# 基础

## 标识符

* 第一个字符必须是字母表中字母或下划线 **\_** 。
* 标识符的其他的部分由字母、数字和下划线组成。
* 标识符对大小写敏感。

## 基本数据类型

在 Python 中，变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。

等号（=）用来给变量赋值。

等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值。

* 多个变量赋值

a=b=c=1

a,b,c=1,2,’runoob’

## 标准数据类型

Python3 中有六个标准的数据类型：

* Number（数字）
* String（字符串）
* List（列表）
* Tuple（元组）
* Set（集合）
* Dictionary（字典）

Python3 的六个标准数据类型中：

* **不可变数据（3 个）：**Number（数字）、String（字符串）、Tuple（元组）；
* **可变数据（3 个）：**List（列表）、Dictionary（字典）、Set（集合）。

## 数字(Number)类型

python中数字有四种类型：整数、布尔型、浮点数和复数。

* **int** (整数), 如 1, 只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。
* **bool** (布尔), 如 True。
* **float** (浮点数), 如 1.23、3E-2
* **complex** (复数), 如 1 + 2j、 1.1 + 2.2j

#### 数字类型转换

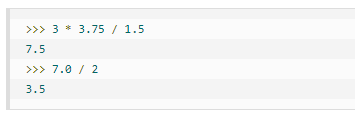
* **int(x)** 将x转换为一个整数。
* **float(x)** 将x转换到一个浮点数。
* **complex(x)** 将x转换到一个复数，实数部分为 x，虚数部分为 0。
* **complex(x, y)** 将 x 和 y 转换到一个复数，实数部分为 x，虚数部分为 y。x 和 y 是数字表达式。
* 在整数除法中，除法 **/** 总是返回一个浮点数，如果只想得到整数的结果，丢弃可能的分数部分，可以使用运算符 **//** ：
* **//** 得到的并不一定是整数类型的数，它与分母分子的数据类型有关系。



* Python 可以使用 **\*\*** 操作来进行幂运算：



* 不同类型的数混合运算时会将整数转换为浮点数：



#### 数学函数

|  |  |
| --- | --- |
| **函数** | **返回值 ( 描述 )** |
| [abs(x)](https://www.runoob.com/python3/python3-func-number-abs.html) | 返回数字的绝对值，如abs(-10) 返回 10 |
| [ceil(x)](https://www.runoob.com/python3/python3-func-number-ceil.html) | 返回数字的上入整数，如math.ceil(4.1) 返回 5 |
| cmp(x, y) | 如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1。 **Python 3 已废弃，使用 (x>y)-(x<y) 替换**。 |
| [exp(x)](https://www.runoob.com/python3/python3-func-number-exp.html) | 返回e的x次幂(ex),如math.exp(1) 返回2.718281828459045 |
| [fabs(x)](https://www.runoob.com/python3/python3-func-number-fabs.html) | 返回数字的绝对值，如math.fabs(-10) 返回10.0 |
| [floor(x)](https://www.runoob.com/python3/python3-func-number-floor.html) | 返回数字的下舍整数，如math.floor(4.9)返回 4 |
| [log(x)](https://www.runoob.com/python3/python3-func-number-log.html) | 如math.log(math.e)返回1.0,math.log(100,10)返回2.0 |
| [log10(x)](https://www.runoob.com/python3/python3-func-number-log10.html) | 返回以10为基数的x的对数，如math.log10(100)返回 2.0 |
| [max(x1, x2,...)](https://www.runoob.com/python3/python3-func-number-max.html) | 返回给定参数的最大值，参数可以为序列。 |
| [min(x1, x2,...)](https://www.runoob.com/python3/python3-func-number-min.html) | 返回给定参数的最小值，参数可以为序列。 |
| [modf(x)](https://www.runoob.com/python3/python3-func-number-modf.html) | 返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示。 |
| [pow(x, y)](https://www.runoob.com/python3/python3-func-number-pow.html) | x\*\*y 运算后的值。 |
| [round(x [,n])](https://www.runoob.com/python3/python3-func-number-round.html) | 返回浮点数 x 的四舍五入值，如给出 n 值，则代表舍入到小数点后的位数。  **其实准确的说是保留值将保留到离上一位更近的一端。** |
| [sqrt(x)](https://www.runoob.com/python3/python3-func-number-sqrt.html) | 返回数字x的平方根。 |

#### 随机数函数

Python包含以下常用随机数函数：

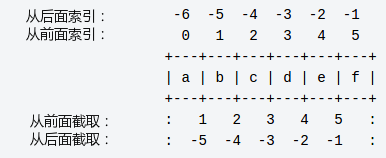
|  |  |
| --- | --- |
| **函数** | **描述** |
| [choice(seq)](https://www.runoob.com/python3/python3-func-number-choice.html) | 从序列的元素中随机挑选一个元素，比如random.choice(range(10))，从0到9中随机挑选一个整数。 |
| [randrange ([start,] stop [,step])](https://www.runoob.com/python3/python3-func-number-randrange.html) | 从指定范围内，按指定基数递增的集合中获取一个随机数，基数默认值为 1 |
| [random()](https://www.runoob.com/python3/python3-func-number-random.html) | 随机生成下一个实数，它在[0,1)范围内。 |
| [seed([x])](https://www.runoob.com/python3/python3-func-number-seed.html) | 改变随机数生成器的种子seed。如果你不了解其原理，你不必特别去设定seed，Python会帮你选择seed。 |
| [shuffle(lst)](https://www.runoob.com/python3/python3-func-number-shuffle.html) | 将序列的所有元素随机排序 |
| [uniform(x, y)](https://www.runoob.com/python3/python3-func-number-uniform.html) | 随机生成下一个实数，它在[x,y]范围内。 |

#### 三角函数

|  |  |
| --- | --- |
| **函数** | **描述** |
| [acos(x)](https://www.runoob.com/python3/python3-func-number-acos.html) | 返回x的反余弦弧度值。 |
| [asin(x)](https://www.runoob.com/python3/python3-func-number-asin.html) | 返回x的反正弦弧度值。 |
| [atan(x)](https://www.runoob.com/python3/python3-func-number-atan.html) | 返回x的反正切弧度值。 |
| [atan2(y, x)](https://www.runoob.com/python3/python3-func-number-atan2.html) | 返回给定的 X 及 Y 坐标值的反正切值。 |
| [cos(x)](https://www.runoob.com/python3/python3-func-number-cos.html) | 返回x的弧度的余弦值。 |
| [hypot(x, y)](https://www.runoob.com/python3/python3-func-number-hypot.html) | 返回欧几里德范数 sqrt(x\*x + y\*y)。 |
| [sin(x)](https://www.runoob.com/python3/python3-func-number-sin.html) | 返回的x弧度的正弦值。 |
| [tan(x)](https://www.runoob.com/python3/python3-func-number-tan.html) | 返回x弧度的正切值。 |
| [degrees(x)](https://www.runoob.com/python3/python3-func-number-degrees.html) | 将弧度转换为角度,如degrees(math.pi/2) ， 返回90.0 |
| [radians(x)](https://www.runoob.com/python3/python3-func-number-radians.html) | 将角度转换为弧度 |

## String（字符串）

索引值以 0 为开始值，-1 为从末尾的开始位置。



加号 **+** 是字符串的连接符， 星号 **\*** 表示复制当前字符串，紧跟的数字为复制的次数。实例如下：

str = 'Runoob' print (str) # 输出字符串

print (str[0:-1]) # 输出第一个到倒数第二个的所有字符

print (str[0]) # 输出字符串第一个字符

print (str[2:5]) # 输出从第三个开始到第五个的字符

print (str[2:]) # 输出从第三个开始的后的所有字符

print (str \* 2) # 输出字符串两次

print (str + "TEST") # 连接字符串



Python 使用反斜杠(\)转义特殊字符，如果你不想让反斜杠发生转义，可以在字符串前面添加一个 r，表示原始字符串

* 1、反斜杠可以用来转义，使用r可以让反斜杠不发生转义。
* 2、字符串可以用+运算符连接在一起，用\*运算符重复。
* 3、Python中的字符串有两种索引方式，从左往右以0开始，从右往左以-1开始。
* 4、Python中的字符串不能改变。

### repr和字符串

* Str()函数可以将数值直接转换成字符串
* Repr()也可以将数值转换为字符串,带有引号的字符串

st=”I will play my file”

print(st)

print(repr(str))

显示结果:

I will play my file

‘I will play my file’

### Input()函数

* Input()函数用于获取用户输入的内容,并返回一个字符串。

### 原始字符串

* 原始字符串以”r”开头,原始字符串不会把把斜线当成转义字符。

### 大小写相关方法

* dir():列出指定类或模块包含的全部内容(包括函数、方法、类、变量等)
* help():查看某个函数或方法的帮助文档。
* title():将每个单词的首字母改大写。
* lower():将整个字符串改为小写。
* upper():将整个字符串改为大写。

