计算机程序设计

刘龙军 副教授

2025年10月27日

目录

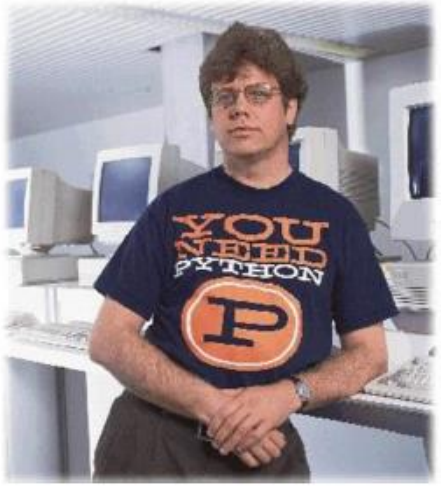
Python 对比学
复习 C/C++ 体系
从两者对比程序设计本质

历史

- Python [ˈpaiθən]，译为“蟒蛇”
- Python语言拥有者是Python Software Foundation(PSF)
- PSF是非盈利组织，致力于保护Python语言开放、开源和发展
- 荷兰人Guido van Rossum（吉多·范·罗苏姆）于1989年发明
- 第一个公开发行版发行于1991年
- 2002年，Python 2.x
- 2008年，Python 3.x
- 2020年，Python 2不再维护

Life is short, you need Python!

人生苦短，我用Python。



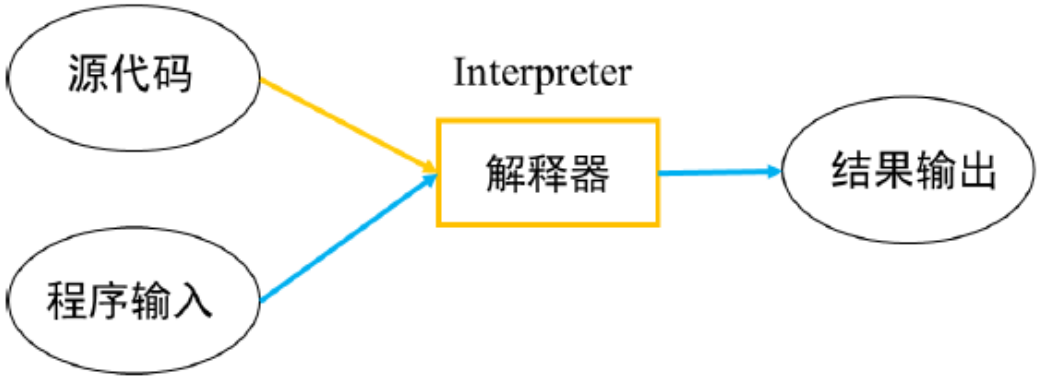
Guido van Rossum

Bruce Eckel

基本特点

Python体系

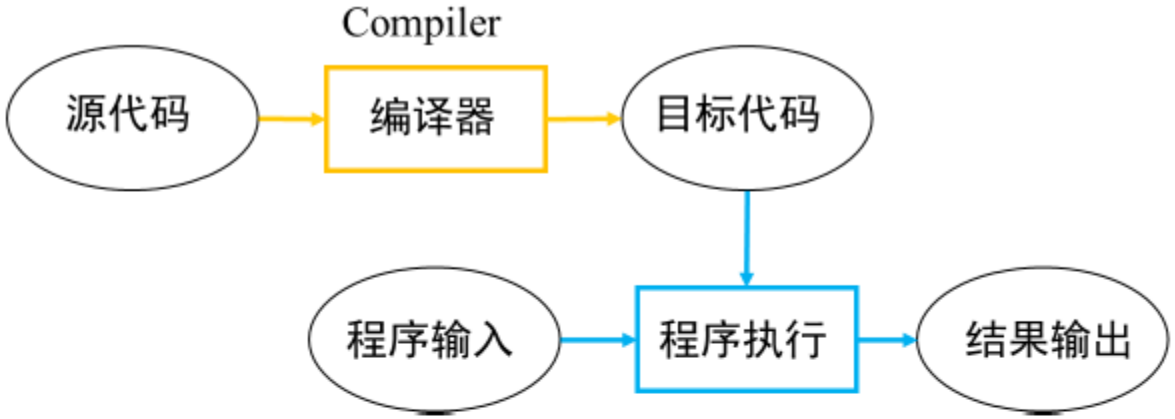
解释性语言



- 执行解释过程的程序叫作解释器
- 将源代码逐条转换成目标代码同时逐条运行的过程
- 每次程序运行时随翻译随执行（类似实时的同声传译）
- 执行程序时需要源代码，维护更灵活
- 更容易跨多个操作系统平台

C/C++体系

编译性语言



- 执行编译过程的程序叫作编译器
- 将源代码一次性转换成目标代码的过程
- 一次性翻译，之后不再需要源代码（类似英文翻译）
- 编译器一次性生成目标代码，优化更充分
- 程序运行速度更快

Hello word

Python体系

```
>>> print( “hello, world” )
hello, world
```

```
C:\work> python hello.py
hello, world
```

1、语法简洁
2、与平台无关
3、开源理念
4、通用灵活
5、类库丰富

1、运行速度慢
2、加密难
3、缩进规则

C/C++体系

```
#include <iostream>

int main(int argc, const char * argv[]) {
    using namespace std;
    cout << "Hello, XJTU!";
    cout << endl;
    cout << "Hello, C++";
    return 0;
}
```

基本语法

Python体系 | C/C++体系

- 用print()在括号中加上字符串，就可以向屏幕上输出指定的文字。比如输出hello, world
- print()函数也可以接受多个字符串，用逗号,隔开，就可以连成一串输出：

```
>>> print('The quick brown fox', 'jumps over', 'the lazy dog')
The quick brown fox jumps over the lazy dog
```

- 遇到逗号,会输出一个空格

```
>>> print('100 + 200 =', 100 + 200)
100 + 200 = 300
```

```
name = input()
print('hello,', name)
```

```
name = input('please enter your name: ')
print('hello,', name)
```

- 对比输出

```
cout << "Hello, XJTU!";
```

```
carrots = 25;
cout << "I have ";
cout << carrots;
cout << " carrots.";
cout << endl;
```

基本语法—程序例子

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

例子

摄氏度和**华氏度**是温度刻画的两种不同体系。中国等世界大多数国家使用摄氏度，以1标准大气压下水的结冰点为0度，沸点为100度，将温度进行等分刻画。

美国、英国等国家使用华氏度，以1标准大气压下水的结冰点为32度，沸点为212度，将温度进行等分刻画。

写一个程序，实现两种温度体系的转换。

输入：带华氏或摄氏标志的温度值

算法：根据温度标志选择适当的温度转换算法

输出：带摄氏或华氏标志的温度值

对比找“不同”与“相同”

基本语法

Python体系 | C/C++体系

观其大略

TempConvert.py

```python
# TempConvert.py

TempStr = input("请输入带有符号的温度值: ")

if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

对比找“不同”与“相同”

```c
#include <stdio.h>
/* 对fahr = 0, 20, ..., 300
 打印华氏温度与摄氏温度对照表 */
int main()
{
        int  fahr, celsius;             //(Fahrenheit华
        int  lower, upper, step;

        lower = 0;      // 温度表的下限
        upper = 300;    // 温度表的上限
        step  = 20;     // 步长
        fahr = lower;

        while (fahr <= upper) {
                celsius = 5 * (fahr-32) / 9;
                printf("%d\t%d\n", fahr, celsius);
                fahr = fahr + step;
        }
        return 0;
}
```

基本语法

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

■ 观其大略

缩进：一行代码开始前的空白区域，表达程序的格式框架

- 严格明确：缩进是语法的一部分，缩进不正确程序运行错误
- 所属关系：表达代码间包含和层次关系的唯一手段
- 长度一致：程序内一致即可，一般用4个空格或1个TAB

```
#TempConvert.py
TempStr = input("请输入带有符号的温度值:")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

缩进的坏处就是“复制 - 粘贴”功能失效了

■ 有区别 不要求

基本语法

Python体系 | C/C++体系

注释

- **单行注释：**
- 以#开头，其后内容为注释

```
# 这里是单行注释
```

- **多行注释：**
- 以'''开头和结尾

```
''' 这是多行注释的第一行
    这是多行注释的第二行'''
```

对比找“不同”与“相同”

- //
- /* */

基本语法

Python体系 | C/C++体系

■ 变量

变量： 程序中用于保存和表示数据的占位符号

- 变量采用标识符(名字) 来表示，关联标识符的过程叫命名。

  TempStr是变量名字

- 可以使用等号(=)向变量赋值或修改值，=被称为赋值符号。

  TempStr = "82F" #向变量TempStr赋值"82F "

- 在使用变量前，需要对其赋值，没有值的变量是没有意义的。
- 同一个变量可以反复赋值，而且可以是不同类型的变量。
- 可以用type()确认变量或者数据的类型。

■ 对比找“不同”与“相同”

■ 定义、初始化、赋值？

基本语法

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- **命名：** 关联标识符的过程

命名规则:

大小写字母、数字、下划线和汉字等字符及组合

如: TempStr, Python_Great, 这是Python课

注意事项:

- 大小写敏感
- 首字符不能是数字
- 不与保留字相同

Python和python是不同变量

123Python是不合法的

- **对比找“不同”与“相同”**

- **相同！**

基本语法

Python体系 | C/C++体系

保留字

- Python语言有30+个保留字(也叫关键字)
- Python 3.9 有36个关键字

  如: if, else, in, class, False, True…
- 保留字是编程语言的基本单词，大小写敏感
- if 是保留字，If 是变量
- 可以用下面的语句查看：

```
import keyword
print(keyword.kwlist)
```

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	or	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

注意，Python区分标识符的大小写，保留字和关键字必须严格区分大小写。

对比找“不同”与“相同”

相同（64）

```
1   auto        break     case      char      const     continue
2   default     do        double    else      enum      extern
3   float       for       goto      if        inline    int
4   long        register  return    short     signed    sizeof
5   static      struct    switch    typedef   union     unsigned
6   void        volatile  while     bool      catch     class
7   const_cast  delete    dynamic_cast explicit friend  goto
8   mutable     namespace new       operator  private   protected
9   public      reinterpret_cast static_cast  template  this      throw
10  try         typeid    typename  using     virtual   wchar_t
```

基本语法

Python体系

C/C++体系

数据类型

- 一、数字型
  - (一)、整型
  - (二)、浮点型
  - (三)、布尔型
  - (四)、复数类型
- 二、字符串
- 三、列表
- 四、元组
- 五、集合
- 六、字典

对比找“不同”与“相同”

二、基本数据类型

1.整型

2.浮点型

3.字符型

4.布尔型

三、类型修饰符

四、派生数据类型

1.数组

2.指针

3.引用

五、标准库类型

六、用户自定义类型

1.结构体（`struct`）

2.枚举（`enum`）

3.类（`class`）

七、空类型（`void`）

基本语法

Python体系 | C/C++体系

输入、输出（简洁）

- input()函数的使用格式：

  <变量> = input(<提示信息字符串>)

- 用户输入的信息以字符串类型保存在<变量>中

  TempStr = input("请输入")

  # TempStr保存用户输入的信息

- print()函数的基本使用格式：

  print(<拟输出字符串或字符串变量>)

- 字符串类型的一对引号仅在程序内部使用，输出无引号

  print("输入格式错误")

  # 向控制台输出 输入格式错误

对比找“不同”与“相同”

- cin　　cout
- scanf　　printf

基本语法

Python体系 | C/C++体系

- 对比找“不同”与“相同”

- 区别

```
print([obj1,…][,sep=' '][,end='\n'][,file=sys.stdout])
>>> print(123)
123
>>> print(123, 'abc',45, 'book')
123 abc 45 book
>>> print(123,'abc',45,'book',sep='#')
123#abc#45#book
>>> print('price');print(100)
price
100
>>> print('price',end='_');print(100)
price_100
```

```
print("转换后的温度是{:.2f}C".format(C))
```

