Python程序设计

第二讲 Python语言基础
简单Python程序的结构

张 华

WHU

简单Python程序的结构

- 程序的构成
- 标识符和关键字
- 常量和变量
- 对象的操作
- 变量的操作
- 表达式和运算符
- 语句
- 函数
- 类和对象
- 模块和包

引例

- 已知三角形的三条边，求三角形的面积。
  - 问题分析
    - 假设三条边长分别为a、b和c，
    - 则三角形的面积 s= $\sqrt{h*(h-a)*(h-b)*(h-c)}$ ，
    - 其中，h为三角形周长的一半
  - 实现

```
import math

a,b,c = eval(input("输入一个三角形的3边的长度："))
h = (a+b+c)/2
s = math.sqrt(h*(h-a)*(h-b)*(h-c))
print("它的面积是", s)
```

Python程序的构成

- 模块
  - 一个Python程序由1个或多个模块构成，每个模块是一个源文件（.py）。模块由语句组成。
- 语句
  - 语句是程序的过程构造块，用于创建对象，给变量赋值，调用函数，控制分支，建立循环等。语句包含表达式。
- 表达式
  - 用于创建和处理对象。
- 对象
  - 一切皆为对象。
  - 程序处理的数据是对象。

Python的内置对象

- 内置对象
  - Python中有许多内置对象可供编程者使用，内置对象可直接使用。
  - 例如
    - 内置的数据类型：数字、字符串、列表等
    - 内置的函数：abs、eval、max等。
- 非内置对象
  - 需要导入模块或模块的元素才能使用。
  - 例如
    - import math
    - 平方根函数sqrt，随机数产生函数random等。

Python的内置对象

常用内置对象

对象类型	类型名称	示例	简要说明
数字	int float complex	1234 3.14, 1.3e5 3+4j	数字大小没有限制，内置支持复数及其运算
字符串	str	'swfu', "I'm student", '''Python ''', r'abc', R'bcd'	使用单引号、双引号、三引号作为定界符，以字母r或R引导的表示原始字符串
字节串	bytes	b'hello world'	以字母b引导，可以使用单引号、双引号、三引号作为定界符，只能包含ASCII字符
列表	list	[1, 2, 3] ['a', 'b', ['c', 2]]	所有元素放在一对方括号中，元素之间使用逗号分隔，其中的元素可以是任意类型
字典	dict	{1:'food' ,2:'taste', 3:'import'}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素形式为“键:值”
元组	tuple	(2, -5, 6) (3,)	不可变，所有元素放在一对圆括号中，元素之间使用逗号分隔，如果元组中只有一个元素的话，后面的逗号不能省略
集合	set frozenset	{'a', 'b', 'c'}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素不允许重复;另外，set是可变的，而frozenset是不可变的

Python的内置对象

常用内置对象

对象类型	类型名称	示例	简要说明
布尔型	bool	True, False	逻辑值，关系运算符、成员测试运算符、同一性测试运算符组成的表达式的值一般为True或False
空类型	NoneType	None	空值
异常	Exception ValueError TypeError		Python内置大量异常类，分别对应不同类型的异常
文件		f = open('data.dat', 'rb')	open是Python内置函数，使用指定的模式打开文件，返回文件对象
其他可迭代对象		生成器对象、range对象、zip对象、enumerate对象、map对象、filter对象等等	具有**惰性求值**的特点，除range对象之外，其他对象中的元素只能看一次
编程单元		**函数**（使用def定义） **类**（使用class定义） **模块**（类型为module）	类和函数都属于**可调用对象**，模块用来集中存放函数、类、常量或其他对象

标识符

- 标识符是指在程序书写中程序员为一些特定对象的命名，包括变量、函数、类、模块和其他对象的名称。
- 命名规则
  - 必须以字母字符（包括中文字符）或下划线开头。
  - 不能有空格以及标点符号（括号、引号、逗号、斜线、反斜线、冒号、句号、问号等等）。
  - 标识符长度任意，大小写敏感。
  - 不能使用关键字。

a_int、str1、_strName、BusNo、锅炉温度 ✓	99var、It'sOK、for ✗

标识符

标识符命名的注意事项

- 以下划线开头的变量在Python中有特殊含义。
- 以双下划线开始和结束的名称通常具有特殊的含义。
  - 例如，__init__为类的构造函数，一般应避免使用。
- 不建议使用系统内置的模块名、类型名或函数名以及已导入的模块名及其成员名作变量名，这将会改变其类型和含义。

