)，将当前位置移到本行开头
\t	水平制表(HT)（跳到下一个TAB位置）
\v	垂直制表(VT)
\\	代表一个反斜线字符\"
\'	代表一个单引号（撇号）字符
\"	代表一个双引号字符
\?	代表一个问号
\0	空字符(NUL)



列表

列表是一种可以包含任何种类的对象、有着多种长短、可随意修改的序列数据。常用中括号' [] ' 或list()函数来创建列表，用' , ' 分割列表里面的每个元素，列表适用切片操作。

操作	实现过程
增加元素	List.insert(index, obj)
删除元素	List.pop(index)
修改元素	List[index]=new
查找元素	List.index(obj)
判断元素存在	obj in List
合并列表	List1+List2
重复列表	List*3

A = [1 , 2 , 3 , 4]
0 1 2 3

A = [1 , 2 , insert , 3 , 4]
0 1 2 3 4

A = [9 , 2 , 3 , 4]
0 1 2 3

列表的其他使用方法

名称	函数说明
List.append(obj)	在列表末尾添加新的对象
List.count(obj)	统计某个元素在列表中出现的次数
List.extend(seq)	在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
List.index(obj)	从列表中找出某个值第一个匹配项的索引位置
List.insert(index, obj)	将对象插入列表
List.pop(obj=List[-1])	移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
List.remove(obj)	移除列表中某个值的第一个匹配项

元组

元组可看作是一种**不可变的列表**，由一对圆括号' ()' 或者tuple()函数创建，' ,' 分割元组里面的每个元素。

元组特点：

元组大小不可更改，既不能增加也不能删除对象。

元组中的对象不可更改

元组的基本操作

创建元组、求长度、合并、重复、关系判断、索引和切片等

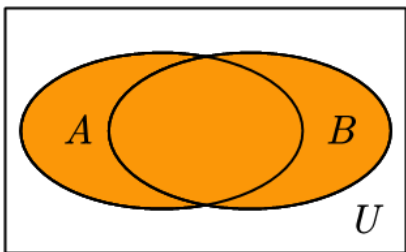
函数名称	函数说明
tuple.count()	记录某个元素在元组中出现的次数。
tuple.index()	获取元素在元组当中第一次出现的位置索引。
sorted()	创建一个对元素进行排序后的列表。
len()	获取元组长度，即元组元素个数。
+	将两个元组合并为一个元组。
*	重复合并同一个元组为一个更长的元组。

集合

集合中的元素具有唯一、无序和不可改变等特点。集合由一对花括号' {} ' 或者set()函数创建，以' ,' 分割集合中的元素。集合支持数学理论中的各种集合运算

函数名称	函数说明
Set.add(x)	往集合插入元素x
Set1.update(Set2)	把集合Set2的元素添加到Set1
Set.remove(x)	删除集合中的元素x
Set.discard(x)	删除指定元素，但是如果集合中没有的话就什么也不做
Set.pop()	随机删除一个，并返回该值
Set1.union(Set2)	Set1和Set2的并集 （元素不重复）
Set1.intersection(Set2)	Set1和Set2的交集
Set1.difference(Set2)	Set1和Set2的差，两个集合都不属于
Set1.issuperSet(Set2)	判断Set1是否是Set2的超集
Set1.symmetric_difference(Set2)	Set1和Set2的对称补集

