Python 语言程序设计

陈 峦 副教授

13880209111，chluan@uestc.edu.cn

研究院大楼316#

第二章 顺序结构

2.1 程序设计概述

- 算法：求解一个问题的方法和步骤。
- 画出算法流程图：读入一批整数，遇到0 结束，统计其中正数和负数的个数。

- 例：求两数之和。
- def f(x,y):
- 　　return x+y
- def main():
- 　　print("f(2,3)=",f(2,3))
- main()
- 结果：
- f(2,3)= 5

2.2 Python程序的书写规则

- 2.2.1 Python程序概述
- C、C++ 要求程序语句代码必须放在函数中。
- Java要求程序语句代码必须放在函数中，函数必须放在类中。
- Python程序对语句代码没有这些限制。

- 例：读入三个数，然后按从小到大的顺序输出。
- 程序如下：

```
x=int(input('input x:')) #输入x的值
y=int(input('input y:')) #输入y的值
z=int(input('input z:')) #输入z的值
if x>y:   #如果x>y,则x和y的值互换
    x,y=y,x
if x>z:   #如果x>z,则x和z的值互换
    x,z=z,x
if y>z:   #如果y>z,则y和z的值互换
    y,z=z,y
print(x,y,z)
```

程序运行结果：

```
input x: 23
input y: -5
input z: 45
-5   23   45
```

- 例：已知$f(x,y)=x^2+y^2$,输入x，y的值，求出对应的函数值。
- 程序如下：
- def f(x,y):
- return x**2+y**2
- print("f(3,4)=",f(3,4))
- 程序运行结果：
- f(3,4)=25

- Python支持定义单行函数，称为lambda函数，可以用在任何需要函数的地方。
- lambda函数是一个可以接收任意多个参数并且返回单个表达式值的函数。

- 例：已知f(x,y)=x2+y2,输入x，y的值，求出对应的函数值。
- 程序如下：
- f=lambda x,y:x**2+y**2
- print("f(3,4)=",f(3,4))
- 程序运行结果：
- f(3,4)=25

例 Fibonacci 数列定义如下：

$$\begin{cases} f_1 = 1 \\ f_2 = 1 \\ f_n = f_{n-1} + f_{n-2} & n > 2 \end{cases}$$

输出 Fibonacci 数列前 50 项之和。

程序如下：

```
a,b=0,1
s=0
for i in range(50):    #i从0变化到49
    s+=b
    a,b=b,a+b
print("s=",s)
```

程序运行结果：

```
s=32951280098
```

如果用整型数据进行计算，在很多程序设计语言中都会产生溢出，而Python支持大数据运算，不会产生溢出。

2.2.2 Python语句缩进规则

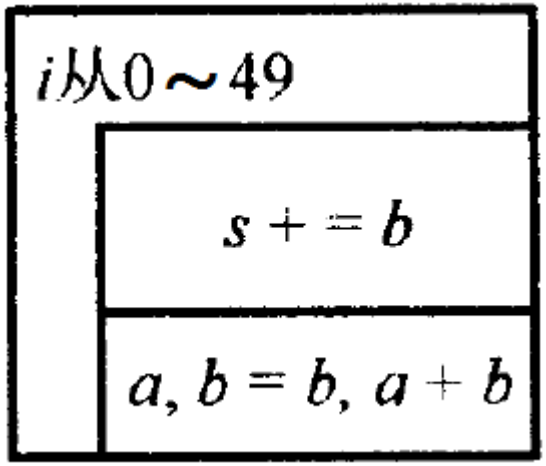
- Python通过语句缩进对齐反映语句之间的逻辑关系，从而区分不同的语句块。
- 缩进可以由任意的空格或制表符组成，缩进的宽度不受限制，一般为四个空格或一个制表符，但在同一程序中不建议混合使用空格和制表符。

- 同一个语句块必须保持一致的缩进量。
- 这是Python语言区别于其他语言的重要特点，Python的语句块不使用像C语言中的大括号({})或其他语言的功能结束语句来控制语句块的开始与结束。

