```
3 * 4, 3 + 4, 3 - 4, 3 / 4               #==> 12, 7, -1, 0.75
3 ** 4, 3 // 4, 3 % 4                    #==> 81, 0, 3
4 > 3, 4 >= 3, 3 == 3.0, 3 != 4, 3 <= 4 #==> True, True, True, True, True
# 运算顺序: 括号, **, {* / // %}, {+ -}, {== != <= < > >=}
min(3, 4), max(3, 4), abs(-10)           #==> 3, 4, 10
sum([1, 2, 3])  # [1, 2, 3]是一个列表      #==> 6

type(3), type(3.0), type("myVariable")   #==> class 'int', class 'float',
                                         #    class 'str'
int("4"+"0"), float(3), str(1 / 2)       #==> 40, 3.0, '0.5'

"double quotes: ', escaped \" \\ \'"     #==> double quotes: ', escaped " \ '
'it\'s "similar" in single quotes '      #==> it's "similar" in single quotes

ord("A"), chr(66)                        #==> 65, 'B'
string = "hello"
# 下列语句也适用于列表
len(string)                              #==> 5
string[0], string[4]    # 得到字符         #==> "h", "o"
string[1:3]             # 得到一个子字符串  #==> "el"
string[:2], string[2:]  # l/r 子字符串     #==> "he", "llo"
string[-1], string[-2:] # 负索引          #==> "o", "lo"
"con" + "cat" + "enation " + str(123)    #==> "concatenation 123"
"boo" * 2                                #==> "booboo"

getLineOfInputAsString = input()         #==>  读入输入 (或者 EOF 错误)
print("takes", 0, "or more arguments")   #==> takes 0 or more arguments
print("using", "custom", "sep", sep=".") #==> using.custom.sep
print("no", "newline", end="bye")        #==> no newlinebye

not True, False or True, False and True  #==> False, True, False
# 运算顺序: 括号, {== !=}, not, and, or

if booleanCondition:
   x                       # 缩进主体
   x                       # 主体中的每一行缩进程度一致
elif anotherCondition:     # elif不必须出现, 使用次数无限制
   x                       # 缩进
else:                      # 可自行选择
   x                       # 缩进

while booleanCondition:
   x                       # 主体
   break                   # 跳出循环 (可选)
   continue                # 从下一次迭代重新开始(可选)

for indexVariable in range(low, highPlus):
   print(indexVariable)                  #==> low, low+1, ..., highPlus-1
# "for item in listOrString:" forall/foreach 循环
# break, continue可以用在for循环
```

```
def nameOfNewFunction(argument1, argument2):
    x                       # 主体
    return y                # (可选;如果你不返还,函数默认返还None)

def remember(bar):          # 写入全局变量
    global saveBar          # 在调用foo(3)后, saveBar = 3
    saveBar = bar           # 即使超出函数的范围

# 这些“切割”指令在列表和range()中十分相似
"0123456789"[::2]           # 切割           #==> "02468"
"0123456789"[::-1]          # 颠倒           #==> "9876543210"
"0123456789"[6:3:-1]                         #==> "654"

x += 1          # 也可以是 -=, /=, *=, %=, **=, //=. Python没有C++-的设定"x++" # 多#
x, y = y, x     # 多个指定
3 < x < 5       # 与"(3 < x) and (x < 5)"一致。可以 连接{< <= > >= == != is}

import math                 # 输入一个模块,使用“.” (英文句号) 来使用math下的函数
print(math.sqrt(2))
from math import sqrt       # 输入一个模块,无需“.”使用方程
print(sqrt(2))
# math模块还有以下函数: pi, exp, log, sin, cos, tan, ceil, floor等等

list = ['zero', 'one', 'two']
list.index('one')                            #==> 1
list.index('three')                          #==> 导致错误
'three' in list, 'zero' in list              #==> False, True
list.count('two')                            #==> 1
del list[1]      # 列表变成 ['zero', 'two']
"string" in "superstring"                    #==> True
"superstring".index("string")                #==> 5

# 其他列表运算函数: append(item), insert(item, index), extend(list),
# remove(value), pop(), pop(index), reverse(), sort()等等

# 一些字符串函数: capitalize(), lower/upper(), islower/isupper(),
# isalpha/isdigit(), center/ljust/rjust(width, fillChar), strip(), split(),
# splitlines(), endswith/startswith(string), find(string), replace(old, new),
# 等等

myList = [11, 99]
actuallyTheSameList = myList # 不是彻底地复制!只是复制了引用
myList is actuallyTheSameList                      #==> True
realCopy = myList[:]         # 或者list(myList), copy.copy(myList), deepcopy
realCopy is myList                           #==> False
```