集合运算操作的简便写法

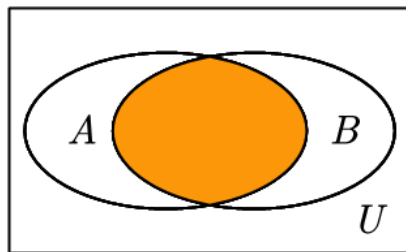


$A \cup B$

' | ' 获取并集

`A.union(B)`

`union`函数取并

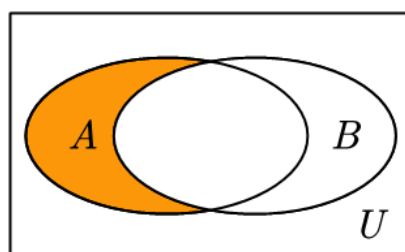


$A \cap B$

' & ' 获取交集

`A.intersection(B)`

`intersection`函数取交

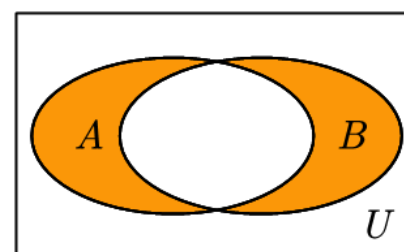


$A - B$

' - ' 来获取差集

`A.difference(B)`

`difference`函数取差



$A \oplus B$

' ^ ' 获取异或集

`A.symmetric_difference(B)`

`symmetric_difference`
函数取异或

字典

字典是一种无序的映射集合，包含一系列的键值对。字典常用 ‘{}’ 表示，也可以通过函数dict () 创建，字典内的值是通过 ‘:’ 来表示，也就是key:value的格式。**key**可以是任何不可变类型，比如整型、浮点型、字符串、元组，并且所有key都是唯一的。**value不限类型**

{*neme:Tony, job:Students*}

操作	实现方法	操作	实现方法
增	Dict.update({'AddKey':'AddValue'})	合	利用增加函数
删	Dict.pop(OldKey)	判 (键)	key in Dict
改	Dict[OldKey]=NewValue	看	Dict.keys()#查看所有键
看	Dict.values()#查看所有值	看	Dict.items()#查看键值对

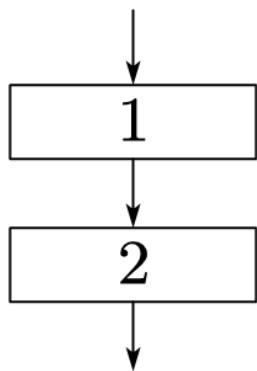
目录

1	Python语法元素
2	Python数据类型
3	Python程序流控制
4	函数、类与参数
5	文件与数据组织

Python常见程序结构

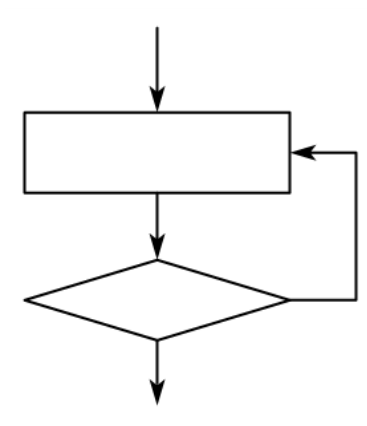
顺序语句

```
*test.py - C:/Users/右雪/Desktop/test.py (3.... - □ ×
File Edit Format Run Options Window Help
Variable=3
Variable1=2
Variable2=Variable1+Variable
print(Variable2)
Ln: 10 Col: 0
```



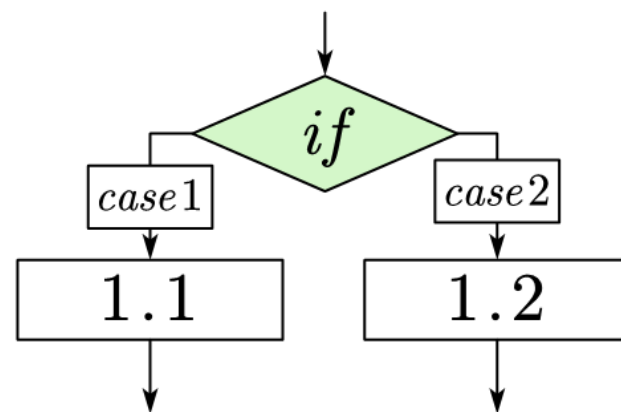
循环语句

```
test.py - C:/Users/右雪/Desktop/test.py (3.8... - □ ×
File Edit Format Run Options Window Help
Object=[1,2,3,4,5,6]
for var in Object:
    print(var)
Ln: 6 Col: 0
```