### 删除空白

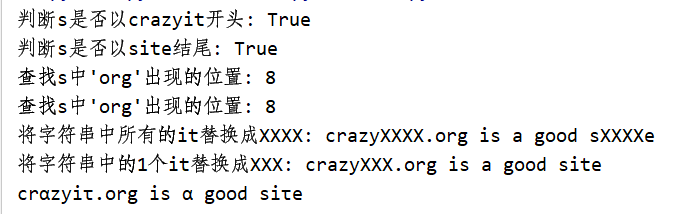
* strip():删除字符串后台的空白
* lstrip():删除字符串前面(左边)的空白
* rstrip():删除字符串后面(右边)的空白

### 查找、替换相关方法

* startswith():判断字符串是否以指定子串开头
* endswith():判断字符串是否以指定子串结尾
* find():查找指定子串在字符串中出现的位置，如果没有找到指定子串，则返回-1
* index():查找指定子串在字符串中出现的位置，如果没有找到指定子串，则引发ValueError错误
* replace():使用指定子串替换字符串中的目标子串
* translate():使用指定的翻译映射表对字符串执行替换

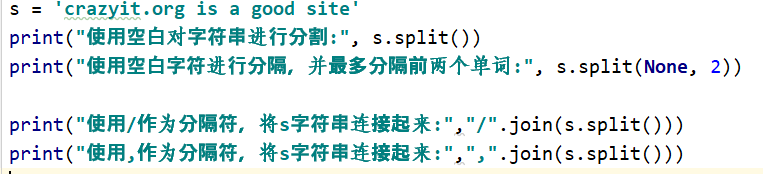


运行结果：

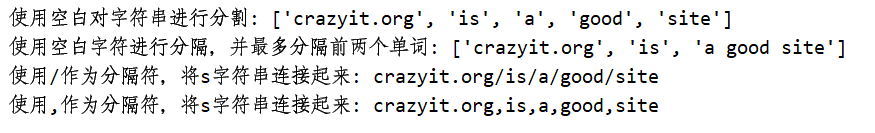


### 分隔、连接方法

* split():将字符串按指定分割符分割成多个短语
* join():将多个短语连接成字符串



运行结果：



### 比较运算符与bool类型

* >：大于，如果运算符前面的值大于后面的值，则返回True,否则返回False;
* >=：大于等于，如果运算符前面的值大于等于后面的值，则返回True,否则返回False
* <：小于，如果运算符前面的值小于后面的值，则返回True,否则返回False
* <=：小于等于，如果运算符前面的值小于等于后面的值，则返回True,否则返回False
* ==：等于，如果运算符前面的值等于后面的值，则返回True,否则返回False
* !=：不等于，如果运算符前面的值不等于后面的值，则返回True,否则返回False
* is：判断两个变更引用的对象是否相同，相同则返回True
* is not：判断两个变更所引用的对象是否不相同，如果不相同则返回True

Python提供了一个全局的id()函数来判断变量所引用的对象的内存地址；

### 逻辑运算符

* and：与，前后两个操作数都是True才返回True,否则返回False
* or：或，只要两个操作数中有一个是True,就返回True，否则返回False
* not：非，只需要一个操作数，如果操作为True，则返回False；如果操作数为False，则返回True

### 三目运算

语法格式：

True\_statements if expression else False\_statements

* 运算规则：先对逻辑表达式expression求值，如果逻辑表达式返回True,则执行并返回True\_statements的值；如果逻辑表达式返回False，则执行并返回False\_statements的值。
* Python支持两种放置方式：
  + 多条语句以英文逗号隔开；每条语句都会执行，程序返回多条语句的返回值组成的元组。

st =print(“crazyit”),’a 大于b’ if a>b else ‘a不大于b’

print(st)

结果：

(None,’a大于b’)

* + 多条语句以分号隔开；每条语句都会执行，程序只返回第一条语句的返回值。
* st =print(“crazyit”); x=20 if a>b else ‘a不大于b’
* print(st)
* print(x)

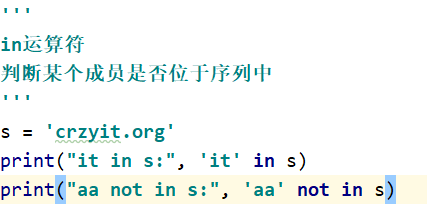
结果：

None

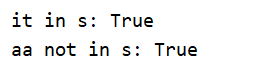
20

### in与not in运算符

* in：用于判断某个成员是否位于序列中
* not in：用于判断某个成员是否不位于序列中



结果：



### 运算符优先级

|  |  |  |
| --- | --- | --- |
| 运算符说明 | Python运算符 | 优先级 |
| 索引运算符 | x[index]或x[index:index2[:index3]] | 18、19 |
| 属性访问 | x.attribute | 17 |
| 乘方 | \*\* | 16 |
| 按位取反 | ~ | 15 |
| 符号运算符 | +或- | 14 |
| 乘、除 | \*、/、//、% | 13 |
| 加、减 | +、- | 12 |
| 位移 | >>、<< | 11 |
| 按位与 | & | 10 |
| 按位异或 | ^ | 9 |
| 按位或 | | | 8 |
| 比较运算符 | ==、!=、>、>=、<、<= | 7 |
| is 运算符 | is、is not | 6 |
| In 运算符 | in、not in | 5 |
| 逻辑非 | not | 4 |
| 逻辑与 | and | 3 |
| 逻辑或 | or | 2 |

### Python转义字符

| **转义字符** | **描述** |
| --- | --- |
| \(在行尾时) | 续行符 |
| \\ | 反斜杠符号 |
| \' | 单引号 |
| \" | 双引号 |
| \a | 响铃 |
| \b | 退格(Backspace) |
| \000 | 空 |
| \n | 换行 |
| \v | 纵向制表符 |
| \t | 横向制表符 |
| \r | 回车 |
| \f | 换页 |
| \oyy | 八进制数，**yy** 代表的字符，例如：**\o12** 代表换行，其中 o 是字母，不是数字 0。 |
| \xyy | 十六进制数，yy代表的字符，例如：\x0a代表换行 |
| \other | 其它的字符以普通格式输出 |

### Python字符串运算符

下表实例变量a值为字符串 "Hello"，b变量值为 "Python"：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **实例** |
| + | 字符串连接 | a + b 输出结果： HelloPython |
| \* | 重复输出字符串 | a\*2 输出结果：HelloHello |
| [] | 通过索引获取字符串中字符 | a[1] 输出结果 **e** |
| [ : ] | 截取字符串中的一部分，遵循**左闭右开**原则，str[0,2] 是不包含第 3 个字符的。 | a[1:4] 输出结果 **ell** |
| in | 成员运算符 - 如果字符串中包含给定的字符返回 True | **'H' in a** 输出结果 True |
| not in | 成员运算符 - 如果字符串中不包含给定的字符返回 True | **'M' not in a** 输出结果 True |
| r/R | 原始字符串 - 原始字符串：所有的字符串都是直接按照字面的意思来使用，没有转义特殊或不能打印的字符。 原始字符串除在字符串的第一个引号前加上字母 **r**（可以大小写）以外，与普通字符串有着几乎完全相同的语法。 | print( r'\n' )  print( R'\n' ) |
| % | 格式字符串 | 请看下一节内容。 |

### Python字符串格式化

|  |  |
| --- | --- |
| **符   号** | **描述** |
| %c | 格式化字符及其ASCII码 |
| %s | 格式化字符串 |
| %d | 格式化整数 |
| %u | 格式化无符号整型 |
| %o | 格式化无符号八进制数 |
| %x | 格式化无符号十六进制数 |
| %X | 格式化无符号十六进制数（大写） |
| %f | 格式化浮点数字，可指定小数点后的精度 |
| %e | 用科学计数法格式化浮点数 |
| %E | 作用同%e，用科学计数法格式化浮点数 |
| %g | %f和%e的简写 |
| %G | %f 和 %E 的简写 |
| %p | 用十六进制数格式化变量的地址 |

格式化操作符辅助指令

|  |  |
| --- | --- |
| **符号** | **功能** |
| \* | 定义宽度或者小数点精度 |
| - | 用做左对齐 |
| + | 在正数前面显示加号( + ) |
| <sp> | 在正数前面显示空格 |
| # | 在八进制数前面显示零('0')，在十六进制前面显示'0x'或者'0X'(取决于用的是'x'还是'X') |
| 0 | 显示的数字前面填充'0'而不是默认的空格 |
| % | '%%'输出一个单一的'%' |
| (var) | 映射变量(字典参数) |
| m.n. | m 是显示的最小总宽度,n 是小数点后的位数(如果可用的话) |