{}表示槽，后续变量填充到槽中

{ :.2f }表示将变量C填充到这个位置时取小数点后2位

基本语法

Python体系

C/C++体系

输出到文件。默认输出到标准输出流，可用file参数指定将数据输出到文件。

```
>>> file1=open(r'd:\data.txt','w')          #打开文件
>>> print(123,'abc',45,'book',file=file1)   #用file参数指定输出文件
>>> file1.close()                           #关闭文件
```

- 对比找“不同”与“相同”

- 不支持，单独指针

基本语法

Python体系 | C/C++体系

赋值

- 赋值语句用来给变量赋予新的数据值

  C=(eval(TempStr[0:-1])-32)/1.8 #右侧运算结果赋给变量C

- 赋值语句右侧的数据类型同时作用于变量

  TempStr=input("") #input()返回一个字符串，TempStr也是字符串

去掉参数最外侧引号并执行余下语句的函数

对比找“不同”与“相同”

相同 有区别

基本语法

Python体系

- **序列赋值**。一次性为多个变量赋值，等号左侧是元组或列表表示的多个变量，等号右侧是元组、列表或字符串等序列表示的数据，按先后顺序依次将数据赋值给变量。

```
>>> x,y=1,2                          # 直接为多个变量赋值
>>> (x,y)=10,20                      #为元组中的变量赋值
>>> [x,y]=30, 'abc'                  #为列表中的变量赋值
```

- 等号右侧为字符串时，Python会将字符串分解为单个字符，依次赋值给各个变量。此时，变量个数和字符个数必须相等

```
>>> (x,y)='ab'                       #用字符串为元组中的变量赋值
>>>((x,y),z)='ab','cd'              #用嵌套的元组为变量赋值
>>> (x,y)='abc'                      #字符个数与变量个数不一致，出错
```

-序列赋值时，可以在变量名之前使用“*”，不带星号的变量仅匹配一个值，剩余的值作为列表赋值给带星号的变量。

```
>>> x,*y='abcd'
>>> x
'a'
>>> y
['b', 'c', 'd']
>>> x,*y=[1,2,'abc','汉字']
>>> x
1
>>> y
[2, 'abc', '汉字']
```

C/C++体系

- **对比找“不同”与“相同”**

- **不支持**

基本语法

Python体系

- **多目标赋值**。用连续的多个等号将同一个数据赋值给多个变量。

```
>>> a=b=c=10                          #将10赋值给变量a、b、c
```

- **增强赋值**。将运算符与赋值相结合的赋值语句。

```
>>> a=5
>>> a+=10                             #增强赋值，等价于a = a + 10
>>> a
15
```

+=	-=	*=	**=
//=	&=	|=	^=
>>=	<<=	/=	%=

C/C++体系

- **对比找“不同”与“相同”**

- **有区别**

基本语法

Python体系 | C/C++体系

语句续行 分格符号

- 通常Python中的一条语句占一行，没有语句结束符号。可使用语句续行符号”\”将一条语句写在多行之中。注意，在“\”符号之后不能有任何其他符号，包括空格和注释。

```
if x < 100 \
   and x>10:
   y = x *5 – 1
```

- 还有一种特殊的续行方式：在使用括号（包括 “()” “[]”和 “{}”等）时，括号中的内容可分多行书写，括号中的注释、空白和换行符都会被忽略。

```
if (x < 100                          #这是续行语句中的注释
   and x>10):
   y = x *5 - 1
else:
   y = 0
```

- Python使用分号作为语句分隔符号，从而将多条语句写在一行。

```
print(100) ; print(2+3)
```

- 使用语句分隔符号分隔的多条语句可视为一条复合语句，Python允许将单独的语句或复合语句写在冒号之后。

```
if x < 100 and x>10 : y = x *5 - 1
else: y = 0;print('x >= 100 或x<=10')
```

对比找“不同”与“相同”

不一样，直接回车或空格

基本语法

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

■ 观其大略　分支判断 函数

TempConvert.py

```python
# TempConvert.py

TempStr = input("请输入带有符号的温度值: ")

if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

■ 对比找“不同”与“相同”

```c
#include <stdio.h>
/* 对fahr = 0, 20, ..., 300
 打印华氏温度与摄氏温度对照表 */
int main()
{
        int  fahr, celsius;             //(Fahrenheit华
        int  lower, upper, step;

        lower = 0;      // 温度表的下限
        upper = 300;    // 温度表的上限
        step  = 20;     // 步长
        fahr = lower;

        while (fahr <= upper) {
                celsius = 5 * (fahr-32) / 9;
                printf("%d\t%d\n", fahr, celsius);
                fahr = fahr + step;
        }
        return 0;
}
```

基本类型

Python体系

C/C++体系

- **int**

例如交互模式下输入如下，使用的就是整型：

```
>>> 51
51
```

整型加法如下：

```
>>> 25+25
50
```

整型减法：

```
>>> 51-50
1
```

整型乘法：

```
>>> 51*2
102
```

整型除法：

```
>>> 153/51
3.0
>>> 155/51
3.0392156862745097
```

分数部分，可以使用地板除（//），整数的地板除（//）永远是整数，即使除不尽。

```
>>> 153//51
3
>>> 155//51
3
```

- 地板除（//）只取结果的整数部分，Python还提供一个余数运算（%），可以得到两个整数相除的余数。

```
>>> 155%51
2
```

- **对比找“不同”与“相同”**

- **相同（除了/）**

基本类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 相同

- 一般的整数常量都是十进制的，也可将整数常量表示为二进制、八进制和十六进制。
- 二进制：以“0b”或“0B”开头，后面跟二进制数字（0或1）。

  0b101、0B11
- 八进制：以“0o”或“0O”开头，后面跟八进制数字（0~7）。

  0o15、0O123。
- 十六进制：以“0x”或“0X”开头，后面跟十六进制数字（0~9、A~F（或a~f））。

  0x12AB、0x12ab。

基本类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 相同

1.2 布尔型常量(bool)：只有True和False两个值。

- 将布尔型常量转换为整数时，True转换为1，False转换为0。
- 将布尔常量转换为字符串时，True转换为“True”，False转换为“False”。

基本类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 相同

1.3 浮点型(float)：浮点型由整数部分与小数部分组成，浮点型也可以使用科学计数法表示。

- 12.5、2.、.5、3.0、1.23e+10、1.23E-10等都是合法的浮点数常量。
- 浮点数存在取值范围，超过取值范围会产生溢出错误（OverflowError）。
- 浮点数取值范围为2.2250738585072014e-308~1.7976931348623157e + 308。
- 整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的，而浮点数运算则可能会有四舍五入的误差。使用sys.float_info可以查看浮点数参数。（扩展：demical模块，fractions模块）

```
>>> 3.3*102
336.59999999999997
>>> 3.3*102+15.5
352.09999999999997
```

```
>>> 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)
```

基本类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”

- 相同

浮点除法：

```
>>> 153/51.0
3.0
>>> 155/51.0
3.0392156862745097
```

浮点地板除：

```
>>> 155//51.0
3.0
>>> 155%51.0
2.0
```

基本类型

Python体系

复数

1.4 复数(complex)：复数由实数部分和虚数部分构成，可以用a + bj，或者complex(a,b) 表示， 复数的实部a和虚部b都是浮点型。如，2+3j、2-3J、2j

```
>>> complex(2,3)
(2+3j)
```

C/C++体系

对比找“不同”与“相同”

complex 类模板

```
#include<complex.h>


int main(int argc, char *argv[])
{
  complex  double  a = 3.0 + 4.0 * _Complex_I;
  __complex__ double b = 4.0 + 5.0 * _Complex_I;
  _Complex  double c = 5.0 + 6.0 * _Complex_I;

  printf("a=%f+%fi\n", creal(a),cimag(a));
  printf("b=%f+%fi\n", creal(b), cimag(b));
  printf("c=%f+%fi\n", creal(c), cimag(c));
```

基本类型

Python体系

- 数据类型转换

1.5 数据类型转换：

int(x)	-将x转换为一个整数。
float(x)	-将x转换到一个浮点数。
complex(x)	-将x转换到一个复数，实数部分为 x，虚数部分为 0。
complex(x, y)	-将 x 和 y 转换到一个复数，实数部分为 x，虚数部分为 y。

C/C++体系

- 对比找“不同”与“相同”

- **基本相同**

基本类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”

操作符	说明	举例
**	幂运算	2**3
~	按位取反	~5
-	负号	-5
*、%、/、//	乘法、求余数、真除法、floor除法	2*3、3%2、5/2、5//2
+、-	加法、减法	2+3、2-3
<<、>>	向左移位、向右移位	3<<2、12>>2
&	按位与	5&2
^	按位异或	5^2
|	按位或	5|2
<、<=、>、>=、==、!=	小于、小于等于、大于、大于等于、相等、不等	2<3、2<=3、2>3、2>=3、2==3、2!=3
not	逻辑非	not True、not 2<3
and	逻辑与	x>5 and x<100
or	逻辑或	x<5 or x>100

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

库

Math 库是Python 提供的内置数学类函数库，支持整数和浮点数运算，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。

引用方法:

1、import math

```
>>> import math
>>> math.ceil(4.01)
5
```

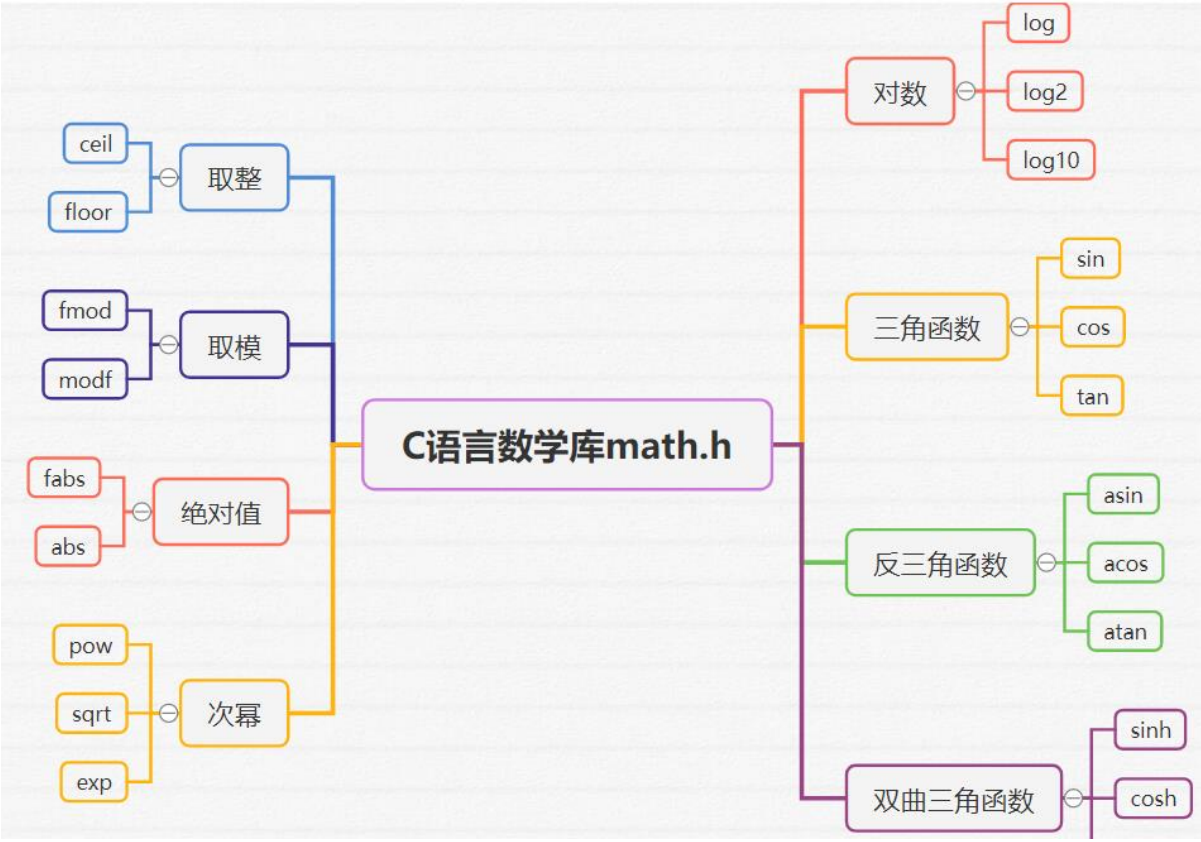
2、from math import <函数名>

