Python 程序开发技术

---第一章:python程序设计基础

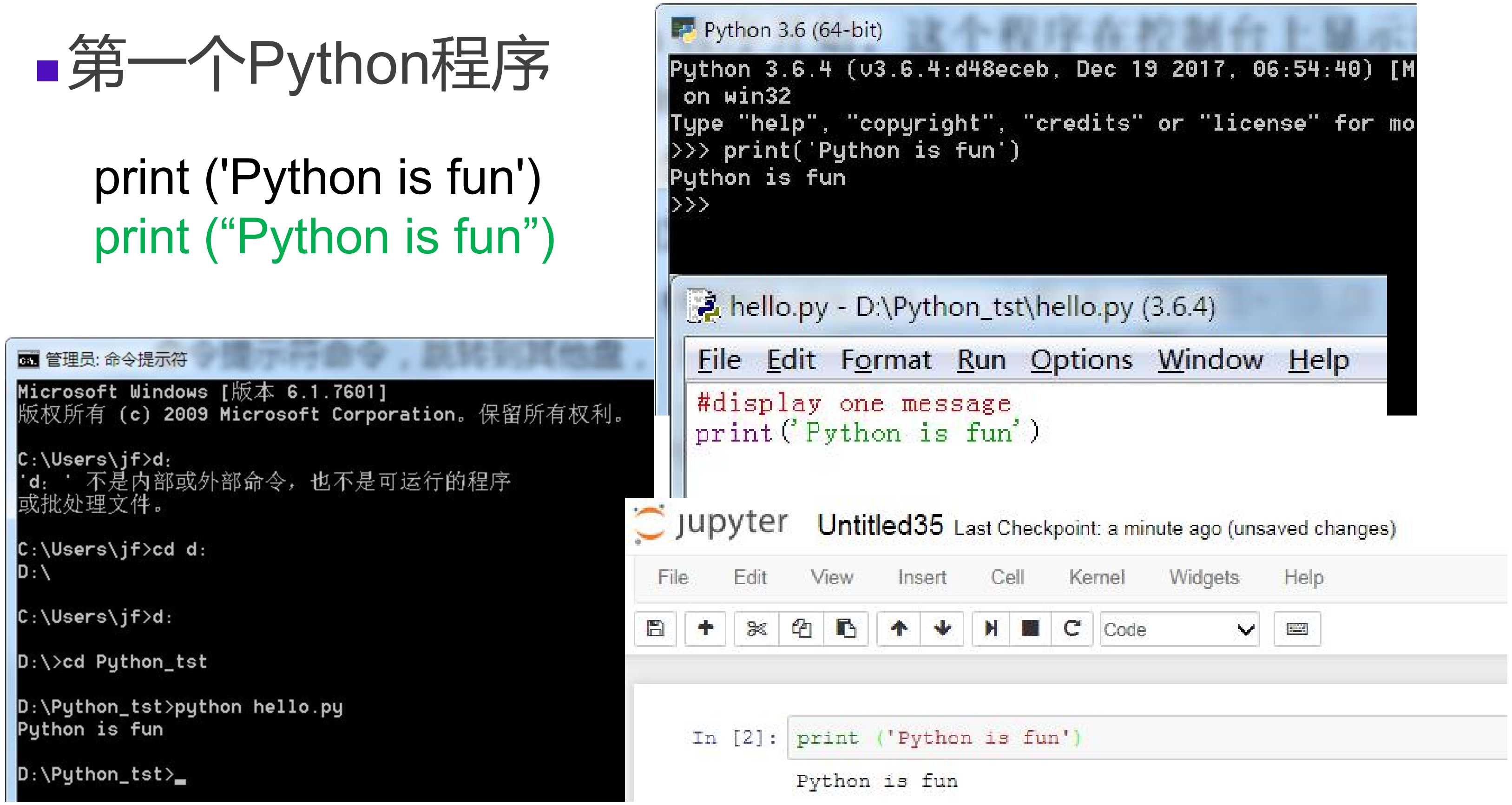
本章节目录

- Python概述
- 基本程序设计
- 数学函数、字符串和对象
- 选择
- 循环
- 函数

Python基础

- 第一个Python程序

  print ('Python is fun')
  print (“Python is fun”)



```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [M
 on win32
Type "help", "copyright", "credits" or "license" for mo
>>> print('Python is fun')
Python is fun
>>>
```

hello.py - D:\Python_tst\hello.py (3.6.4)
File Edit Format Run Options Window Help

```
#display one message
print('Python is fun')
```

```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\jf>d:
'd: ' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\jf>cd d:
D:\

C:\Users\jf>d:

D:\>cd Python_tst

D:\Python_tst>python hello.py
Python is fun

D:\Python_tst>_
```

jupyter Untitled35 Last Checkpoint: a minute ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Code

```
In [2]: print ('Python is fun')

        Python is fun
```

Python基础

- 注释
  - # 行注释
  - ''' 段注释

```
# This program displays Welcome to Python

''' This program displays Welcome to Python and
    Python is fun
'''
```

- 缩进、标点

```
# Display two messages
  print("Welcome to Python")
print("Python is fun")
```

```
# Display two messages
print("Welcome to Python").
print("Python is fun"),
```

Python基础

- 使用Python完成算术运算
  - 加、减、乘、除

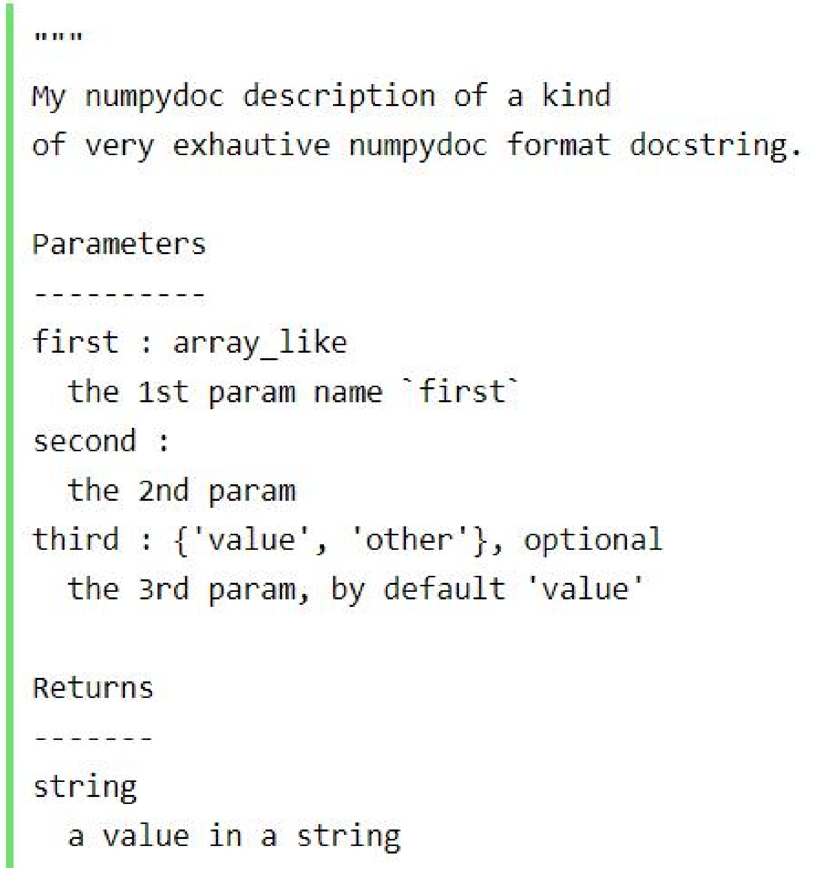
```
print(x + y)
print(x - y)
print(x * y)
print(x / y)
```

```
# Compute expression
print((10.5 + 2 * 3) / (45 - 3.5))
```

Python基础

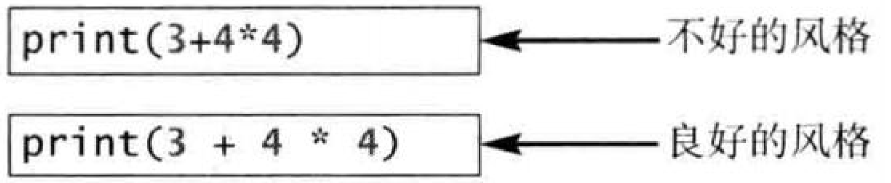
- 程序设计风格和文档
  - 恰当的注释和注释风格

```
#!/usr/bin/env python
# -*-coding:utf-8-*-
```



```
"""
My numpydoc description of a kind
of very exhautive numpydoc format docstring.

Parameters
----------
first : array_like
  the 1st param name `first`
second :
  the 2nd param
third : {'value', 'other'}, optional
  the 3rd param, by default 'value'

Returns
-------
string
  a value in a string
```

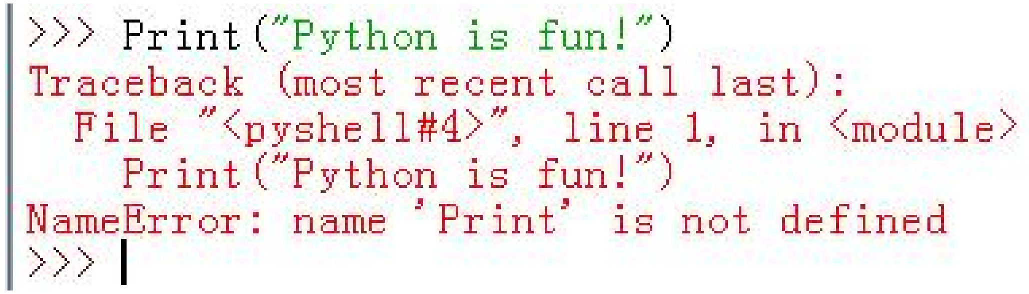
- 恰当的空格



```
print(3+4*4)
print(3 + 4 * 4)
```

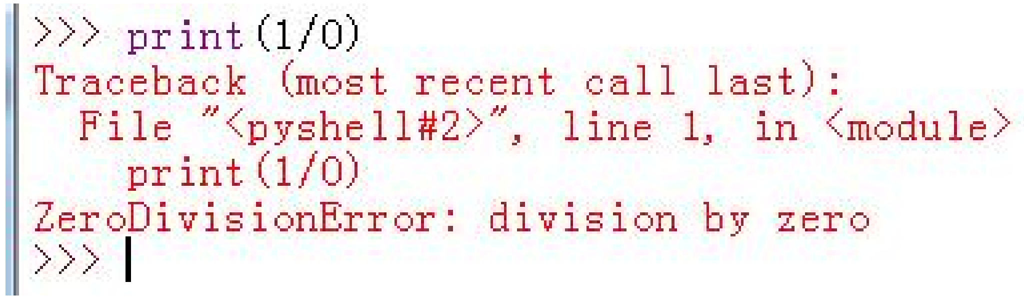
Python基础

- 程序设计错误
  - 语法错误



```
>>> Print("Python is fun!")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    Print("Python is fun!")
NameError: name 'Print' is not defined
>>> 
```

  - 运行时错误



```
>>> print(1/0)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(1/0)
ZeroDivisionError: division by zero
>>> 
```

Python基础

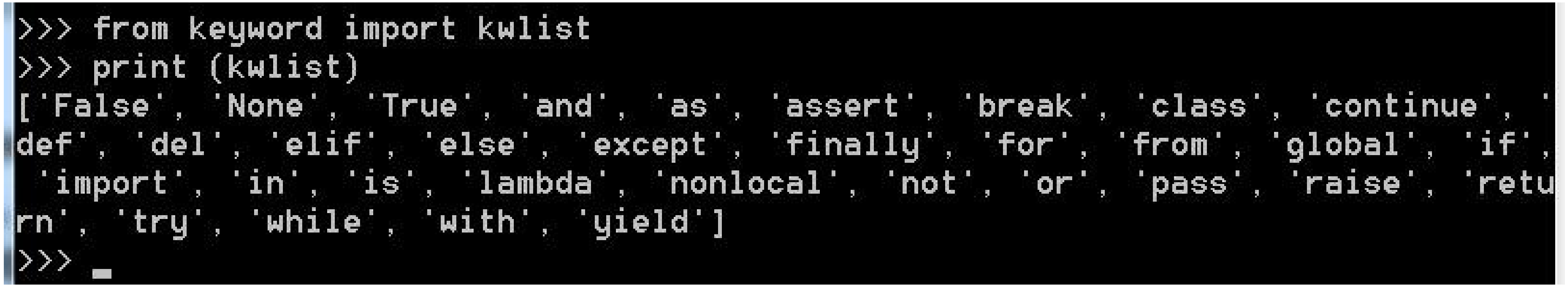
- 逻辑错误

```
# Convert Fahrenheit to Celsius
print("Fahrenheit 35 is Celsius degree ")
print(5 / 9 * 35 - 32)
```

```
Fahrenheit 35 is Celsius degree
-12.555555555555554
```

Python基础

- Python3中所有保留字

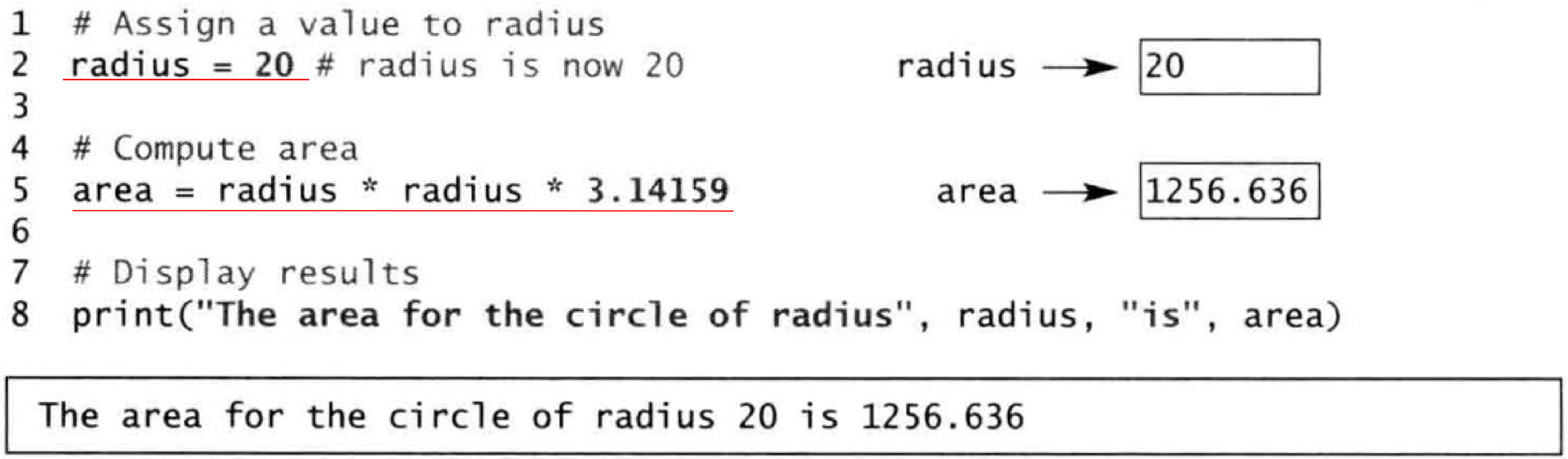


```
>>> from keyword import kwlist
>>> print (kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', '
def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'retu
rn', 'try', 'while', 'with', 'yield']
>>> _
```

基本程序设计

- 例程：计算一个圆的面积
  - 从用户获取圆的半径
  - 利用公式　面积= π *半径*半径（***变量，描述性名字***）
  - 显示结果

基本程序设计



```python
# Assign a value to radius
radius = 20 # radius is now 20

# Compute area
area = radius * radius * 3.14159

# Display results
print("The area for the circle of radius", radius, "is", area)
```

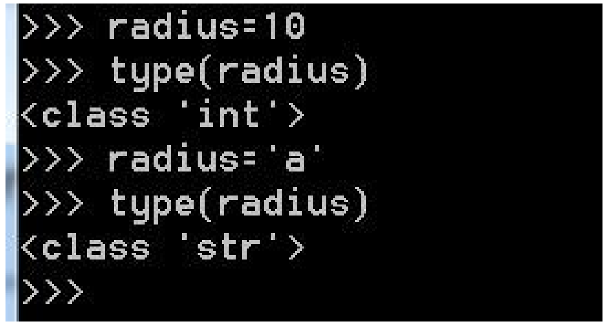
$$\begin{tikzcd} \text{radius} \arrow[r] & \boxed{20} \end{tikzcd}$$

$$\begin{tikzcd} \text{area} \arrow[r] & \boxed{1256.636} \end{tikzcd}$$

```
The area for the circle of radius 20 is 1256.636
```

基本程序设计

- 明确变量的数据类型



```
>>> radius=10
>>> type(radius)
<class 'int'>
>>> radius='a'
>>> type(radius)
<class 'str'>
>>>
```

- print语句 `print(item1, item2, ..., itemk)`

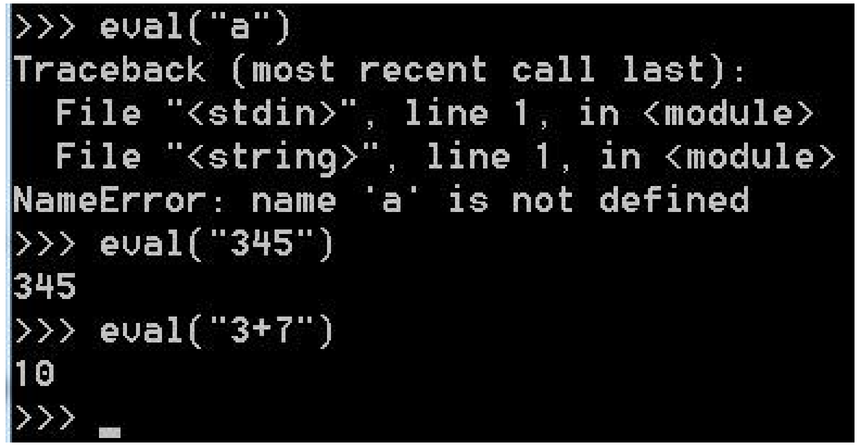
基本程序设计

- 从控制台读取输入
  - input函数 `variable = input("Enter a value: ")`
  - 例程

```
# Prompt the user to enter a radius
radius = eval(input("Enter a value for radius: "))

# Compute area
area = radius * radius * 3.14159

# Display results
print("The area for the circle of radius", radius, "is", area)
```

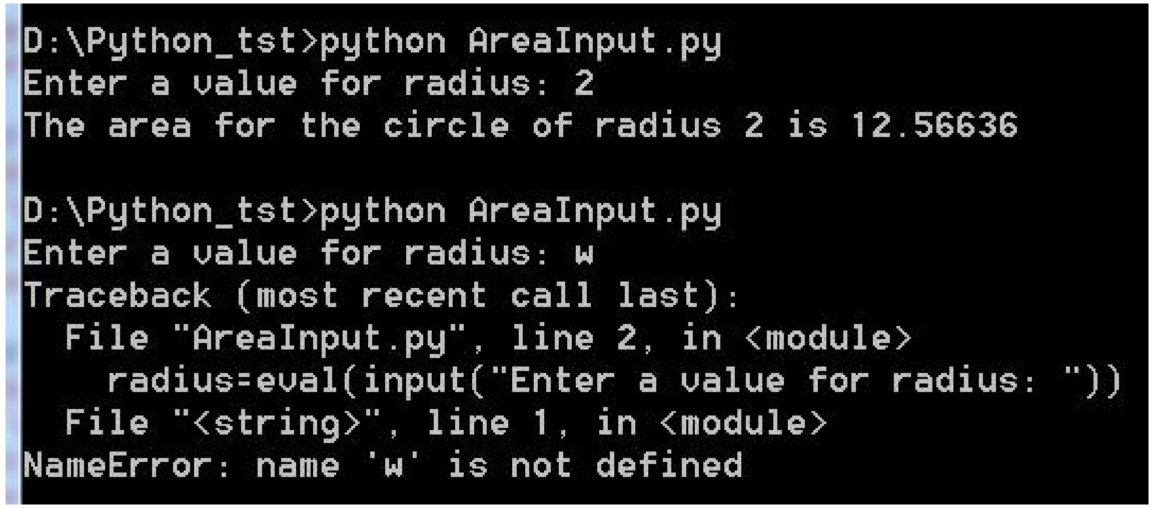
基本程序设计

- eval函数



```
>>> eval("a")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'a' is not defined
>>> eval("345")
345
>>> eval("3+7")
10
>>> _
```

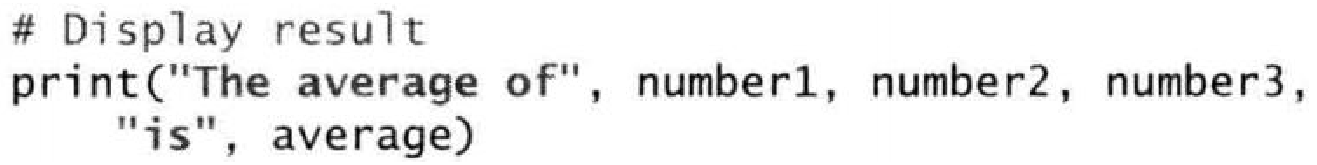
- 运行时错误



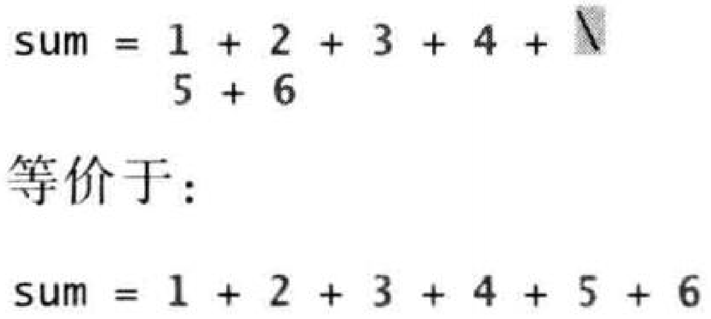
```
D:\Python_tst>python AreaInput.py
Enter a value for radius: 2
The area for the circle of radius 2 is 12.56636

D:\Python_tst>python AreaInput.py
Enter a value for radius: w
Traceback (most recent call last):
  File "AreaInput.py", line 2, in <module>
    radius=eval(input("Enter a value for radius: "))
  File "<string>", line 1, in <module>
NameError: name 'w' is not defined
```

基本程序设计

- 程序换行



```
# Display result
print("The average of", number1, number2, number3,
    "is", average)
```



```
sum = 1 + 2 + 3 + 4 + \
      5 + 6
```

等价于：

```
sum = 1 + 2 + 3 + 4 + 5 + 6
```

基本程序设计

- 标识符
  - 是由字母、数字、下划线构成的序列
  - 必须以字母或下划线开头
  - 不能是关键字
  - 可以为任意长度
  - 区分大小写
  - 变量名要小写，而多词连接时，骆驼拼写法。

如 area，numberOfStudents

基本程序设计

- 变量、赋值语句、赋值表达式
  - 赋值运算符“=”

```
variable = expression
```

  - 例程1

```
y = 1                         # Assign 1 to variable y
radius = 1.0                  # Assign 1.0 to variable radius
x = 5 * (3 / 2) + 3 * 2       # Assign the value of the expression to x
x = y + 1                     # Assign the addition of y and 1 to x
area = radius * radius * 3.14159 # Compute area
```