### Python 的字符串内建函数

Python 的字符串常用内建函数如下：

|  |  |
| --- | --- |
| **序号** | **方法及描述** |
| 1 | [capitalize()](https://www.runoob.com/python3/python3-string-capitalize.html) 将字符串的第一个字符转换为大写 |
| 2 | [center(width, fillchar)](https://www.runoob.com/python3/python3-string-center.html) 返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格。 |
| 3 | [count(str, beg= 0,end=len(string))](https://www.runoob.com/python3/python3-string-count.html)返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数 |
| 4 | [bytes.decode(encoding="utf-8", errors="strict")](https://www.runoob.com/python3/python3-string-decode.html) Python3 中没有 decode 方法，但我们可以使用 bytes 对象的 decode() 方法来解码给定的 bytes 对象，这个 bytes 对象可以由 str.encode() 来编码返回。 |
| 5 | [encode(encoding='UTF-8',errors='strict')](https://www.runoob.com/python3/python3-string-encode.html) 以 encoding 指定的编码格式编码字符串，如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace' |
| 6 | [endswith(suffix, beg=0, end=len(string))](https://www.runoob.com/python3/python3-string-endswith.html) 检查字符串是否以 obj 结束，如果beg 或者 end 指定则检查指定的范围内是否以 obj 结束，如果是，返回 True,否则返回 False. |
| 7 | [expandtabs(tabsize=8)](https://www.runoob.com/python3/python3-string-expandtabs.html) 把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是 8 。 |
| 8 | [find(str, beg=0, end=len(string))](https://www.runoob.com/python3/python3-string-find.html) 检测 str 是否包含在字符串中，如果指定范围 beg 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1 |
| 9 | [index(str, beg=0, end=len(string))](https://www.runoob.com/python3/python3-string-index.html)跟find()方法一样，只不过如果str不在字符串中会报一个异常. |
| 10 | [isalnum()](https://www.runoob.com/python3/python3-string-isalnum.html)如果字符串至少有一个字符并且所有字符都是字母或数字则返 回 True,否则返回 False |
| 11 | [isalpha()](https://www.runoob.com/python3/python3-string-isalpha.html)如果字符串至少有一个字符并且所有字符都是字母则返回 True, 否则返回 False |
| 12 | [isdigit()](https://www.runoob.com/python3/python3-string-isdigit.html)如果字符串只包含数字则返回 True 否则返回 False.. |
| 13 | [islower()](https://www.runoob.com/python3/python3-string-islower.html)如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是小写，则返回 True，否则返回 False |
| 14 | [isnumeric()](https://www.runoob.com/python3/python3-string-isnumeric.html)如果字符串中只包含数字字符，则返回 True，否则返回 False |
| 15 | [isspace()](https://www.runoob.com/python3/python3-string-isspace.html)如果字符串中只包含空白，则返回 True，否则返回 False. |
| 16 | [istitle()](https://www.runoob.com/python3/python3-string-istitle.html)如果字符串是标题化的(见 title())则返回 True，否则返回 False |
| 17 | [isupper()](https://www.runoob.com/python3/python3-string-isupper.html)如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大写，则返回 True，否则返回 False |
| 18 | [join(seq)](https://www.runoob.com/python3/python3-string-join.html)以指定字符串作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串 |
| 19 | [len(string)](https://www.runoob.com/python3/python3-string-len.html)返回字符串长度 |
| 20 | [ljust(width[, fillchar])](https://www.runoob.com/python3/python3-string-ljust.html)返回一个原字符串左对齐,并使用 fillchar 填充至长度 width 的新字符串，fillchar 默认为空格。 |
| 21 | [lower()](https://www.runoob.com/python3/python3-string-lower.html)转换字符串中所有大写字符为小写. |
| 22 | [lstrip()](https://www.runoob.com/python3/python3-string-lstrip.html)截掉字符串左边的空格或指定字符。 |
| 23 | [maketrans()](https://www.runoob.com/python3/python3-string-maketrans.html)创建字符映射的转换表，对于接受两个参数的最简单的调用方式，第一个参数是字符串，表示需要转换的字符，第二个参数也是字符串表示转换的目标。 |
| 24 | [max(str)](https://www.runoob.com/python3/python3-string-max.html)返回字符串 str 中最大的字母。 |
| 25 | [min(str)](https://www.runoob.com/python3/python3-string-min.html)返回字符串 str 中最小的字母。 |
| 26 | [replace(old, new [, max])](https://www.runoob.com/python3/python3-string-replace.html)把 将字符串中的 str1 替换成 str2,如果 max 指定，则替换不超过 max 次。 |
| 27 | [rfind(str, beg=0,end=len(string))](https://www.runoob.com/python3/python3-string-rfind.html)类似于 find()函数，不过是从右边开始查找. |
| 28 | [rindex( str, beg=0, end=len(string))](https://www.runoob.com/python3/python3-string-rindex.html)类似于 index()，不过是从右边开始. |
| 29 | [rjust(width,[, fillchar])](https://www.runoob.com/python3/python3-string-rjust.html)返回一个原字符串右对齐,并使用fillchar(默认空格）填充至长度 width 的新字符串 |
| 30 | [rstrip()](https://www.runoob.com/python3/python3-string-rstrip.html)删除字符串字符串末尾的空格. |
| 31 | [split(str="", num=string.count(str))](https://www.runoob.com/python3/python3-string-split.html)num=string.count(str)) 以 str 为分隔符截取字符串，如果 num 有指定值，则仅截取 num+1 个子字符串 |
| 32 | [splitlines([keepends])](https://www.runoob.com/python3/python3-string-splitlines.html)按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。 |
| 33 | [startswith(substr, beg=0,end=len(string))](https://www.runoob.com/python3/python3-string-startswith.html)检查字符串是否是以指定子字符串 substr 开头，是则返回 True，否则返回 False。如果beg 和 end 指定值，则在指定范围内检查。 |
| 34 | [strip([chars])](https://www.runoob.com/python3/python3-string-strip.html)在字符串上执行 lstrip()和 rstrip() |
| 35 | [swapcase()](https://www.runoob.com/python3/python3-string-swapcase.html)将字符串中大写转换为小写，小写转换为大写 |
| 36 | [title()](https://www.runoob.com/python3/python3-string-title.html)返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle()) |
| 37 | [translate(table, deletechars="")](https://www.runoob.com/python3/python3-string-translate.html)根据 str 给出的表(包含 256 个字符)转换 string 的字符, 要过滤掉的字符放到 deletechars 参数中 |
| 38 | [upper()](https://www.runoob.com/python3/python3-string-upper.html)转换字符串中的小写字母为大写 |
| 39 | [zfill (width)](https://www.runoob.com/python3/python3-string-zfill.html)返回长度为 width 的字符串，原字符串右对齐，前面填充0 |
| 40 | [isdecimal()](https://www.runoob.com/python3/python3-string-isdecimal.html) 检查字符串是否只包含十进制字符，如果是返回 true，否则返回 false。 |

## 列表和元组

### 创建列表和元组

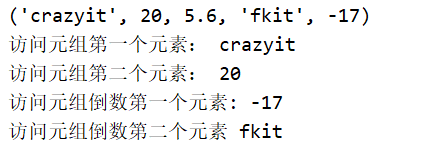
创建列表和元组的语法  
区别：  
 创建列表使用方括号  
 创建元组使用圆括号  
  
创建列表的语法格式如下:  
[ele1,ele2,ele3,...]  
创建元组的语法格式如下：  
(ele1,ele2,ele3,...)

*# 使用方括号定义列表*my\_list = [**'crazyit'**, 20, **'Python'**]  
print(my\_list)  
*# 使用圆括号定义元组*my\_tuple = (**'crazyit'**, 20, **'Python'**)  
print(my\_tuple)

### 列表和元组通用方法

* 通过索引访问元素，索引从0开始

a\_tuple = (**'crazyit'**, 20, 5.6, **'fkit'**, -17)  
print(a\_tuple)  
print(**"访问元组第一个元素："**, a\_tuple[0])  
print(**"访问元组第二个元素："**, a\_tuple[1])  
print(**"访问元组倒数第一个元素:"**, a\_tuple[-1])  
print(**"访问元组倒数第二个元素"**, a\_tuple[-2])



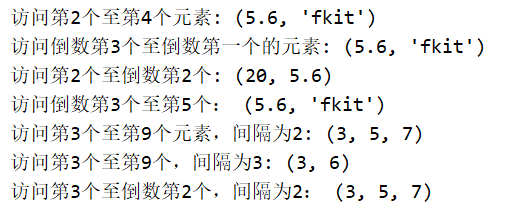
* 子序列

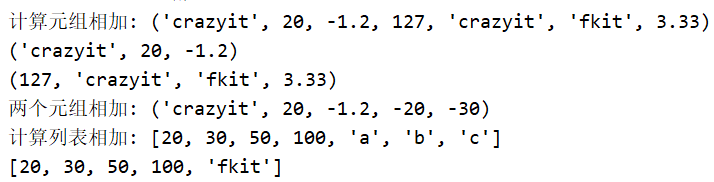
语法格式：

[start:end:step]