```
>>> from math import fabs
>>> fabs(-3.55)
3.55
```

dir(math)查看有哪些函数，help(math)或help(math.ceil)查看帮助

对比找“不同”与“相同”

基本相同 #include<math.h>



基本类型

Python体系

2. 字符串

由0个或多个字符组成的有序字符序列

- 字符串由一对**单引号**或一对**双引号**表示

  "请输入带有符号的温度值: "或者 'C'

- 由一对**三单引号**或**三双引号**表示，可表示多行字符串

  ''' Python

  语言 '''

- 如果希望在字符串中包含双引号或单引号呢?

  '这里有个双引号(")' 或者 "这里有个单引号(')"

- 如果希望在字符串中既包括单引号又包括双引号呢

  ''' 这里既有单引号(')又有双引号 (") '''

C/C++体系

对比找“不同”与“相同”

- 在C语言中，字符串是由字符数组表示的通常以空字符 \0 结尾。

1、用双引号括起来的**字符串常量**，如"CHINA"，"C++ program"

2、存放于**字符数组**中，以"\0"字符（ASCII码为0）结尾。

3、string对象。string是C++标准模板库里的一个类，专门用于处理字符串

```
char str1[] = {'h', 'e', 'l', 'l', 'o'};//方式1：逐个字符初始化
```

```
char title[] = "Prison Break";
char hero[100] = "Michael Scofield";
char prisonName[100];
char response[100];
```

```
string s1 = "123", s2;
s2 += s1;
s1 = "abc";
s1 += "def";
```

基本类型

Python体系

C/C++体系

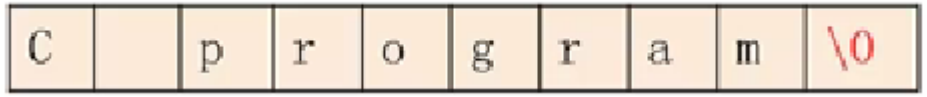
西安交通大学 XI'AN JIAOTONG UNIVERSITY

字符串的序号

正向递增序号 和 反向递减序号

- 对比找“不同”与“相同”

- 不相同



基本类型

Python体系

2.2 字符串操作符

in

- 字符串是字符的有序集合，可用in操作符判断字符串包含关系

```
>>> x='abcdef'
>>> 'a' in x
True
>>> 'cde' in x
True
>>> '12' in x
False
```

- 加法运算将多个字符串合并。

```
>>> '12'+'34'+'56'
'123456'
```

C/C++体系

- 对比找“不同”与“相同”

- **不相同**

基本类型

Python体系 | C/C++体系

如果想让字符串之间有空格，可以建一个空字符变量，插在相应的字符串之间让它们隔开，或是在字符串中加入相应的空格。交互模式下输入如下：

```
>>> string1='hello'
>>> string2='world'
>>> space=' '
>>> print(string1+space+string2)
hello world
```

或者

```
>>> string1='hello'
>>> string2=' world'
>>> print(string1+string2)
hello world
```

- 对比找“不同”与“相同”
- **基本相同**

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

星号

- 星号用于将字符串复制多次以构成新的字符串。

```
>>> '12' * 3
'121212'
```

逗号

- 在使用逗号分隔字符时，会创建用字符串组成的元组。

```
>>> x='abc','def'
>>> x
('abc', 'def')
>>> type(x)
<class 'tuple'>
```

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

- 索引指通过偏移量来定位字符串中的单个字符。

```
>>> x='abcdef'
>>> x[0]                    #索引第1个字符
'a'
>>> x[-1]                   #索引最后1个字符
'f'
>>> x[3]                    #索引第4个字符
'd'
```

C/C++体系

- 对比找“不同”与“相同”
- 有区别

基本类型

Python体系

索引可获得指定位置的单个字符，但不能通过索引来修改字符串，因为字符串对象不允许被修改。

```
>>> x='abcd'
>>> x[0]='1'  #试图修改字符串中的指定字符，出错
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    x[0]='1'
TypeError: 'str' object does not support item assignment
```

C/C++体系

- 对比找“不同”与“相同”

- 本相同

基本类型

Python体系

C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

2.4 字符串的切片

- 字符串的切片也称分片，它利用索引范围从字符串中获得连续的多个字符（即子字符串）。
- 字符串切片的基本格式如下。

  x[start : end]

- 返回字符串x中从偏移量start开始、到偏移量end之前的子字符串。
- start和end参数均可省略，start默认为0，end默认为字符串长度

- **对比找“不同”与“相同”**

- **不相同**

基本类型

Python体系

```
>>> x='abcdef'
>>> x[1:4]                #返回偏移量为1到3的字符
'bcd'
>>> x[1:]                 #返回偏移量为1到末尾的字符
'bcdef'
>>> x[:4]                 #返回从字符串开头到偏移量为3的字符
'abcd'
>>> x[:-1]                #除最后一个字符，其他字符全部返回
'abcde'
>>> x[:]                  #返回全部字符
'abcdef'
```

C/C++体系

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

默认情况下，切片是返回字符串中的多个连续字符，可以通过步长参数来跳过中间的字符，其基本格式如下。

x[start : end : step]

用这种格式切片时，会依次跳过中间step-1个字符，step默认为1。

```
>>> x='0123456789'
>>> x[1:7:2]              #返回偏移量为1、3、5的字符
'135'
>>> x[::2]                #返回偏移量为偶数的全部字符
'02468'
>>> x[7:1:-2]             #返回偏移量为7、5、3的字符
'753'
>>> x[::-1]               #将字符串反序返回
'9876543210'
```

C/C++体系

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

2.5 迭代字符串

- 字符串是有序的字符集合，可用for循环迭代处理字符串。

```
>>> for a in 'abc':          #变量a依次表示字符串中的每个字符
...     print(a)
...
a
b
c
```

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

求字符串长度 len

- 字符串长度指字符串中包含的字符个数，可用len()函数获得字符串长度。

```
>>> len('abcdef')
6
```

字符串转换 str

- 可用str()函数将非字符串数据转换为字符串。

```
>>> str(1.23)              #将浮点数转换为字符串
'1.23'
>>> str([1,2,3])           #将列表转换为字符串
'[1, 2, 3]'
>>> str(True)              #将布尔常量转换为字符串
'True'
```

C/C++体系

- 对比找“不同”与“相同”
- strlen(定义在<string.h〉头文件中)

基本类型

Python体系 | C/C++体系

- 字符串处理方法调用格式：字符串.方法( )

capitalize()

- 将字符串第一个字母大写，其余字母小写，返回新的字符串。

```
>>> 'this is Python'.capitalize()
'This is python'
```

count(sub[, start[, end]])

- 返回字符串sub在当前字符串的[start, end]范围内出现的次数，省略范围时会查找整个字符串。

```
>>> 'abcabcabc'.count('ab')          #在字符中统计ab出现的次数
3
>>> 'abcabcabc'.count('ab',2)        #从第3个字符开始统计
2
```

- 对比找“不同”与“相同”
- 之前ppt

基本类型

Python体系

C/C++体系

find(sub[, start[, end]])

- 在当前字符串的[start, end]范围内查找子字符串sub，返回sub第一次出现的位置，没有找到时返回-1。

```
>>> x='abcdabcd'
>>> x.find('ab')
0
>>> x.find('ab',2)
4
>>> x.find('ba')
-1
```

对比找“不同”与“相同”

`size_type find(const string& str, size_type pos = 0) const`：查找子串的位置。

`size_type find(const char* s, size_type pos = 0) const`：查找C风格字符串的位置。

`size_type find(char c, size_type pos = 0) const`：查找字符的位置。

基本类型

Python体系

isalnum()

- 当字符串不为空且不包含任何非数字或字母（包括各国文字）的字符时返回True，否则返回False。

```
>>> '123'.isalnum()
True
>>> '123a'.isalnum()
True
>>> '123#asd'.isalnum()                    #包含非数字或字母的字符
False
>>> ''.isalnum()                           #空字符串，返回False
False
>>> '中国'.isalnum()
True
```

C/C++体系

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

lower()

- 将字符串中的字母全部转换成小写。

```
>>> 'This is ABC'.lower()
'this is abc'
```

upper()

- 将字符串中的字母全部转换成大写。

```
>>> 'This is ABC'.upper()
'THIS IS ABC'
```

C/C++体系

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系

replace(old, new[, count])

- 将当前字符串包含的old字符串替换为new字符串，省略count时会替换全部old字符串。指定count时，最多替换count次。

```
>>> x='ab12'*4
>>> x
'ab12ab12ab12ab12'
>>> x.replace('12','000')                    #全部替换
'ab000ab000ab000ab000'
>>> x.replace('12','00',2)                   #替换2次
'ab00ab00ab12ab12'
```

C/C++体系

■ 对比找“不同”与“相同”

`string& replace(size_type pos, size_type count, const char* s)`：替换子串。

`string& replace(size_type pos, size_type count, const string& str)`：替换子串。

`string& replace(size_type pos, size_type count, size_type count2, char c)`：替换子串为多个字符。

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 对比找“不同”与“相同”

方法	说明
str.casefold()	返回原字符串消除大小写的副本。
str.center(width[, fillchar])	返回长度为 width 的字符串，原字符串在其正中
str.endswith(suffix[, start[, end]])	如果字符串以指定的 suffix 结束返回 True，否则返回 False。
str.isdigit()	如果字符串中的所有字符都是数字，并且至少有一个字符，返回 True ，否则返回 False 。
str.partition(sep)	在 sep 首次出现的位置拆分字符串，返回一个 3 元组，其中包含分隔符之前的部分、分隔符本身，以及分隔符之后的部分。
str.startswith(prefix[, start[, end]])	如果字符串以指定的 prefix 开始则返回 True，否则返回 False。
str.split(sep=None, maxsplit=- 1)	返回一个由字符串内单词组成的列表，使用 sep 作为分隔字符串。
str.zfill(width)	返回原字符串的副本，在左边填充 ASCII '0' 数码使其长度变为 width。

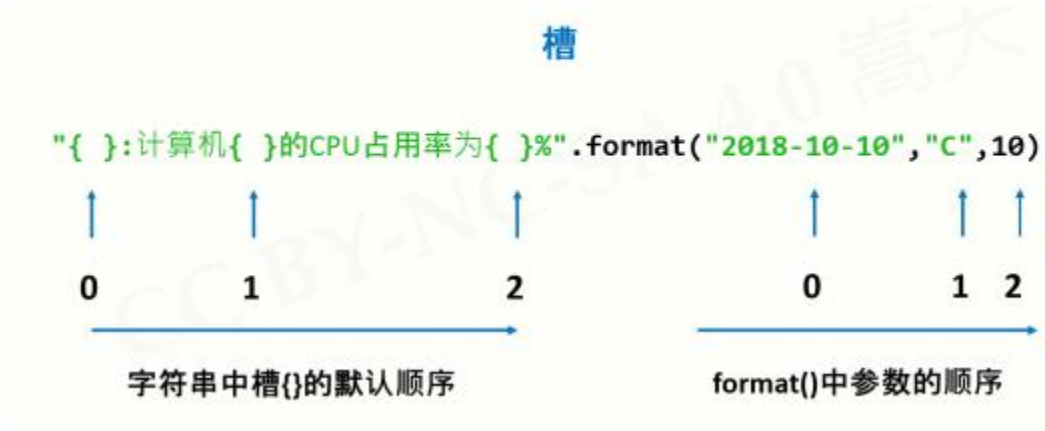
- 不相同

基本类型

Python体系

- 字符串格式化使用.format()方法，用法如下：

<模板字符串>.format(<逗号分隔的参数>)



字符串格式化使用.format()方法，用法如下：

<模板字符串>.format(<逗号分隔的参数>)

C/C++体系

- 对比找“不同”与“相同”

- 不相同

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

format()方法的格式控制

槽内部对格式化的配置方式

{ <参数序号> : <格式控制标记>}

:	<填充>	<对齐>	<宽度>	< , >	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽设定的输出宽度	数字的千位分隔符	浮点数小数精度 或 字符串最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