例：下面两段程序的含义是截然不同的。

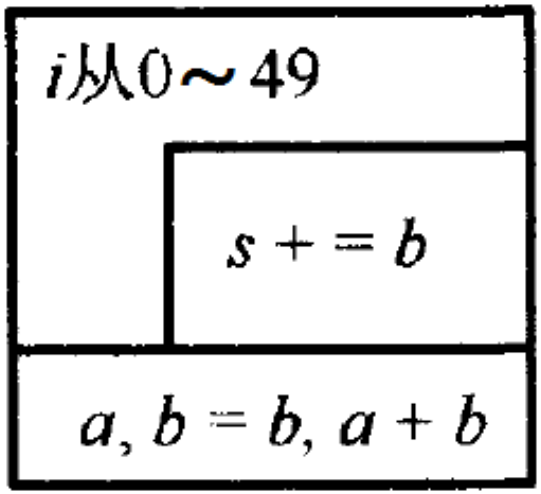
程序段一：

```
for i in range(0,50):    #i从0变化到49
    s+=b
    a,b=b,a+b
```



程序段二：

```
for i in range(0,50):    #i从0变化到49
    s+=b
a,b=b,a+b
```



1、设置语句块的缩进量

在程序编辑窗口，首先选中语句块：

（1）语句块的批量缩进

Format→Indent Region，

或按快捷键Ctl+]。

（2）取消缩进

Format→Dedent Region，

或按快捷键Ctrl+[。

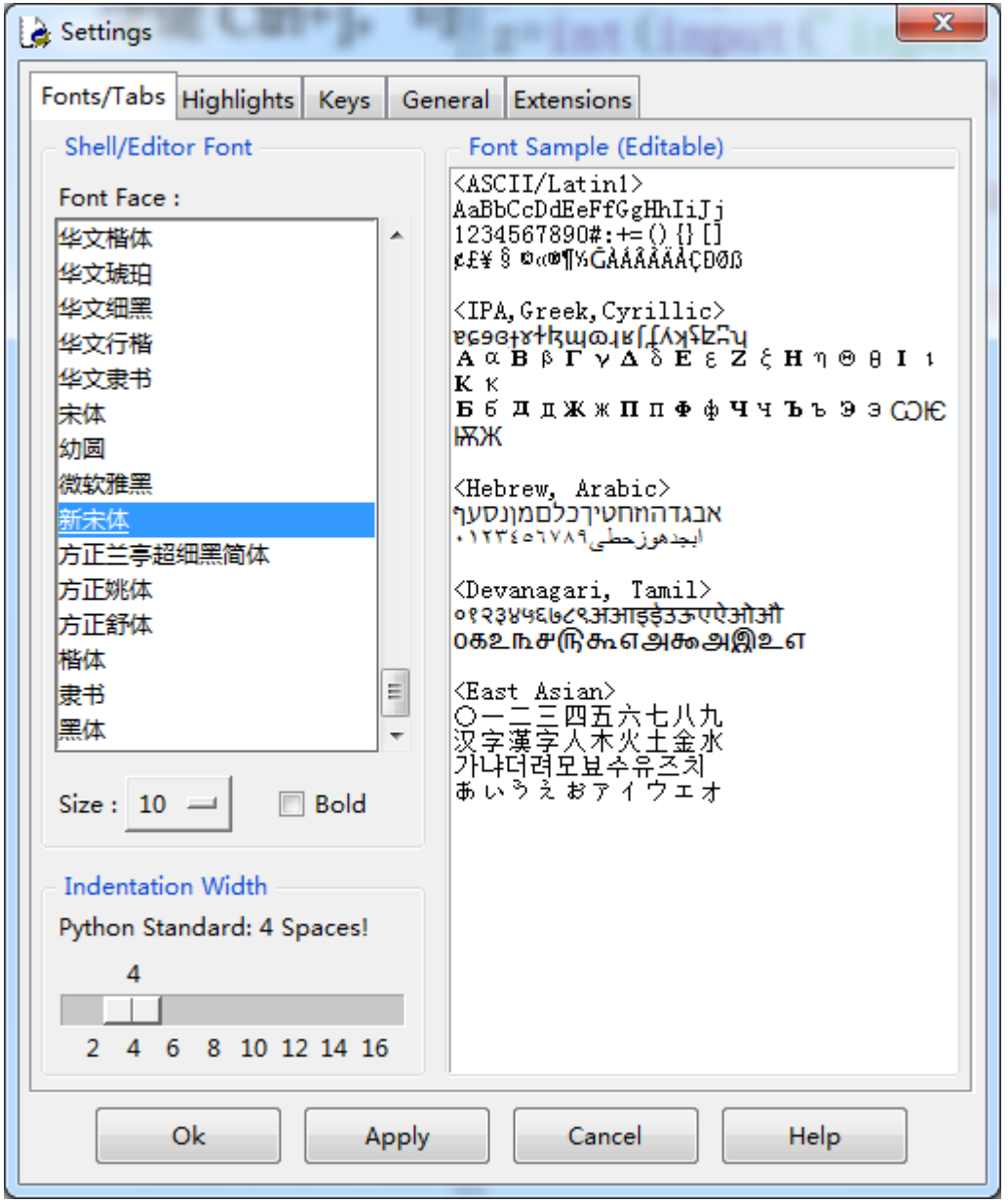
2、设置默认缩进量

在程序编辑窗口：

Options→Configure IDLE，

出现设置对话框，

默认缩进宽度(Indentation Width)是4个空格。



2.2.3 Python语句行与注释

- 在Python中，语句行从解释器提示符后的第一列开始，前面不能有任何空格，否则会产生语法错误。
- 每个语句行以回车符结束。
- 可以在同一行中使用多条语句，语句之间使用分号分隔。

- 例：
- **>>>x='f=';f=100;printf(x,f)**
- **f= 100**
- 如果语句行太长，可以使用反斜杠将一行语句分为多行显示。
- 例：
- **>>>t=1+2+3+4**
- **+5+6**
- **>>>t**
- **21**

- 如果在语句中包含小括号、中括号或大括号，则不需要使用多行续行符。
- 例：
- **>>>def f(**
- **　　):return 120**
- **#提示：回车，空一行表示结束输入。**
- **>>>f()**
- **120**

2. 注释

- 注释对程序的执行没有任何影响，目的是对程序做解释说明，以增强程序的可读性。
- 在程序调试阶段，有时需要暂时不执行某些语句，这时可以给这些语句加注释符号，相当于对这些语句做逻辑删除，需要执行时，再去掉注释符号即可。

- 程序中的单行注释采用#开头。
- 注释可以从任意位置开始，可以在语句行末尾，也可以独立成行。
- 对于多行注释，一般推荐使用多个#开头的多行注释，也可采用三引号（实际上是用三引号括起来的一个多行字符串，起到注释的作用）。
- 注意：注释行是不能使用反斜杠\续行的。

- 在Python程序编辑窗口，首先选中语句块，然后选择Format →Comment Out Region命令，或按快捷键Alt+3，可以进行语句块的批量注释。