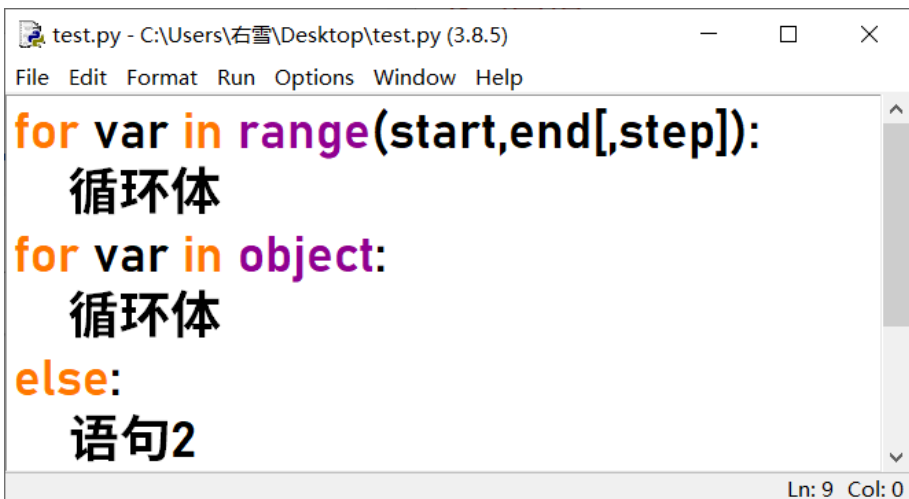
分支语句

```
test.py - C:/Users/右雪/Desktop/test.py (3.8... - □ ×
File Edit Format Run Options Window Help
Variable=3
if Variable>2:
    print('大于')
Ln: 5 Col: 0
```



Python中的循环

for循环

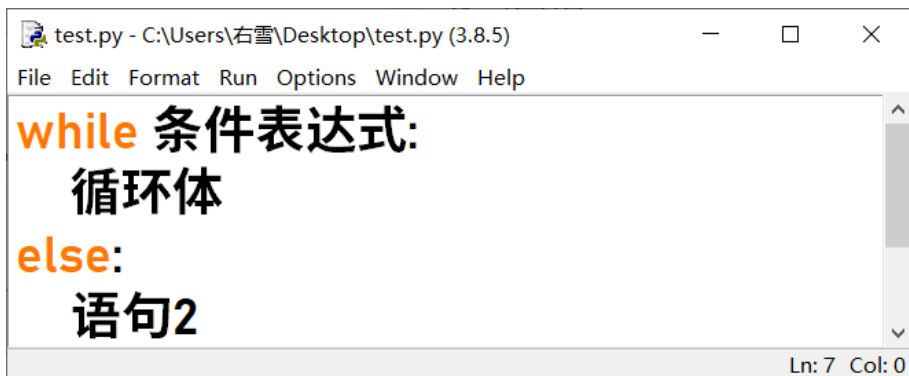


The screenshot shows a Python IDE window titled "test.py - C:\Users\右雪\Desktop\test.py (3.8.5)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following syntax for for loops:

```
for var in range(start,end[,step]):  
    循环体  
for var in object:  
    循环体  
else:  
    语句2
```

The status bar at the bottom right indicates "Ln: 9 Col: 0".

While循环



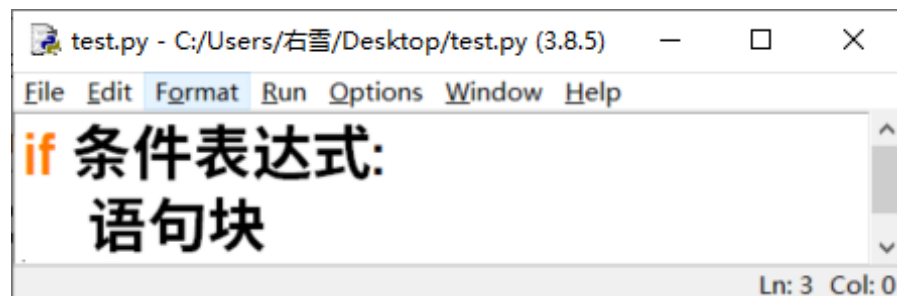
The screenshot shows a Python IDE window titled "test.py - C:\Users\右雪\Desktop\test.py (3.8.5)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following syntax for while loops:

```
while 条件表达式:  
    循环体  
else:  
    语句2
```

The status bar at the bottom right indicates "Ln: 7 Col: 0".