```
>>>"{0:=^20}".format("PYTHON")
'=======PYTHON======='
>>>"{0:*>20}".format("BIT")
'*****************BIT'
>>>"{:10}".format("BIT")
'BIT       '
```

: 引导符号

```
>>>"{0:,.2f}".format(12345.6789)
'12,345.68'
>>>"{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425)
'110101001,Σ,425,651,1a9,1A9'
>>>"{0:e},{0:E},{0:f},{0:%}".format(3.14)
'3.140000e+00,3.140000E+00,3.140000,314.000000%'
```

对比找“不同”与“相同”

- ios::dec 以10进制表示整数
- ios::hex 以16进制表示整数
- ios::oct 以8进制表示整数
- ios::showbase 为整数添加一个表示其进制的前缀
- ios::internal 在符号位和数值的中间插入需要数量的填充字符以使串两端对齐
- ios::left 在串的末尾插入填充字符以使串居左对齐
- ios::right 在串的前面插入填充字符以使串居右对齐
- ios::boolalpha 将bool类型的值以true或flase表示，而不是1或0
- ios::fixed 将符点数按照普通定点格式处理（非科学计数法）
- ios::scientific 将符点数按照科学计数法处理（带指数域）
- ios::showpoint 在浮点数表示的小数中强制插入小数点（默认情况是浮点数表示的整数不显示小数点）
- ios::showpos 强制在正数前添加+号
- ios::skipws 忽略前导的空格（主要用于输入流，如cin）
- ios::unitbuf 在插入（每次输出）操作后清空缓存
- ios::uppercase 强制大写字母

基本类型

Python体系

C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

附：序列类型

序列是具有先后关系的一组元素

- 序列是一维元素向量，元素类型可以不同
- 元素间由序号引导，通过下标访问序列的特定元素

序列是一个基类类型

序列类型
- **字符串类型**
- **列表类型**
- **元组类型**

- **对比找“不同”与“相同”**

- **不相同**

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 对比找“不同”与“相同”

序列类型

序号的定义

- 不相同

基本类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 对比找“不同”与“相同”

通用操作符

操作符及应用	描述
x in s	如果x是序列s的元素，返回True，否则返回False
x not in s	如果x是序列s的元素，返回False，否则返回True
s + t	连接两个序列s和t
s*n 或 n*s	将序列s复制n次
s[i]	索引，返回s中的第i个元素，i是序列的序号
s[i: j] 或 s[i: j: k]	切片，返回序列s中第i到j以k为步长的元素子序列

- 不相同

基本类型

Python体系 | C/C++体系

函数和方法	描述
len(s)	返回序列s的长度
min(s)	返回序列s的最小元素，s中元素需要可比较
max(s)	返回序列s的最大元素，s中元素需要可比较
s.index(x) 或 s.index(x, i, j)	返回序列s从i开始到j位置中第一次出现元素x的位置
s.count(x)	返回序列s中出现x的总次数

- 对比找“不同”与“相同”

- 不相同

控制结构

- 对比找“不同”与“相同”

1、顺序结构

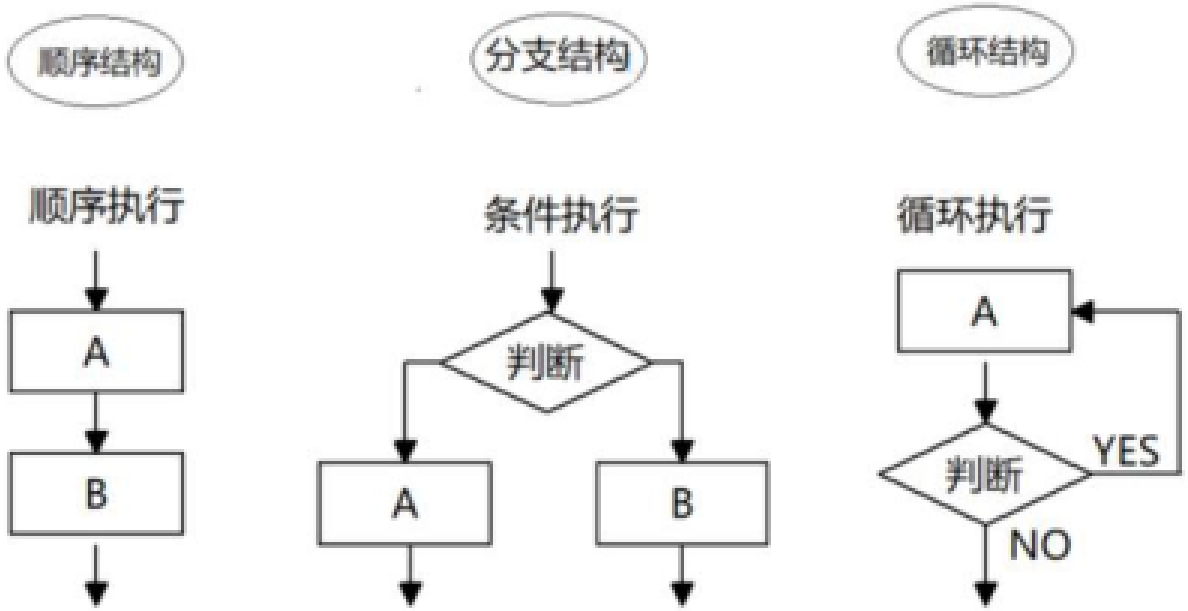
顺序结构的程序按语句的先后顺序依次执行各条语句。

2、分支结构

分支结构指程序根据条件执行不同的代码块。

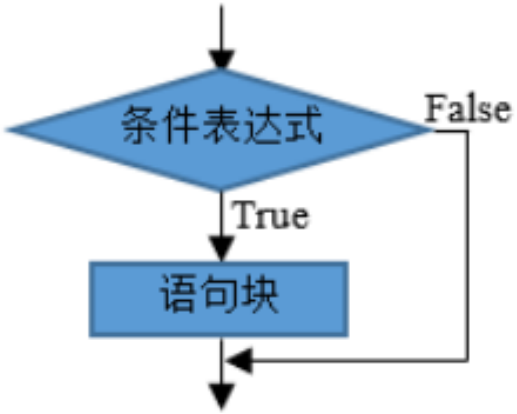
3、循环结构

循环结构指程序根据条件重复执行同一个代码块。



控制结构

Python体系 | C/C++体系



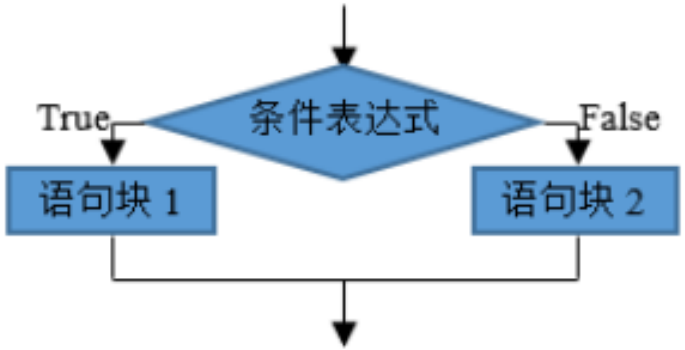

```
if <条件表达式>:
    <语句块>
```

```
guess = eval(input())
if  guess == 99:
    print("猜对了")
```

- 对比找“不同”与“相同”
- 基本相同

控制结构

Python体系 | C/C++体系




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

```
guess = eval(input())

if guess == 99:
    print("猜对了")
else :
    print("猜错了")
```

- 对比找“不同”与“相同”

- 基本相同

控制结构

Python体系 | C/C++体系

三元表达式： **<表达式1> if <条件表达式> else <表达式2>**

```
guess = eval(input())
print("猜{}了".format("对" if guess==99 else "错"))
```

列表三元表达式： **[表达式1 , 表达式2] [条件表达式]**

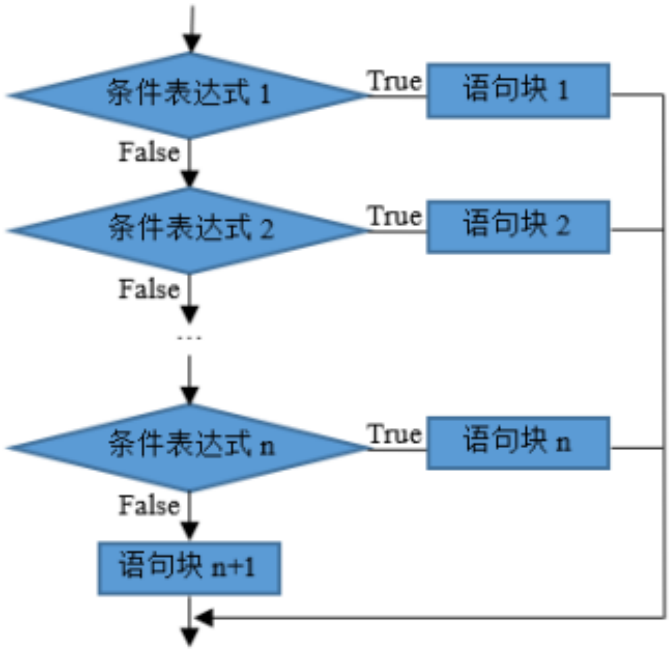
- 当条件表达式计算结果为False时，将表达式1的值作为三元表达式的值；
- 否则，将表达式2的值作为三元表达式的值。

```
>>> x=5
>>> y=10
>>> [x,y][x<y]          #x<y结果为True，返回y的值
10
```

- **对比找“不同”与“相同”**
- **基本相同**

控制结构

Python体系 | C/C++体系




```
if <条件表达式1>:
    <语句块1>
elif <条件表达式2>:
    <语句块2>
    ......
elif <条件表达式n>:
    <语句块n>
else:
    <语句块n+1>
```

- 对比找“不同”与“相同”
- 基本相同

控制结构

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

条件判断

操作符	数学符号	描述
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
! =	≠	不等于

条件组合

操作符及使用	描述
x and y	两个条件x和y的逻辑**与**
x or y	两个条件x和y的逻辑**或**
not x	条件x的逻辑**非**

- **对比找“不同”与“相同”**
- **基本相同**

控制结构

对比找“不同”与“相同”

Python 3.10 beta 版的发布，终于将 switch-case 语句纳入其中。格式如下：

```
match subject:
    case <pattern_1>:
        <action_1>
    case <pattern_2>:
        <action_2>
    case <pattern_3>:
        <action_3>
    case _:
        <action_wildcard>
```

```
http_code = "418"
match http_code:
    case "200":
        print("OK")
        do_something_good()
    case "404":
        print("Not Found")
        do_something_bad()
    case "418":
        print("I'm a teapot")
        make_coffee()
    case _:
        print("Code not found")
```

控制结构

Python体系

C/C++体系

3.1 for循环：

- 遍历某个结构形成的循环运行方式。
- 由保留字for和in组成。
- 从遍历结构中逐一提取元素，放在循环变量中，并执行一次语句块。
- 完整遍历所有元素后结束。

```
for <循环变量> in <遍历结构> :
    <语句块>
```

- 对比找“不同”与“相同”

- 不相同

控制结构

Python体系 | C/C++体系

```
for  i  in  range(N) :
     <语句块>
```

```
for i in range(5):
    print("Hello:", i)
```

```
Hello: 0
Hello: 1
Hello: 2
Hello: 3
Hello: 4
```

```
for  i  in  range(M,N,K) :
     <语句块>
```

```
for i in range(1,6,2):
    print("Hello:", i)
```

```
Hello: 1
Hello: 3
Hello: 5
```

`range(n)`：从0数到n. 不包含n

`range(m, n)`：从m数到n, 不包含n

`range(m, n, s)`：从m数到n, 不包含n, 每次的间隔是s

- 对比找“不同”与“相同”
- 不相同

控制结构

Python体系 | C/C++体系

遍历循环：

- 字符串遍历循环：遍历字符串每个字符，产生循环