字符集

- Python语言使用的字符来自UTF-8编码字符。

关键字

- 关键字是预定义的保留的标识符，也称保留字。
  - Python 3有35个关键字

```
>>>help()                                                    Python 3.7.6
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more
help.

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

数据对象

常量

- 这些数据是不会改变的，也称为字面常量。
- 例如，整数389，浮点数23.56，字符串'hello'，

变量

- 将数据存储在内存中，然后用一个名称来引用内存空间，这个名称称为变量，其值是可以变化的。

```
>>> a=3.1415926
>>> a
3.1415926
>>> a=3.1415
>>> a
3.1415
```

数据对象

- 每个数据对象有标识、类型和值。
  - 标识（identity）
    - 用于唯一标识一个对象，通常对应于对象在计算机内存中的位置。使用内置函数id(obj1)可返回对象obj1的标识。
    - 通过内置的id()函数，可以获取一个对象唯一的id标识（CPython的实现为内存存放位置）。

```
>>> x=12
>>> id(x)
1671916960
>>> id(12)
1671916960
```

数据对象

- 每个数据对象有标识、类型和值。
  - 类型（type）
    - 用于表示对象所属的数据类型（类），数据类型用于限定对象的取值范围，以及允许执行的处理操作。
    - 使用内置函数**type(obj1)**可以返回对象**obj1**所属的数据类型。
  - 值（value）
    - 用于表示对象的数据类型的值。
    - 使用内置函数**print(obj1)**可返回对象**obj1**的值。

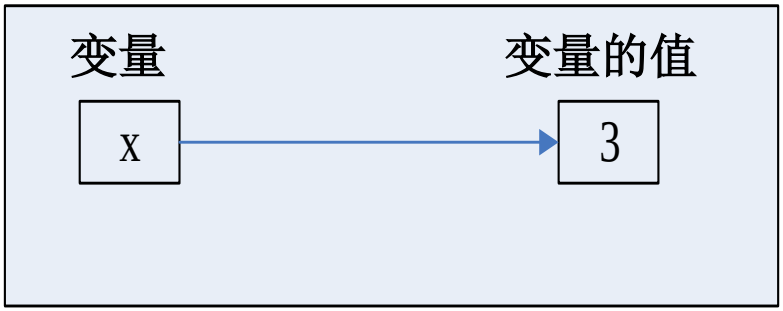
```
>>> type(x)
<class 'int'>
>>> print(x)
12
```

变量和对象的引用

变量

- 在Python中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。
  - 这一点适用于Python任意类型的对象。
  - 赋值语句的执行过程是：首先把等号右侧表达式的值计算出来，然后在内存中寻找一个位置把值存放进去，最后创建变量并指向这个内存地址。

```
>>> x=3
```



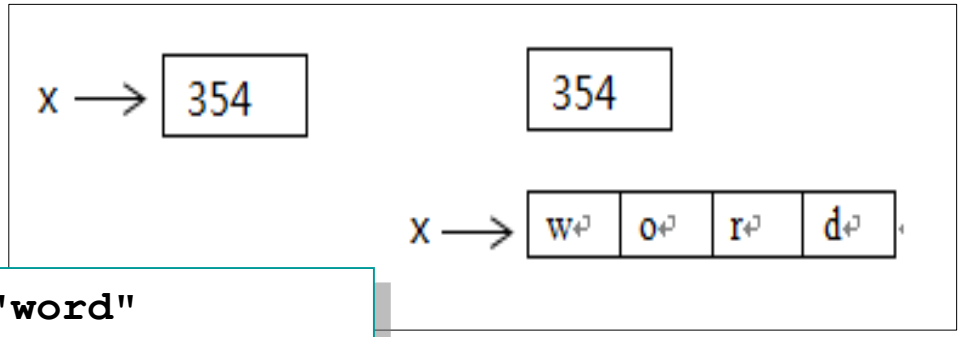
- Python中的变量并不直接存储值，而是存储了值的内存地址或者引用，这也是变量类型随时可以改变的原因。

动态类型编程语言

Python是动态类型语言

- 变量不需要显式声明数据类型。
- 根据变量的赋值，Python解释器自动确定其数据类型。
- 通过标识符和赋值运算符，可以指定某个变量指向某个对象，即引用该对象。