  - 例程2

```
i = j = k = 1
```

基本程序设计

- 同时赋值

```
var1, var2, ..., varn = exp1, exp2, ..., expn
```

  - 例程1：交换变量的值

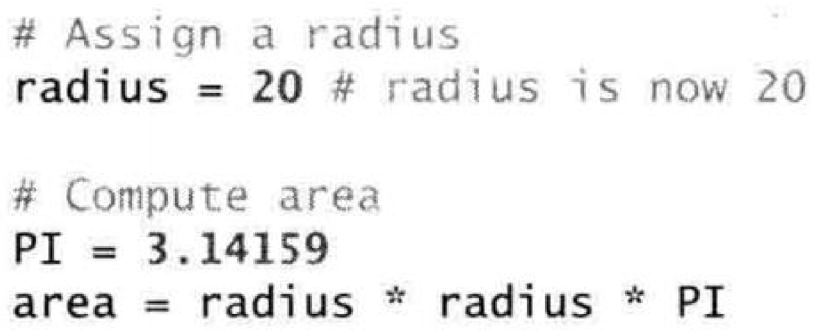
```
>>> x, y = y, x # Swap x with y
```

  - 例程2：同时输入

```
# Prompt the user to enter three numbers
number1, number2, number3 = eval(input(
    "Enter three numbers separated by commas: "))

# Compute average
average = (number1 + number2 + number3) / 3
```

基本程序设计

- 定名常量：固定数据
  - 全部大写字母命名
  - python中不存在绝对的常量

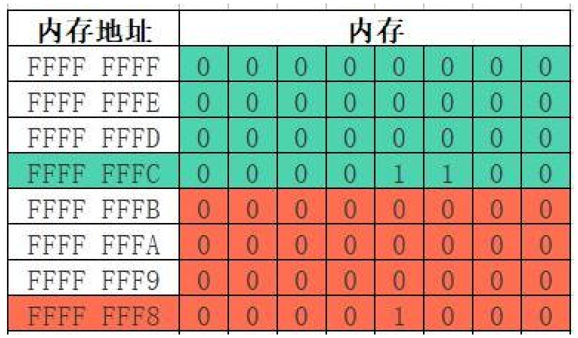


```
# Assign a radius
radius = 20 # radius is now 20

# Compute area
PI = 3.14159
area = radius * radius * PI
```

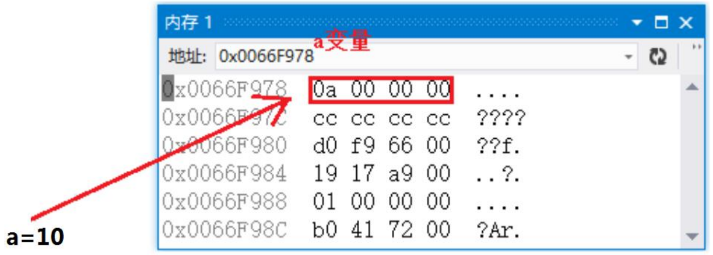
基本程序设计

- 数值数据类型和运算符
  - 两种数值类型：整数、实数
  - 例： 字面量为1.0　　字面量为1

byte



内存地址	内存							
FFFF FFFF	0	0	0	0	0	0	0	0
FFFF FFFE	0	0	0	0	0	0	0	0
FFFF FFFD	0	0	0	0	0	0	0	0
FFFF FFFC	0	0	0	0	1	1	0	0
FFFF FFFB	0	0	0	0	0	0	0	0
FFFF FFFA	0	0	0	0	0	0	0	0
FFFF FFF9	0	0	0	0	0	0	0	0
FFFF FFF8	0	0	0	0	1	0	0	0



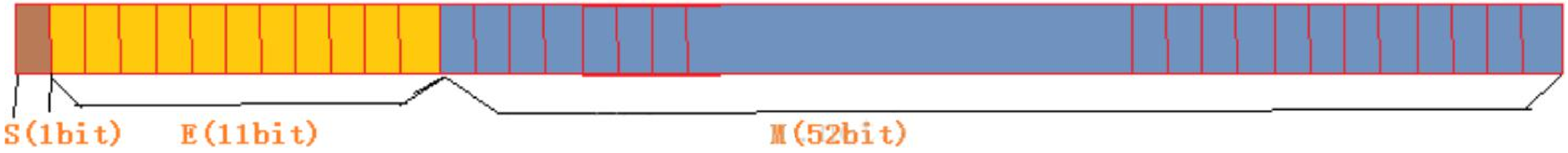
a=10

内存 1

地址: 0x0066F978 a变量

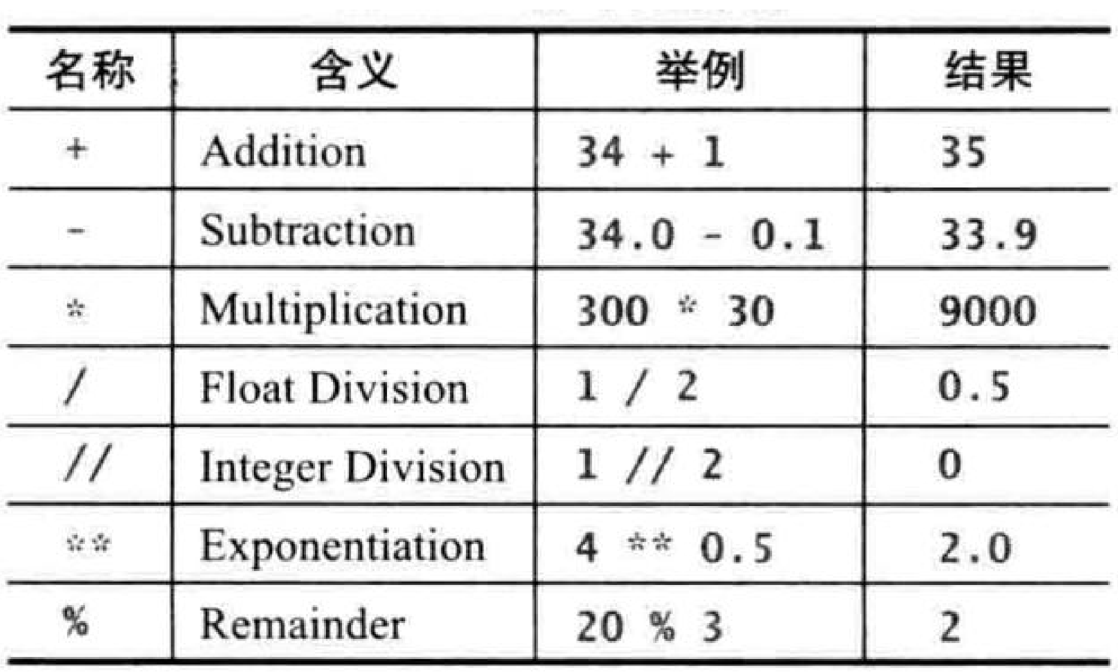
```
0x0066F978  0a 00 00 00  ....
0x0066F97C  cc cc cc cc  ????
0x0066F980  d0 f9 66 00  ??f.
0x0066F984  19 17 a9 00  ..?.
0x0066F988  01 00 00 00  ....
0x0066F98C  b0 41 72 00  ?Ar.
```

浮点数： (-1)^S*M*2^E



基本程序设计

- 运算符



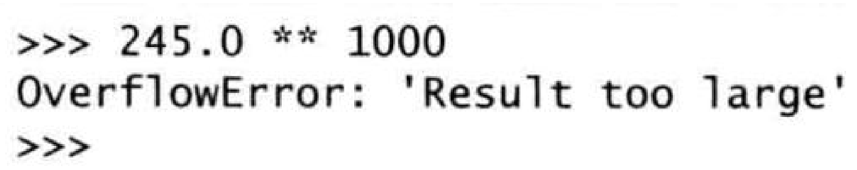
名称	含义	举例	结果
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Float Division	1 / 2	0.5
//	Integer Division	1 // 2	0
**	Exponentiation	4 ** 0.5	2.0
%	Remainder	20 % 3	2

基本程序设计

- / 执行浮点数除法，产生小数；// 执行整数除法，舍去小数。
- 科学计数法

$$1.234\ 56 \times 10^{-2} \longleftrightarrow 1.234\ 56\text{E-}2$$

- 变量太大，数据溢出



```
>>> 245.0 ** 1000
OverflowError: 'Result too large'
>>>
```

基本程序设计

- 运算表达式

$$\frac{3+4x}{5}-\frac{10(y-5)(a+b+c)}{x}+9\left(\frac{4}{x}+\frac{9+x}{y}\right)$$

$$\updownarrow$$

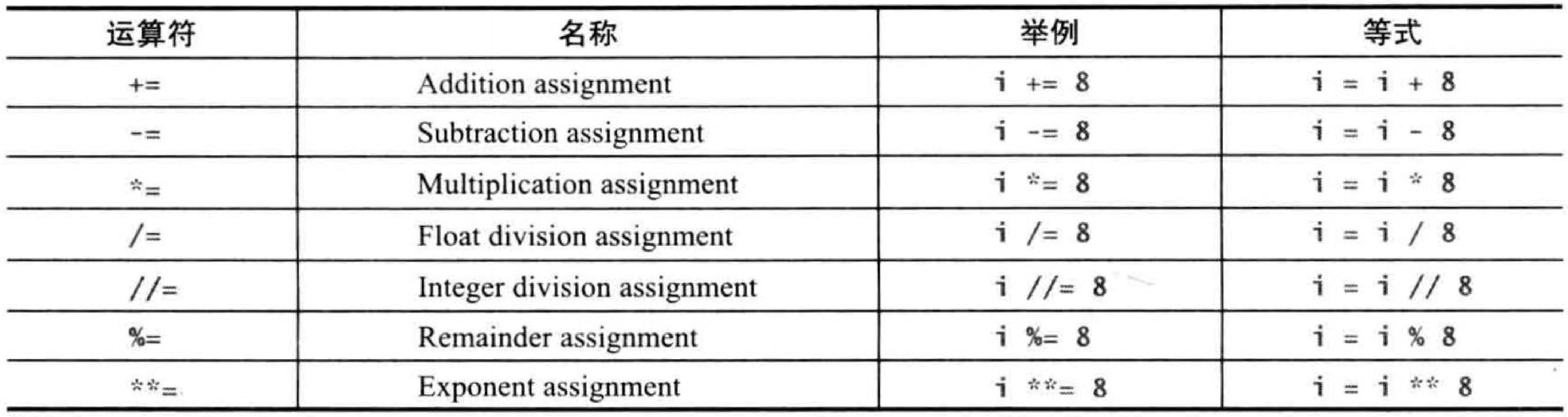
```
(3 + 4 * x) / 5 - 10 * (y - 5) * (a + b + c) / x +
9 * (4 / x + (9 + x) / y)
```

- 运算符优先级

( ) >** > * / // > + -

基本程序设计

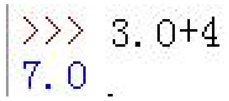
- 增强型赋值运算符



运算符	名称	举例	等式
+=	Addition assignment	i += 8	i = i + 8
-=	Subtraction assignment	i -= 8	i = i - 8
*=	Multiplication assignment	i *= 8	i = i * 8
/=	Float division assignment	i /= 8	i = i / 8
//=	Integer division assignment	i //= 8	i = i // 8
%=	Remainder assignment	i %= 8	i = i % 8
**=	Exponent assignment	i **= 8	i = i ** 8

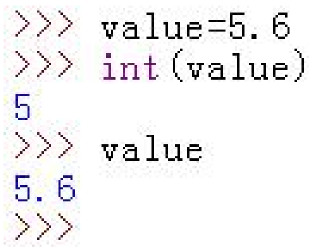
基本程序设计

- 类型转换和四舍五入
  - 类型转换：

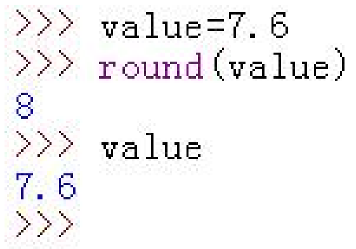


```
>>> 3.0+4
7.0
```

  - int(value)函数和round(value)函数



```
>>> value=5.6
>>> int(value)
5
>>> value
5.6
>>>
```



```
>>> value=7.6
>>> round(value)
8
>>> value
7.6
>>>
```

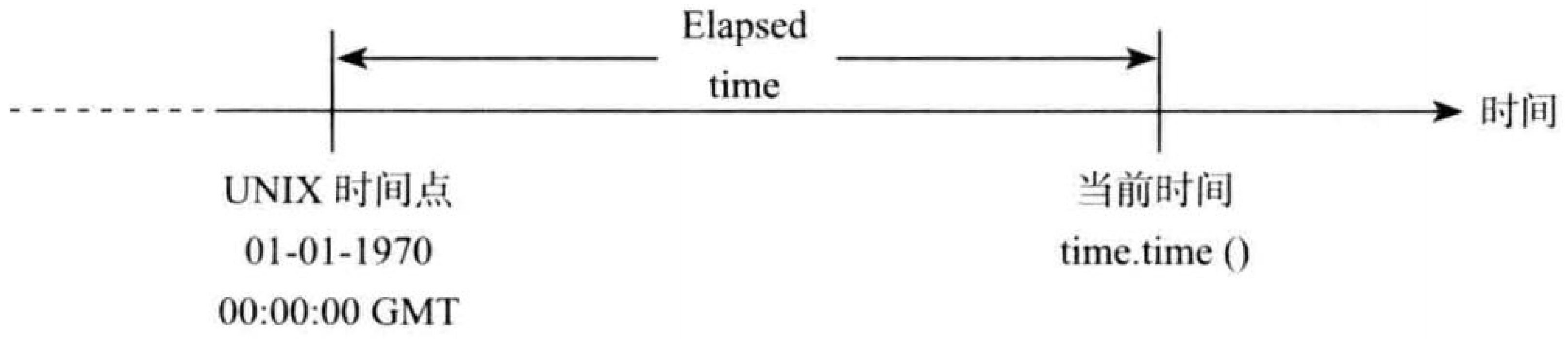
  - int()函数与eval()函数

int('3.4')　　eval('3.4')
int('003')　　eval('003')

基本程序设计

- 时间显示
  - time模块中的time()函数

```
import time

currentTime = time.time() # Get current time
```



基本程序设计

- 软件开发流程

键入：利率，贷款额度，期限
输出：每月还贷数，总还款数

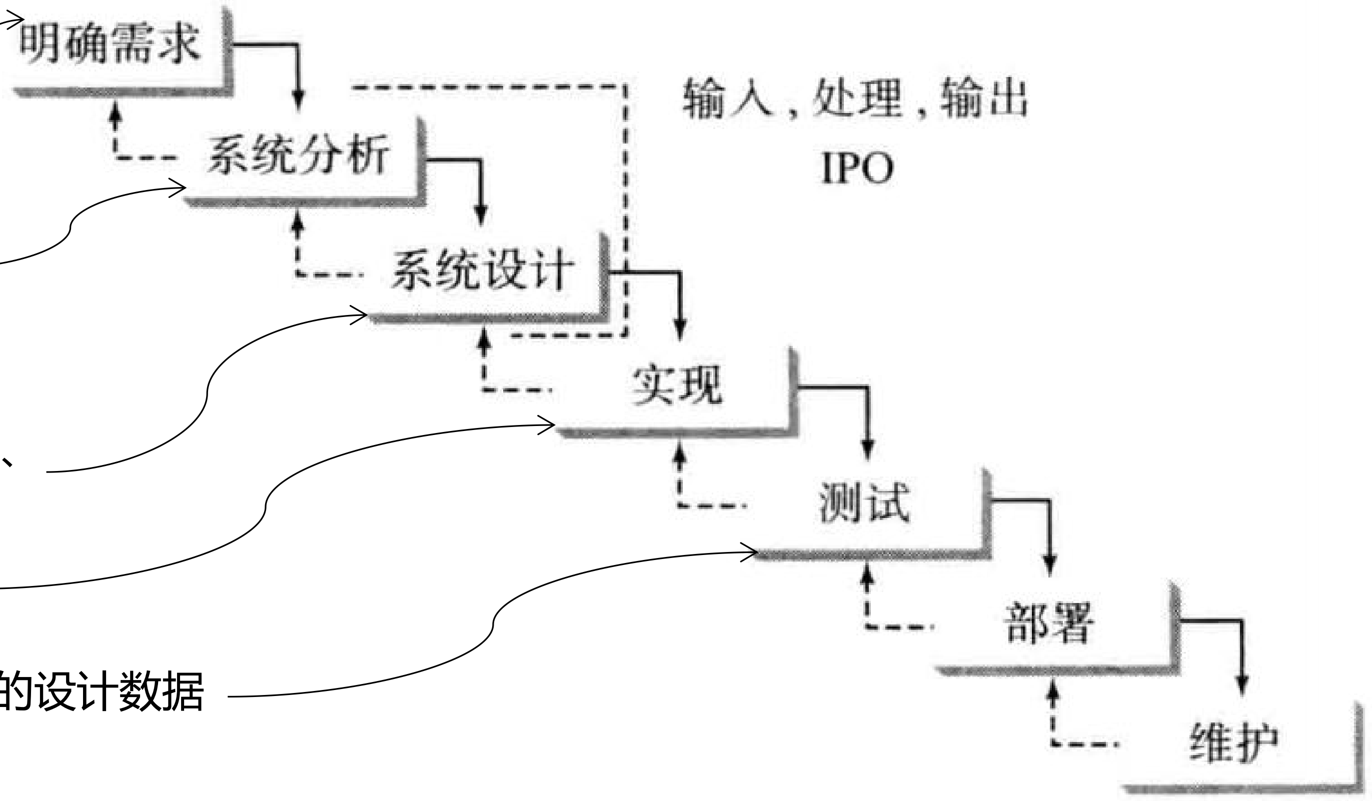
$$月供 = \frac{贷款数 \times 月利率}{1 - \frac{1}{(1+月利率)^{年限 \times 12}}}$$

$$总还款数 = 月供 \times 年限 \times 12$$

提示用户输入、输入格式的处理、利用公式计算、显示

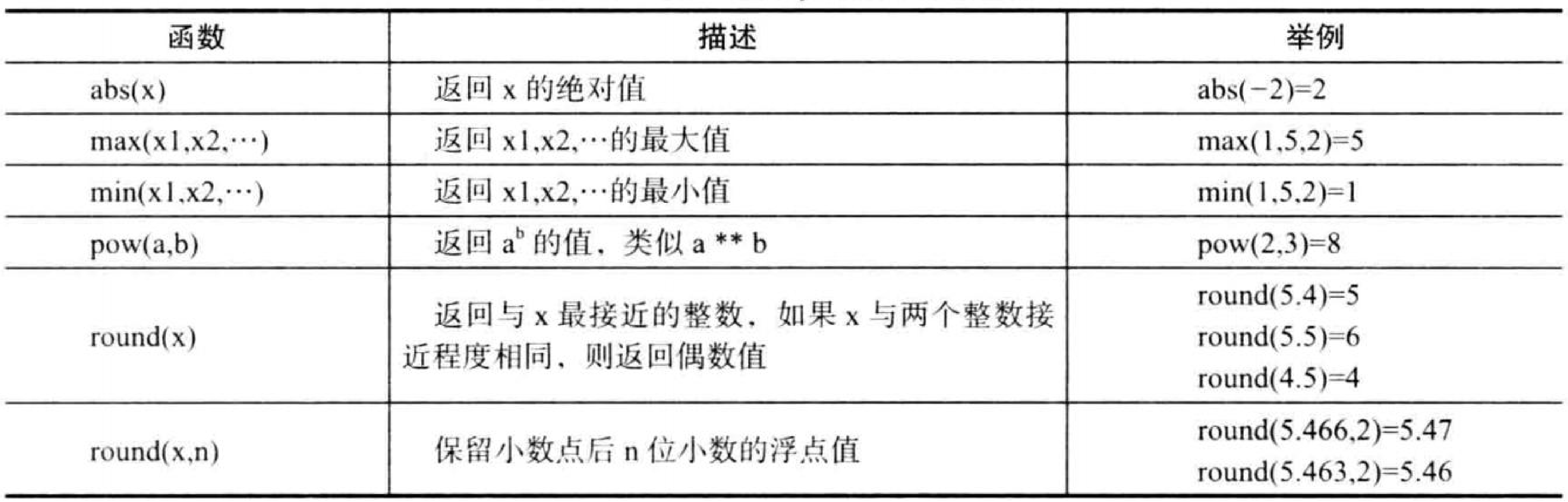
选用编程语言，转化为程序

设计能涵盖所有情况的设计数据



数学函数、字符串和对象

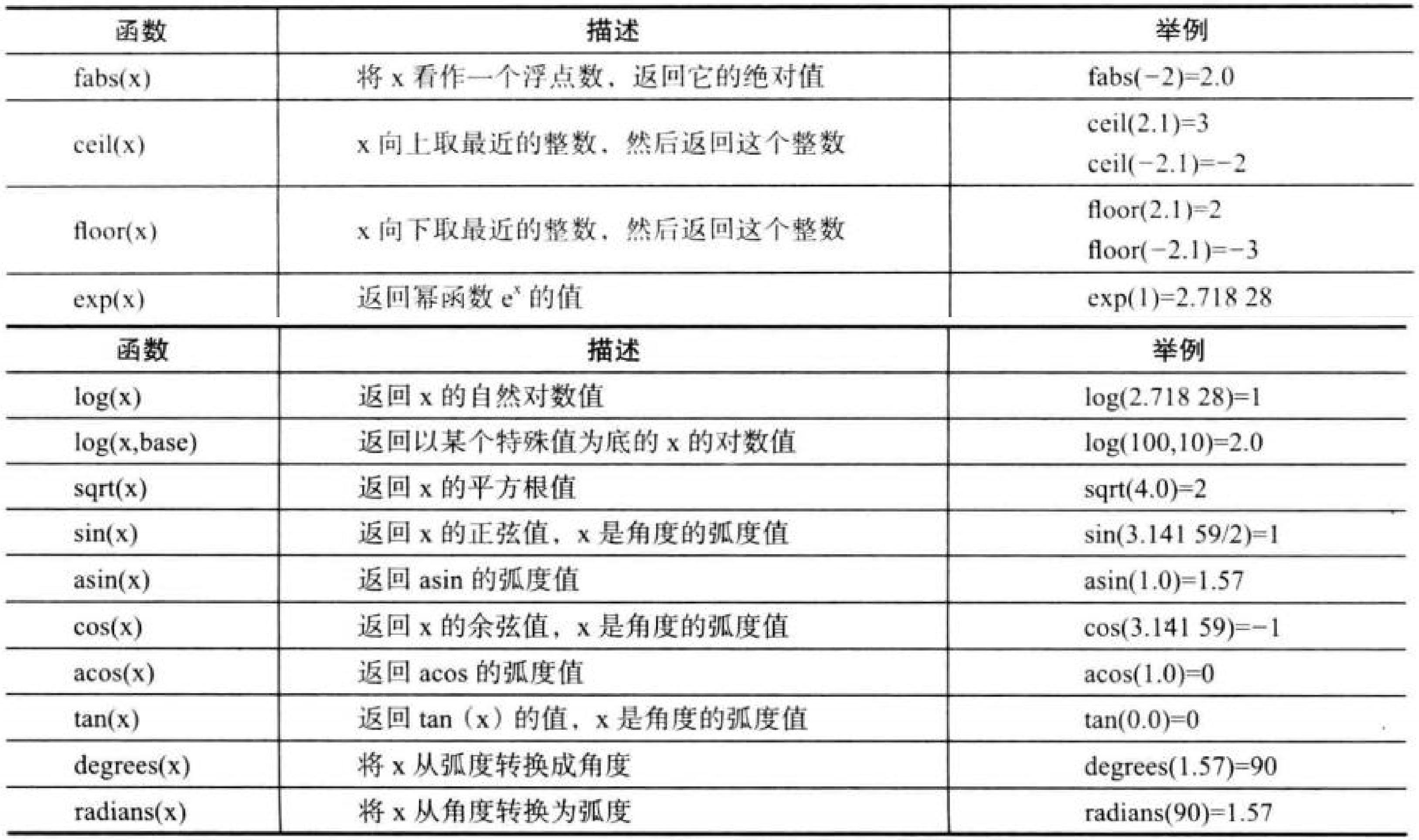
- 常见的python函数
  - 内置函数



函数	描述	举例
abs(x)	返回 x 的绝对值	abs(−2)=2
max(x1,x2,···)	返回 x1,x2,···的最大值	max(1,5,2)=5
min(x1,x2,···)	返回 x1,x2,···的最小值	min(1,5,2)=1
pow(a,b)	返回 a^b 的值，类似 a ** b	pow(2,3)=8
round(x)	返回与 x 最接近的整数，如果 x 与两个整数接近程度相同，则返回偶数值	round(5.4)=5 round(5.5)=6 round(4.5)=4
round(x,n)	保留小数点后 n 位小数的浮点值	round(5.466,2)=5.47 round(5.463,2)=5.46