```
for c in s :
    <语句块>
```

```
for c in "Python123":
    print(c, end=",")
```

```
P,y,t,h,o,n,1,2,3,
```

- 列表遍历循环：遍历列表的每个元素，产生循环

```
for item in ls :
    <语句块>
```

```
for item in [123, "PY", 456] :
    print(item, end=",")
```

```
123,PY,456,
```

- 对比找“不同”与“相同”

- 不相同

控制结构

Python体系

- 元组遍历循环：遍历元组的每个元素，产生循环

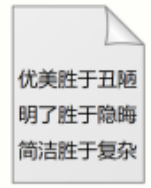
```
for  item  in  tp:
     <语句块>
```

```
for x in (1,2,3,(4,5)):
    print(x)
```

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

- 文件遍历循环：遍历文件的每行，产生循环

```
for  line  in  fi  :
     <语句块>
```



优美胜于丑陋
明了胜于隐晦
简洁胜于复杂

C/C++体系

- 对比找“不同”与“相同”

- 不相同

控制结构

Python体系

- 可在for循环中用多个变量来迭代序列对象。

```
>>> for (a,b) in ((1,2),(3,4),(5,6)):
...     print(a,b)
...
1 2
3 4
5 6
```

- # 等价于 for a,b in ((1,2),(3,4),(5,6)):

- 与赋值语句类似，可以用“*”表示给变量赋值一个列表。

```
>>> for (a,*b) in ((1,2,'abc'),(3,4,5)):
...     print(a,b)
...
1 [2, 'abc']
3 [4, 5]
```

C/C++体系

- **对比找“不同”与“相同”**

- **不相同**

控制结构

Python体系 | C/C++体系

```
while  <条件> :
    <语句块>
```

```
>>> a = 3
>>> while a > 0 :
        a = a + 1
        print(a)

4
5
        (CTRL + C退出执行)
…
```

- 对比找“不同”与“相同”

- 不相同

控制结构

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

3.3 循环控制保留字： break 和 continue

- break跳出并结束当前**整个**循环，执行循环后的语句
- continue结束**当次**循环，继续执行后续次数循环
- break和continue可以与for和while循环搭配使用

```
>>> for c in "PYTHON" :
        if c == "T" :
            break
        print(c, end="")

PY
```

```
>>> for c in "PYTHON" :
        if c == "T" :
            continue
        print(c, end="")

PYHON
```

- 对比找“不同”与“相同”
- 基本相同

控制结构

Python体系 | C/C++体系

- 对比找“不同”与“相同”

示例代码：

```
while True:
    n=eval(input('请输入一个正整数：'))
    if n==-1:
        break                          #输入为-1时结束程序
    if n<0:
        continue
    #计算n的阶乘
    s=1
    for x in range(2,n+1):             #当n不是整数时，会发生TypeError异常
        s*=x
    print('%s!=' % n,s)
```

- 基本相同

控制结构

Python体系 | C/C++体系

- 异常指程序在运行过程中发生的错误，异常会导致程序意外终止。
- 异常处理可捕捉程序中发生的异常，执行相应的处理代码，避免程序意外终止。
- 程序中的语法错误不属于异常。

```
try:
    可能引发异常的代码
except 异常类型名称:
    异常处理代码
else:
    没有发生异常时执行的代码
finally:
    不管是否发生异常，都会执行的代码
```

- 对比找“不同”与“相同”
- **基本相同**

控制结构

Python体系 | C/C++体系

```
while True:
    try:
        n=eval(input('请输入一个正整数：'))
        if n==-1:
            break                          #输入为-1时结束程序
        if n<0:
            continue
        #计算n的阶乘
        s=1
        for x in range(2,n+1):             #当n不是整数时，会发生TypeError异常
            s*=x
        print('%s!=' % n,s)
    except TypeError:                      #异常处理
        print('输入数据错误，必须是正整数！')
```

- 对比找“不同”与“相同”

- 基本相同

对比找“不同”与“相同”

- AttributeError：访问对象属性出错时引发的异常，例如访问不存在的属性或属性不支持赋值等。
- EOFError：使用input()函数读文件，遇到文件结束标志EOF时发生的异常。文件对象的read()和readline()方法遇到EOF时返回空字符串，不会引发异常。
- ImportError：导入模块出错引发的异常。
- IndexError：使用序列对象的下标超出范围时引发的异常。
- StopIteration：迭代器没有可执行迭代的迭代元素引发的异常。
- IndentationError：使用了不正确的缩进引发的异常。
- TabError：同时使用Tab键和空格导致缩进不一致引发的异常。
- TypeError：在运算或函数调用时，使用了不兼容的类型引发的异常。
- ZeroDivisionError：除数为0时引发的异常。

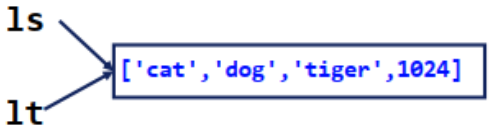
复合数据类型

Python体系

列表 List

- 列表是一种序列类型，支持索引、分片和合并等操作。
- 使用方括号 [] 或list() 创建，元素间用逗号 , 分隔。
- 列表中各元素类型可以不同，无长度限制。
- 列表是可变的。列表的长度可变，即可添加或删除列表成员，列表元素的值也可改变。
- 每个列表元素存储的是对象的引用，而不是对象本身，类似于C/C++的指针数组。

```
>>> ls = ["cat", "dog", "tiger", 1024]
>>> ls
['cat', 'dog', 'tiger', 1024]
>>> lt = ls
>>> lt
```



方括号 [] 真正创建一个列表，赋值仅传递引用

C/C++体系

对比找“不同”与“相同”

不同

容器库

容器库是类模板与算法的汇集，允许程序员简单地访问常见数据结构，例如队列、链表和栈。有三类容器——顺序容器、关联容器和无序关联容器——每种都被设计为支持不同组的操作。

容器管理为其元素分配的存储空间，并提供直接或间接地通过迭代器（拥有类似指针属性的对象）访问它们的函数。

大多数容器拥有至少几个常见的成员函数，并共享功能。特定应用的最佳容器不仅依赖于提供的功能，还依赖于对于不同工作量的效率。

顺序容器

顺序容器实现能按顺序访问的数据结构。

array (C++11 起)	静态的相接数组 (类模板)
vector	动态的相接数组 (类模板)
deque	双端队列 (类模板)
forward_list (C++11 起)	单链表 (类模板)
list	双链表 (类模板)

复合数据类型

Python体系

■ 列表 List

- 列表对象可以用列表常量或list()函数来创建。

```
>>> []                              #创建空列表对象
[]
>>> list()                          #创建空列表对象
[]
>>> [1,2,3]                         #用同类型数据创建列表对象
[1, 2, 3]
>>> [1,2,('a','abc'),[12,34]]       #用不同类型的数据创建列表对象
[1, 2, ('a', 'abc'), [12, 34]]
```

C/C++体系

■ 对比找“不同”与“相同”

■ 不同

```
list( const list& other );                                          (5)
list( const list& other, const Allocator& alloc );                  (5)  (C+
list( list&& other );                                               (6)  (C+
list( list&& other, const Allocator& alloc );                       (7)  (C+
list( std::initializer_list<T> init,
      const Allocator& alloc = Allocator() );                       (8)  (C+
```

复合数据类型

Python体系 | C/C++体系

- 列表 List

- 对比找“不同”与“相同”
- 不同

```
>>> list('abcd')                    #用可迭代对象创建列表对象
['a', 'b', 'c', 'd']
>>> list(range(-2,3))               #用连续整数创建列表对象
[-2, -1, 0, 1, 2]
>>> list((1,2,3))                   #用元组创建列表对象
[1, 2, 3]
>>> [x+10 for x in range(5)]        #用解析结构创建列表对象
[10, 11, 12, 13, 14]
```

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 列表 List

3.2 求长度

- 可用len()函数获得列表长度。

```
>>> len([])
0
>>> len( [ 1, 2, ('a', 'abc'), [3, 4] ])
4
```

3.3 合并

- 加法运算可用于合并列表。

```
>>> [1,2]+['abc',20]
[1, 2, 'abc', 20]
```

- 对比找“不同”与“相同”
- 不同

```
#include <list>
#include <iostream>

int main()
{
    std::list<int> nums {1, 3, 5, 7};

    std::cout << "nums contains " << nums.size() << " elements.\n";
}
```

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 列表 List

3.4 重复

- 乘法运算可用于创建具有重复值的列表。

```
>>> [1,2]*3
[1, 2, 1, 2, 1, 2]
```

3.5 关系判断

- 可用in操作符判断对象是否属于列表。

```
>>> 2 in [1,2,3]
True
>>> 'a' in [1,2,3]
False
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 列表 List

3.6 迭代

- 迭代操作可用于遍历列表元素。

```
>>> x=[1,2,('a','abc'),[12,34]]
>>> for a in x:print(a)
...
1
2
('a', 'abc')
[12, 34]
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 列表 List

3.7 索引

- 与字符串类似，可通过位置来索引列表元素，也可通过索引修改列表元素。

```
>>> x=[1,2,['a','b']]
>>> x[0]                    #输出列表的第1个数据
1
>>> x[2]                    #输出列表的第3个数据
['a', 'b']
>>> x[-1]                   #用负数从列表末尾开始索引
['a', 'b']
>>> x[2]=100                #修改列表的第3个数据
>>> x
[1, 2, 100]
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系

- 列表 List

C/C++体系

- 对比找“不同”与“相同”
- 不同

3.8 分片

- 与字符串类似，可通过分片来获得列表中的连续多个数据，也可通过分片将连续多个数据替换成新的数据。

```
>>> x=list(range(10))          #创建列表对象
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
 >>> x[2:5]                    #返回分片列表
[2, 3, 4]
>>> x[2:]                      #省略分片结束位置时，分片直到列表结束
[2, 3, 4, 5, 6, 7, 8, 9]
>>> x[:5]                      #省略分片开始位置时，分片从第1个数据开始
[0, 1, 2, 3, 4]
```

复合数据类型

Python体系 | C/C++体系

- 列表 List

```
>>> x[3:10:2]                    #指定分片时偏移量步长，步长为2
[3, 5, 7, 9]
>>> x[3:10:-2]                   #步长为负数时按相反顺序获得数据
[]
>>> x[10:3:-2]
[9, 7, 5]
>>> x[2:5]='abc'                 #通过分片替换多个数据
>>> x
[0, 1, 'a', 'b', 'c', 5, 6, 7, 8, 9]
>>> x[2:5]=[10,20]               #通过分片替换多个数据
>>> x
[0, 1, 10, 20, 5, 6, 7, 8, 9]
```

- 对比找“不同”与“相同”
- 不同

```
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

复合数据类型

Python体系 | C/C++体系

- 列表 List

3.9 添加单个/多个数据

- append()方法用于在列表末尾添加一个数据。

```
>>> x=[1,2]
>>> x.append('abc')
>>> x
[1, 2, 'abc']
```

- extend()方法用于在列表末尾添加多个数据，参数为可迭代对象。

```
>>> x=[1,2]
>>> x.extend(['a','b'])  #用列表对象作参数
>>> x
[1, 2, 'a', 'b']
>>> x.extend('abc')  #用字符串作参数时，每个字符作为一个数据
>>> x
[1, 2, 'a', 'b', 'a', 'b', 'c']
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 列表 List

- 对比找“不同”与“相同”
- 不同

3.10 插入数据

- insert()方法用于在指定位置插入数据。

```
>>> x=[1,2,3]
>>> x.insert(1,'abc')
>>> x
[1, 'abc', 2, 3]
```