```
>>> x=354
>>> type(x)
<class 'int'>
>>> id(x)
34539888
```




```
>>> x="word"
>>> type(x)
<class 'str'>
>>> id(x)
33407296
```

强类型编程语言

Python是强类型语言

- 每个变量指向的对象均属于某个数据类型，只支持该类型允许的运算操作。
- Python解释器会根据赋值或运算来自动推断变量的类型。

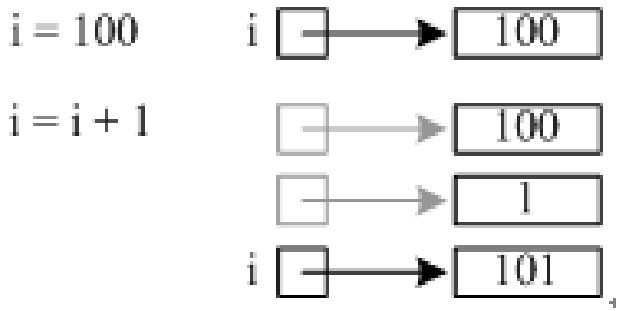
```
>>> x='word'
>>> x+3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
```

数据对象

对象占用的内存

- Python程序运行时，在内存中会创建各种对象（位于堆内存中），通过赋值语句，将对象绑定到变量（位于栈内存中），通过变量引用对象，进行各种操作。
- 多个变量可以引用同一个对象。
- 如果一个对象不再被任何有效作用域中的变量引用，则会通过自动垃圾回收机制，回收该对象占用的内存。

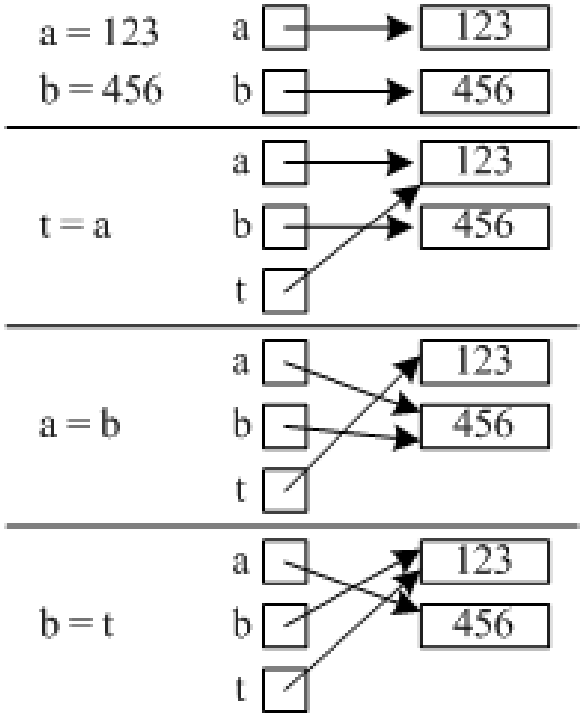
```
>>> x = 3
>>> id(x)
1786684560
>>> y = x
>>> id(y)
1786684560
```



数据对象

示例：交换两个变量的值

```
>>> a=123
>>> b=456
>>> t=a
>>> a=b
>>> b=t
>>> a
456
>>> b
123
```



另一种写法

```
>>> a,b=123,456
>>> a,b=b,a
>>> a,b
(456, 123)
```

对象操作

- 对象的值比较（==）
  - ==运算符判断两个变量指向的对象的值是否相同。
- 引用判别（is）
  - is运算符判断两个变量是否指向同一对象。

```
>>> x='abc'
>>> y=x
>>> z='abcd'
>>> print(x==y)
True
>>> print(x is y)
True
>>> print(x==z)
False
>>> print(x is z)
False
```

不可变对象和可变对象

对象的可变性

不可变对象（immutable）

- 不可变对象的值不能被修改。例如：int、str、complex

```
>>> x='abc'
>>> x[0]='x'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

不可变对象和可变对象

对象的可变性

- 可变对象（mutable）
  - 可变对象的值可以被修改。例如：list、dict

```
>>> y=list('abc')
>>> y[0]='x'
>>> y
['x', 'b', 'c']
```

- Python对象的可变性取决于其数据类型的设计，即是否允许改变其值。

变量的操作

- 链式赋值
  - 变量名 = 变量名 = 表达式（包括字面常量）
  - 例如：a=b=c=0
- 复合赋值
  - 变量名 复合赋值运算符 表达式（包括字面常量）
  - 复合赋值运算符

    +=，-=，*=，/=，//=，…
  - 例如：
    - a+=3　　相当于　　a=a+3（把a现有的值+3赋给a）
    - x*=y+7　　相当于　　x=x*(y+7)

变量的操作

- 系列解包赋值
  - 将系列数据类型解包为对应相同个数的变量
  - 例如：
    - x,y,z = 1,2,3
    - x,y = y,x
- 删除变量
  - del 变量名
  - 例如：
    - del z