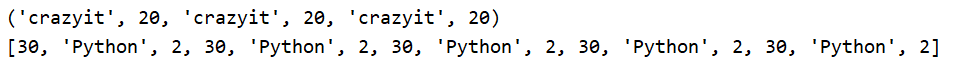
* + 上面语法中start、end两个索引值可使用正数或负数，负数表示从倒数开始,step表示步长

*# 子序列*print(**"访问第2个至第4个元素:"**, a\_tuple[2:4])  
print(**"访问倒数第3个至倒数第一个的元素:"**, a\_tuple[-3:-1])  
print(**"访问第2个至倒数第2个:"**, a\_tuple[1:-2])  
print(**"访问倒数第3个至第5个："**, a\_tuple[-3:4])  
b\_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)  
print(**"访问第3个至第9个元素，间隔为2:"**, b\_tuple[2:8:2])  
print(**"访问第3个至第9个，间隔为3:"**, b\_tuple[2:8:3])  
print(**"访问第3个至倒数第2个，间隔为2："**, b\_tuple[2:-2:2])



print(**"+++++++++++++++加法+++++++++++++++"**)  
a\_tuple = (**'crazyit'**, 20, -1.2)  
b\_tuple = (127, **'crazyit'**, **'fkit'**, 3.33)  
*# 计算元组相加*sum\_tuple = a\_tuple + b\_tuple  
print(sum\_tuple)  
print(a\_tuple)  
print(b\_tuple)  
*# 两个元组相加*print(a\_tuple + (-20, -30))  
a\_list = [20, 30, 50, 100]  
b\_list = [**'a'**, **'b'**, **'c'**]  
*# 计算列表相加*sum\_list = a\_list + b\_list  
print(sum\_list)  
print(a\_list + [**'fkit'**])  


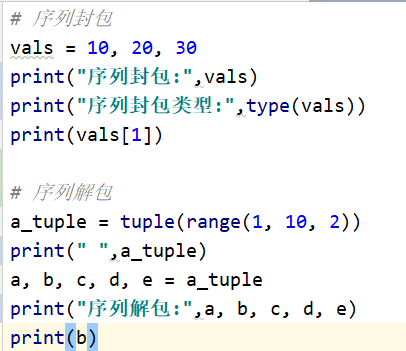
print(**"++++++++++++乘法++++++++++++"**)  
a\_tuple = (**'crazyit'**, 20)  
*# 执行乘法*mul\_tuple = a\_tuple \* 3  
print(mul\_tuple)  
a\_list = [30, **'Python'**, 2]  
mul\_list = a\_list \* 5  
print(mul\_list)



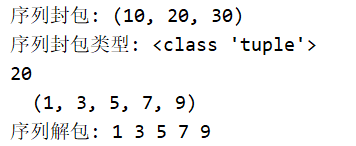
print(**"++++++++长度、最大值和最小值+++++++++++++"**)  
*# 元素都是数值的元组*a\_tuple = (29, 49, 13, 22, 33, 10, 4, 102)  
print(**"元素都是数值，最大值："**, max(a\_tuple))  
print(**"元素都是数值，最小值:"**, min(a\_tuple))  
print(**"元素都是数值，长度:"**, len(a\_tuple))  
b\_tuple = (  
 **'crazyit'**, **'Codes'**, **'Hibernate'**, **'struts'**, **'Spring'**, **'Appium'**, **'Ruby'**, **'Seleinum'**, **'Java'**, **'C++'**, **'Python'**, **'fkit'**)  
print(**"元素都是字符串，最大值:"**, max(b\_tuple))  
print(**"元素都是字符串，最小值:"**, min(b\_tuple))  
print(**"元素都是字符串，长度:"**, len(b\_tuple))

### 序列解包和封包

* 程序把多个值赋给一个变量时，Python会自动将多个值封装成元组，称为序列封包
* 程序允许将序列(元组和列表)直接赋值给多个变量，此时序列的各元素会被依次赋值给每个变量(要求序列的元素个数和变量个数相等)，称为序列解包



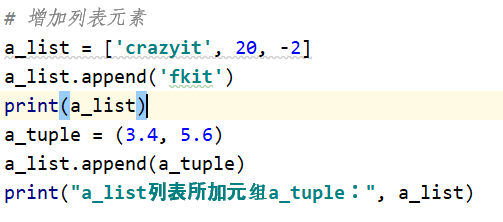
结果：



### 创建列表

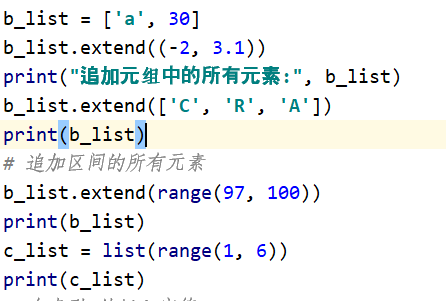
*# 创建列表*a\_tuple = (**'crazyit'**, 20, -1.2)  
print(**"将元组转换成列表:"**, list(a\_tuple))  
a\_range = range(1, 5)  
print(a\_range)  
b\_list = list(a\_range)  
print(**"将区间转换成列表:"**, b\_list)  
c\_list = list(range(4, 20, 3))  
print(c\_list)

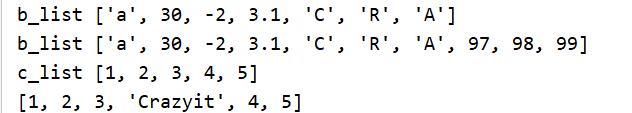
### 列表添加元素append()



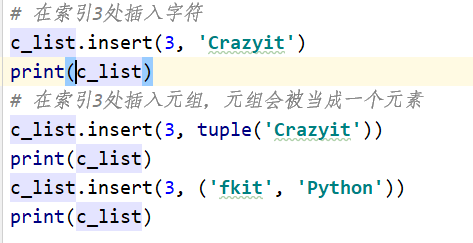


### 列表追加元素extend()

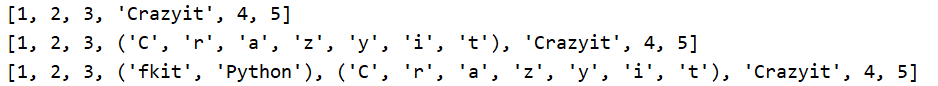




### 列表插入字符insert()

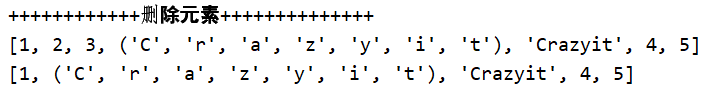


结果:

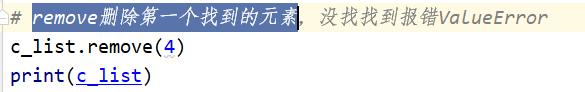


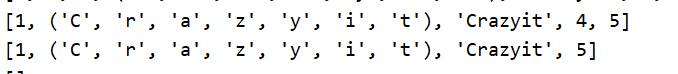
### 删除元素del

*# 删除元素*print(**"++++++++++++删除元素++++++++++++++"**)  
**del** c\_list[3]  
print(c\_list)  
**del** c\_list[1:3]  
print(c\_list)  
*# del c\_list 清空列表*



### remove删除第一个找到的元素





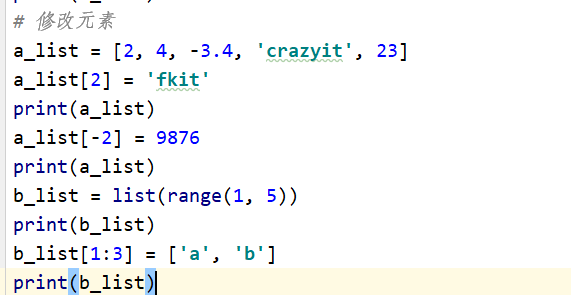
说明:第一列为删除前的元素列表;第二列为删除元素’4’之后的列表

### 清空列表clear()

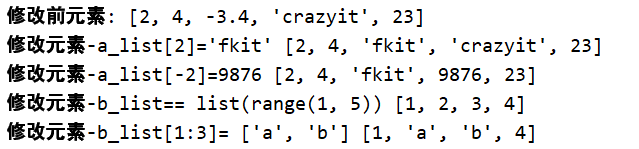
c\_list.clear()  
print(c\_list)