Python中的分支结构

单分支

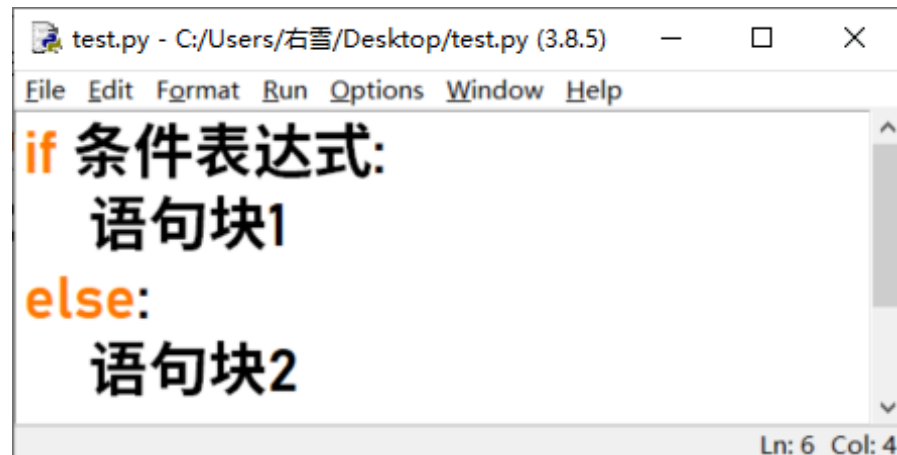


The screenshot shows a window titled "test.py - C:/Users/右雪/Desktop/test.py (3.8.5)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code content is:

```
if 条件表达式:  
    语句块
```

The status bar at the bottom indicates "Ln: 3 Col: 0".

双分支

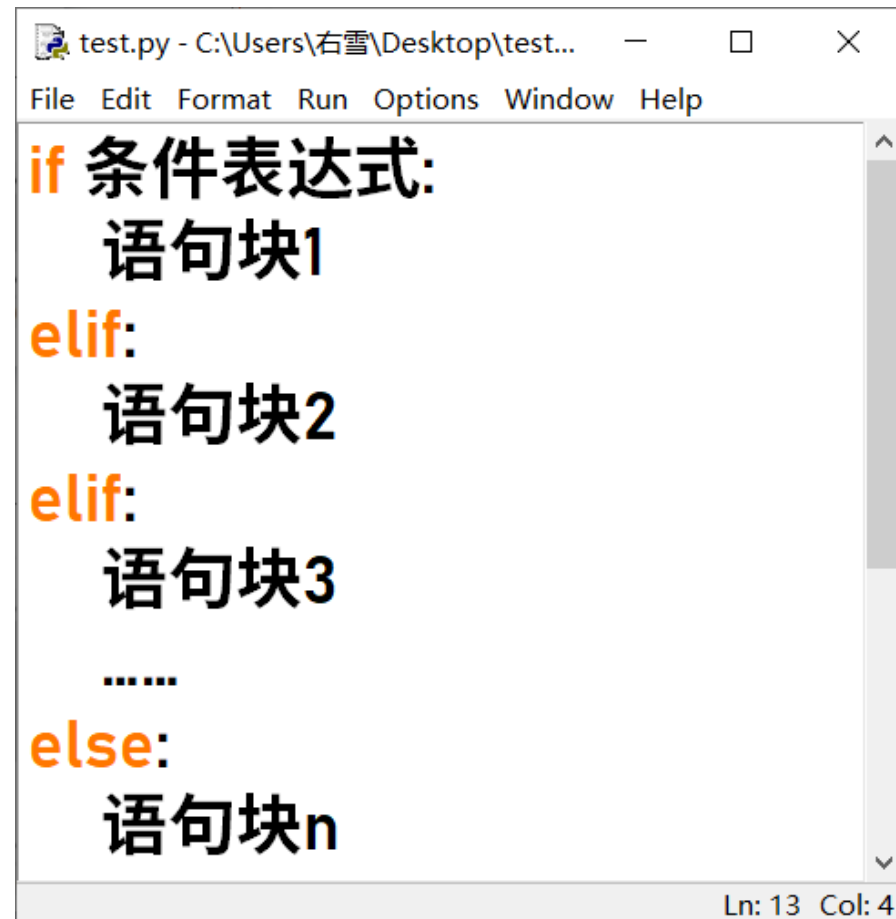


The screenshot shows a window titled "test.py - C:/Users/右雪/Desktop/test.py (3.8.5)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code content is:

```
if 条件表达式:  
    语句块1  
else:  
    语句块2
```

The status bar at the bottom indicates "Ln: 6 Col: 4".

多分支



The screenshot shows a window titled "test.py - C:\Users\右雪\Desktop\test...". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code content is:

```
if 条件表达式:  
    语句块1  
elif:  
    语句块2  
elif:  
    语句块3  
.....  
else:  
    语句块n
```

The status bar at the bottom indicates "Ln: 13 Col: 4".