表达式和运算符

表达式的组成

- 操作数、运算符和圆括号按一定规则组成表达式。
- 运算符的优先级控制各个运算符的计算顺序。

表达式的书写规则

- 表达式从左到右在同一个基准上书写。
  - 例如，数学公式a^2+b^2应该写为：a**2+b**2
- 乘号不能省略。
  - 例如，数学公式ab（表示a乘以b）应写为：a*b
- 括号必须成对出现，而且只能使用圆括号；圆括号可以嵌套使用。

表达式和运算符

运算符

- 运算符用于在表达式中对一个或多个操作数进行计算并返回结果值。
- 表达式计算顺序取决于运算符的优先级和结合性。
- 可以使用圆括号“()”强制改变运算顺序。
- 例如：
  - **(a+b+c)/3　　不能写成　　a+b+c/3**
  - **x*y/z　　相当于　　(x*y)/z**
- **Python运算符表**
  - 详见教材第28页

语句

Python的语句

- 语句是Python程序的过程构造块，用于定义函数、定义类、创建对象、变量赋值、调用函数、控制分支、创建循环等。
- Python语句分为简单语句和复合语句。
- 简单语句包括：
  - 表达式语句、赋值语句、assert语句、pass空语句、del语句、return语句、yield语句、raise语句、break语句、continue语句、import语句、global语句、nonlocal语句等。
- 复合语句包括：
  - if语句、while语句、for语句、try语句、with语句、函数定义、类定义等。

语句的书写规则

Python语句的书写规则

- 使用换行符分隔，一般情况下，一行一条语句。
- 从第一列开始，前面不能有任何空格，否则会产生语法错误。
- 注释可以从任意位置开始。
- 复合语句构造体必须缩进。
- 反斜杠（\）用于一个代码跨越多行的情况。
  - 如果语句太长，可以使用续行符（\）
- 分号（;）用于在一行书写多条语句。

语句的书写规则

复合语句的书写规则

- 复合语句由头部语句和构造体语句块组成。
- 构造体语句块由一条或多条语句组成。
  - 头部语句由相应的关键字开始，构造体语句块则为下一行开始的一行或多行缩进代码。
  - 通常，缩进是相对头部语句缩进四个空格，也可以是任意空格，但同一构造体代码块的多条语句缩进的空格数必须一致对齐。如果语句不缩进，或缩进不一致，将导致编译错误。
  - 如果条件语句、循环语句、函数定义和类定义比较短，可以放在同一行。

注释

注释用来对代码进行说明。

单行注释

- 以符号#开始，表示本行#之后的内容为注释。

```
# this is my first Python Program
prtint('Hi, Python!') # print a string
```

多行注释（实际上是多行字符串）

- 包含在一对三引号'''... '''或"""... """之间且不属于任何语句的内容将被解释器认为是注释。

```
''' this is my first Python Program
    print a string
'''
prtint('Hi, Python!')
```

函数

- 函数是可以重复调用的代码块。
  - 定义函数时，可以声明函数的参数，即形式参数，简称形参。

```
def 函数名([形参列表]):
    函数体
```

  - 调用函数时，需要提供函数需要的参数的值，即实际参数，简称实参。

```
函数名([实参列表])
```

  - 函数可以有返回值，即计算结果。

函数

- Python语言提供了海量的内置函数、标准库函数、第三方模块函数。
  - 内置函数，例如dir()、type()、id()、help()、len()等。
  - 标准库函数，例如math库的sqrt()。
  - 第三方模块函数，例如Pandas库中的DataFrame。

函数

模块函数

- 通过import语句，可以导入模块module，然后使用下面的形式调用模块中的函数。

  module.function(arguments)

- 一般，每个import语句只导入一个模块，最好按标准库、扩展库、自定义库的顺序依次导入。

```
import csv
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
```

```
datetime.datetime.today()
```

函数

模块函数

- 也可以用from module import …的方式导入模块中的一个常量、函数和类，然后直接调用模块中的函数。

  function(arguments)

```
from datetime import datetime
datetime.today()
```

- 还可以用from module import *的方式导入模块中的所有元素。

```
from datetime import *
datetime.today()
```

类和对象

- 类和对象是面向对象程序设计的两个基本概念。
  - 创建类对象

    class 类名:

    ····类体

  - 实例对象的创建和调用

    anObject = 类名(参数列表)

    anObject.对象方法 或 anObject.对象属性

```
>>> astr = str('abcdef')
>>> astr.upper()
'ABCDEF'
```

模块和包

- **模块**
  - 包含Python代码的源文件（通常包含用户自定义的变量、函数和类）称之为模块，其扩展名为.py。
- **包**
  - 包是模块的层次性组织结构。
  - 功能相近的模块可以组织成包。