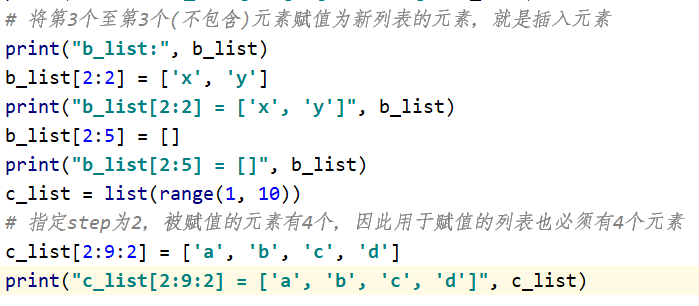


### 修改元素

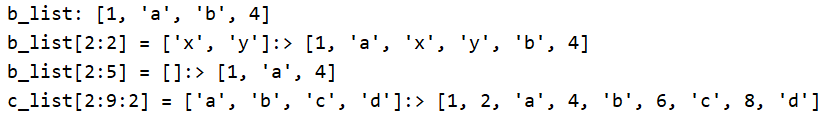


运行结果:



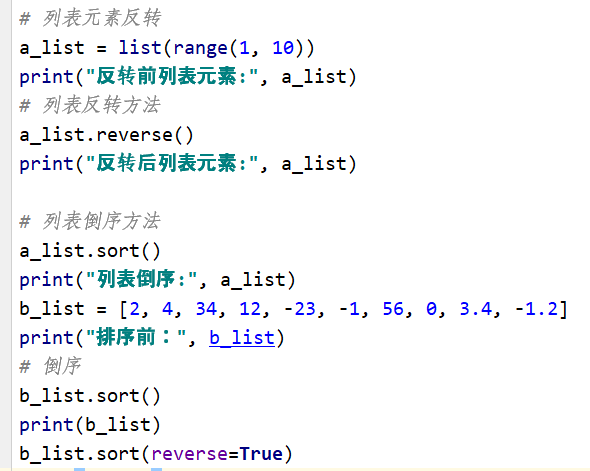


结果:

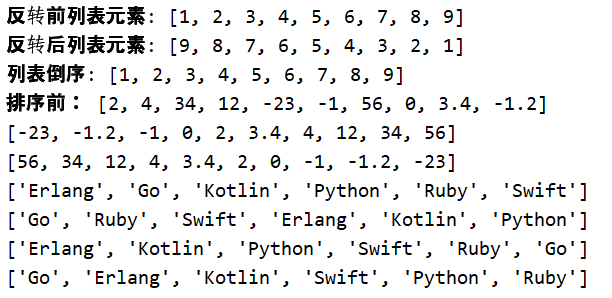


### 其他方法

count():用于统计列表中某个元素出现的次数  
index():用于判断某个元素在列表中出现的位置  
pop():用于寺将列表当成“栈”使用，实现元素出栈功能  
reverse():用于将列表元素反向存放  
sort():用于对列表元素排序



结果:



### reversed-反转序列

该方法是将列表,元组,区间进行序列反转

### sorted-排序,默认按照从小到大

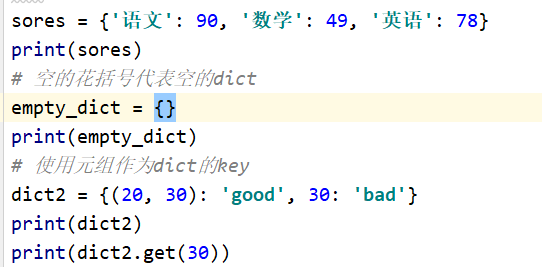
reverse=True(从大到小排序)

c\_list.sort(key=len, reverse=True)

## 字典

### 常用方法

### 创建字典



注意：元组可以作为dict的key,但列表不能作为dict的key.这是因为dict要求key必须是不可变类型，但列表是可变类型

* 使用关键字创建字典

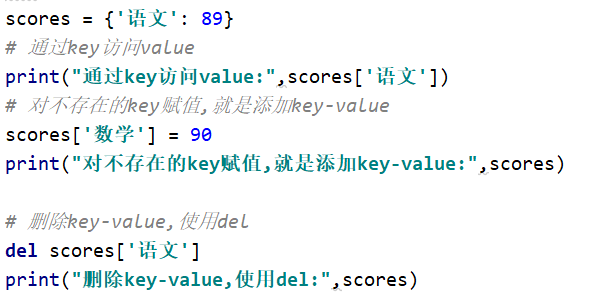
dict1=dict(spinach=1.39,cabbage=2.59’)

print(dict1)

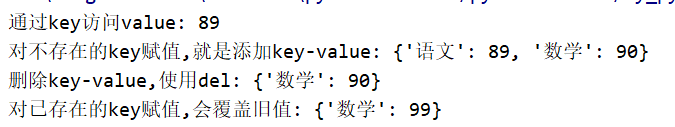
结果：{‘spinach’:1.39,’cabbage’:2.59}

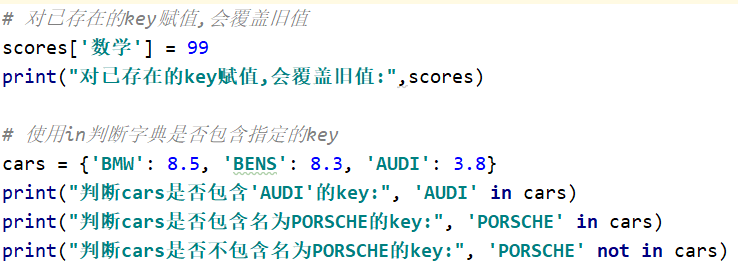
### 字典的基本用法

* 通过key访问value
* 通过key添加key-value对
* 通过key删除key-value对
* 通过key修改key-value对
* 通过key判断指定key-value对

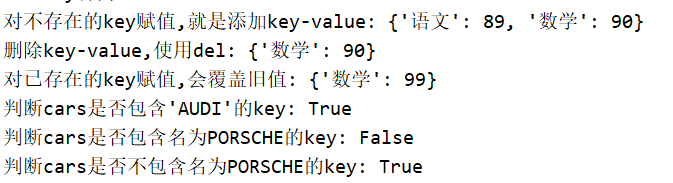


结果：





结果：

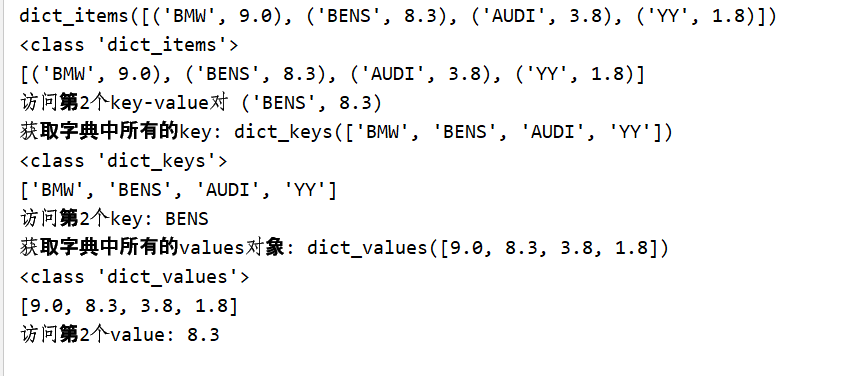


### 字典的常用方法

* clear():用于清空字典中所有的key-value对
* get()方法:用来获取value

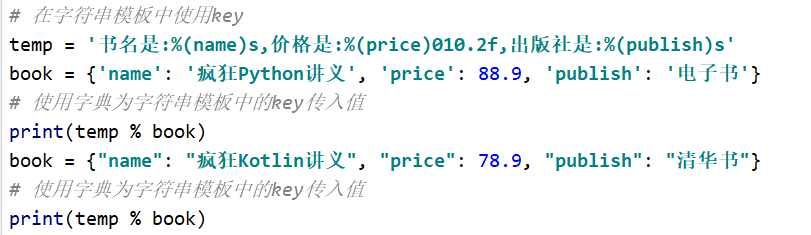
使用get()方法访问不存在的key时,不会报错,但返回None

* update():更新已有的key-value对,如果不存在,则进行添加
* items():用于获取字典中所有的key-value对,返回dict\_items
* keys():用于获取字典所有的的key,返回dict\_keys
* values():用于获取字典所有的value,返回dict\_values
* ims = cars.items()  
  print(ims)  
  print(type(ims))  
  print(list(ims))  
  print(**"访问第2个key-value对"**, list(ims)[1])  
  kys = cars.keys()  
  print(**"获取字典中所有的key:"**, kys)  
  print(type(kys))  
  print(list(kys))  
  *# 访问第2个key*print(**"访问第2个key:"**, list(kys)[1])  
  vals = cars.values()  
  print(**"获取字典中所有的values对象:"**, vals)  
  print(type(vals))  
  print(list(vals))  
  print(**"访问第2个value:"**, list(vals)[1])



* pop():用于获取指定key对应的value,并删除这个key-value对
* popitem():用于随机弹出字典中的一个key-value对,并删除
* setdefault():用于根据key来获取value,如果key不存在,则设置一个默认的value并返回,然后再返回该value,如果存在则直接返回value
* fromkeys():使用给定的多个key创建字典,key对应的value都是None

### 字典格式化字符串





## 流程控制

### If分支结构

第一种形式:

If expression :

Statement…

第二种形式:

If expression :

Statement…

else :

Statement…

第三种形式

If expresseion :

Statement…

elif expression :

Statements…

else :

statements…

说明:False,None,0,””,[],{},()会被当作False处理

Pass语句就是空语句,有时候程序需要一个位,放一条语句,但又不希望这条语句做任何事情,此时就可以通过pass语句来实现,通过pass语句,可以使程序更加完整.