列表推导式

列表推导式

列表推导式可以内部嵌套**多个列表推导式、判断语句、循环语句**。

```
List=[]
```

```
for var in object:
```

```
    List.append( 计算结果 )
```

```
List =[ 计算结果 for var in object]
```

```
List=[]
```

```
for var in object:
```

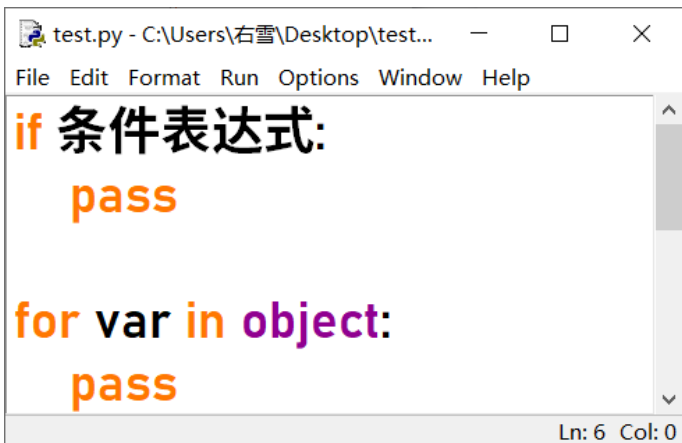
```
    if 判断语句 :
```

```
        List.append( 计算结果 )
```

```
List =[ 计算结果 for var in object if 判断语句 ]
```

pass、break和continue

pass



The screenshot shows a Python IDE window titled 'test.py - C:\Users\右雪\Desktop\test...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains two code blocks. The first block is an 'if' statement with the condition '条件表达式:' followed by an indented 'pass' statement. The second block is a 'for' loop with 'for var in object:' followed by an indented 'pass' statement. The status bar at the bottom right indicates 'Ln: 6 Col: 0'.

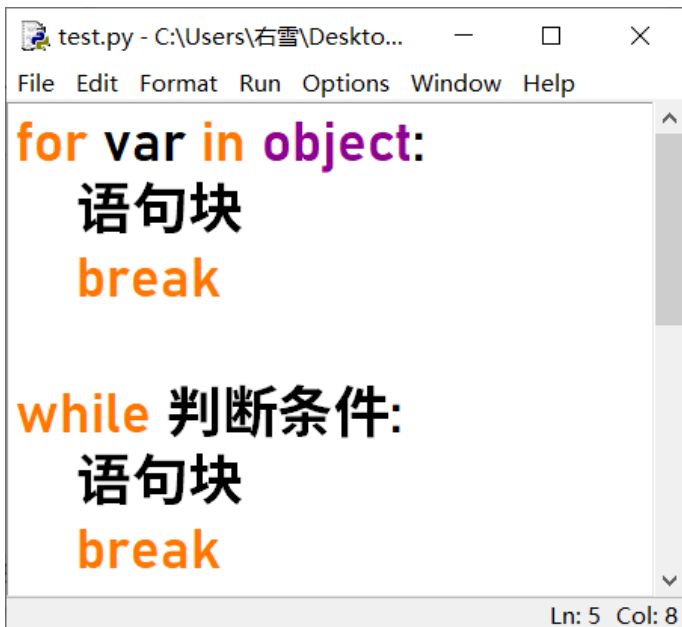
```
test.py - C:\Users\右雪\Desktop\test...
File Edit Format Run Options Window Help

if 条件表达式:
    pass

for var in object:
    pass

Ln: 6 Col: 0
```

break



The screenshot shows a Python IDE window titled 'test.py - C:\Users\右雪\Desktop...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains two code blocks. The first block is a 'for' loop with 'for var in object:' followed by an indented block of code and a 'break' statement. The second block is a 'while' loop with the condition '判断条件:' followed by an indented block of code and a 'break' statement. The status bar at the bottom right indicates 'Ln: 5 Col: 8'.