3.11 按值删除数据

- remove()方法用于删除列表中的指定值。如果有重复值，则删除第1个。

```
>>> x=[1,2,2,3]
>>> x.remove(2)
>>> x
[1, 2, 3]
```

复合数据类型

Python体系 | C/C++体系

- 列表 List

- 对比找“不同”与“相同”
- 不同

3.12 按位置删除数据

- pop()方法用于删除指定位置的对象，省略位置时删除列表最后一个对象，同时返回删除对象。

```
>>> x=[1,2,3,4]
>>> x.pop()                    #删除并返回最后一个对象
4
 >>> x
[1, 2, 3]
>>> x.pop(1)                   #删除并返回偏移量为1的对象
2
>>> x
[1, 3]
```

复合数据类型

Python体系 | C/C++体系

- 列表 List

3.13 用del语句删除

- 可用del语句删除列表中的指定数据或分片。

```
>>> x=[1,2,3,4,5,6]
>>> del x[0]                    #删除第1个数据
>>> x
[2, 3, 4, 5, 6]
>>> del x[2:4]                  #删除偏移量为2、3的数据
>>> x
[2, 3, 6]
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系

- 列表 List

3.14 删除全部数据

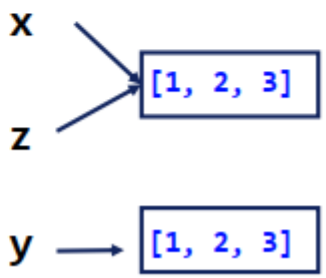
- clear()方法可删除列表中的全部数据。

```
>>> x=[1,2,3]
>>> x.clear()
>>> x
[]
```

3.15 复制列表

- copy()方法可以复制列表对象。

```
>>> x=[1,2,3]
>>> z=x
>>> y=x.copy()
>>> y
[1, 2, 3]
```



C/C++体系

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

列表 List

3.16 列表排序

- sort()方法可将列表排序，默认是升序排列。

```
>>> x=[10,2,30,5]
>>> x.sort()                          #对数字列表排序
>>> x
[2, 5, 10, 30]

>>> x=['bbc', 'abc', 'BBC', 'Abc']
>>> x.sort()                          #对字符串列表排序
>>> x
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

列表 List

- 若列表包含多种类型，则会出错。

```
>>> x=[1,5,3,'bbc','abc','BBC']
>>> x.sort()                                    #对混合类型列表排序时出错
Traceback (most recent call last):
  File "<pyshell#115>", line 1, in <module>
    x.sort()
TypeError: unorderable types: str() < int()
```

对比找“不同”与“相同”

不同

复合数据类型

Python体系 | C/C++体系

列表 List

- sort()方法通过按顺序使用“<”运算符比较列表元素实现排序，它还支持自定义排序。
- 可用key参数指定一个函数，sort()方法将列表元素作为参数调用该函数，函数返回值代替列表元素完成排序。

```
>>> def getv(a):                #返回字典中“键:值”对中的值
...     b=list(a.values())
...     return b
...
>>> x=[{'price':20},{'price':2},{'price':12}]
>>> x.sort(key=getv)            #按列表中每个字典的第1个“键:值”对中的值排序
>>> x
[{'price': 2}, {'price': 12}, {'price': 20}]
```

对比找“不同”与“相同”

不同

复合数据类型

Python体系 | C/C++体系

- 列表 List

- 对比找“不同”与“相同”
- 不同

sort()方法默认按从小到大排序，还可用reverse参数指定按从大到小排序。

```
>>> b=[12,5,9,8]
>>> b.sort(reverse=True)                    #从大到小排序
>>> b
[12, 9, 8, 5]
>>> x.sort(key=getv,reverse=True)       #从大到小排序
>>> x
[{'price': 20}, {'price': 12}, {'price': 2}]
```

控制结构

- 对比找“不同”与“相同”

定义空列表lt
```
>>> lt = []
```
向lt新增5个元素
```
>>> lt += [1,2,3,4,5]
```
修改lt中第2个元素
```
>>> lt[2] = 6
```
向lt中第2个位置增加一个元素
```
>>> lt.insert(2, 7)
```
从lt中第1个位置删除一个元素
```
>>> del lt[1]
```
删除lt中第1-3位置元素
```
>>> del lt[1:4]
```

判断lt中是否包含数字0
```
>>> 0 in lt
```
向lt新增数字0
```
>>> lt.append(0)
```
返回数字0所在lt中的索引
```
>>> lt.index(0)
```
lt的长度
```
>>> len(lt)
```
lt中最大元素
```
>>> max(lt)
```
清空lt
```
>>> lt.clear()
```

复合数据类型

Python体系

■ 元组 Tuple

4. 元组（Tuple）

- 元组是一种序列类型，继承了序列类型的全部通用操作，可通过位置进行索引和分片。
- 元组可包含任意类型的对象。
- 元组可以看作是不可变的列表。
- 元组的大小不能改变，即不能为元组添加对象，也不能删除元组中的对象。元组中的对象也不能改变。
- 可以使用小括号 () 或 tuple() 创建，元素间用逗号，分隔。
- 可以使用或不使用小括号。
- 元组中存储的是对象的引用，而不是对象本身。

C/C++体系

■ 对比找“不同”与“相同”

■ 不同

标准库头文件 <tuple>

此头文件是通用工具库的一部分。

类

tuple (C++11)	实现固定大小的容器，它保有类型可以相异的元素 (类模板)
tuple_size	在编译时获得 tuple 的大小 (类模板特化)
tuple_element	获得指定元素的类型 (类模板特化)
std::uses_allocator<std::tuple> (C++11)	特化 std::uses_allocator 类型特性 (类模板特化)

常量

ignore	用 tie 解包 tuple 时用来跳过元素的占位符 (常量)

函数

make_tuple	创建一个类型由参数类型定义的tuple对象 (函数模板)
tie	创建左值引用的一个 tuple ，或解包 tuple 为独立对象 (函数模板)
forward_as_tuple	创建右值引用的 tuple (函数模板)
tuple_cat	通过连接任意数量的元组来创建一个tuple (函数模板)
std::get(std::tuple)	tuple 访问指定的元素 (函数模板)
operator== operator!= operator< operator<= operator> operator>=	按字典顺序比较 tuple 中的值 (函数模板)
std::swap(std::tuple) (C++11)	特化 std::swap 算法 (函数模板)
apply (C++17)	以参数的元组调用函数 (函数模板)
make_from_tuple (C++17)	以参数的元组构造对象 (函数模板)

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

■ 元组 Tuple

4.1 创建元组

- 可用元组常量或tuple()方法来创建元组对象。

```
>>> ()                          #创建空元组对象
()
>>> tuple()                     #创建空元组对象
()
>>> (2,)                        #包含一个对象的元组，不能缺少逗号
(2,)
>>> (1,2.5,'abc',[1,2])         #包含不同类型对象的元组
(1, 2.5, 'abc', [1, 2])
>>> 1,2.5,'abc',[1,2]           #元组常量可以省略括号
(1, 2.5, 'abc', [1, 2])
```

■ 对比找“不同”与“相同”

■ 不同

复合数据类型

Python体系 | C/C++体系

- 元组 Tuple

```
>>> (1,2,('a','b'))                    #元组可以嵌套元组
(1, 2, ('a', 'b'))

>>> tuple('abcd')                      #用字符串创建元组，可迭代对象均可用于创建元组
('a', 'b', 'c', 'd')

>>> tuple([1,2,3])                     #用列表创建元组
(1, 2, 3)

>>> tuple(x*2 for x in range(5))       #用解析结构创建元组
(0, 2, 4, 6, 8)
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

- 元组 Tuple

4.2 求长度

- len()函数可用于获得元组长度。

```
>>> len((1,2,3,4))
4
```

4.3 合并

- 加法运算可用于合并多个元组。

```
>>> (1,2)+('ab','cd')+(2.45,)
(1, 2, 'ab', 'cd', 2.45)
```

4.4 重复

- 乘法运算可用于合并多个重复的元组。

```
>>> (1,2)*3
(1, 2, 1, 2, 1, 2)
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

元组 Tuple

4.5 迭代

- 可用迭代方法遍历元组中的各个对象。

```
>>> for x in (1,2.5,'abc',[1,2]):print(x)
...
1
2.5
abc
[1, 2]
```

4.6 关系判断

- in操作符可用于判断对象是否属于元组。

```
>>> 2 in (1,2)
True
>>> 5 in (1,2)
False
```

对比找“不同”与“相同”

不同

复合数据类型

Python体系 | C/C++体系

- 元组 Tuple

- 对比找“不同”与“相同”
- 不同

4.7 索引和分片

- 可通过位置对元组对象进行索引和分片。

```
>>> x=tuple(range(10))
>>> x
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> x[1]
1
>>> x[-1]
9
>>> x[2:5]
(2, 3, 4)
```

```
>>> x[2:]
(2, 3, 4, 5, 6, 7, 8, 9)
>>> x[:5]
(0, 1, 2, 3, 4)
>>> x[1:7:2]
(1, 3, 5)
>>> x[7:1:-2]
(7, 5, 3)
```

复合数据类型

Python体系 | C/C++体系

- 元组 Tuple

4.8 count()方法

- count()方法用于返回指定值在元组中出现的次数。

```
>>> x=(1,2)*3
>>> x
(1, 2, 1, 2, 1, 2)
>>> x.count(1)                    #返回1在元组中出现的次数
3
>>> x.count(3)                    #元组不包含指定值时，返回0
0
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 元组 Tuple

4.9 index(value,[start,[end]])方法

- 未用start和end指定范围时，返回指定值在元组中第一次出现的位置。
- 指定范围时，返回指定值在指定范围内第一次出现的位置。

```
>>> x=(1,2,3)*3
>>> x
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>> x.index(2)                    #默认查找全部元组
1
>>> x.index(2,2)                  #从偏移量2到元组末尾查找
4
>>> x.index(2,2,7)                #在范围[2:7]内查找
4
>>> x.index(5)                    #如果元组不包含指定的值，则出错
Traceback (most recent call last):
  File "<pyshell#171>", line 1, in <module>
    x.index(5)
ValueError: tuple.index(x): x not in tuple
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 集合 set

5. 集合（Sets）

- 集合是多个元素的无序组合，每个元素唯一，不存在相同元素。
- 集合类型与数学中的集合概念一致。
- 集合元素不可更改，不能是可变数据类型。
- 集合用大括号 {} 表示，元素间用逗号分隔，建立集合类型用 {} 或 set()。
- 建立空集合类型，必须使用set()。
- 集合支持数学理论中的各种集合运算。

- 对比找“不同”与“相同”
- 不同

标准库头文件 <set>

此头文件是容器库的一部分。

包含

<initializer_list> (C++11)	

类

set	唯一键的集合，按照键排序 (类模板)
multiset	键的集合，按照键排序 (类模板)

函数

operator== operator!= operator< operator<= operator> operator>=	按照字典顺序比较 set 中的值 (函数模板)
std::swap(std::set)	特化 std::swap 算法 (函数模板)
operator== operator!= operator< operator<= operator> operator>=	按照字典顺序比较 multiset 中的值 (函数模板)
std::swap(std::multiset)	特化 std::swap 算法 (函数模板)

复合数据类型

Python体系 | C/C++体系

- 集合 set

5.1 集合常量

- 集合常量用大括号表示，也可用内置的set()函数创建集合对象。

```
>>> x={1,2,3}                #直接使用集合常量
>>> x
{1, 2, 3}
>>> type(x)                  #测试集合对象的类型名称
<class 'set'>
>>> set({1,2,3})             #用集合常量做参数创建集合对象
{1, 2, 3}
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

■ 集合 set

```
>>> set([1,2,3])              #用列表常量做参数创建集合对象
{1, 2, 3}
>>> set('123abc')             #用字符串常量做参数创建集合对象
{'a', '3', 'b', 'c', '2', '1'}
>>> set()                     #创建空集合
set()
```

- 集合中的元素不允许有重复值，在创建集合对象时，Python会自动去掉重复值。