数学函数、字符串和对象

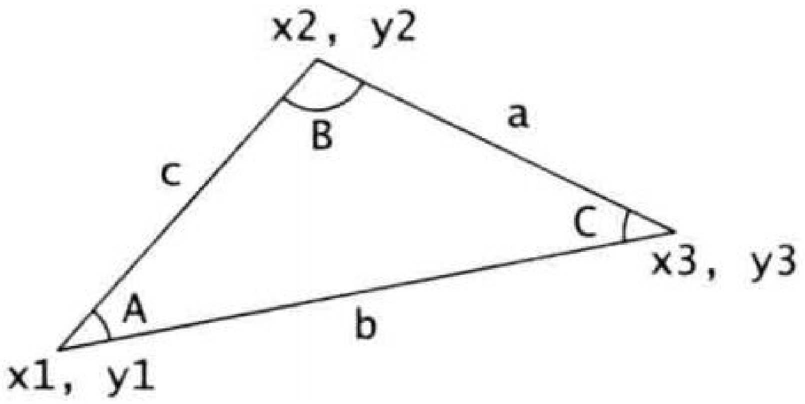
- math模块的函数



函数	描述	举例
fabs(x)	将 x 看作一个浮点数，返回它的绝对值	fabs(−2)=2.0
ceil(x)	x 向上取最近的整数，然后返回这个整数	ceil(2.1)=3 ceil(−2.1)=−2
floor(x)	x 向下取最近的整数，然后返回这个整数	floor(2.1)=2 floor(−2.1)=−3
exp(x)	返回幂函数 e^x 的值	exp(1)=2.718 28
log(x)	返回 x 的自然对数值	log(2.718 28)=1
log(x,base)	返回以某个特殊值为底的 x 的对数值	log(100,10)=2.0
sqrt(x)	返回 x 的平方根值	sqrt(4.0)=2
sin(x)	返回 x 的正弦值，x 是角度的弧度值	sin(3.141 59/2)=1
asin(x)	返回 asin 的弧度值	asin(1.0)=1.57
cos(x)	返回 x 的余弦值，x 是角度的弧度值	cos(3.141 59)=−1
acos(x)	返回 acos 的弧度值	acos(1.0)=0
tan(x)	返回 tan（x）的值，x 是角度的弧度值	tan(0.0)=0
degrees(x)	将 x 从弧度转换成角度	degrees(1.57)=90
radians(x)	将 x 从角度转换为弧度	radians(90)=1.57

数学函数、字符串和对象

- 访问math模块中的Pi 和 e： math.pi math.
- 例程：利用math库函数求三角形的三个角



```
import math

x1, y1, x2, y2, x3, y3 = eval(input("Enter three points: "))

a = math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3))
b = math.sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3))
c = math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))

A = math.degrees(math.acos((a * a - b * b - c * c) / (-2 * b * c)))
B = math.degrees(math.acos((b * b - a * a - c * c) / (-2 * a * c)))
C = math.degrees(math.acos((c * c - b * b - a * a) / (-2 * a * b)))
```

数学函数、字符串和对象

- 注：输入和输出

```
eval(input("Enter three points: "))
```

```
input("Enter six coordinates of three points separated by commas\
like x1, y1, x2, y2, x3, y3: ")
```

```
print("The three angles are ", round(A * 100) / 100.0,
    round(B * 100) / 100.0, round(C * 100) / 100.0)
```

```
round (A, 2 )
```

数学函数、字符串和对象

- 字符串和字符
  - 字符=一个字符的字符串

```
letter = 'A' # Same as letter = "A"
numChar = '4' # Same as numChar = "4"
message = "Good morning" # Same as message = 'Good morning'
```

  - 约定：双引号括住多个字符构成的字符串；单引号括住单个字符的字符串或空字符串。

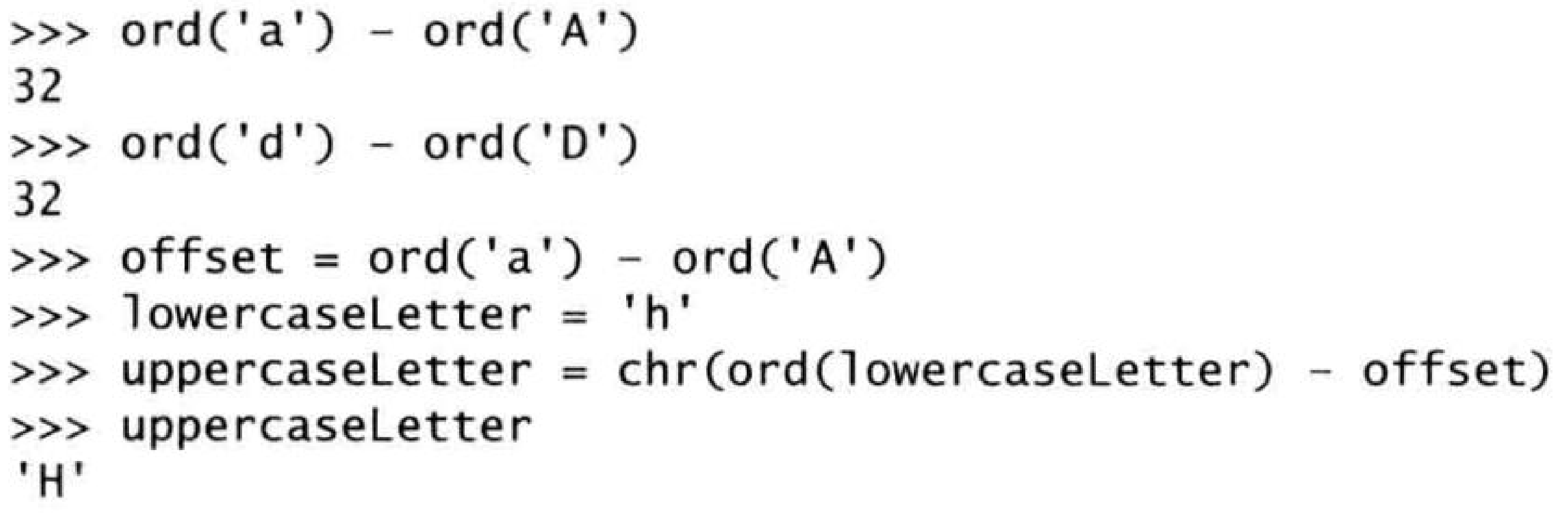
数学函数、字符串和对象

- 字符编码
  - ASCII码
  - 统一码（Unicode），以“\u”开始
    - UTF----“Unicode Character Set Transformation Format”
    - 例程：

```
import turtle

turtle.write("\u6B22\u8FCE \u03b1 \u03b2 \u03b3")

turtle.done()
```

数学函数、字符串和对象

- ord(ch)函数：返回字符ch的ASCII码。
- chr(code)：返回code所代表的字符。

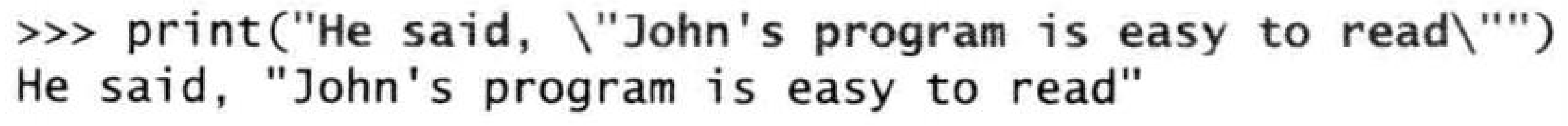


```
>>> ord('a') - ord('A')
32
>>> ord('d') - ord('D')
32
>>> offset = ord('a') - ord('A')
>>> lowercaseLetter = 'h'
>>> uppercaseLetter = chr(ord(lowercaseLetter) - offset)
>>> uppercaseLetter
'H'
```

数学函数、字符串和对象

- 转义序列:由“\”和紧跟其后的字母或数字组成

He said，“John's program is easy to read.”



```
>>> print("He said, \"John's program is easy to read\"")
He said, "John's program is easy to read"
```

- 字符转义序列如： \b \t \n \f \r \\ \' \”

数学函数、字符串和对象

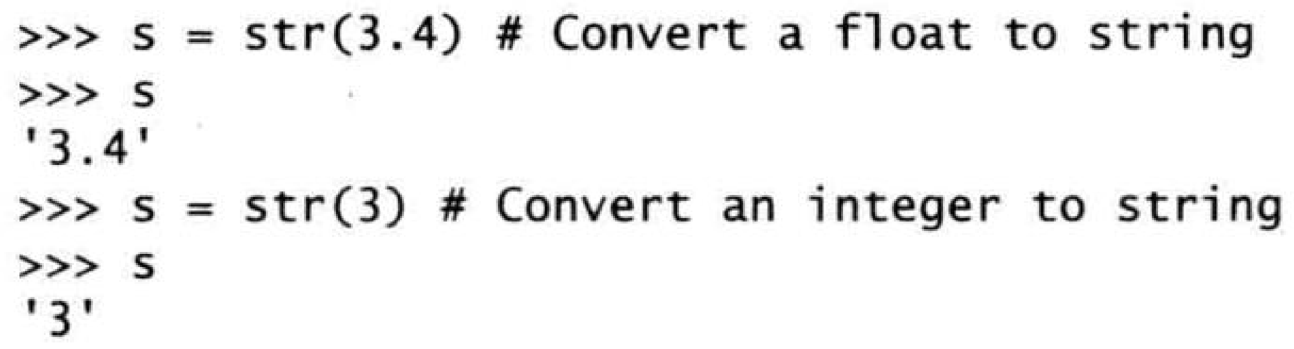
- 不换行打印

```
print(item1, item2, ..., end = "anyendingstring")
```

```
radius = 3
print("The area is", radius * radius * math.pi, end = ' ')
print("and the perimeter is", 2 * radius * math.pi)
```

数学函数、字符串和对象

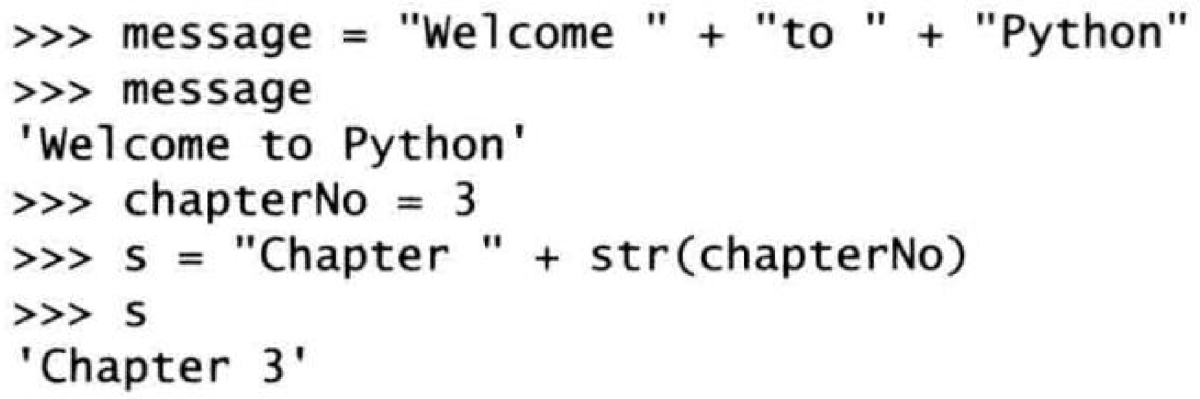
- 函数str()：将一个数字转化成字符串



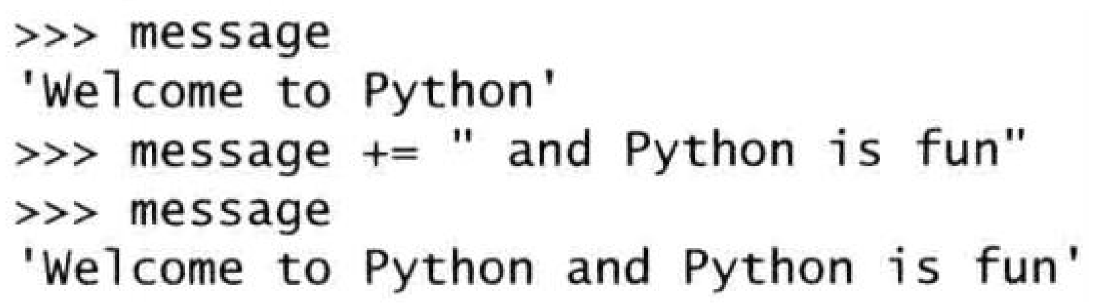
```
>>> s = str(3.4) # Convert a float to string
>>> s
'3.4'
>>> s = str(3) # Convert an integer to string
>>> s
'3'
```

数学函数、字符串和对象

- 字符串连接操作：“+”、“+=”



```
>>> message = "Welcome " + "to " + "Python"
>>> message
'Welcome to Python'
>>> chapterNo = 3
>>> s = "Chapter " + str(chapterNo)
>>> s
'Chapter 3'
```



```
>>> message
'Welcome to Python'
>>> message += " and Python is fun"
>>> message
'Welcome to Python and Python is fun'
```

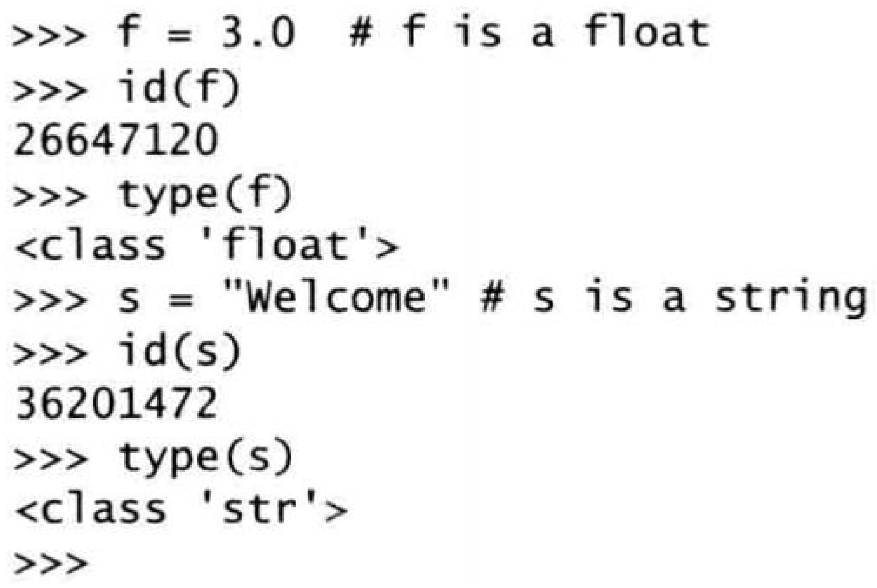
数学函数、字符串和对象

- 从控制台读取字符串

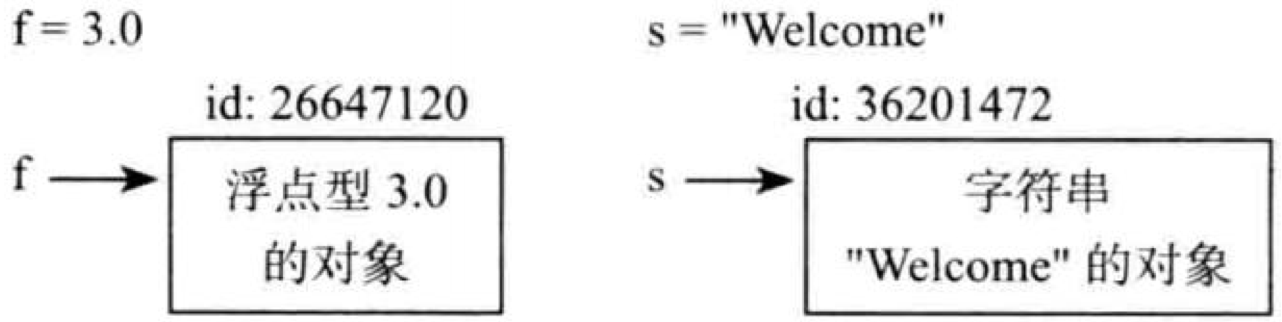
```
s1 = input("Enter a string: ")
s2 = input("Enter a string: ")
s3 = input("Enter a string: ")
print("s1 is " + s1)
print("s2 is " + s2)
print("s3 is " + s3)
```

数学函数、字符串和对象

- 对象和方法简介
  - 所有数据都是对象

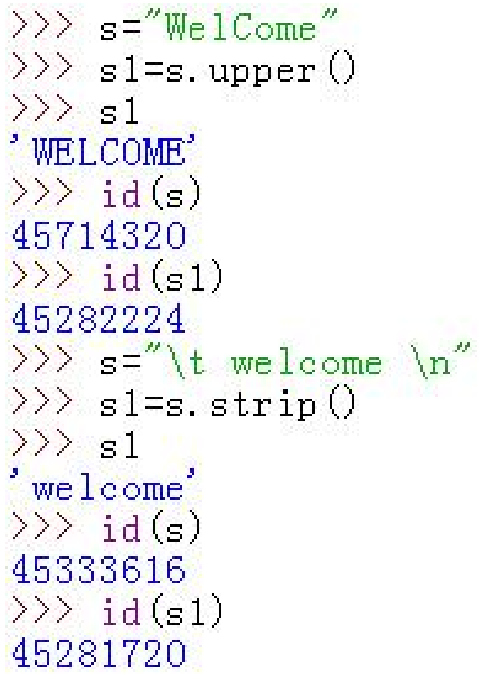


```
>>> f = 3.0  # f is a float
>>> id(f)
26647120
>>> type(f)
<class 'float'>
>>> s = "Welcome" # s is a string
>>> id(s)
36201472
>>> type(s)
<class 'str'>
>>>
```



数学函数、字符串和对象

- 方法：对象所用的函数 object.method()
- 字符串类型的方法举例：

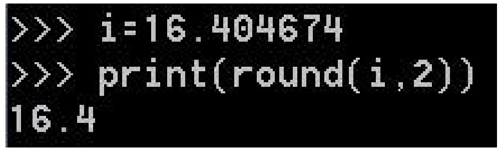


```
>>> s="WelCome"
>>> s1=s.upper()
>>> s1
'WELCOME'
>>> id(s)
45714320
>>> id(s1)
45282224
>>> s="\t welcome \n"
>>> s1=s.strip()
>>> s1
'welcome'
>>> id(s)
45333616
>>> id(s1)
45281720
```

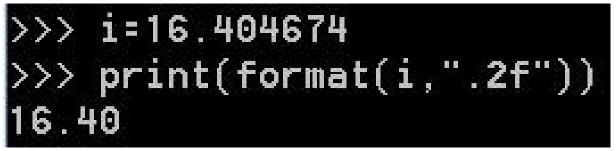
数学函数、字符串和对象

- 格式化数字和字符串
  - format()函数用于金融计算

```
format(item, format-specifier)
```



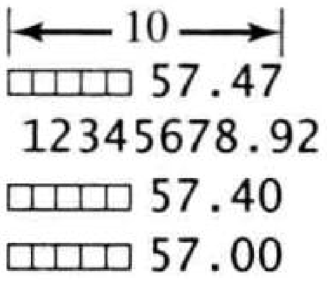
```
>>> i=16.404674
>>> print(round(i,2))
16.4
```



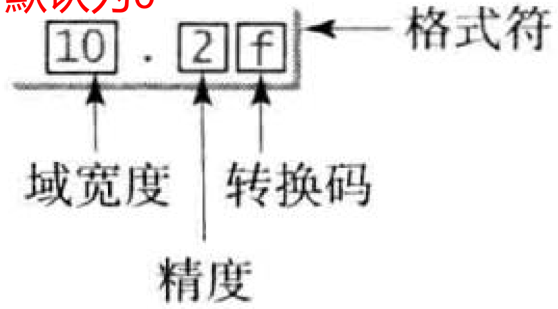
```
>>> i=16.404674
>>> print(format(i,".2f"))
16.40
```

  - 格式化浮点数 “ *width.precisionf*”

```
print(format(57.467657, "10.2f"))
print(format(12345678.923, "10.2f"))
print(format(57.4, "10.2f"))
print(format(57, "10.2f"))
```

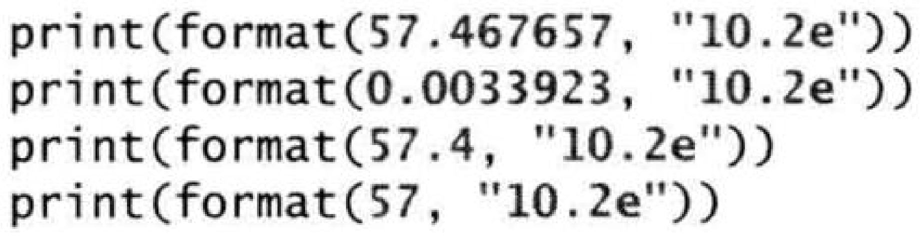


默认为0

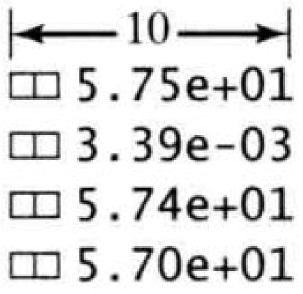


数学函数、字符串和对象

- 用科学计数法格式化

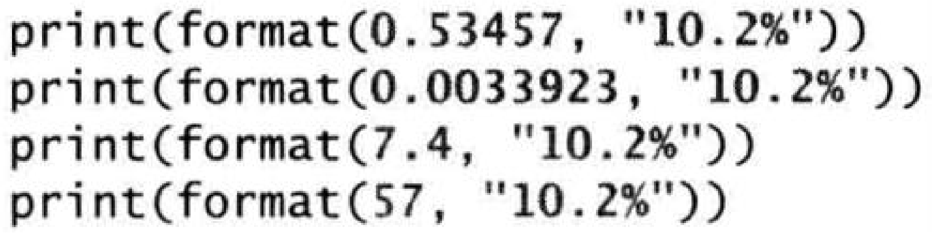


```
print(format(57.467657, "10.2e"))
print(format(0.0033923, "10.2e"))
print(format(57.4, "10.2e"))
print(format(57, "10.2e"))
```

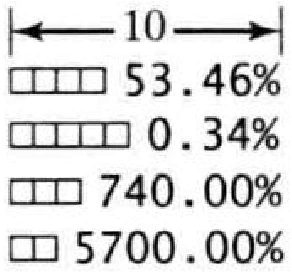


```
  5.75e+01
  3.39e-03
  5.74e+01
  5.70e+01
```

- 格式化成百分数



```
print(format(0.53457, "10.2%"))
print(format(0.0033923, "10.2%"))
print(format(7.4, "10.2%"))
print(format(57, "10.2%"))
```

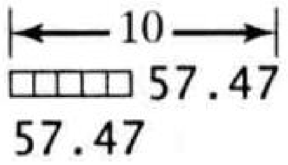


```
    53.46%
     0.34%
   740.00%
  5700.00%
```

数学函数、字符串和对象

- 调整对齐方式

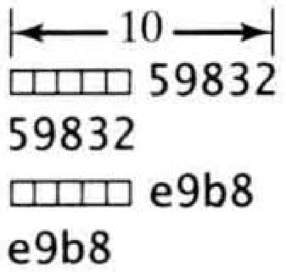
```
print(format(57.467657, "10.2f"))
print(format(57.467657, "<10.2f"))
```



```
□□□□□ 57.47
57.47
```

- 格式化整数

```
print(format(59832, "10d"))
print(format(59832, "<10d"))
print(format(59832, "10x"))
print(format(59832, "<10x"))
```