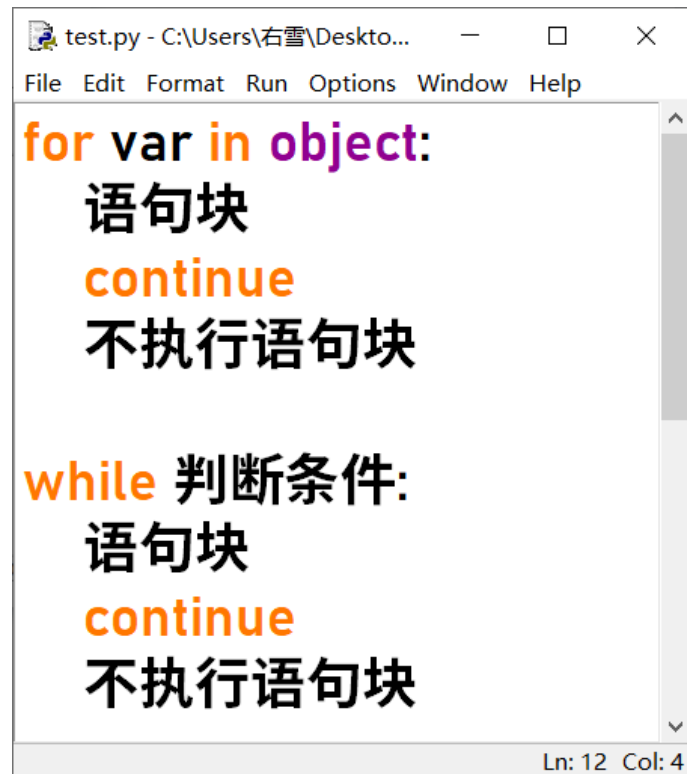
```
test.py - C:\Users\右雪\Desktop...
File Edit Format Run Options Window Help

for var in object:
    语句块
    break

while 判断条件:
    语句块
    break

Ln: 5 Col: 8
```

continue



The screenshot shows a Python IDE window titled 'test.py - C:\Users\右雪\Desktop...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains two code blocks. The first block is a 'for' loop with 'for var in object:' followed by an indented block of code and a 'continue' statement. The second block is a 'while' loop with the condition '判断条件:' followed by an indented block of code and a 'continue' statement. The status bar at the bottom right indicates 'Ln: 12 Col: 4'.

```
test.py - C:\Users\右雪\Desktop...
File Edit Format Run Options Window Help

for var in object:
    语句块
    continue
    不执行语句块

while 判断条件:
    语句块
    continue
    不执行语句块

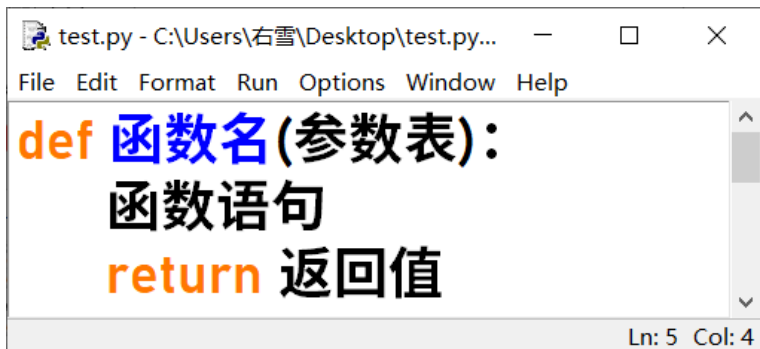
Ln: 12 Col: 4
```

目录

1	Python语法元素
2	Python数据类型
3	Python程序流控制
4	函数、类与模块
5	文件与数据组织

函数

def函数

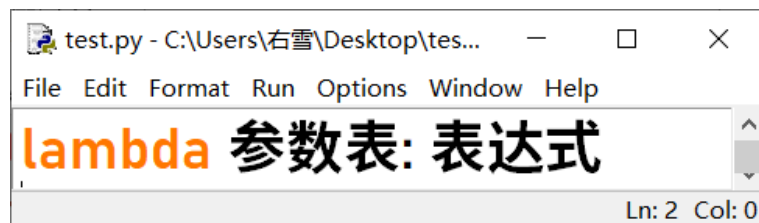


The screenshot shows a window titled "test.py - C:\Users\右雪\Desktop\test.py...". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following text:

```
def 函数名(参数表):  
    函数语句  
    return 返回值
```

The status bar at the bottom right indicates "Ln: 5 Col: 4".

lambda函数



The screenshot shows a window titled "test.py - C:\Users\右雪\Desktop\tes...". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following text:

```
lambda 参数表: 表达式
```

The status bar at the bottom right indicates "Ln: 2 Col: 0".