```
>>> {1,1,2,2}
{1, 2}                        数据去重
>>> set([1,1,2,2])
{1, 2}
```

■ 对比找“不同”与“相同”

■ 不同

复合数据类型

Python体系 | C/C++体系

- 集合 set
- 对比找“不同”与“相同”
- 不同

5.2 集合运算

- 集合对象支持求长度、判断包含、求差集、求并集、求交集、求对称差和比较等运算。

```
>>> x={1,2,'a','bc'}
>>> y={1,'a',5}
>>> len(x)                  #求长度：计算集合中元素的个数
4
>>> 'a' in x                #判断包含：判断集合是否包括数据
True
>>> x – y                   #求差集：用属于x但不属于y的元素创建新集合
{2, 'bc'}
```

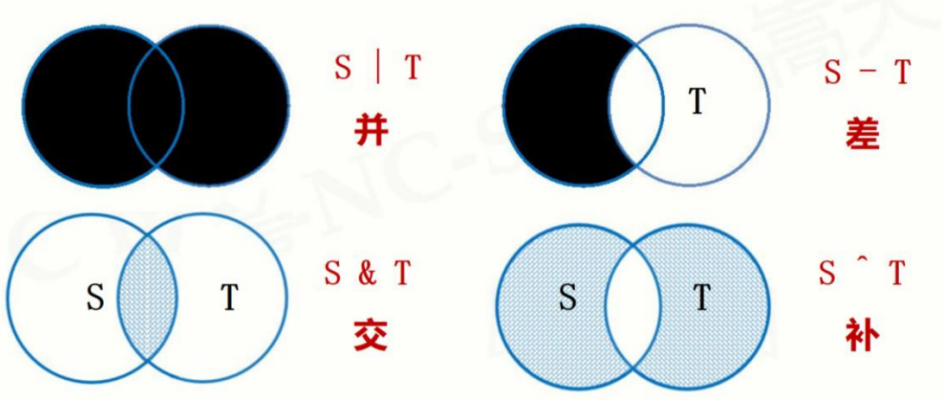
复合数据类型

Python体系 | C/C++体系

- 集合 set

```
>>> x={1,2,'a','bc'}
>>> y={1,'a',5}
>>> x | y
{1, 2, 'a', 'bc', 5}
>>> x & y
{1, 'a'}
>>> x ^ y
{2, 5, 'bc'}
>>> x < y
False
>>> {1,2}<x
True
```

- 对比找“不同”与“相同”
- 不同



操作符及应用	描述
S | T	返回一个新集合，包括在集合S和T中的所有元素
S - T	返回一个新集合，包括在集合S但不在T中的元素
S & T	返回一个新集合，包括同时在集合S和T中的元素
S ^ T	返回一个新集合，包括集合S和T中的非相同元素
S <= T 或 S < T	返回True/False，判断S和T的子集关系
S >= T 或 S > T	返回True/False，判断S和T的包含关系

复合数据类型

Python体系 | C/C++体系

- 集合 set
- 对比找“不同”与“相同”
- 不同

5.3 集合操作

- 集合对象元素的值不支持修改
- 可以复制集合、为集合添加或删除元素

```
>>> x={1,2}
>>> y=x.copy()                    #复制集合对象
>>> y
{1, 2}
>>> x.add('abc')                  #为集合添加一个元素
>>> x
{1, 2, 'abc'}
```

复合数据类型

Python体系 | C/C++体系

- 集合 set

- 对比找“不同”与“相同”
- 不同

```
>>> x.update({10,20})                    #为集合添加多个元素
>>> x
{1, 2, 10, 20, 'abc'}
>>> x.remove(10)                         #从集合中删除指定元素
>>> x
{1, 2, 20, 'abc'}
>>> x.remove(50)                         #删除不存在元素时会报错
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 50
>>> x.discard(20)                        #从集合中删除指定元素
>>> x
{1, 2, 'abc'  }
>>> x.discard(50)                        #删除不存在元素时不报错
```

复合数据类型

Python体系 | C/C++体系

- 集合 set

```
>>> x.pop()          #从集合中随机删除一个元素，并返回该元素。
1
>>> x
{2, 'abc'}
>>> x.clear()        #删除集合中的全部元素
>>> x
set()
```

- 集合可用for循环执行迭代操作。

```
>>> x={1,2,3}
>>> for a in x:print(a)
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

集合 set

- 对比找“不同”与“相同”
- 不同

- 集合元素是不可改变的，因此不能将可变对象放入集合中。
- 集合、列表和字典对象均不能加入集合。元组可以作为一个元素加入集合。

```
>>> x={1,2}
>>> x
{1, 2}
>>> x.add({1})                                    #不能将集合对象加入集合
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    x.add({1})
TypeError: unhashable type: 'set'
```

复合数据类型

Python体系 | C/C++体系

- 集合 set

- 对比找“不同”与“相同”
- 不同

```
>>> x.add([1,2,3])                          #不能将列表对象加入集合
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    x.add([1,2,3]
TypeError: unhashable type: 'list'
>>> x.add({'Mon':1})                        #不能将字典对象加入集合
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    x.add({'Mon':1})
TypeError: unhashable type: 'dict'
>>> x.add((1,2,3))                          #可以将元组加入集合
>>> x
{1, 2, (1, 2, 3)}
```

复合数据类型

Python体系 | C/C++体系

- 集合 set
- 对比找“不同”与“相同”
- 不同

冻结集合

- Python提供了一种特殊的集合——冻结集合（frozenset）
- 冻结集合是一个不可改变的集合，可将其作为其他集合的元素。
- 冻结集合的输出格式与普通集合不同：

```
>>> x=frozenset([1,2,3])                    #创建冻结集合
>>> print(x)                                #输出冻结集合
frozenset({1, 2, 3})

>>> y=set([4,5])
>>> y.add(x)                                #将冻结集合作为元素加入另一个集合
>>> y
{frozenset({1, 2, 3}), 4, 5}
```

复合数据类型

Python体系 | C/C++体系

西安交通大学 XI'AN JIAOTONG UNIVERSITY

■ 字典 Dictionary

6. 字典（Dictionary）

- 字典是一种无序的映射集合，包含一系列的“键:值”对。
- 字典常量用大括号表示，例如，{'name':'John','age':25,'gender':'male’ }。
- 字典的键名称通常采用字符串，也可以用数字、元组等不可变的类型。
- 字典的值可以是任意类型。
- 通过键来访问映射的值，而不是通过位置来索引。
- 字典长度可变，可为字典添加或删除“键:值”对。
- 字典可以任意嵌套，即键映射的值可以是一个字典。
- 字典存储的是对象的引用，而不是对象本身。

■ 对比找“不同”与“相同”

■ 不同

标准库头文件 <map>

此头文件是容器库的一部分。

类

map	键值对的集合，按照键排序，键是唯一的 (类模板)
multimap	键值对的集合，按照键排序 (类模板)

函数

operator== operator!= operator< operator<= operator> operator>=	按照字典顺序比较 map 中的值 (函数模板)
std::swap(std::map)	特化 std::swap 算法 (函数模板)
operator== operator!= operator< operator<= operator> operator>=	按照字典顺序比较 multimap 中的值 (函数模板)
std::swap(std::multimap)	特化 std::swap 算法 (函数模板)

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

6.1 创建字典

```
>>> {}                                          #创建空字典
{}
>>> dict()                                      #创建空字典
{}
>>> {'name':'John','age':25,'gender':'male'}          #使用字典常量
{'gender': 'male', 'age': 25, 'name': 'John'}
>>> {'book':{'Python编程':100,'C++入门':99}}      #使用嵌套的字典
{'book': {'C++入门': 99, 'Python编程': 100}}
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> {1:'one',2:'two'}                          #将数字作为键
{1: 'one', 2: 'two'}
>>> {(1,3,5):10,(2,4,6):50}                    #将元组作为键
{(1, 3, 5): 10, (2, 4, 6): 50}
>>> dict(name='John',age=25)                   #使用赋值格式的“键:值”对创建字典
{'age': 25, 'name': 'John'}
>>> dict([('name','John'),('age',25)])         #使用包含“键:值”对元组的列表创建字典
{‘age’: 25, 'name': 'John'}
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> dict.fromkeys(['name','age'])          #创建无映射值的字典，默认值为
None
{'age': None, 'name': None}
 >>> dict.fromkeys(['name','age'],0)       #创建值相同的字典
{'age': 0, 'name': 0}
 >>> dict.fromkeys('abc')                  #使用字符串创建无映射值的字典
{'a': None, 'b': None, 'c': None}
 >>> dict.fromkeys('abc',10)               #使用字符串和映射值创建字典
{'a': 10, 'b': 10, 'c': 10}
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> dict.fromkeys('abc',(1,2,3))
{'a': (1, 2, 3), 'b': (1, 2, 3), 'c': (1, 2, 3)}
>>> dict(zip(['name','age'],['John',25]))    #使用zip解析“键:值”列表创建字典
{'name': 'John', 'age': 25}
>>> x={}                                     #先创建一个空字典
>>> x['name']='John'                         #通过赋值添加“键:值”对
>>> x['age']=25
>>> x
{'age': 25, 'name': 'John'}
```

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

6.2 求长度

- len()函数可返回字典长度，即“键:值”对的个数

```
>>> len({'name':'John','age':25,'gender':'male'})
3
```

6.3 关系判断

- in操作符可用于判断字典是否包含某个键

```
>>> 'name' in {'name':'John','age':25,'gender':'male'}
True
>>> 'date' in {'name':'John','age':25,'gender':'male'}
False
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary
- 对比找“不同”与“相同”
- 不同

6.4 索引

- 字典可通过键来索引其映射的值。

```
>>> x={'book':{'Python编程':100,'C++入门':99},'publish':'人民邮电出版社'}
>>> x['book']
{'C++入门': 99, 'Python编程': 100}
>>> x['publish']
'人民邮电出版社'
>>> x['book']['Python编程’ ]          #用两个键索引嵌套的字典元素
100
```

复合数据类型

Python体系 | C/C++体系

■ 字典 Dictionary

■ 对比找“不同”与“相同”

■ 不同

- 可通过索引修改映射值。

```
>>> x=dict(name='John',age=25)
>>> x
{'age': 25, 'name': 'John'}
>>> x['age']=30                    #修改映射值
>>> x
{'age': 30, 'name': 'John'}
>>> x['phone']='17055233456'   #为不存在的键赋值，为字典添加“键:值”对
>>> x
{'phone': '17055233456', 'age': 30, 'name': 'John'}
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

也可通过索引删除键值对。

```
>>> x={'name':'John','age':25}
>>> del x['name']          #删除“键:值”对
>>> x
{'age': 25}
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

6.5 字典常用方法

clear() #删除全部字典对象

```
>>> x=dict(name='John',age=25)
>>> x.clear()
>>> x
{}
```

copy() #复制字典对象

```
>>> x={'name':'John','age':25}
>>> y=x                                  #直接赋值时，x和y引用同一个字典
>>> y
{'name': 'John', 'age': 25}
>>> y['name']='Curry'                    #通过y修改字典
>>> x,y                                  #显示结果相同
({'age': 25, 'name': 'Curry'}, {'age': 25, 'name': 'Curry'})
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> y is x                                  #判断是否引用相同对象
True
>>> y=x.copy()                              #y引用复制的字典
>>> y['name']='Python'                      #此时不影响x的引用
>>> x,y
({'age': 25, 'name': 'Curry'}, {'age': 25, 'name': 'Python'})
>>> y is x                                  #判断是否引用相同对象
False
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

get(key[, default])

- get()方法返回键key映射的值。如果键key不存在，返回空值。
- 可用default参数指定键不存在时的返回值。

```
>>> x={'name':'John','age':25}
>>> x.get('name')                    #返回映射值
'John'
>>> x.get('addr')                    #不存在的键返回空值
>>> x.get('addr','xxx’)              #不存在的键返回指定值
'xxx'
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

pop(key[, default])

- pop()方法从字典中删除“键:值”对，并返回映射值。