### While循环循环

语法格式:

[init\_statements]

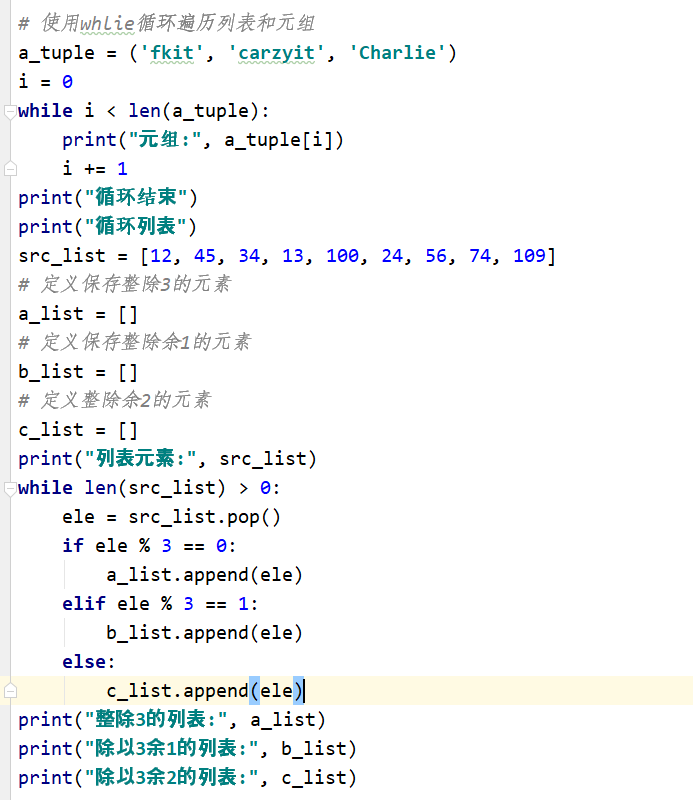
While test\_expression:

Body\_statements

[iteration\_statements]

count\_i = 0  
**while** count\_i < 10:  
 print(**"count\_i:"**, count\_i)  
 count\_i += 1  
print(**"循环结束"**)

循环遍历列表和元组



### For-in循环

s\_max = input(**"请输入您想计算的阶乘:"**)  
mx = int(s\_max)  
reslut = 1  
**for** num **in** range(1, mx + 1):  
 reslut \*= num  
print(reslut)

* for-in循环遍历列表和元组

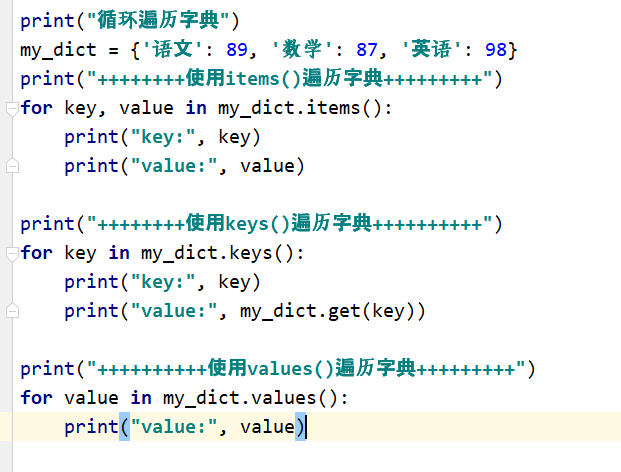
a\_tuple = (**'crazyit'**, **'fkit'**, **'Charlie'**)  
**for** ele **in** a\_tuple:  
 print(**"当前元素是:"**, ele)

#遍历数组

a\_list = [330, 1.4, 50, **'crazyit'**, -3.5]  
**for** num **in** range(0, len(a\_list)):  
 print(**"第%d个元素是:%s"** % (num, a\_list[num]))

* 遍历字典
* Items():返回字典所有key-value对的列表.
* Keys():返回字典中所有key的列表
* Values():返回返回字典所有value的列表

print(**"循环遍历字典"**)  
my\_dict = {**'语文'**: 89, **'数学'**: 87, **'英语'**: 98}  
print(**"++++++++使用items()遍历字典+++++++++"**)  
**for** key, value **in** my\_dict.items():  
 print(**"key:"**, key)  
 print(**"value:"**, value)  
  
print(**"++++++++使用keys()遍历字典++++++++++"**)  
**for** key **in** my\_dict.keys():  
 print(**"key:"**, key)  
 print(**"value:"**, my\_dict.get(key))  
  
print(**"++++++++++使用values()遍历字典+++++++++"**)  
**for** value **in** my\_dict.values():  
 print(**"value:"**, value)



### for表达式

格式:

[表达式 for 循环计数器 in 可迭代对象]

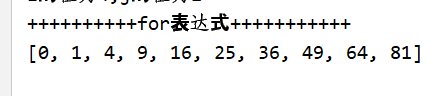
For表达式与普通for循环的区别:

* 在for关键字之前定义一个表达式,该表达式通常会包含循环计数器
* for表达式没有循环体,因此不需要冒号

for表达式最终返回的是列表,因此for表达式也被称为列表推导式.

print(**"++++++++++for表达式+++++++++++"**)  
a\_range = range(10)  
a\_list = [x \* x **for** x **in** a\_range]  
print(a\_list)

运行结果:



还可以在for表达式后面添加if条件,这样for表达式将只迭代那些符合条件的元素,例如:

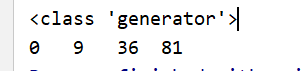
b\_list = [x \* x **for** x **in** a\_range **if** x % 2 == 0]  
print(b\_list)



For表达式使用圆括号,返回的是生成器,称为生成器推导式.

c\_generator = (x \* x **for** x **in** a\_range **if** x % 3 == 0)

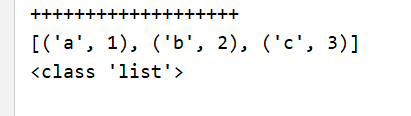
print(type(c\_generator))  
**for** i **in** c\_generator:  
 print(i,end=”\t)



For表达式,for之前的为循环体,for之后的循环条件

### 常用工具函数zip

a = [**'a'**, **'b'**, **'c'**]  
b = [1, 2, 3]  
c = [x **for** x **in** zip(a, b)]  
print(c)  
print(type(c))



# 函数

## 定义函数

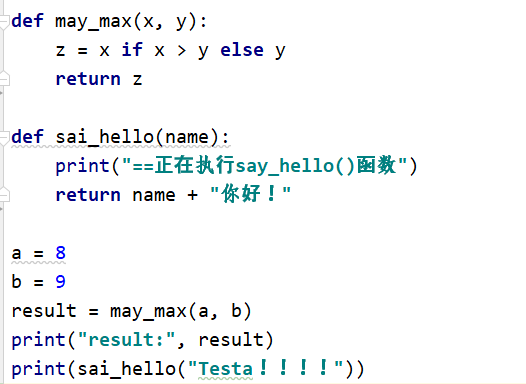
格式：

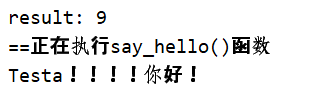
def 函数名[形参列表]:

//由零条到多条可执行语句组成的函数

[return [返回值]]

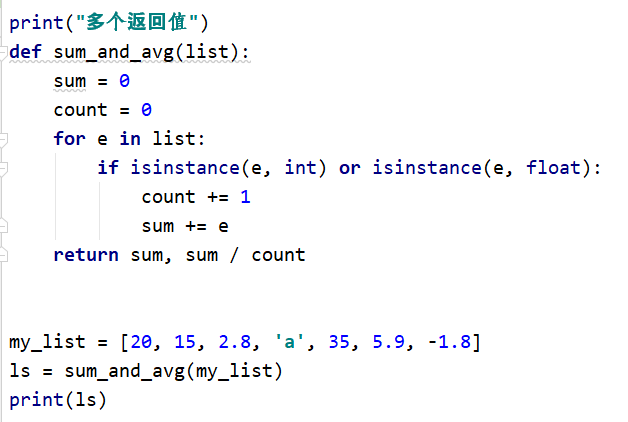
* 函数名:合法的标识符,每个单词的字母全部小写,单词与单词之间使用下画线隔开。
* 开能列表:用于定义函数可以接收的参数。多个形参用逗号隔开