类

类：创建对象的基础，描述了所创建对象共有的属性和方法，可使用class语句定义。

属性：对象的公共属性可以提前绑定self，并使用_init_初始化。self必不可少且必须位于其他参数前面。

方法：类中的函数称为方法，适用于一切函数的知识。

实例化：把用类创建对象的过程称为实例化。是将一个抽象的概念类，具体到该类实物的过程

注意：

init前后必须是双下划线



The screenshot shows a Python IDE window titled 'test.py - C:\Users\右雪\Desktop\test.py (3.8.5)'. The code defines a class and two methods. Red dashed boxes and lines connect the text blocks on the left to specific parts of the code:

- A box around the `class` keyword and the class name (类名) connects to the definition of a class.
- A box around the `def __init__` method signature and its body connects to the explanation of the `__init__` method.
- A box around the `def` keyword and the first method name (类方法1) connects to the general definition of a class method.
- A box around the `def` keyword and the second method name (类方法2) connects to the general definition of a class method.
- A box around the instantiation examples connects to the general definition of instantiation.

```
class 类名:
    def __init__(self, 属性1, ..., 属性n):
        self.属性1=属性值1
        ...
        self.属性n=属性值n

    def 类方法1(self):
        执行语句
        return 返回值

    def 类方法2(self, 方法的形参):
        执行语句
        return 返回值

实例=类名(参数1, ..., 参数n)
实例.参数
实例.f1(方法的形参)
```

Ln: 18 Col: 0

模块

模块

- 内置模块
- 第三方模块
- 自定义模块

使用方法

import 模块名称

import 模块名称 as 新名称

from 模块名称 import 导入对象的名称

from 模块名称 import 导入对象的名称 as 新名称

from 模块名称 import *

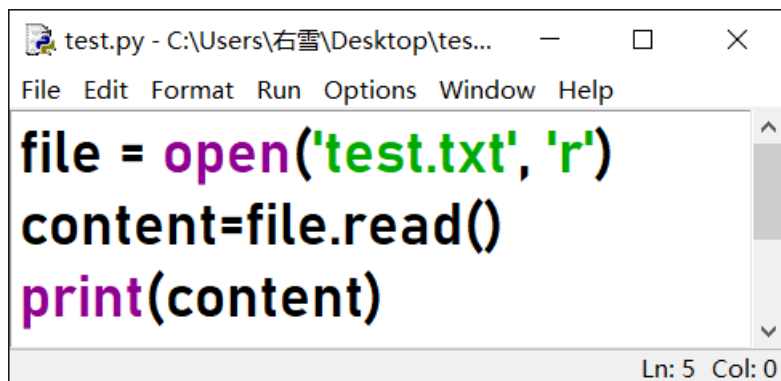
目录

1	Python语法元素
2	Python数据类型
3	Python程序流控制
4	函数、类与参数
5	文件与数据组织

文件读写

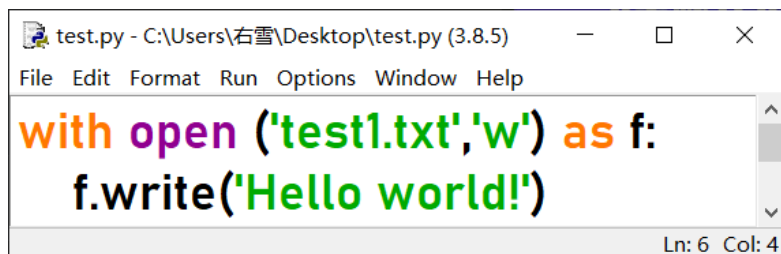
在python, 使用open函数, 可以打开一个已经存在的文件, 或者创建一个新文件:

open(文件名, 访问模式)



```
test.py - C:\Users\右雪\Desktop\tes...  
File Edit Format Run Options Window Help  
file = open('test.txt', 'r')  
content=file.read()  
print(content)  
Ln: 5 Col: 0
```

使用write()可以完成向文件写入数据



```
test.py - C:\Users\右雪\Desktop\test.py (3.8.5)  
File Edit Format Run Options Window Help  
with open ('test1.txt','w') as f:  
    f.write('Hello world!')  
Ln: 6 Col: 4
```



打开文件

访问模式	说明
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
w	打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
r+	可读、可写，文件不存在也会报错，写操作时会覆盖
w+	可读，可写，文件不存在先创建，会覆盖
a+	可读、可写，文件不存在先创建，不会覆盖，追加在末尾



Thank you!