```
>>> x={'name':'John','age':25}
>>> x.pop('name')                    #删除键并返回映射值
'John'
>>> x
{'age': 25}
```

复合数据类型

Python体系 | C/C++体系

■ 字典 Dictionary

- 若键不存在，则返回default。
- 若键不存在且未指定default参数时，删除键会出错。

```
>>> x.pop('gender','xxx')          #删除不存在的键，返回default参数值
'xxx'
>>> x.pop('gender')                #删除不存在的键，未指定default参数，出错
Traceback (most recent call last):
  File "<pyshell#252>", line 1, in <module>
    x.pop('gender')
KeyError: 'gender'
```

■ 对比找“不同”与“相同”

■ 不同

字典 Dictionary

popitem()

- pop()方法从字典中删除“键:值”对，并返回映射值。
- popitem()方法从字典删除“键:值”对，同时返回“键:值”对元组。
- 空字典调用该方法会产生KeyError错误。

```
>>> x={'name':'John','age':25}
>>> x.popitem()                    #删除“键:值”对并返回元组
('age', 25)
>>> x                              #x中剩余一个“键:值”对
{'name': 'John'}
```

对比找“不同”与“相同”

不同

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> x.popitem()                    #删除“键:值”对并返回元组
('name', 'John’)
>>> x                              #x为空字典
{}
>>> x.popitem()                    #空字典产生KeyError错误
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    x.popitem()
KeyError: 'popitem(): dictionary is empty'
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

update(other)

- update()方法用于为字典添加“键:值”对。
- 参数other可以是另一个字典或用赋值格式表示的元组。
- 若字典已存在同名的键，则映射值被覆盖。

```
>>> x={'name':'John','age':25}
>>> x.update({'age':30,'gender':'male'})  #添加“键:值”对，并覆盖同名键的映射值
>>> x                                     #age的映射值已被修改
{'gender': 'male', 'age': 30, 'name': 'John'}
>>> x.update(code=110,address='NewStreet')    #添加“键:值”对
>>> x
{'gender': 'male', 'address': 'NewStreet', 'age': 30, 'code': 110, 'name': 'Mike'}
```

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

6.6 字典视图

items()

- items()方法返回“键:值”对视图。

```
>>> x={'name':'John','age':25}
>>> y=x.items()                    #返回“键:值”对视图
>>> y                              # “键:值”对视图为dict_items对象
dict_items([('age', 25), ('name', 'John')])
>>> list(y)                        #将“键:值”对视图转换为列表
[('age', 25), ('name', 'John')]
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

```
>>> for a in y:print(a)                    #迭代“键:值”对视图
…
('age', 25)
('name', 'John')
>>> x['age']=30                            #修改字典
>>> x
{'age': 30, 'name': 'John'}
>>> y                                      #从显示结果可以看出视图反映了字典中的修改内容
dict_items([('age', 30), ('name', 'John')])
```

复合数据类型

Python体系 | C/C++体系

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

keys()

- keys()方法返回字典中所有键的视图。

```
>>> x={'name':'John','age':25}
>>> y=x.keys()                          #返回键的视图
>>> y                                   #显示键视图，键视图为dict_keys对象
dict_keys(['name',  'age'])
>>> x['gender']='male'                  #为字典添加“键:值”对
>>> x
{'name': 'John', 'age': 25, 'gender': 'male'}
>>> y                                   #显示结果说明键视图包含了新添加的键
dict_keys(['gender', 'age', 'name'])
>>> list(y)                             #将键视图转换为列表
['gender', 'age', 'name']
```

- 字典 Dictionary

- 对比找“不同”与“相同”
- 不同

values()

- values()方法返回字典中全部值的视图。

```
>>> x={'name':'John','age':25}
>>> y=x.values()                    #返回字典的值视图
>>> y                               #显示值视图，值视图为dict_values对象
dict_values([25, 'John'])
 >>> x['gender']='male'             #添加“键:值”对
>>> y                               #值视图包含了新添加的值
dict_values(['male', 25, 'John'])
>>> list(y)                         #将值视图转换为列表
['male', 25, 'John']
```

复合数据类型

Python体系 | C/C++体系

■ 字典 Dictionary

■ 对比找“不同”与“相同”

■ 不同

键视图的集合操作

- 键视图支持各种集合运算，“键:值”对视图和值视图不支持集合运算。

```
>>> x={'a':1,'b':2}
>>> kx=x.keys()                    #返回x的键视图
>>> y={'b':3,'c':4}
>>> ky=y.keys()                    #返回y的键视图
>>> kx-ky                          #求差集
{'a’}
>>> kx|ky                          #求并集
{'a', 'b', 'c’}
>>> kx&ky                          #求交集
{'b’}
>>> kx^ky                          #求对称差集
{'a', 'c'}
```

Python体系

- 迭代

C/C++体系

- 对比找“不同”与“相同”
- 不同

7. 迭代

- 字符串、列表、元组和字典等对象均支持迭代操作，可使用迭代器遍历对象。
- 字符串、列表、元组和字典等对象没有自己的迭代器，可调用iter()函数生成迭代器。
- 对迭代器调用next()函数即可遍历对象。
- next()函数依次返回可迭代对象的元素，无数据返回时，会产生StopIteration异常

复合数据类型

Python体系 | C/C++体系

- 迭代

```
>>> d=iter([1,2,3])          #为列表生成迭代器
>>> next(d)                  #返回第1个数据
1
>>> next(d)                  #返回第2个数据
2
>>> next(d)                  #返回第3个数据
3
>>> next(d)                  #无数据可返回，产生异常
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 迭代

```
>>> d=iter((1,2,(3,4)))          #使用迭代器迭代元组
>>> next(d)
1
>>> next(d)
2
>>> next(d)
(3, 4)
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 迭代

```
>>> d=iter('abc')                    #使用迭代器迭代字符串
>>> next(d)
'a'
>>> next(d)
'b'
>>> next(d)
'c'
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 迭代

```
>>> d=iter({'name':'John','age':25})      #使用迭代器迭代字典，字典只能迭代键
>>> next(d)
'name'
>>> next(d)
'age’
>>> d=iter({'name':'John','age':25}.keys())        #迭代字典keys方法返回对象
>>> next(d)
‘name'
>>> next(d)
‘age'
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

Python体系 | C/C++体系

- 迭代

```
>>> d=iter({'name':'John','age':25}.values())      #迭代字典values方法返回对象
>>> next(d)
'John'
>>> next(d)
25
>>> d=iter({'name':'John','age':25}.items())       #迭代字典items方法返回对象
>>> next(d)
('name', 'John')
>>> next(d)
('age', 25 )
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

迭代

- 也可用next()函数来迭代文件对象。

```
>>> mf=open('code.txt')
>>> next(mf)
'one第一行\n'
>>> next(mf)
'two第二行\n'
>>> next(mf)
'three第三行xxx'
>>> next(mf)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

对比找“不同”与“相同”

不同

- 列表解析

- 对比找“不同”与“相同”
- 不同

8. 列表解析

- 列表解析与循环的概念紧密相关，先通过下面的例子了解如何使用for循环来修改列表。

```
>>> t=[1,2,3,4]
>>> for x in range(4):
...     t[x]=t[x]+10
...
>>> t
[11, 12, 13, 14]
```

- 使用列表解析来代替上面例子的for循环。

```
>>> t=[1,2,3,4]
>>> t=[x+10 for x in t]
>>> t
[11, 12, 13, 14]
```

列表解析

- 列表解析的基本结构如下。

  表达式 for 变量 in 可迭代对象 if 表达式

带条件的列表解析

- 可以在列表解析中使用if设置筛选条件。

```
>>> [x+10 for x in range(10) if x%2==0]     #用if筛选偶数
[10, 12, 14, 16, 18]
```

多重解析嵌套

- 列表解析支持嵌套。

```
>>> [x+y for x in (10,20) for y in (1,2,3)]
[11, 12, 13, 21, 22, 23]
```

- 嵌套时，Python对第1个for循环中的每个x，执行嵌套for循环。

对比找“不同”与“相同”

- 不同

复合数据类型

列表解析

- 用下面的嵌套的for循环来生成上面的列表。

```
>>> a=[]
>>> for x in (10,20):
...     for y in (1,2,3):
...         a.append(x+y)
...
>>> a
[11, 12, 13, 21, 22, 23]
```

- 对嵌套的解析，也可以分别使用if执行筛选。

```
>>> [x+y for x in (10,20) if x>10 for y in (1,2,3) if y%2==1]
[21, 23]
```

对比找“不同”与“相同”

不同

复合数据类型

- 列表解析

列表解析用于生成元组

```
>>> tuple(x*2 for x in range(5))
(0, 2, 4, 6, 8)
>>> tuple(x*2 for x in range(10) if x%2==1)
(2, 6, 10, 14, 18)
```

列表解析用于生成集合

```
>>> {x for x in range(10)}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
>>> {x for x in range(10) if x%2==1}
{1, 3, 5, 9, 7}
```

- 对比找“不同”与“相同”
- 不同

复合数据类型

- 列表解析

列表解析用于生成字典

```
>>> {x:ord(x) for x in 'abcd'}
{'d': 100, 'a': 97, 'b': 98, 'c': 99}
>>> {x:ord(x) for x in 'abcd' if ord(x)%2==0}
{'d': 100, 'b': 98}
```

列表解析用于文件时，每次从文件读取一行数据

```
>>> [x for x in open('code.txt')]
['one第一行\n', 'two第二行\n', 'three第三行']
>>> [x.strip() for x in open('code.txt')]
['one第一行', 'two第二行', 'three第三行']
>>> [x.strip() for x in open('code.txt') if x[0]=='t']
['two第二行', 'three第三行']
```

- 对比找“不同”与“相同”
- 不同

- 对比找“不同”与“相同”
- 不同

zip()、map()和filter()

zip()函数

- zip()函数用于创建zip对象，其参数为多个可迭代对象。
- 生成zip对象时，每次从可迭代对象中取一个值组成一个元组，直到可迭代对象中的值取完，生成的zip对象包含了一系列的元组

```
>>> x=zip((1,2,3),(10,20,30))                    #用两个元组参数来生成zip对象
>>> x
<zip object at 0x00672508>
>>> next(x)                                      #返回zip对象中的下一个值
(1, 10)
>>> next(x)
(2, 20)
>>> next(x)
(3, 30)
```

复合数据类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 不同

```
>>> next(x)                          #已无对象，产生StopIteration异常
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
>>> x=zip('abc',(1,2,3))             #使用一个字符串和一个元组作参数
>>> next(x)
('a', 1)
>>> next(x)
('b', 2)
>>> next(x)
('c', 3)
>>> x=zip((1,2),'ab',[5,6])          #使用多个可迭代对象作参数
>>> next(x)
(1, 'a', 5)
>>> next(x)
(2, 'b', 6)
```

复合数据类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 不同

map()函数

- map()函数用于将函数映射到可迭代对象，对可迭代对象中的每个元素应用该函数，函数返回值包含在生成的map对象中。

```
>>> x=map(ord,'abc’)                    #用ord函数返回各个字符的ASCII码，生成map对象
>>> x
<map object at 0x00D9A4B0>
>>> next(x)                             #返回map对象中的下一个值
97
>>> next(x)
98
>>> next(x)
99
>>> list(map(ord,'abc'))                #用map对象生成列表
[97, 98, 99]
```

复合数据类型

Python体系 | C/C++体系

- 对比找“不同”与“相同”
- 不同

filter()函数

- filter()函数与map()函数类似，filter()函数用指定函数处理可迭代对象。
- 若函数返回值为真，则对应可迭代对象元素包含在生成的filter对象。

```
>>> x=filter(bool,(1,-1,0,'ab','',(),[],{},(1,2),[1,2],{1,2},{'a':1}))  #筛选出可转换为真的对象
>>> x
<filter object at 0x00D9A570>
>>> next(x)                                    #返回filter对象中的下一个值
1
>>> list(x)                                    #将迭代器转换为列表，不包含已迭代的值
[-1, 'ab', (1, 2), [1, 2], {1, 2}, {'a': 1}]
```

本章结束！