- 如果要解除注释，则选择Format→Uncomment Region命令，或按快捷键Alt+4。

2.3 赋值语句

- Python的赋值和一般高级语言的赋值有很大的不同，它是数据对象的一个引用。

2.3.1 赋值语句的一般格式

- 赋值号：=
- 赋值语句格式：变量=表达式
- 赋值的意义：先计算表达式的值，然后使该变量指向该数据对象。
- 赋值号左边必须是变量，右边则是表达式。
- 变量可以理解为数据对象的别名。
- 一个变量通过赋值可以指向不同类型的对象。

- 例：
- >>>a=5
- >>>b=8
- >>>a=b
- 结果：a和b同时指向了数值8。
- Python是动态类型语言，不需要预先定义变量类型，变量的类型和值在赋值时被初始化。
- 例：
- >>>x=67.2
- >>>x="ABCD"

- Python中的赋值并不是直接将一个值赋给一个变量的，而是通过引用传递的，在赋值时，不管这个对象是新创建的还是一个已经存在的，都是将该对象的引用（并不是值）赋值给变量。
- 赋值语句是没有返回值的。
- 例：
- 表达式“(x=10)+20”是错误的。

2.3.2 复合赋值语句

- 在程序设计中，经常遇到在变量已有值的基础上做某种修正的运算。例：x=x+5.0。
- 这类运算的特点：变量既是运算对象，又是赋值对象。
- 为避免对同一存储对象的地址重复计算，Python提供了12种复合赋值运算符。

- Python有12种复合赋值运算符:
- +=、-=、*=、/=、//=、%=、**=、<<=、>>=、&=、|=、^=
- 例:
- x+=5.0　　#等价于 x=x+5.0
- x*=u+v　　#等价于 x=x*(u+v)

2.3.3 多变量赋值

1. 链式赋值

- 链式赋值语句的一般形式：
- 变量1=变量2=……=变量n=表达式
- 等价于：
- 变量n=表达式
- ……
- 变量1=变量2
- 链式赋值用于为多个变量赋同一个值。

- 例:
- >>>a=b=10
- >>>id(a)
- 1471472544
- >>>id(b)
- 1471472544
- >>>a
- 10
- >>>b
- 10

创建一个值为10的整型对象，将该对象的同一个引用赋值给a和b，即a和b均指向数据对象10。

2. 同步赋值

- 同步赋值的一般形式为：
- 变量1,变量2,……,变量n=表达式1,表达式2,……,表达式n
- 赋值号左边变量的个数与右边表达式的个数要一致。
- 首先计算右边n个表达式的值，
- 然后同时将表达式的值赋值给左边的n个变量。
- 这并非等同于简单地将多个单一赋值语句进行组合。