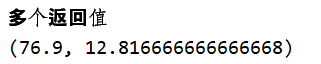




## 多个返回值

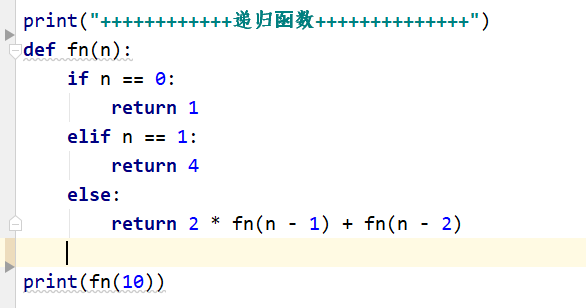
* 程序需要有多个返回值时，可以将多个值包装成列表进行返回。





## 递归函数

* 在一个函数体内，调用它本身，被称为递归函数。递归函数包含了一种隐式的循环，它会重复执行某段代码，但这种重复执行无须循环控制。

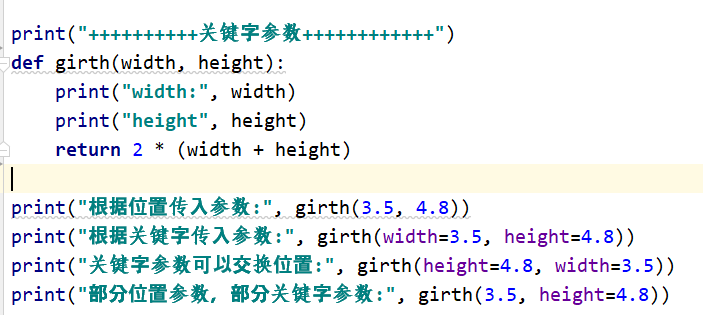


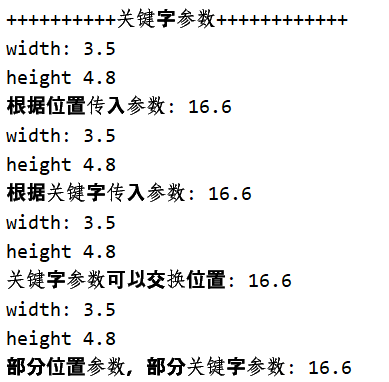


## 函数的参数

### 关键字参数

* 按照形参位置传入的参数被称为位置参数。
* 按照参数名来传入参数值被称为关键字参数。



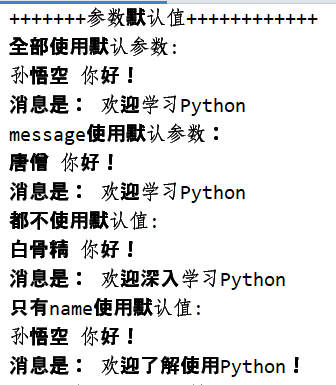


### 参数默认值

* 语法：

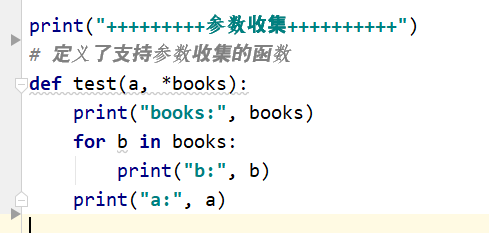
形参名=默认值

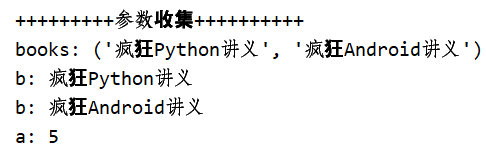




### 参数收集

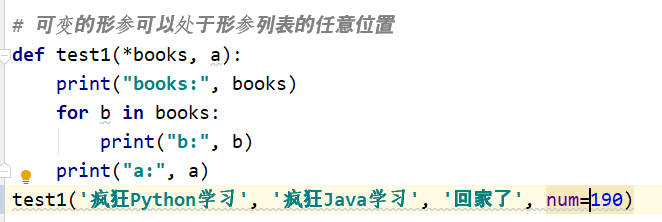
* Python允许在形参面前添加一个星号(\*),表示该参数可接收多个参数值，多个参数值被当成元组传入

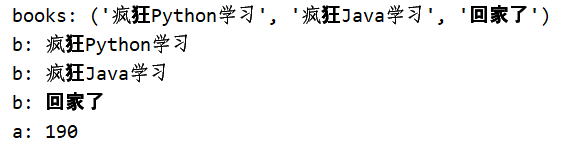




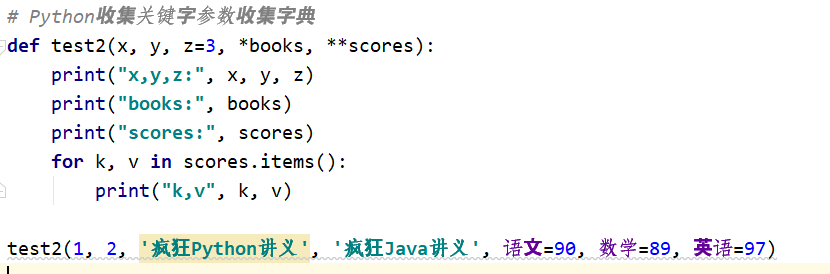
* 可变的形参收集

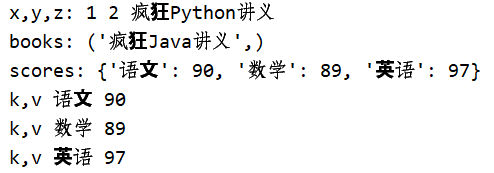
如果第一个参数为可变的形参，由于该参数可接收个数不等的参数值，因此后面的参数传入必须使用关键字参数



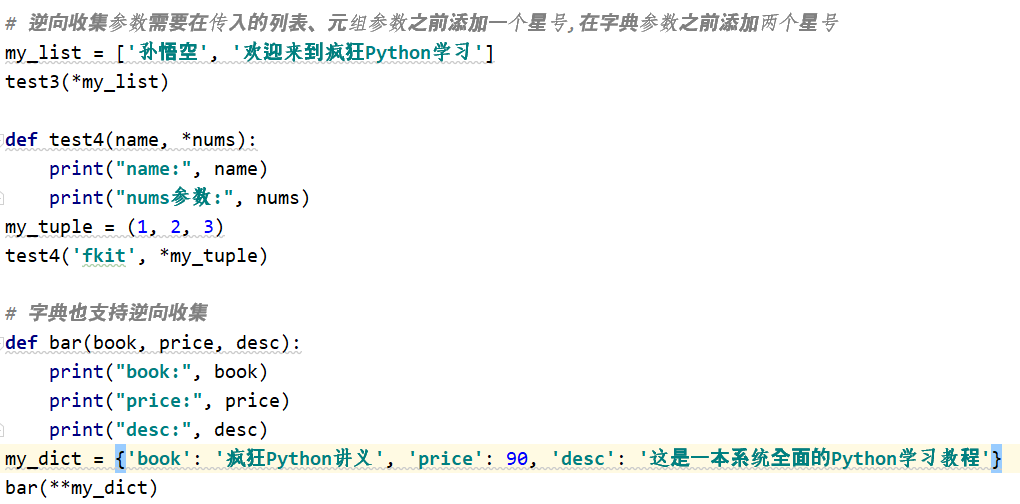


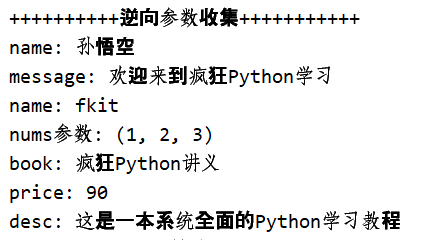
* 关键字参数收集，在形参数前面加两个星号(\*\*)





* 逆向参数收集，在程序已有列表、元组、字典等对象的前提下，把它们的元素拆开后传给函数的参数





### 变量的作用域

* 局部变量:在函数中定义的变量，包括参数，称为局部变量
* 全局变量:在函数外，全局范围内定义的变量，称为全局变量

Python提供了三个工具函数来获取指定范围内的”变量字典”

* globals():该函数返回全局变量范围内所有变量组成的”变量字典”
* locals():该函数返回当前局部范围内所有变量组成的”变量字典”
* vars(object):获取在指定对象范围内所有变量组成的”变量字典”,如果不传入object参数,vars()和locals()的作用完全相同。

globals()和locals()的特点：

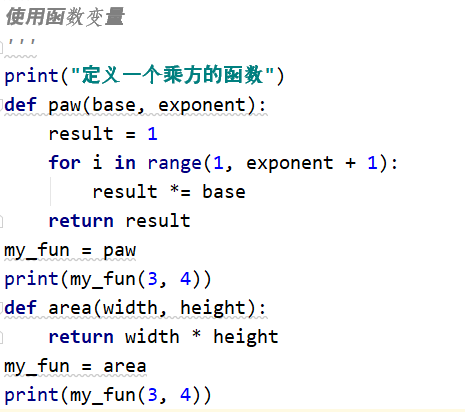
* locals()总是获取当前局部范围内所有变量组成的”变量字典”,如果在全局范围内调用locals()函数，同样获取全局范围内所有变量组成的”变量字典”
* 声明全局变量-global