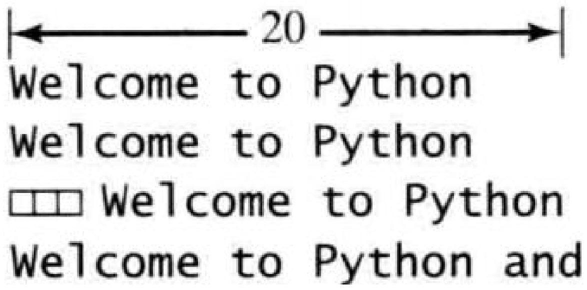


```
□□□□□ 59832
59832
□□□□□ e9b8
e9b8
```

数学函数、字符串和对象

- 格式化字符串

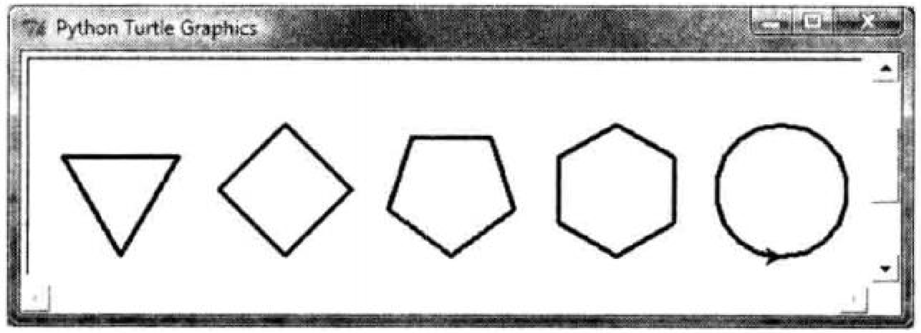
```
print(format("Welcome to Python", "20s"))
print(format("Welcome to Python", "<20s"))
print(format("Welcome to Python", ">20s"))
print(format("Welcome to Python and Java", ">20s"))
```



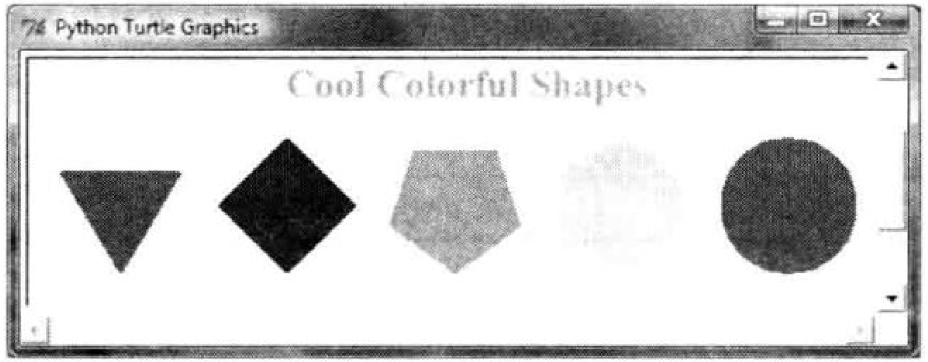
```
Welcome to Python
Welcome to Python
□□□Welcome to Python
Welcome to Python and Java
```

数学函数、字符串和对象

- 图形绘制
  - 使用Turtle绘图（略）



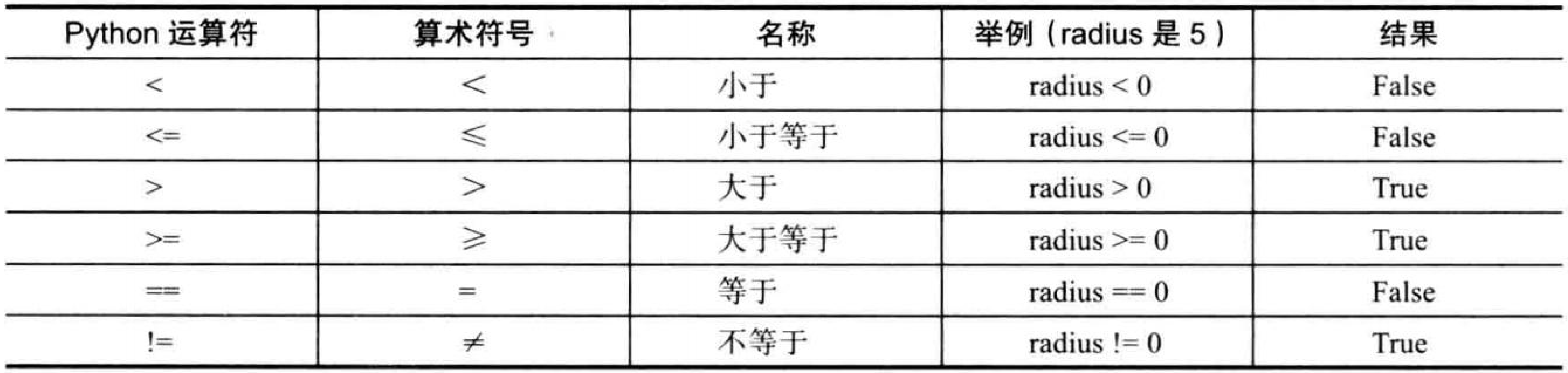
请见教材66页



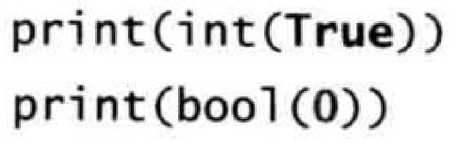
请见教材68页

选择结构

- Python比较运算符



Python 运算符	算术符号	名称	举例（radius 是 5）	结果
<	<	小于	radius < 0	False
<=	≤	小于等于	radius <= 0	False
>	>	大于	radius > 0	True
>=	≥	大于等于	radius >= 0	True
==	=	等于	radius == 0	False
!=	≠	不等于	radius != 0	True



```
print(int(True))
print(bool(0))
```

选择结构

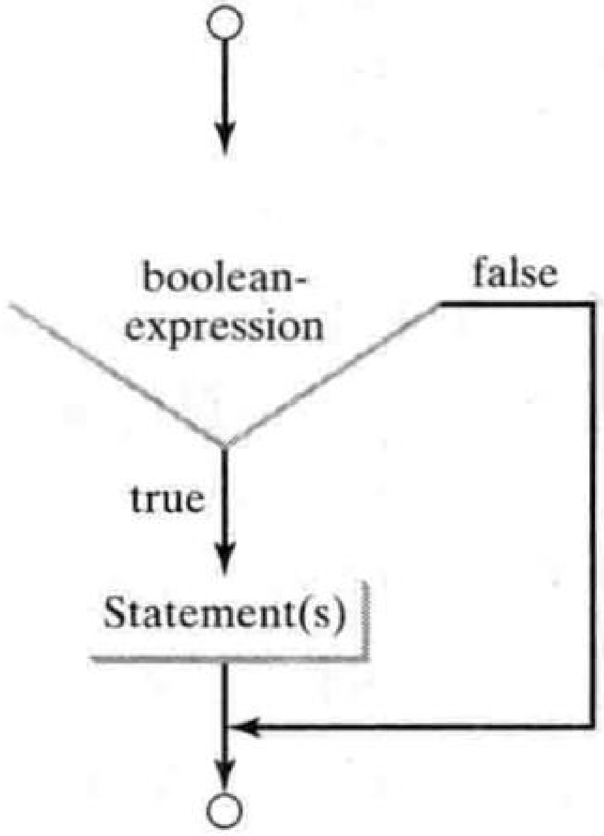
- 产生随机数字
  - random模块中的randint(a,b)、random()、randrange(a,b)

```
>>> import random
>>> random.random()
0.34343
>>> random.random()
0.20119
>>> random.randint(0, 1)
0
>>> random.randint(0, 1)
1
>>> random.randrange(0, 1) # This will always be 0
0
```

选择结构

- if语句
  - 单向if语句

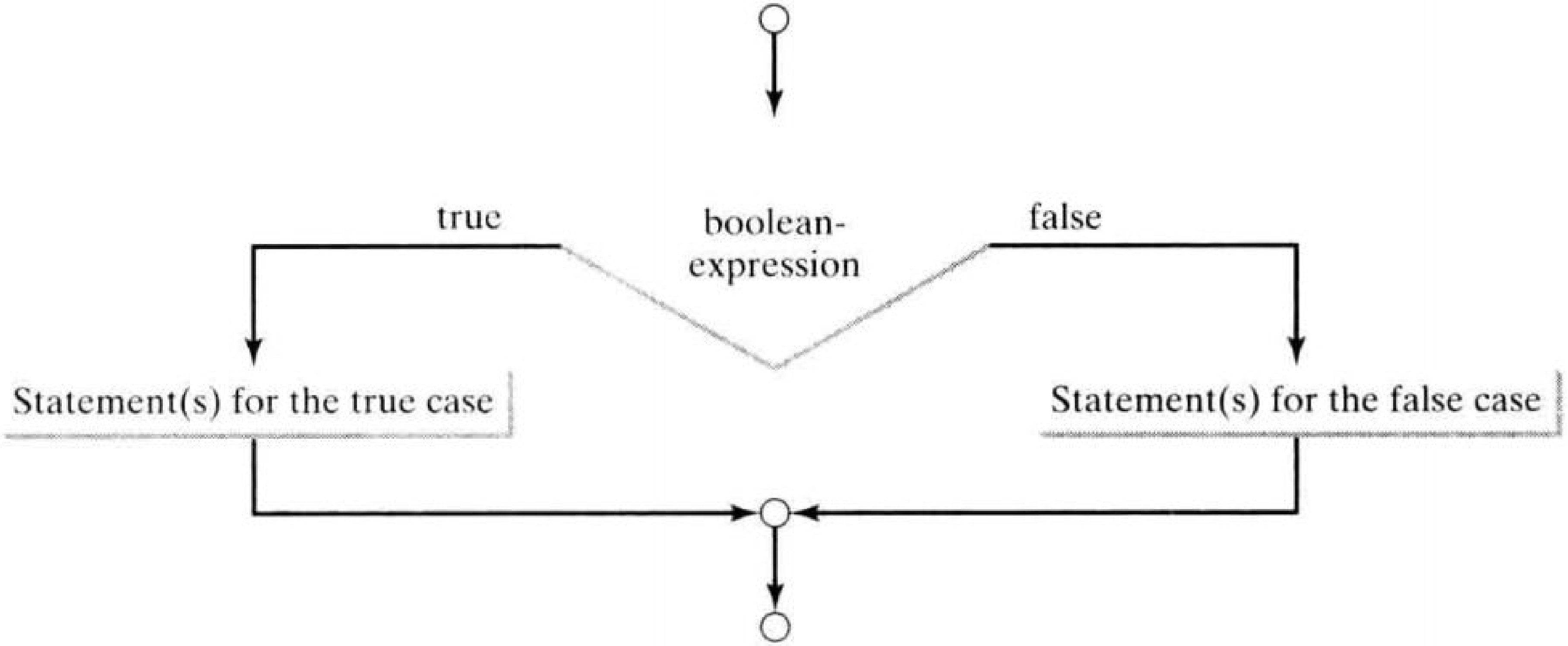
```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area is",area )
```



选择结构

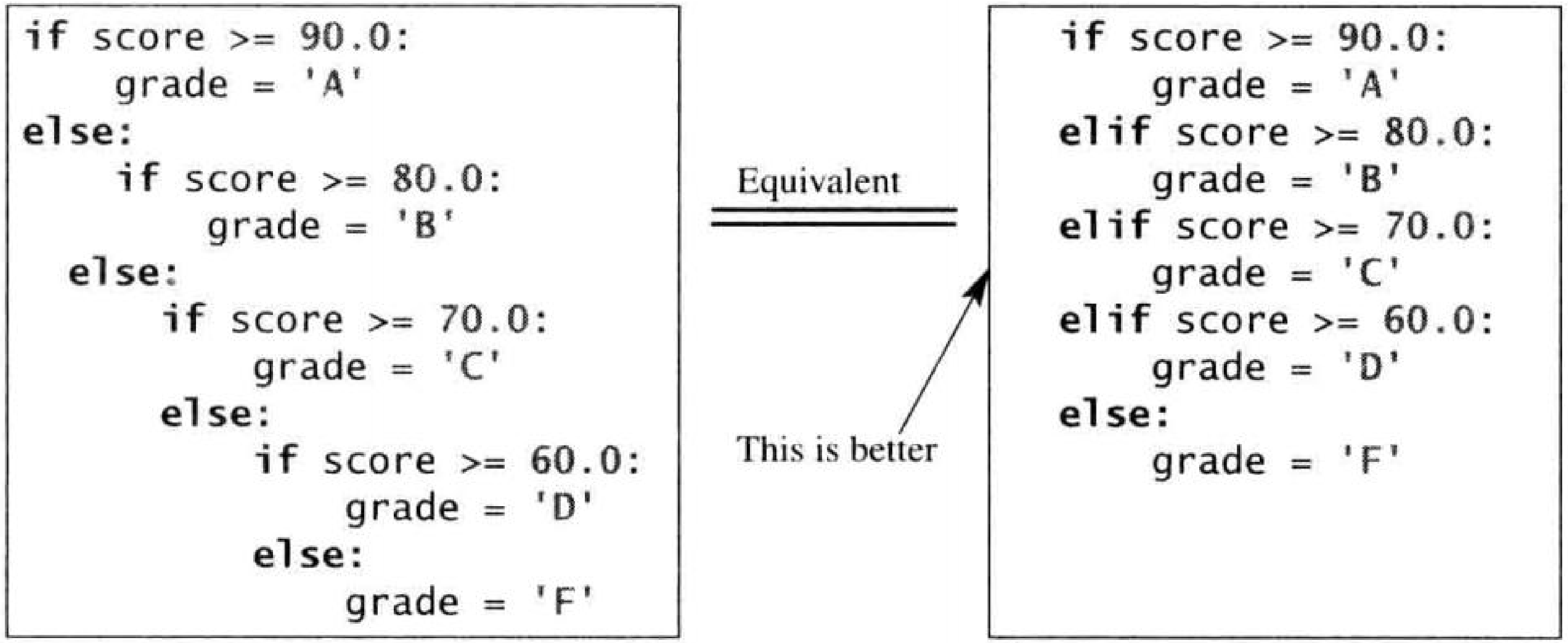
- 双向if-else语句

```
if boolean-expression:
    statement(s)-for-the-true-case
else:
    statement(s)-for-the-false-case
```


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

选择结构

- 嵌套if和多向if-elif-else语句

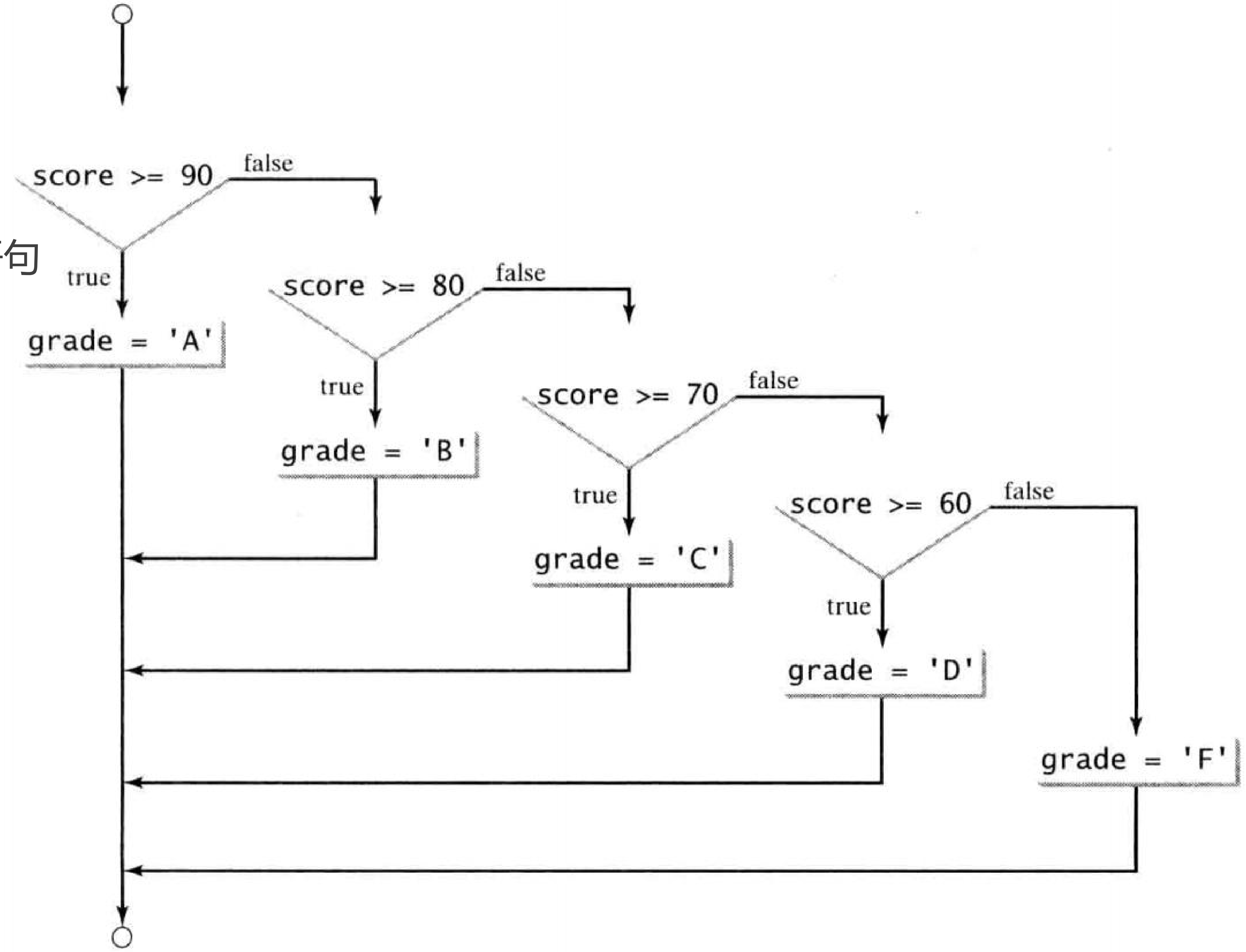


```
if score >= 90.0:
    grade = 'A'
else:
    if score >= 80.0:
        grade = 'B'
  else:
        if score >= 70.0:
            grade = 'C'
        else:
            if score >= 60.0:
                grade = 'D'
            else:
                grade = 'F'
```

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

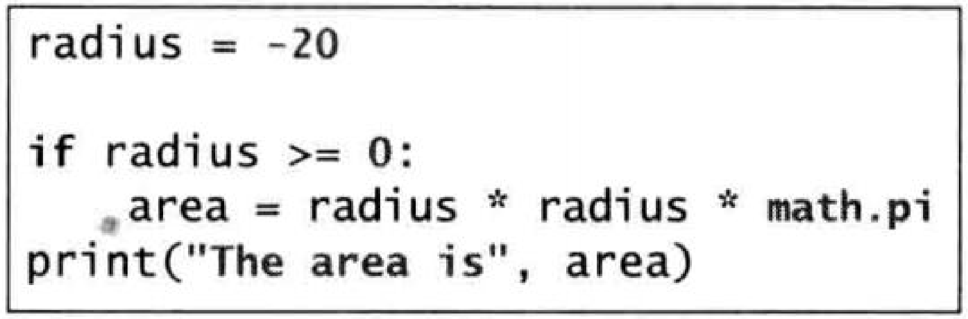
选择结构

- 嵌套if和多向if-elif-else语句

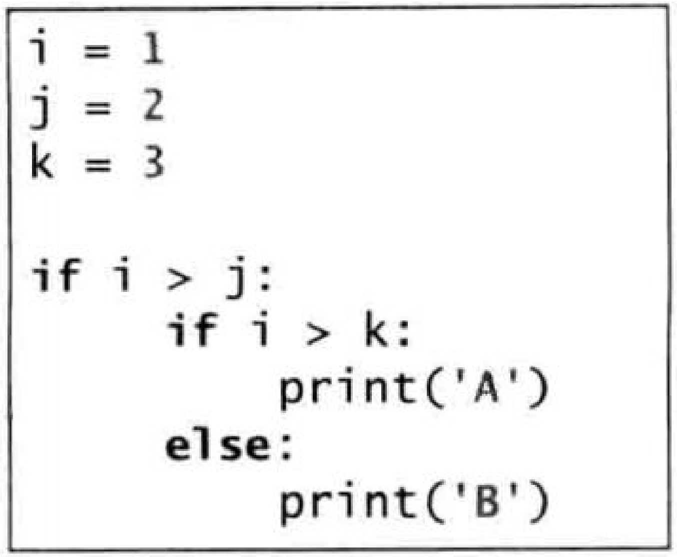


选择结构

- 选择语句中的常见错误
  - 不正确的缩进



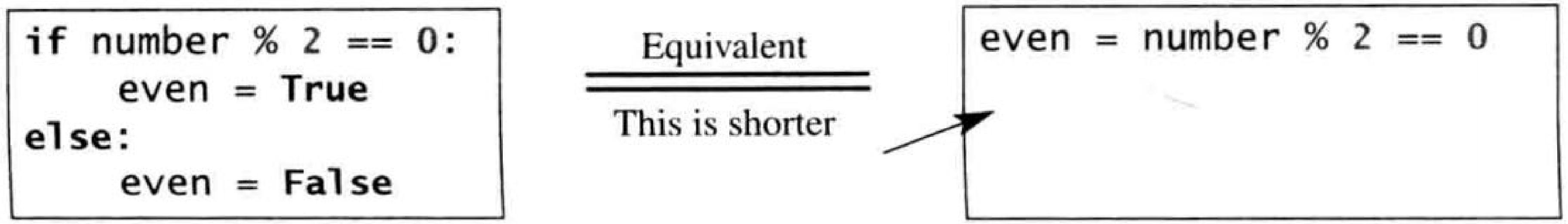
```
radius = -20

if radius >= 0:
    area = radius * radius * math.pi
print("The area is", area)
```



```
i = 1
j = 2
k = 3

if i > j:
    if i > k:
        print('A')
    else:
        print('B')
```

选择结构

- 代码的简化

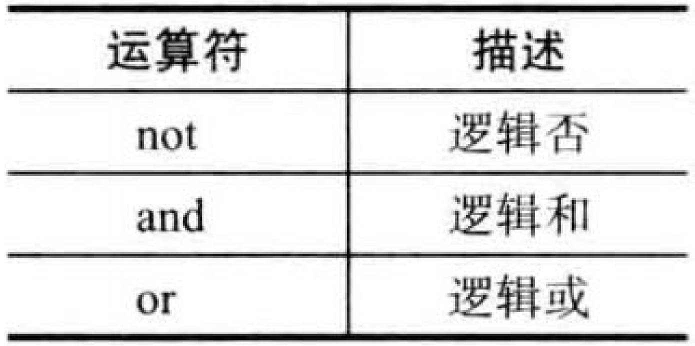


```
if number % 2 == 0:
    even = True
else:
    even = False
```