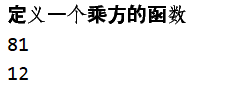
### 局部函数

* 在函数体内定义函数，称为局部函数，局部函数对外隐藏，局部函数只能在其封闭函数内有效，其封闭函数也可以返回局部函数，以便程序在其他作用域中使用局部函数。
* nonlocal用于声明访问当前函数所在函数内的局部变量

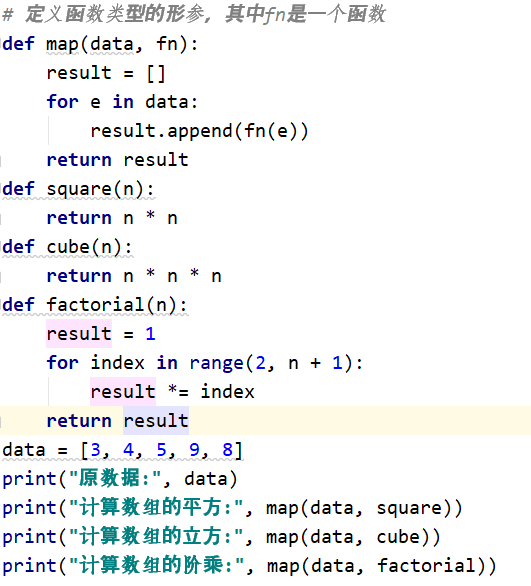
### 函数高级内容

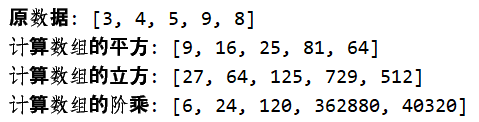
* 使用函数变量
  + Python函数也是一种值,所有函数都是Function对象。



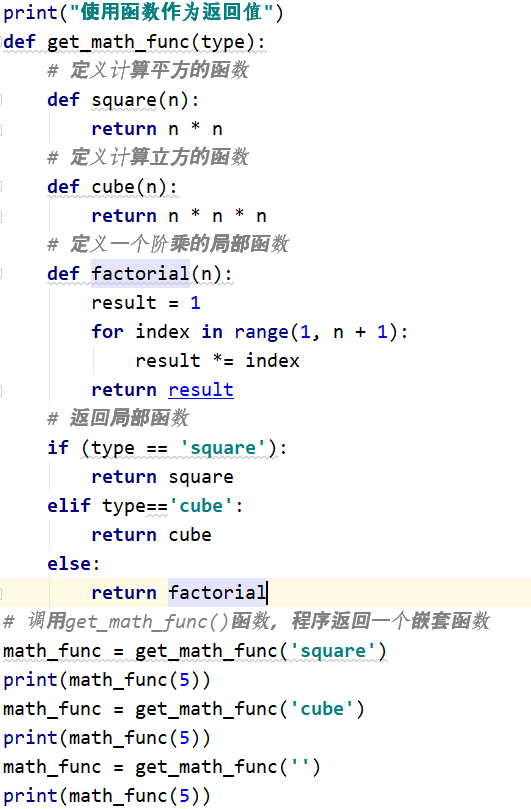


* 使用函数作为形参



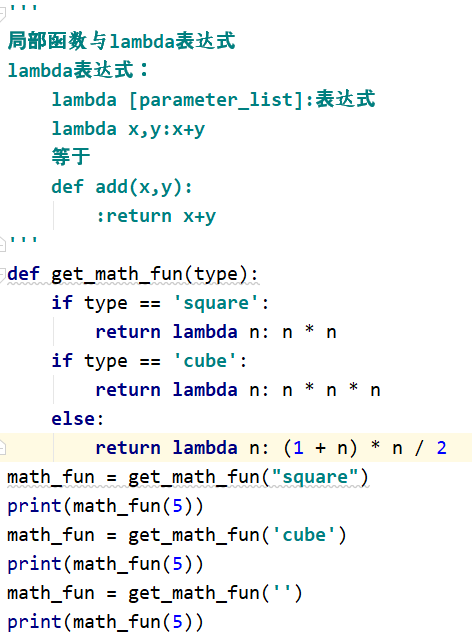


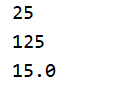
* 使用函数作为返回值





### 局部函数与lambda表达式



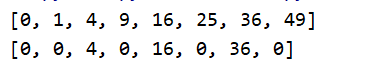
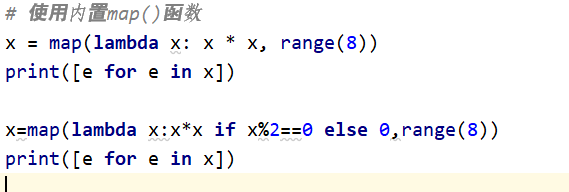


### lambda函数表达式

语法格式:

lambda [parameter\_list]:表达式

* lambda表达式必须使用lambda关键字
* 在lambda关键字之后，冒号左边的是参数列表，可以没有参数，也可以有多个参数。如果有多个参数，则需要用逗号隔开，冒号右边是该lambda表达式的返回值。
* 表达式：lambda x,y:x+y
* 对于单行函数，使用lambda表达式可以省去定义函数的过程
* 对于不需要多次复用的函数，使用lambda表达式可以在用完之后立即释放，提高了性能



# 类和对象

## 定义类

Python定义类的简单语法如下：

class 类名:

执行语名

零到多个类变量

零到多个方法…

* 类名只要是一个合法的标识符即可,类名必须是由一个或多个有意义的单词连缀而成，每个单词首字母大写,其他字母全部小写,单词与单词之间不要使用任务分隔符
* Python是一门动态语言,它可以动态增加或删除包含的类变量—程序在实体中为新变量赋值就是增加类变量,程序也可以在任何地方为已有的类增加类变量;程序可通过del语句删除已有的类变量
* Python对象的实例也可以动态的增加或删除—只要对新实例变量赋值就是增加实例变量,程序也可通过del语句删除已有对象的实例变量。
* 类定义的方法默认是实例方法，定义实例方法的方法与定义函数的方法基本相同,只是实例方法的第一个参数 会被绑定到方法的调用者(该类的实例)—因此实例方法至少应该定义一个self参数
* 实例方法中,\_\_init\_\_方法，这个方法称为构造方法，用于构造该类的对象,Python通过调用该方法返回该类的对象(无须使用new)
* Python为自动为类定义一个包含self参数的默认构造方法

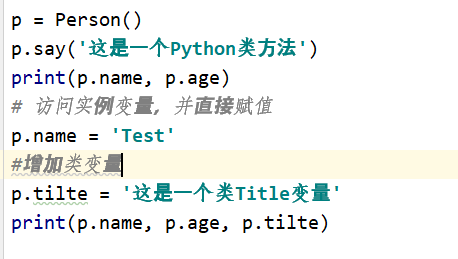


## 对象的产生和使用

* 调用类的构造方法，返回一个类对象

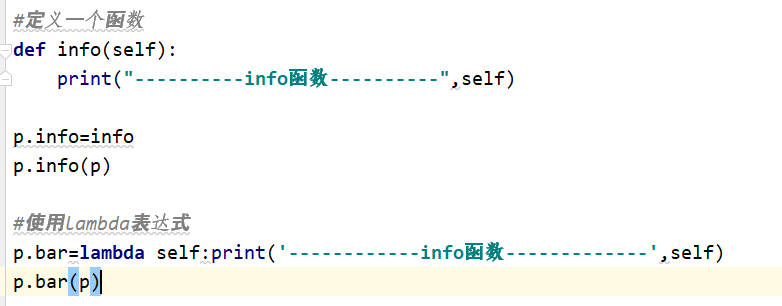
p=Person()

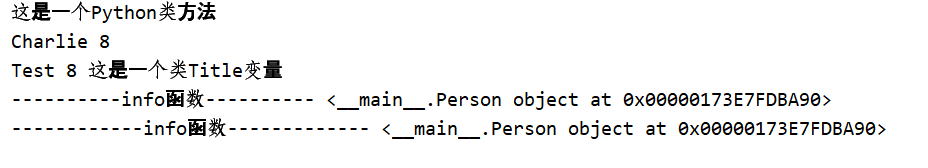
* 操作对象的实例变量(包括访问实例变量的值、添加实例变量、删除实例变量)
* 操作对象的方法



## 对象的动态性

* 动态添加方法,Python不会自动将方法调用者绑定到它们的第一个参数,因此程序需要手动为第一个参数传入参数值





* 如果希望动态增加的方法也能自动绑定到第一个参数,则可借助于types模块下的MethodType进行包装.