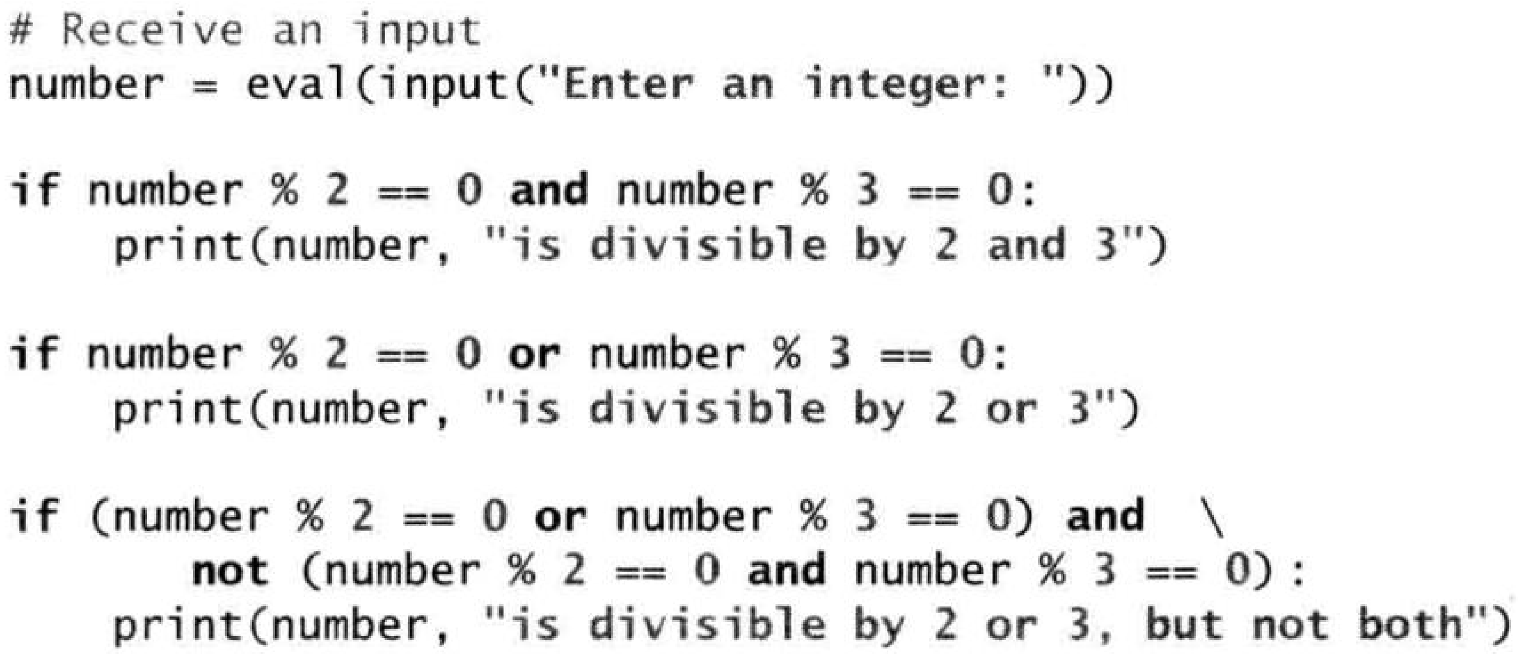
```
even = number % 2 == 0
```

选择结构

- 逻辑运算符（布尔运算符）



运算符	描述
not	逻辑否
and	逻辑和
or	逻辑或



```
# Receive an input
number = eval(input("Enter an integer: "))

if number % 2 == 0 and number % 3 == 0:
    print(number, "is divisible by 2 and 3")

if number % 2 == 0 or number % 3 == 0:
    print(number, "is divisible by 2 or 3")

if (number % 2 == 0 or number % 3 == 0) and  \
       not (number % 2 == 0 and number % 3 == 0):
    print(number, "is divisible by 2 or 3, but not both")
```

选择结构

- 条件表达式

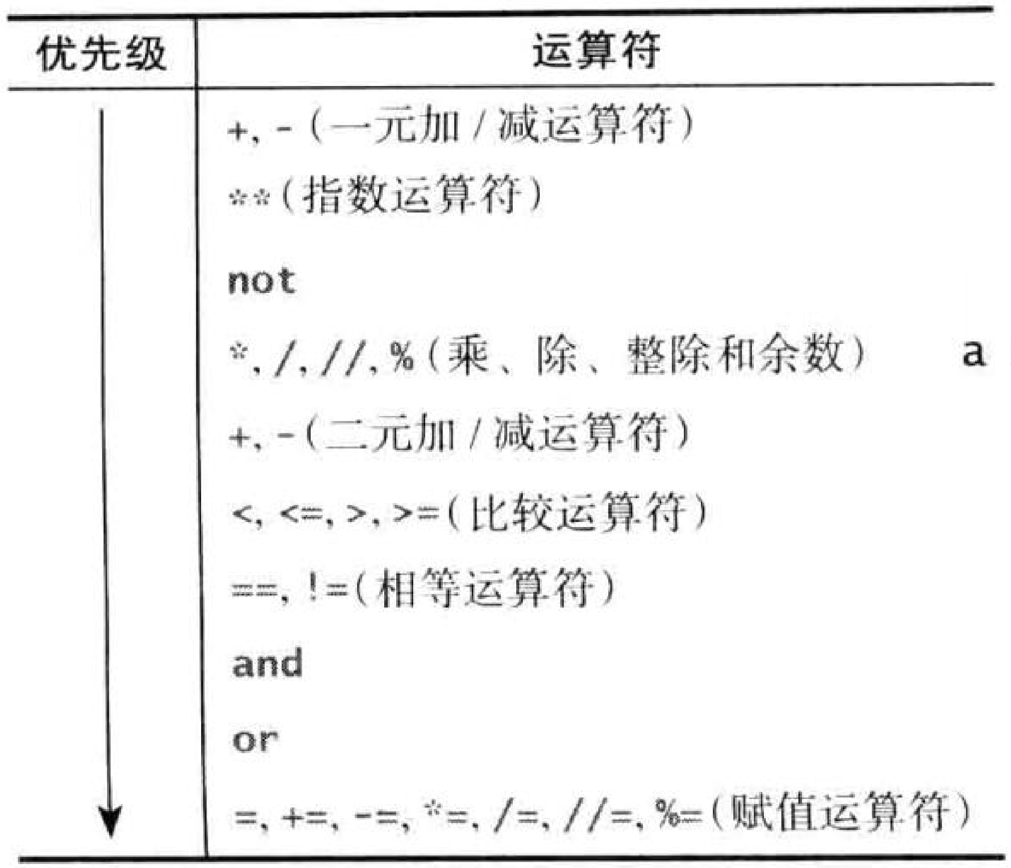
```
expression1 if boolean-expression else expression2
```

```
if x > 0:
    y = 1
else:
    y = -1
```

```
y = 1 if x > 0 else -1
```

选择结构

- 运算符的优先级和从左往右的结合顺序



优先级	运算符
↓	+, -（一元加 / 减运算符）
	**（指数运算符）
	not
	*, /, //, %（乘、除、整除和余数）
	+, -（二元加 / 减运算符）
	<, <=, >, >=（比较运算符）
	==, !=（相等运算符）
	and
	or
	=, +=, -=, *=, /=, //=, %=（赋值运算符）

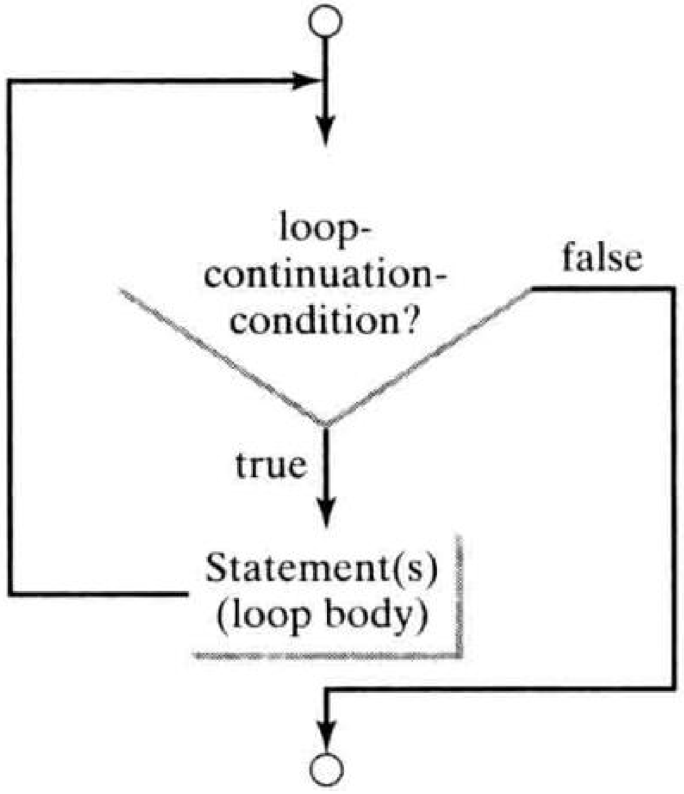
$$a - b + c - d \overset{\text{相等于}}{=\!=\!=\!=} ((a - b) + c) - d$$

循环结构

- while循环

```
while loop-continuation-condition:
    # Loop body
    Statement(s)
```

```
while loop-continuation-condition:
    Statements
    Additional statements for controlling the loop
```



循环结构

- 常见错误1：

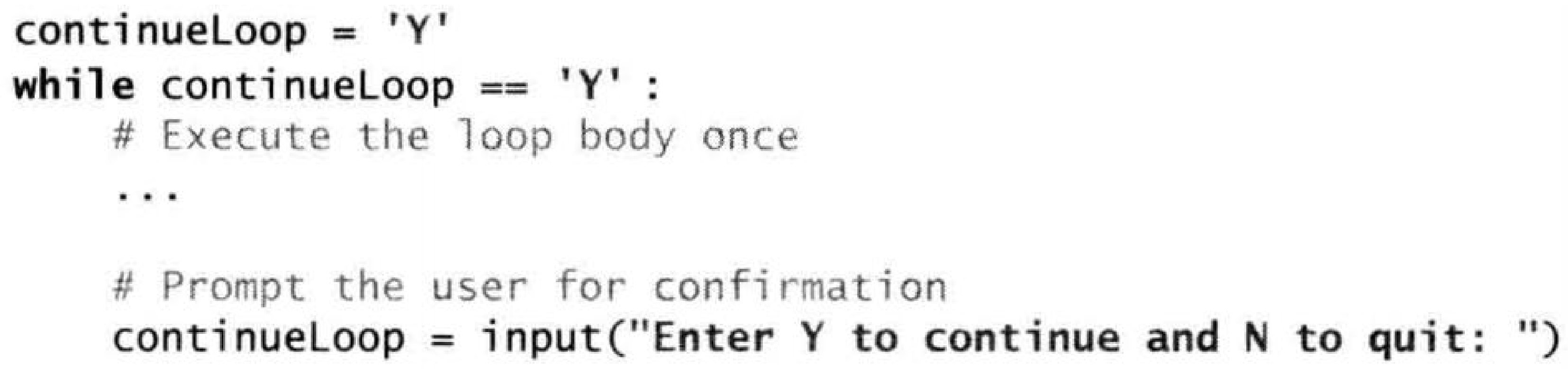
```
sum = 0
i = 1
while i < 10:
    sum = sum + i
i = i + 1
```

- 常见错误2：

```
count = 0
while count <= 100 :
    print("Programming is fun!")
    count = count + 1
```

循环结构

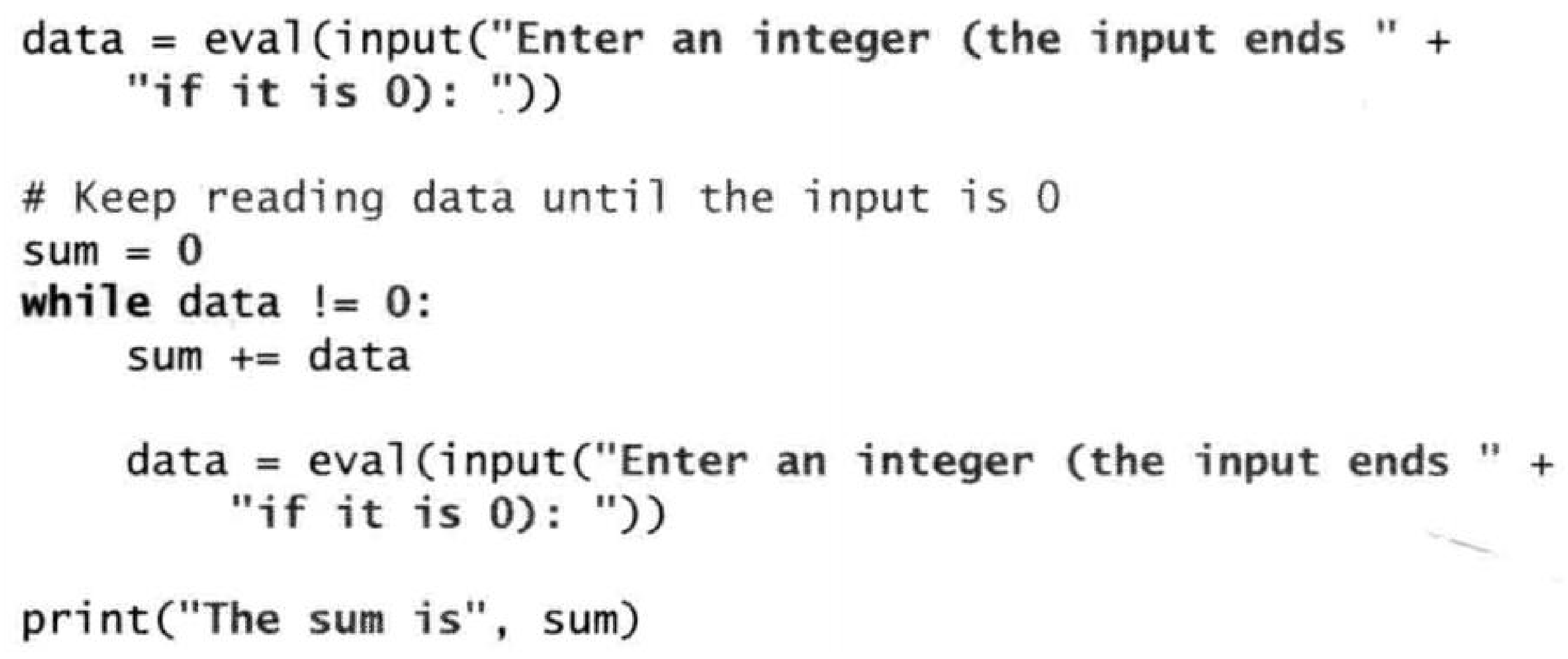
- 用户确认循环控制



```
continueLoop = 'Y'
while continueLoop == 'Y' :
    # Execute the loop body once
    ...

    # Prompt the user for confirmation
    continueLoop = input("Enter Y to continue and N to quit: ")
```

循环结构

- 哨式控制



```
data = eval(input("Enter an integer (the input ends " +
    "if it is 0): "))

# Keep reading data until the input is 0
sum = 0
while data != 0:
    sum += data

    data = eval(input("Enter an integer (the input ends " +
        "if it is 0): "))

print("The sum is", sum)
```

循环结构

- 常见错误3：

```
item = 1
sum = 0

while item != 0:  # No guarantee item will be 0
    sum += item
    item -= 0.1

 print(sum)
```

循环结构

- 输入输出重定向指令

```
python SentinelValue.py < input.txt

python Script.py > output.txt
```

或者

```
python SentinelValue.py < input.txt > output.txt
```

循环结构

- for循环
  - 计数器控制的循环

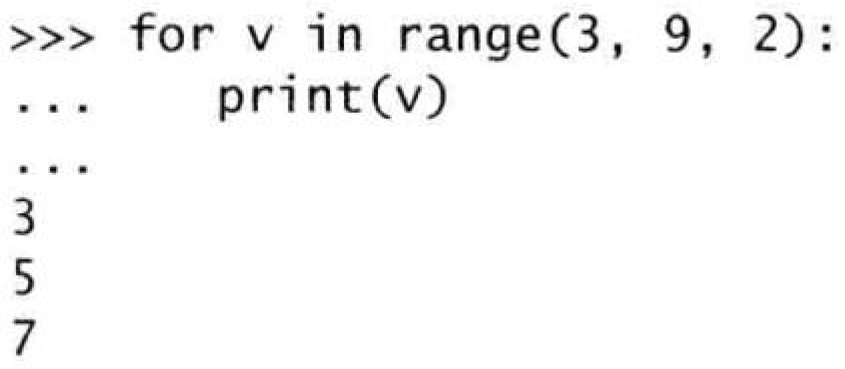
```
i = initialValue  # Initialize loop-control variable
while i < endValue:
    # Loop body
    ...
    i += 1  # Adjust loop-control variable
```

↕

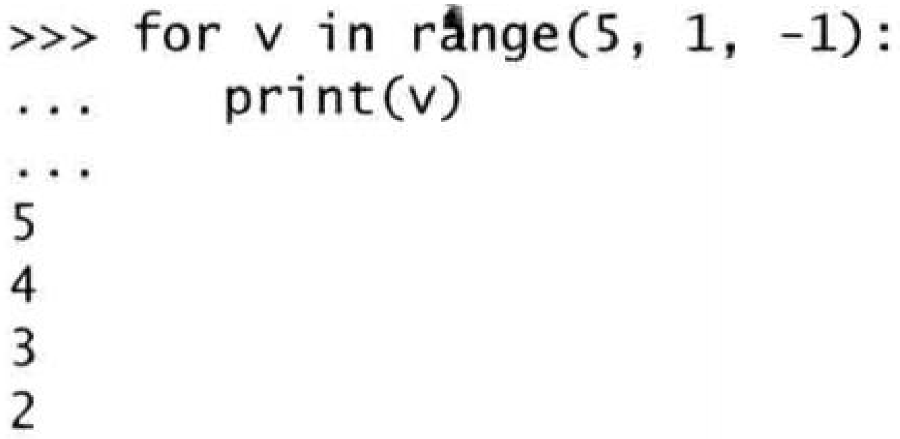
```
for i in range(initialValue, endValue):
    # Loop body
```

循环结构

- 例程：



```
>>> for v in range(3, 9, 2):
...     print(v)
...
3
5
7
```



```
>>> for v in range(5, 1, -1):
...     print(v)
...
5
4
3
2
```

循环结构

- 循环嵌套

```
for i in range(1000):
    for j in range(1000):
        for k in range(1000):
            Perform an action
```

```
# Display table body
for i in range(1, 10):
    print(i, "|", end = '')
    for j in range(1, 10):
        # Display the product and align properly
        print(format(i * j, "4d"), end = '')
    print()  # Jump to the new line
```

循环结构

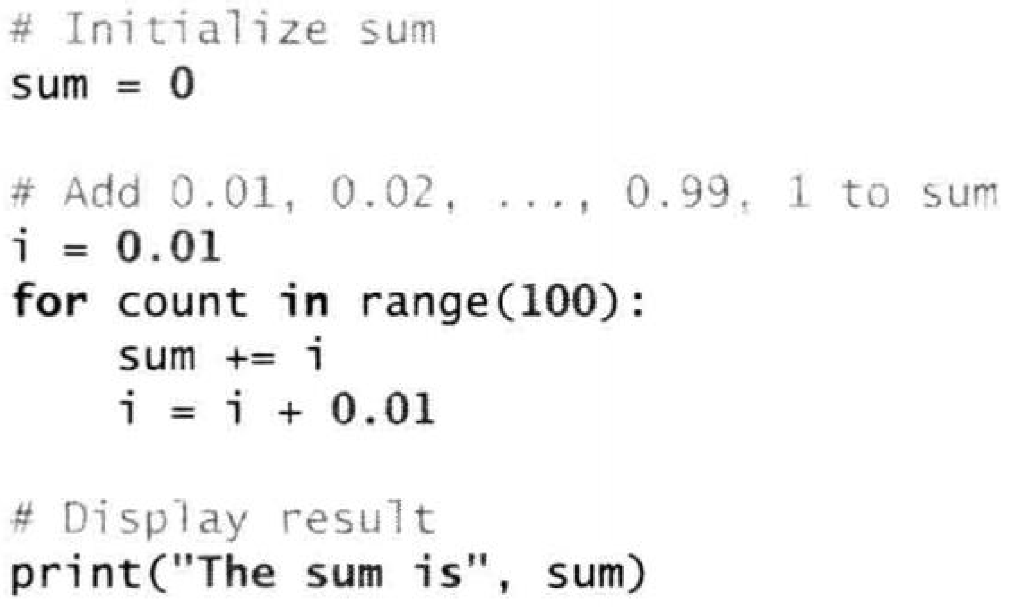
- 最小化数值错误

```
# Initialize sum
sum = 0

# Add 0.01, 0.02, ..., 0.99, 1 to sum
i = 0.01
while i <= 1.0:
    sum += i
    i = i + 0.01

# Display result
print("The sum is", sum)
```

```
The sum is 49.5
```

循环结构

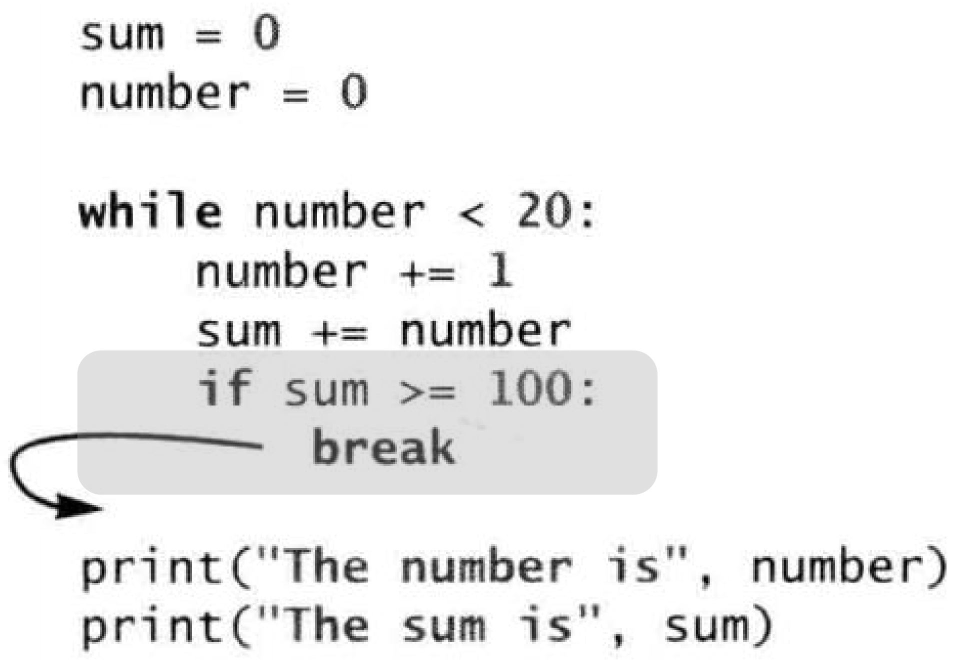
- 消除浮点数运算误差



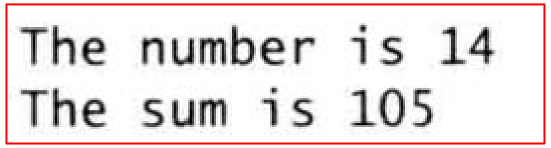
```
# Initialize sum
sum = 0

# Add 0.01, 0.02, ..., 0.99, 1 to sum
i = 0.01
for count in range(100):
    sum += i
    i = i + 0.01

# Display result
print("The sum is", sum)
```

循环结构

- 关键字break和continue
  - break终止循环



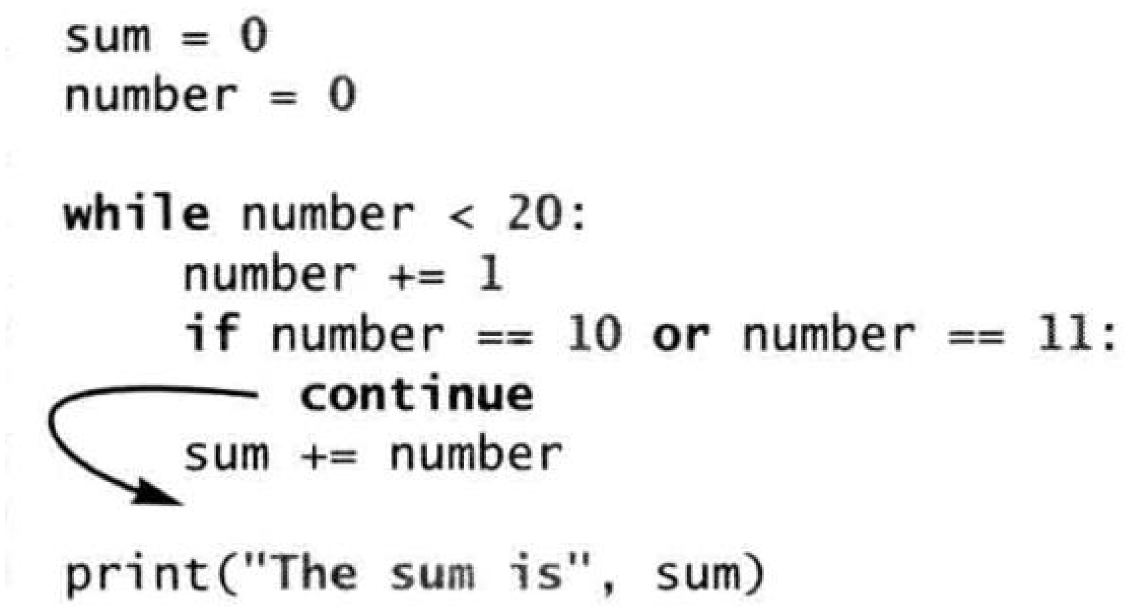
```
sum = 0
number = 0

while number < 20:
    number += 1
    sum += number
    if sum >= 100:
        break

print("The number is", number)
print("The sum is", sum)
```



```
The number is 14
The sum is 105
```

循环结构

- continue终止当前迭代

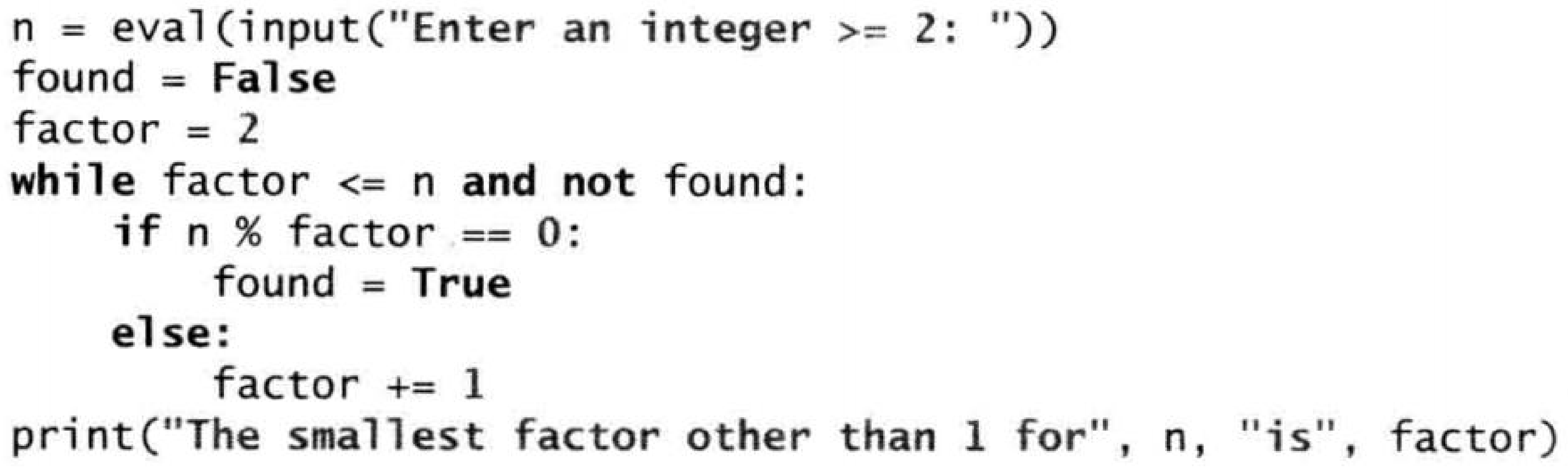


```
sum = 0
number = 0

while number < 20:
    number += 1
    if number == 10 or number == 11:
        continue
    sum += number

print("The sum is", sum)
```

```
The sum is 189
```

循环结构

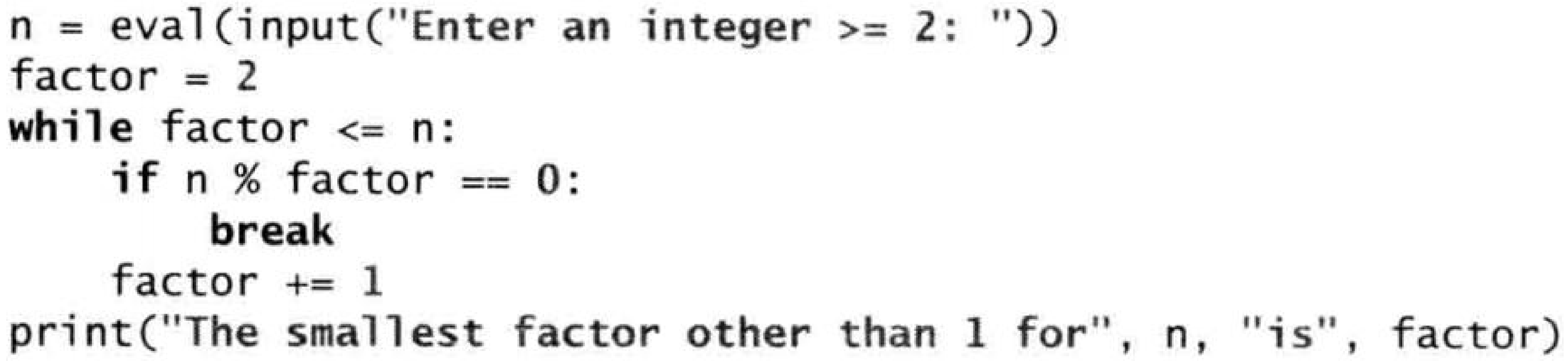
- 例程对比：



```
n = eval(input("Enter an integer >= 2: "))
found = False
factor = 2
while factor <= n and not found:
    if n % factor == 0:
        found = True
    else:
        factor += 1
print("The smallest factor other than 1 for", n, "is", factor)
```

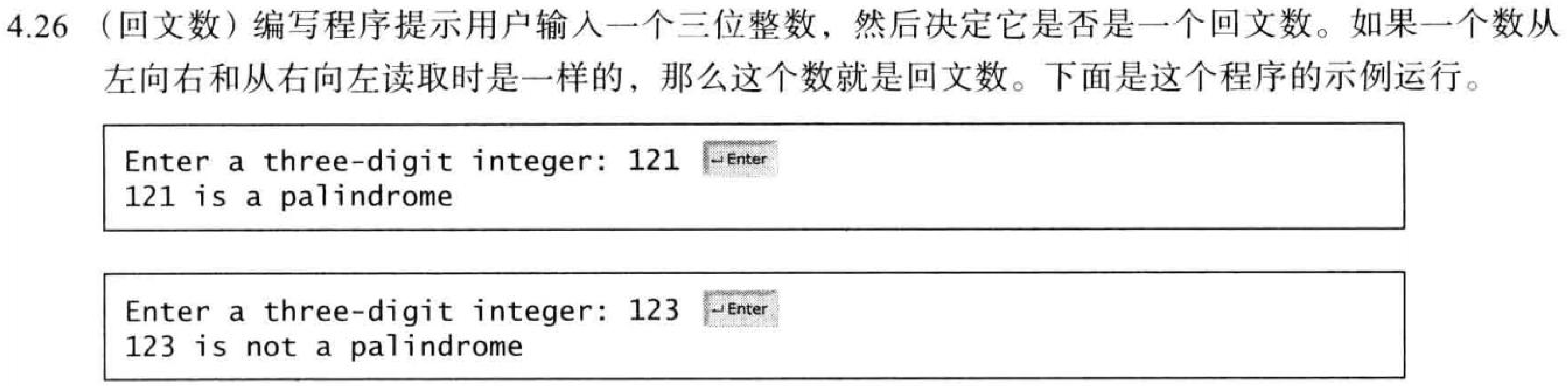
循环结构

- 例程对比：加入break



```
n = eval(input("Enter an integer >= 2: "))
factor = 2
while factor <= n:
    if n % factor == 0:
        break
    factor += 1
print("The smallest factor other than 1 for", n, "is", factor)
```

程序作业

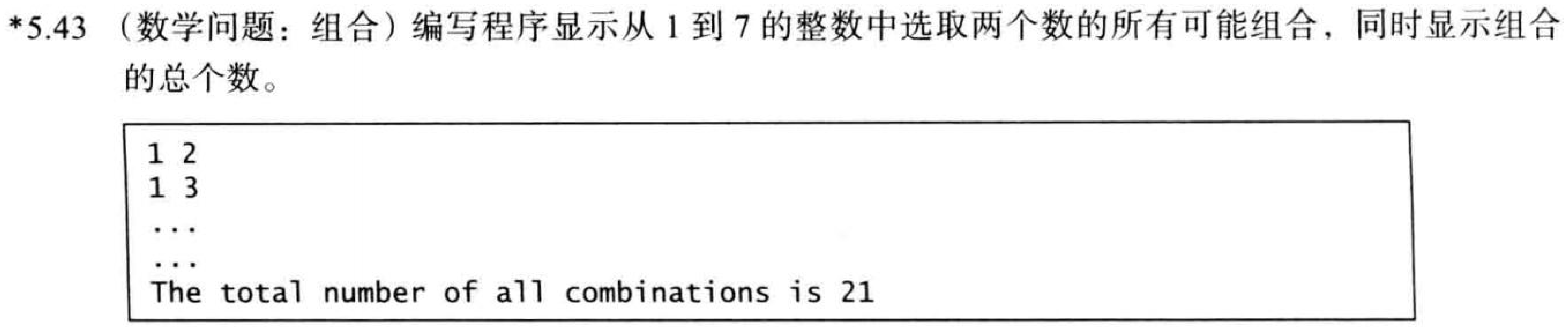


4.26 （回文数）编写程序提示用户输入一个三位整数，然后决定它是否是一个回文数。如果一个数从左向右和从右向左读取时是一样的，那么这个数就是回文数。下面是这个程序的示例运行。

```
Enter a three-digit integer: 121 ↵Enter
121 is a palindrome
```

```
Enter a three-digit integer: 123 ↵Enter
123 is not a palindrome
```

程序作业



*5.43 （数学问题：组合）编写程序显示从 1 到 7 的整数中选取两个数的所有可能组合，同时显示组合的总个数。

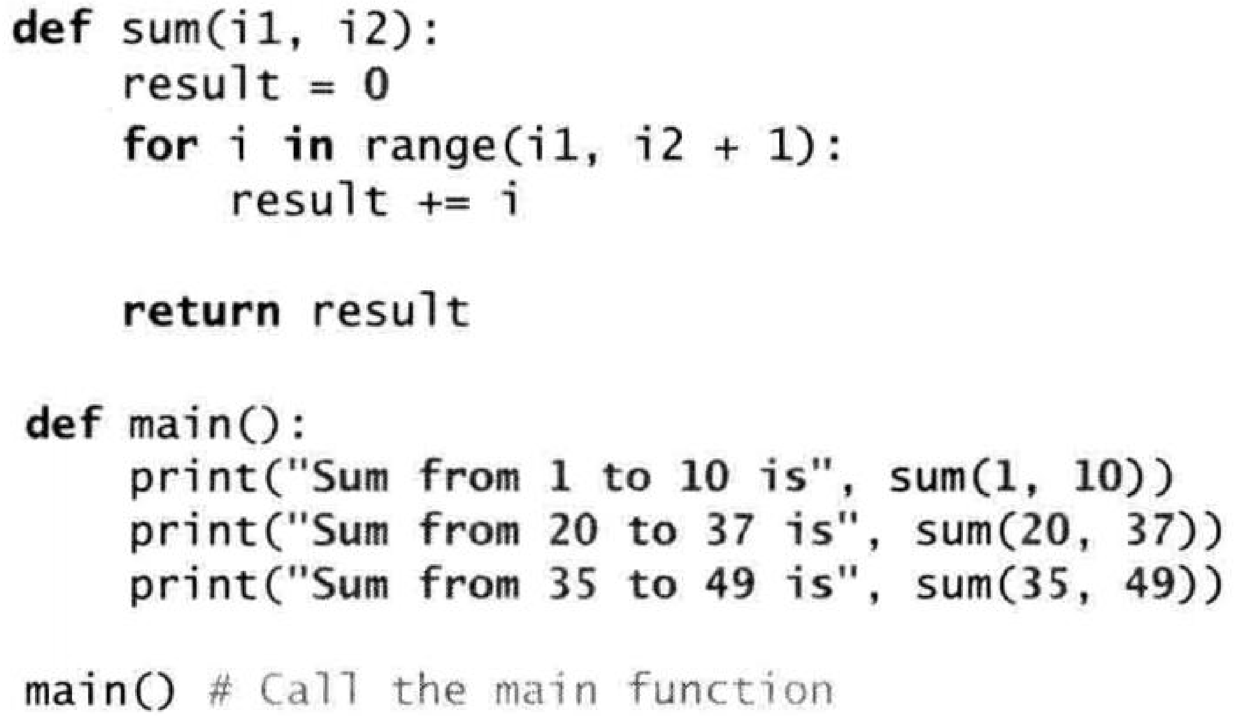
```
1 2
1 3
...
...
The total number of all combinations is 21
```

函数

- 函数：用来定义可重用代码、组织和简化代码

```
sum = 0
for i in range(1, 11):
    sum += i
print("Sum from 1 to 10 is", sum)

sum = 0
for i in range(20, 38):
    sum += i
print("Sum from 20 to 37 is", sum)

sum = 0
for i in range(35, 50):
    sum += i
print("Sum from 35 to 49 is", sum)
```

函数

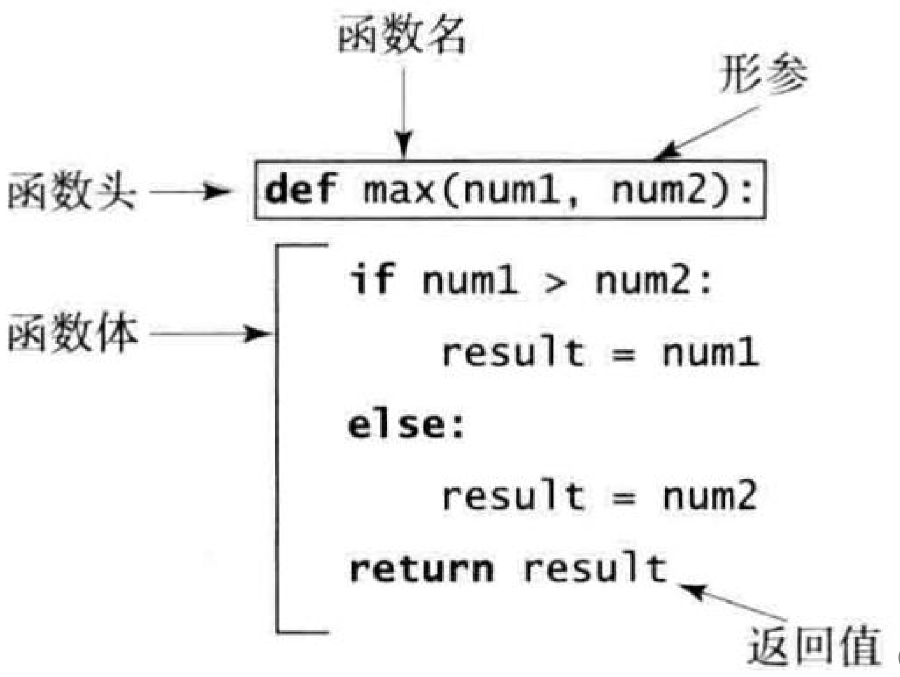


```
def sum(i1, i2):
    result = 0
    for i in range(i1, i2 + 1):
        result += i

    return result

def main():
    print("Sum from 1 to 10 is", sum(1, 10))
    print("Sum from 20 to 37 is", sum(20, 37))
    print("Sum from 35 to 49 is", sum(35, 49))

main() # Call the main function
```

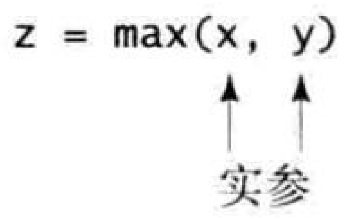
函数

- 定义和调用一个函数

```
def functionName(list of parameters)
    # Function body
```



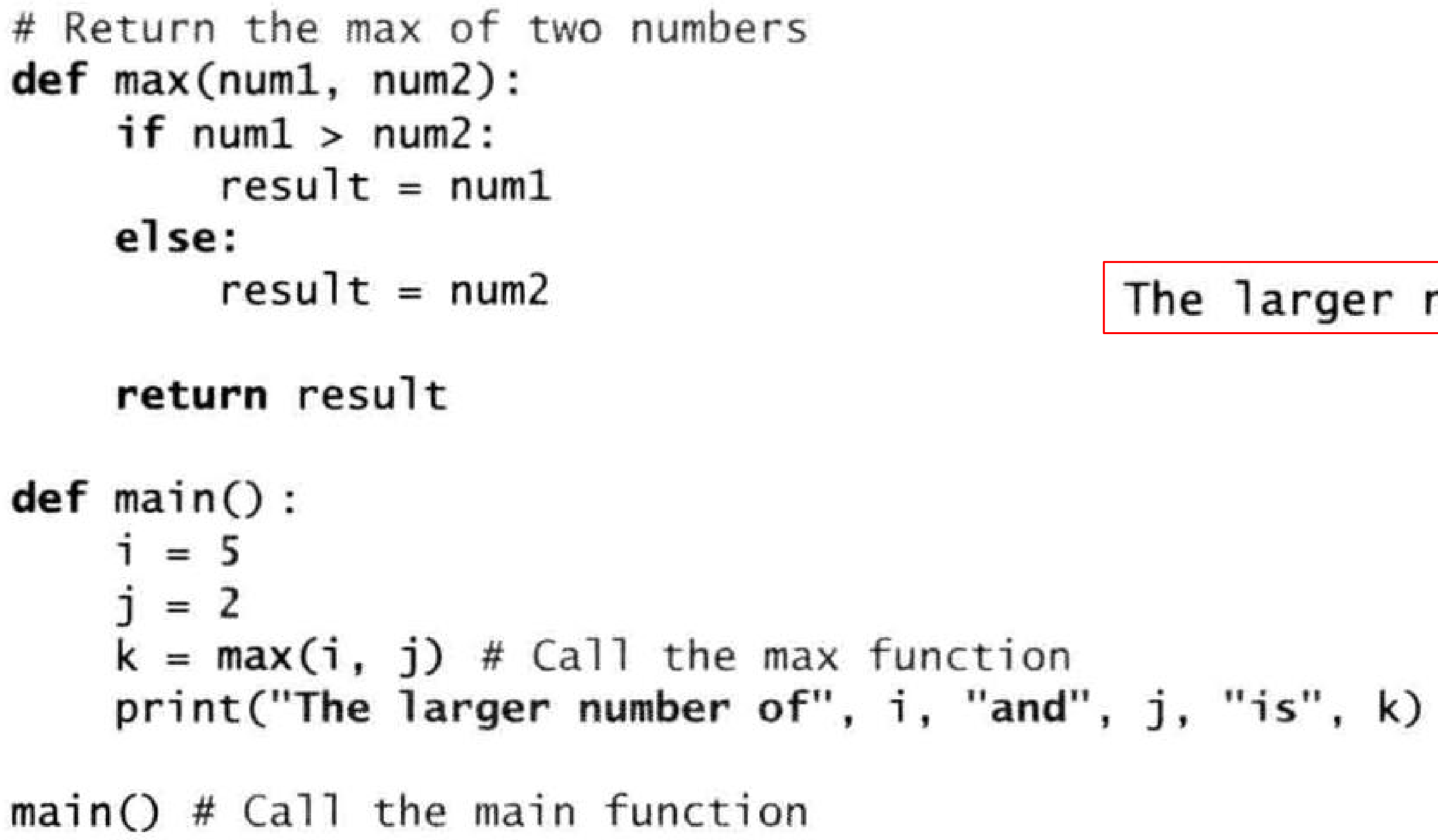
```
def max(num1, num2):
    if num1 > num2:
        result = num1
    else:
        result = num2
    return result
```



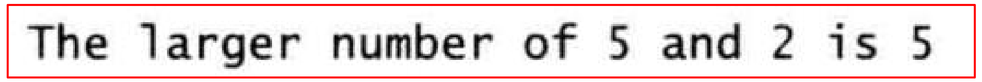
```
z = max(x, y)
```

函数

- main()函数调用

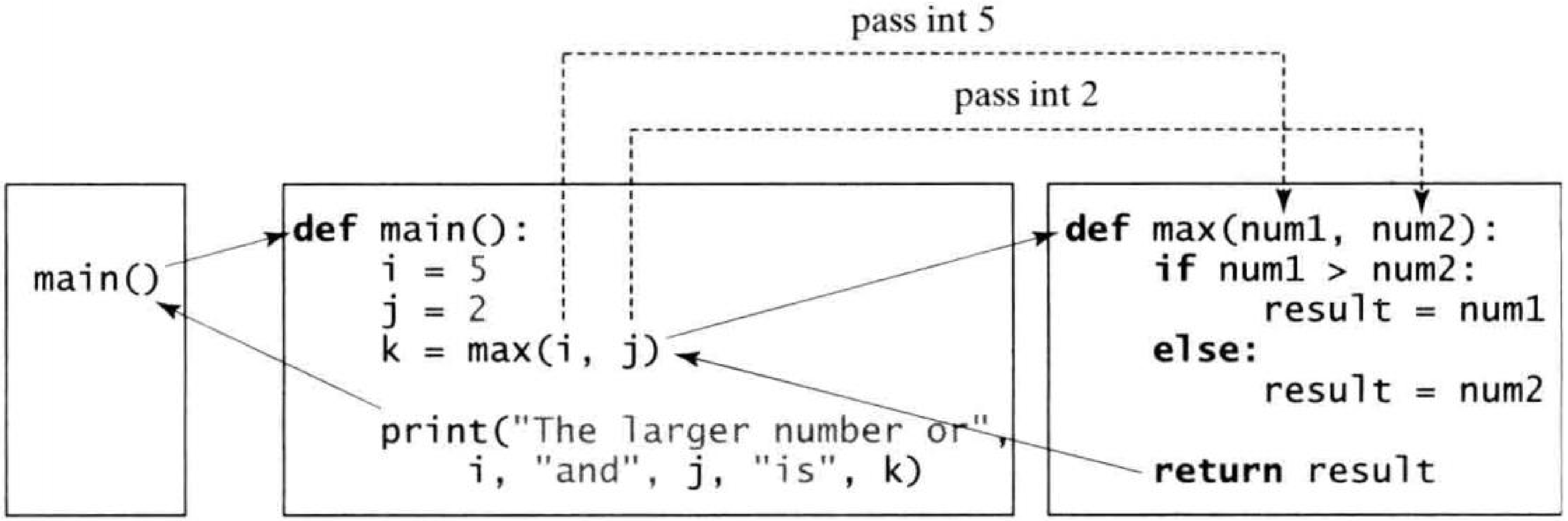


```
# Return the max of two numbers
def max(num1, num2):
    if num1 > num2:
        result = num1
    else:
        result = num2

    return result

def main():
    i = 5
    j = 2
    k = max(i, j) # Call the max function
    print("The larger number of", i, "and", j, "is", k)

main() # Call the main function
```



```
The larger number of 5 and 2 is 5
```

函数

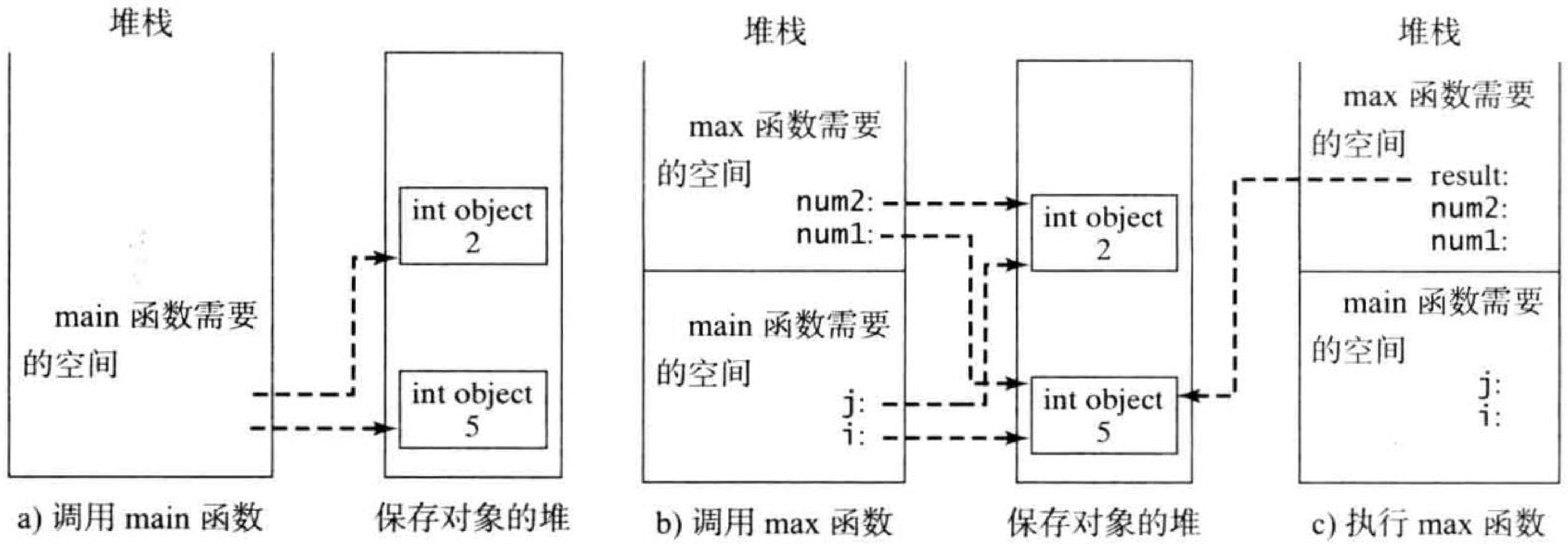


```
def main():
    i = 5
    j = 2
    k = max(i, j)

    print("The larger number of",
        i, "and", j, "is", k)
```

```
def max(num1, num2):
    if num1 > num2:
        result = num1
    else:
        result = num2

    return result
```

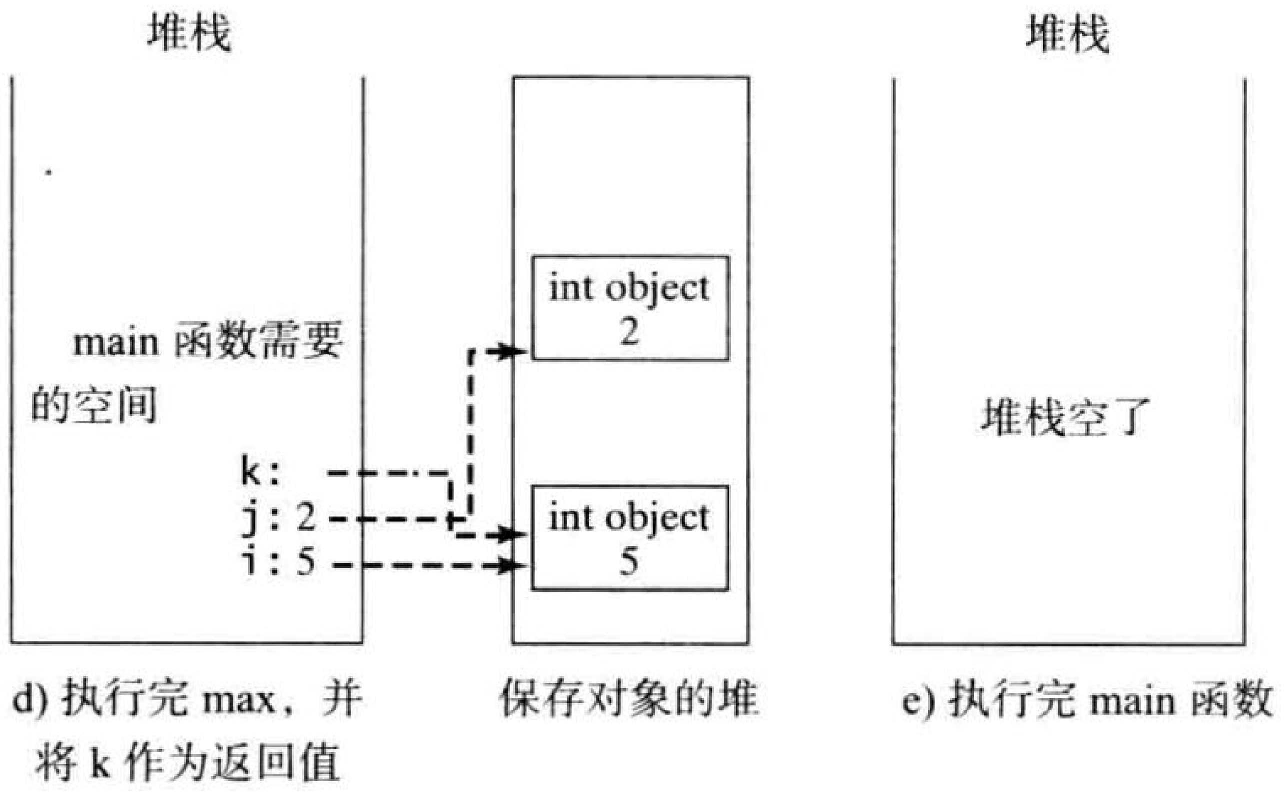
函数

- 调用栈
  - 激活记录，后进先出



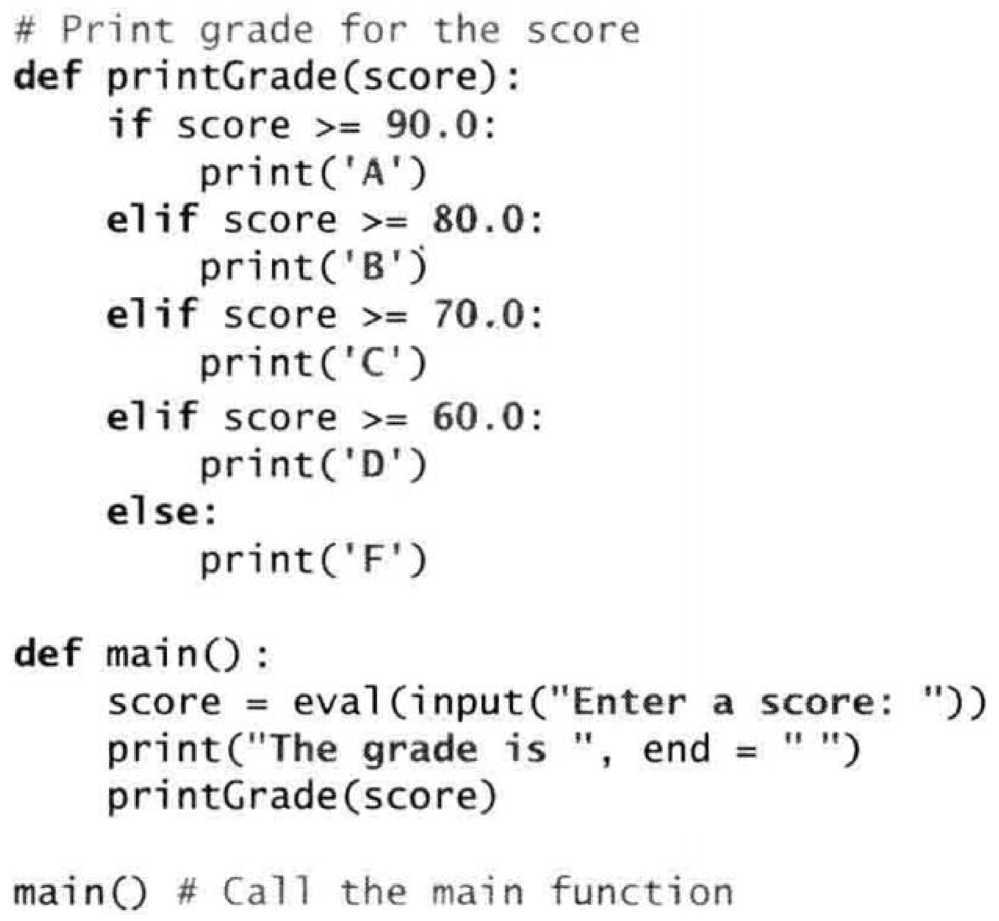
函数

- Python自动清空堆中对象



函数

- 无返回值的函数（None函数）的使用
  - 当做一个语句调用



```
# Print grade for the score
def printGrade(score):
    if score >= 90.0:
        print('A')
    elif score >= 80.0:
        print('B')
    elif score >= 70.0:
        print('C')
    elif score >= 60.0:
        print('D')
    else:
        print('F')

def main():
    score = eval(input("Enter a score: "))
    print("The grade is ", end = " ")
    printGrade(score)

main() # Call the main function
```

函数

- 赋值给变量，不指向任何对象

```
def sum(number1, number2):
    total = number1 + number2

print(sum(1, 2))
```

```
None
```

- return语句：改变函数正常流程

```
# Print grade for the score
def printGrade(score):
    if score < 0 or score > 100:
        print("Invalid score")
        return # Same as return None
```

函数

- 实参：位置参数和关键字参数

定义：

```
def nPrintln(message, n):
    for i in range(n):
        print(message)
```

调用1：

```
nPrintln('a',3)
```

调用2：

```
nPlintln(n=5,message="good")
```

- 位置参数不能出现在关键字参数之后

函数

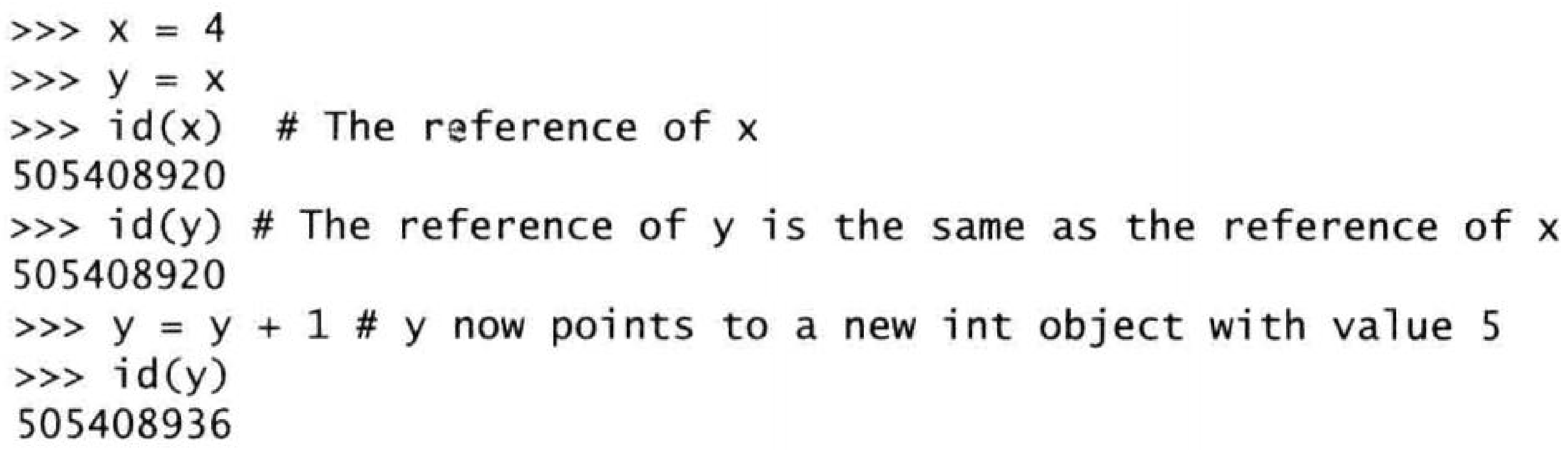
- 通过传引用来传递参数
  - 实参的引用值被传递给形参

```
def main():
    x = 1
    print("Before the call, x is", x)
    increment(x)
    print("After the call, x is", x)

def increment(n):
    n += 1
    print("\tn inside the function is", n)

main() # Call the main function
```

```
Before the call, x is 1
        n inside the function is 2
After the call, x is 1
```

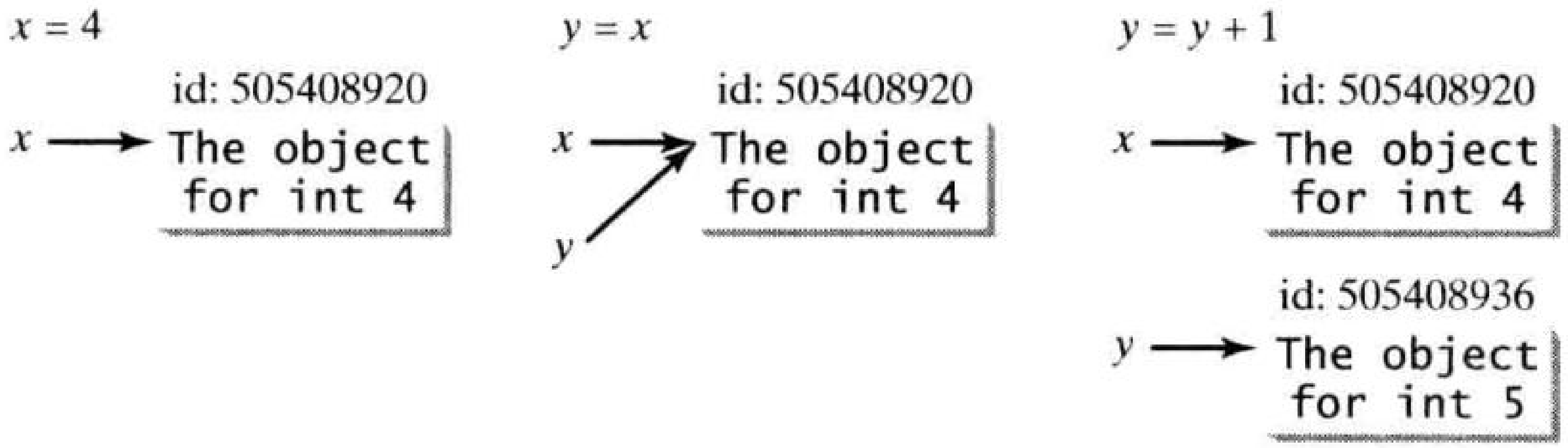
函数

- 不可变对象



```
>>> x = 4
>>> y = x
>>> id(x)  # The reference of x
505408920
>>> id(y) # The reference of y is the same as the reference of x
505408920
>>> y = y + 1 # y now points to a new int object with value 5
>>> id(y)
505408936
```

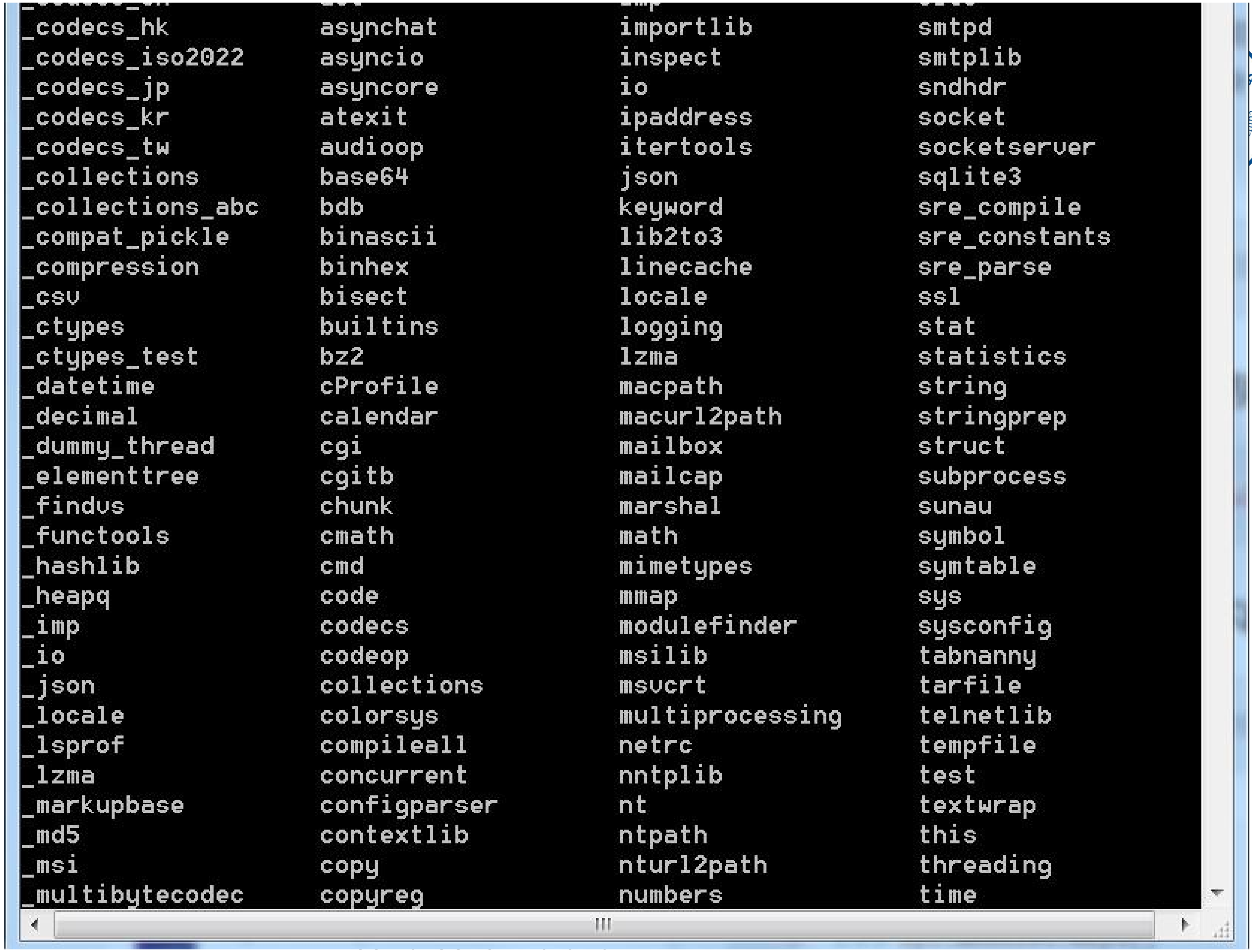
函数



$x = 4$
id: 505408920
x
The object for int 4
$y = x$
id: 505408920
x
y
The object for int 4
$y = y + 1$
id: 505408920
x
The object for int 4
id: 505408936
y
The object for int 5

函数

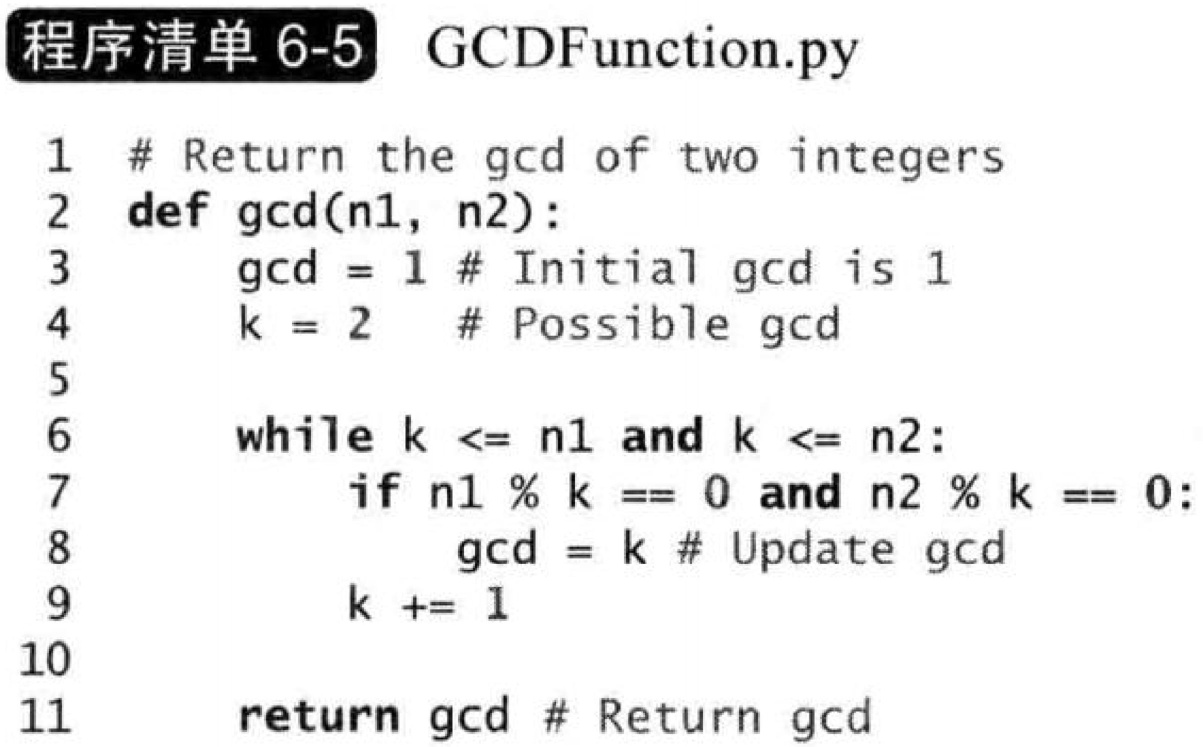
- 模块化代码
  - Python中的模块



```
_codecs_hk          asynchat            importlib           smtpd
_codecs_iso2022     asyncio             inspect             smtplib
_codecs_jp          asyncore            io                  sndhdr
_codecs_kr          atexit              ipaddress           socket
_codecs_tw          audioop             itertools           socketserver
_collections        base64              json                sqlite3
_collections_abc    bdb                 keyword             sre_compile
_compat_pickle      binascii            lib2to3             sre_constants
_compression        binhex              linecache           sre_parse
_csv                bisect              locale              ssl
_ctypes             builtins            logging             stat
_ctypes_test        bz2                 lzma                statistics
_datetime           cProfile            macpath             string
_decimal            calendar            macurl2path         stringprep
_dummy_thread       cgi                 mailbox             struct
_elementtree        cgitb               mailcap             subprocess
_findvs             chunk               marshal             sunau
_functools          cmath               math                symbol
_hashlib            cmd                 mimetypes           symtable
_heapq              code                mmap                sys
_imp                codecs              modulefinder        sysconfig
_io                 codeop              msilib              tabnanny
_json               collections         msvcrt              tarfile
_locale             colorsys            multiprocessing     telnetlib
_lsprof             compileall          netrc               tempfile
_lzma               concurrent          nntplib             test
_markupbase         configparser        nt                  textwrap
_md5                contextlib          ntpath              this
_msi                copy                nturl2path          threading
_multibytecodec     copyreg             numbers             time
```

函数

- 自定义模块



程序清单 6-5 GCDFunction.py

```python
# Return the gcd of two integers
def gcd(n1, n2):
    gcd = 1 # Initial gcd is 1
    k = 2   # Possible gcd

    while k <= n1 and k <= n2:
        if n1 % k == 0 and n2 % k == 0:
            gcd = k # Update gcd
        k += 1

    return gcd # Return gcd
```

函数

```
from GCDFunction import gcd # Import the gcd function

# Prompt the user to enter two integers
n1 = eval(input("Enter the first integer: "))
n2 = eval(input("Enter the second integer: "))

print("The greatest common divisor for", n1,
    "and", n2, "is", gcd(n1, n2))
```

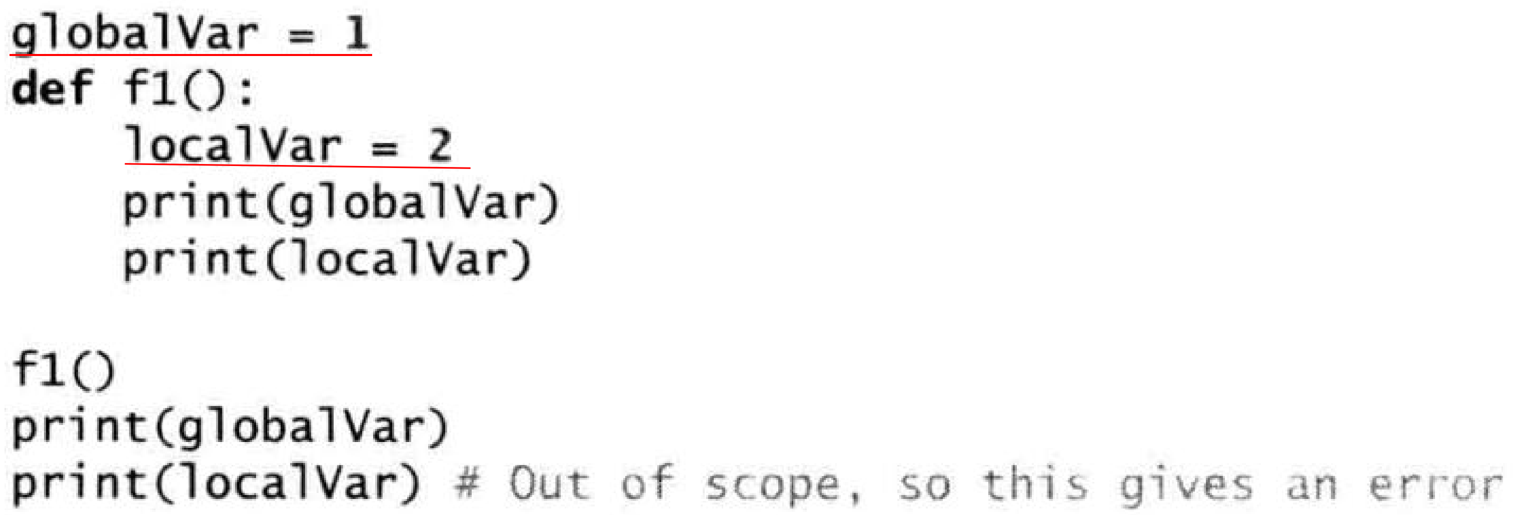
或：

```
import GCDFunction

GCDFuntion.gcd
```

函数

- 变量的作用域：该变量在程序中可以被引用的范围
  - 局部变量：函数内部定义、内部访问
  - 全局变量：函数之外创建、被所有函数访问

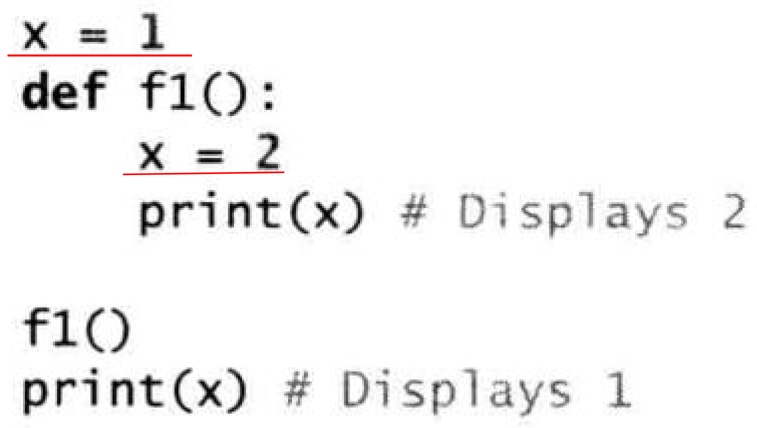
例程1：全局变量与局部变量的适用范围



```
globalVar = 1
def f1():
    localVar = 2
    print(globalVar)
    print(localVar)

f1()
print(globalVar)
print(localVar) # Out of scope, so this gives an error
```

函数

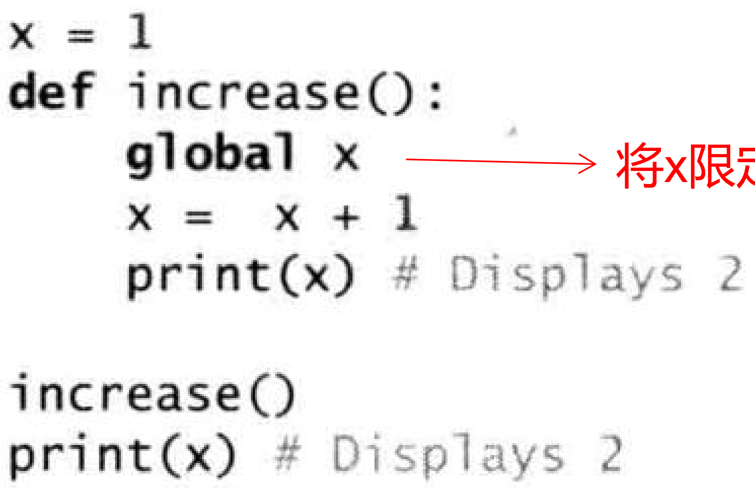
例程2：全局变量与局部变量重名



```
x = 1
def f1():
    x = 2
    print(x) # Displays 2

f1()
print(x) # Displays 1
```

函数

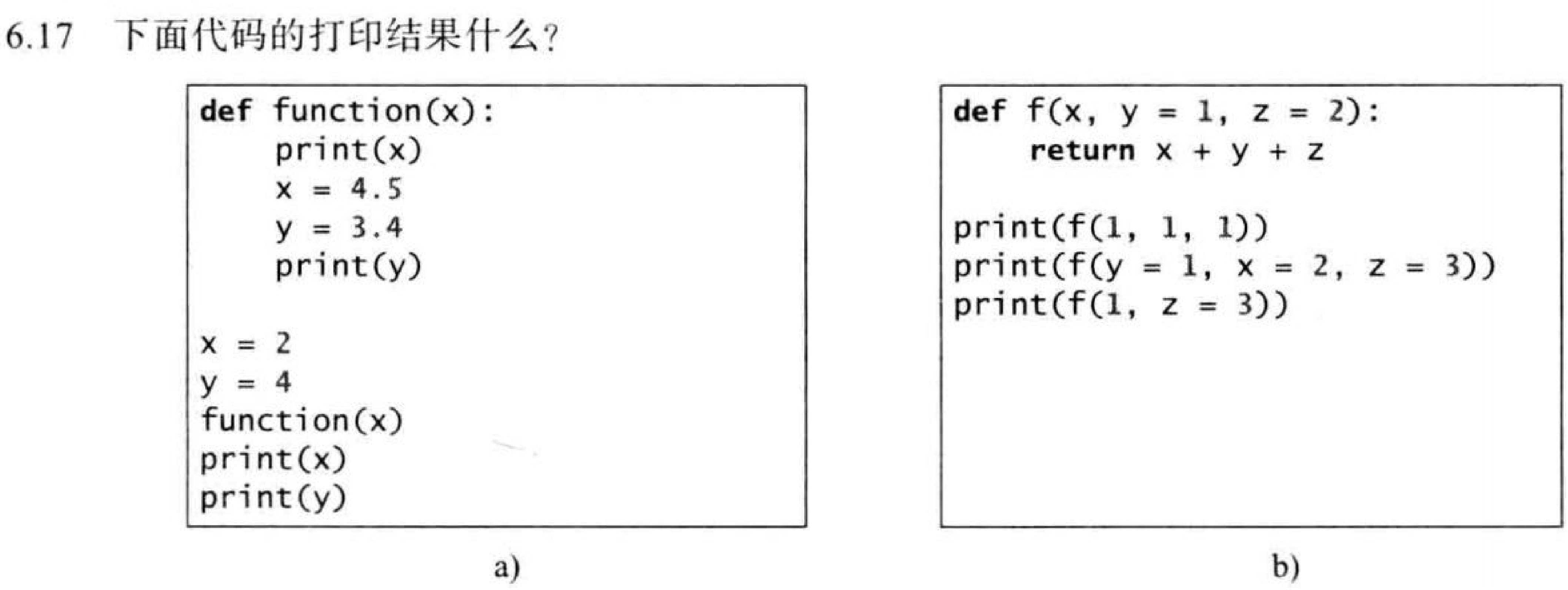
- 将一个局部变量绑定为全局



```
x = 1
def increase():
    global x
    x =  x + 1
    print(x) # Displays 2

increase()
print(x) # Displays 2
```

将x限定在函数中

函数



6.17 下面代码的打印结果什么？

```
def function(x):
    print(x)
    x = 4.5
    y = 3.4
    print(y)

x = 2
y = 4
function(x)
print(x)
print(y)
```

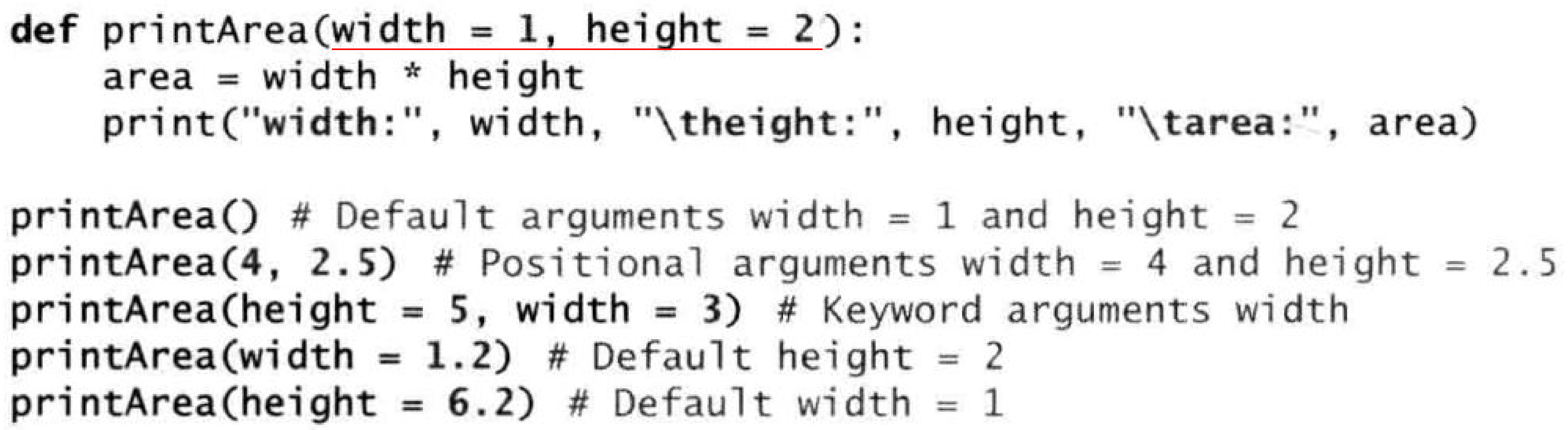
a)

```
def f(x, y = 1, z = 2):
    return x + y + z

print(f(1, 1, 1))
print(f(y = 1, x = 2, z = 3))
print(f(1, z = 3))
```

b)

函数

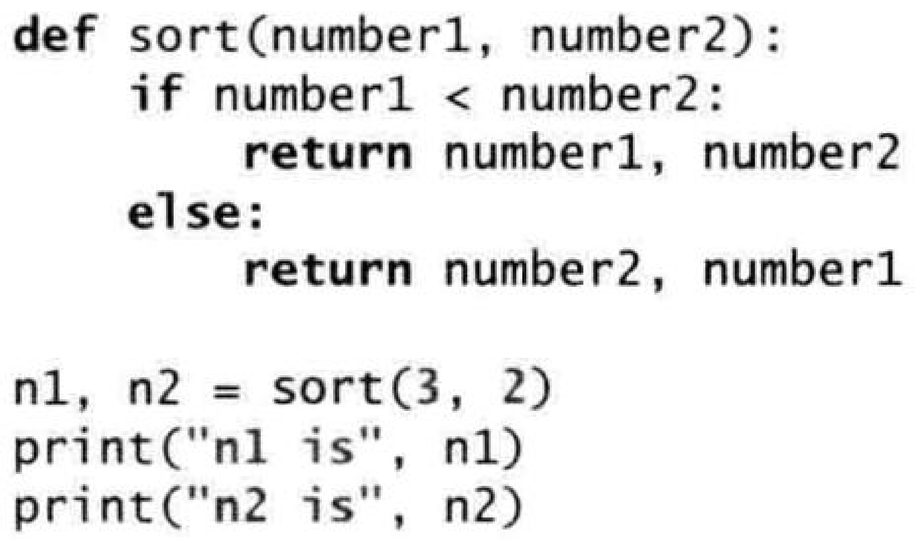
- 默认参数



```
def printArea(width = 1, height = 2):
    area = width * height
    print("width:", width, "\theight:", height, "\tarea:", area)

printArea() # Default arguments width = 1 and height = 2
printArea(4, 2.5) # Positional arguments width = 4 and height = 2.5
printArea(height = 5, width = 3) # Keyword arguments width
printArea(width = 1.2) # Default height = 2
printArea(height = 6.2) # Default width = 1
```

函数

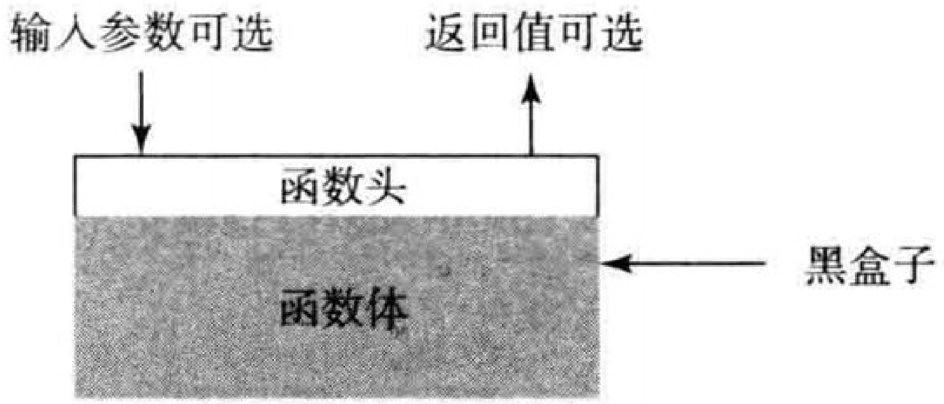
- 返回多个值



```
def sort(number1, number2):
    if number1 < number2:
        return number1, number2
    else:
        return number2, number1

n1, n2 = sort(3, 2)
print("n1 is", n1)
print("n2 is", n2)
```

函数

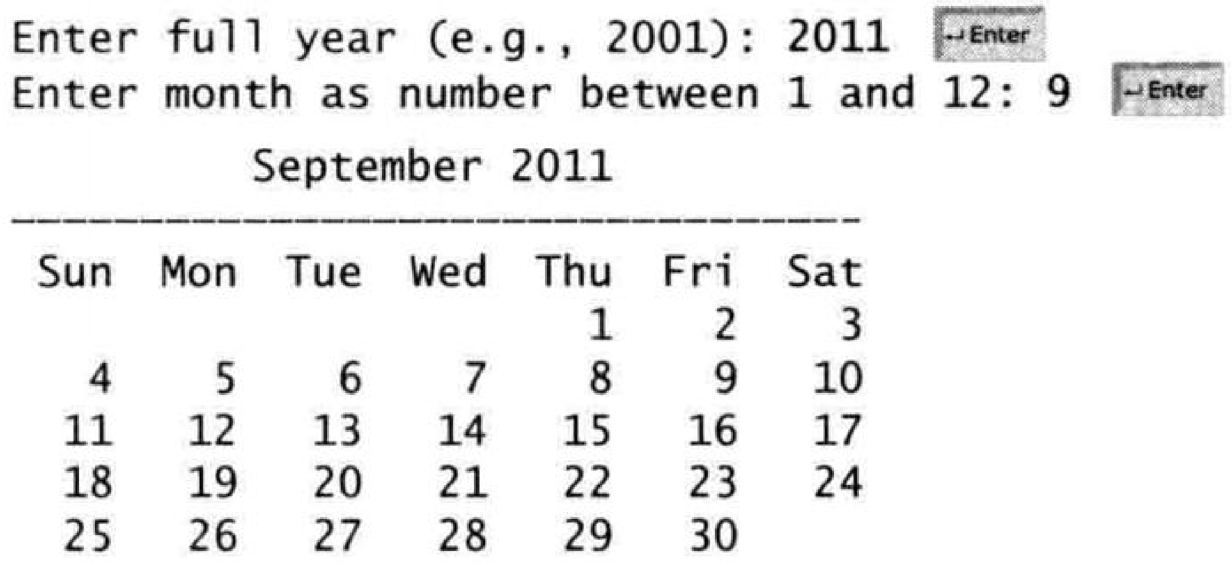
- 函数抽象和逐步求精
  - 函数的抽象：将函数的使用和函数的实现分开来



  - 逐步求精：分治策略

函数

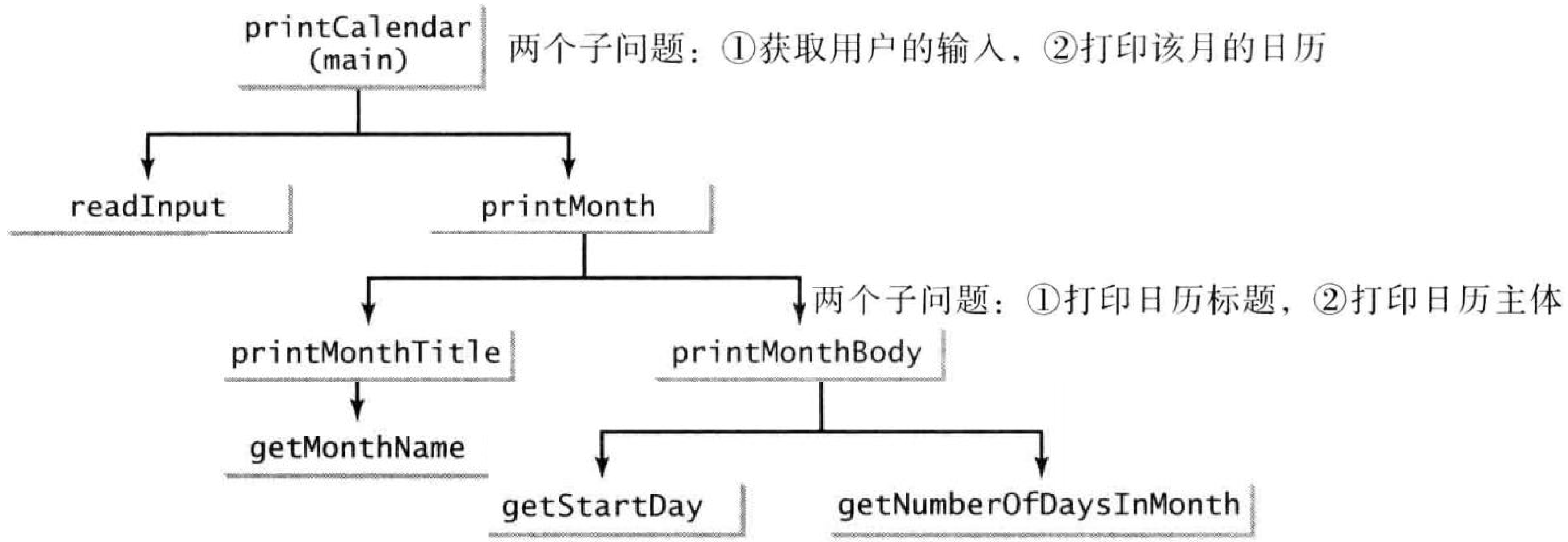
- 例程：显示给定年月的日历



```
Enter full year (e.g., 2001): 2011 ↵Enter
Enter month as number between 1 and 12: 9 ↵Enter
         September 2011
-----------------------------------
 Sun  Mon  Tue  Wed  Thu  Fri  Sat
                       1    2    3
   4    5    6    7    8    9   10
  11   12   13   14   15   16   17
  18   19   20   21   22   23   24
  25   26   27   28   29   30
```

函数

- 自顶向下设计


printCalendar (main)
两个子问题：①获取用户的输入，②打印该月的日历
readInput
printMonth
两个子问题：①打印日历标题，②打印日历主体
printMonthTitle
printMonthBody
getMonthName
getStartDay
getNumberOfDaysInMonth

函数

- getSartDay：这个月第一天是星期几？

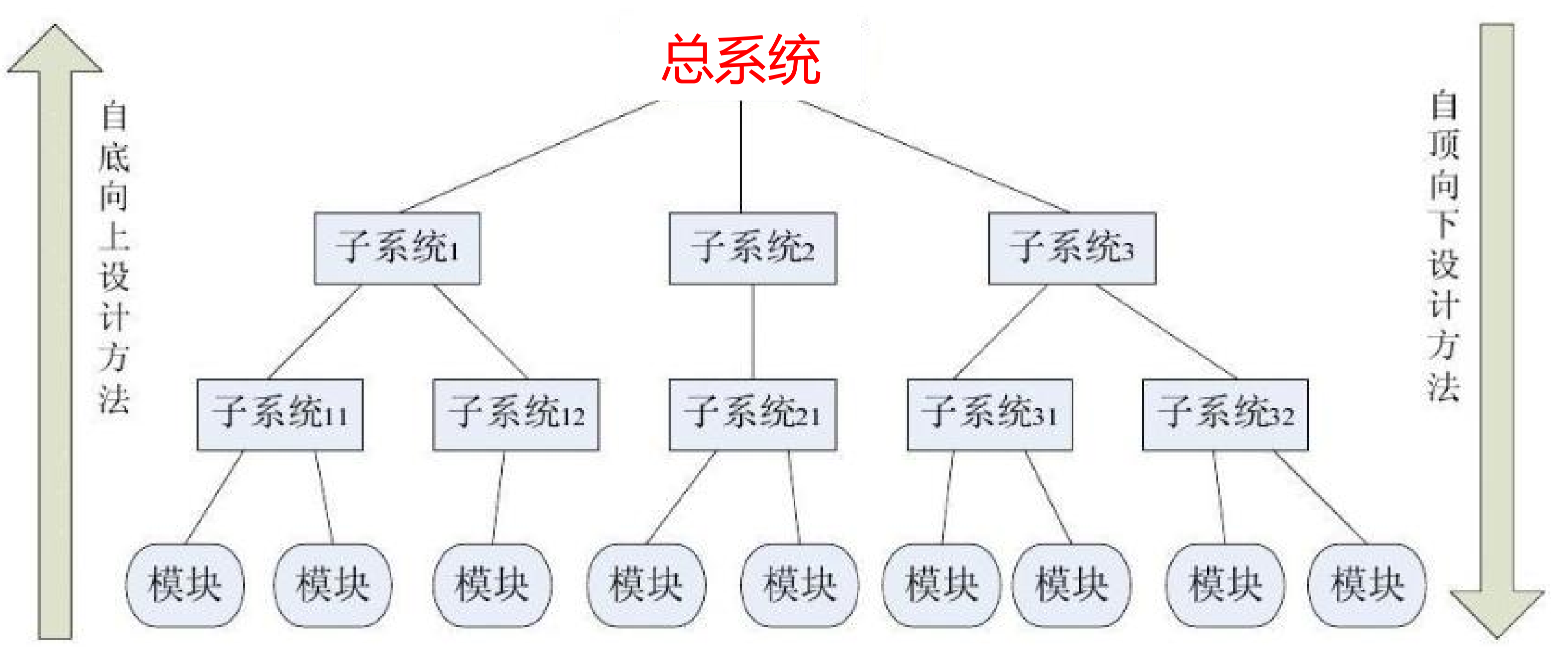
  totalNumberOfDays+startDay1800) %7 假设已知1800年1月1日是星期几

- getTotalNumberOfDays：本月第一天和距离1800.1.1共有多少天？

  isLeapYear 和 getNumberOfDaysInMonth

函数

- 自顶向下和自底向上


总系统
自底向上设计方法
自顶向下设计方法
子系统1
子系统2
子系统3
子系统11
子系统12
子系统21
子系统31
子系统32
模块