- 例：
- >>>a,b,c=10,20,30
- >>>a
- 10
- >>>b
- 20
- >>>c
- 30

- 例：
- **>>>x,x=11,22**
- **>>>x**
- **22**
- 将右边表达式的各值从左到右依次赋值给左边各变量（先做**x=11**，后做**x=22**）。

- 同步赋值有先后顺序，但不是传统意义上的单一赋值语句的先后执行。
- 例：
- **>>>x=34**
- **>>>x,y=12,x**
- **>>>x**
- **12**
- **>>>y**
- **34**
- 先计算右边表达式的各值，再将它们从左到右依次赋值给左边各变量。

- 要交换a，b两个变量的值，一般需要一个中间变量：
- **>>>t=a**
- **>>>a=b**
- **>>>b=t**
- 如果采用同步赋值，一个语句即可完成：
- 例：
- **>>>a,b=10,20**
- **>>>a,b=b,a**
- **>>>a**
- **20**
- **>>>b**
- **10**
- 不需中间变量即可交换两个变量的值,优雅、简洁。

2.4 数据输入/输出

- 程序可以从键盘读取数据，也可以从文件读取数据；
- 程序的运行结果可以输出到屏幕上，也可以保存到文件中便于以后使用。
- 标准输入输出是指通过键盘和屏幕的输入输出，即控制台输入输出。

1.标准输入

Python用内置函数input()实现标准输入，其调用格式：

input([提示字符串])

如果有“提示字符串”，则原样显示，提示用户输入数据。

input()函数从标准输入设备(键盘)读取一行数据，并返回一个字符串（去掉结尾的换行符）。

例：

```
>>>a=input("Please input a:")
Please input a:2
```

- input()函数把输入的内容当成字符串，如果要输入数值数据，可以使用类型转换函数将字符串转换为数值。
- 例：
- >>>x=input()
- 12
- >>>x
- '12'
- >>>x=int(input())
- 12
- >>>x
- 12

- 使用input()函数可以给多个变量赋值。
- 例:
- >>>x,y=eval(input())
- 3,4
- >>>x
- 3
- >>>x+y
- 7
- 从键盘输入“3,4”，input()函数返回字符串“3,4”。
- Eval()函数将去掉字符串最外侧的引号。
- 经过eval()函数处理，字符串“3,4”变成由3和4组成的元组。

- >>>a=eval(“1.2+3.4”)
- >>>print(a)
- 4.6
- >>>a=eval(“’Python’”)
- >>>print(a)
- Python

```
>>> a=eval("Python")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'Python' is not defined
```

```
>>> a=2
>>> b=eval("a")
>>> print(b)
2
>>> id(b)
494527264
>>> id(a)
494527264
>>>
```

```
>>> a=2
>>> b=eval("'a'")
>>> print(b)
a
>>> id(b)
19440384
>>> id(a)
494527264
>>>
```

- 例：
- >>>eval('3,4')
- (3,4)
- 语句“eval('3,4')”等价于“x,y=3,4”或“x,y=(3,4)”。
- 例：
- >>>x,y=(3,4)
- >>>x
- 3
- >>>y
- 4

2.标准输出

- 有两种输出方式：使用表达式和使用print()函数。
- 直接使用表达式可以输出该表达式的值。
- 例：
- >>>x=123
- >>>x+45
- 168
- >>>x
- 123
- 使用表达式语句输出，输出形式简单，一般用于检查变量的值。

- 常用输出方法是用print()函数，其调用格式为：
- print([输出项1,输出项2,……,输出项n][,sep=分隔符][,end=结束符])
- 其中，输出项之间以逗号分隔，没有输出项时输出一个空行。
- sep表示输出时各输出项之间的分隔符（默认以空格分隔）。(提示：separator，分隔符)
- end表示结束符（默认以回车换行结束）。
- print()函数从左至右求每一个输出项的值，并将各输出项的值依次显示在屏幕的同一行上。

- 例：
- >>>print(10,20)
- 10 20
- >>>print(10,20,sep=',')
- 10,20
- >>>print(10,20,sep=',',end='*')
- 10,20*
- 第三次print()函数调用时。以“*”作为结束符，并且不换行。

- print()函数输出文本时默认会在最后增加一个换行，使下一条命令提示符和语句显示在下一行。
- 如果不希望在本行输出最后增加这个换行，或者希望本行输出文本后增加其他内容，可以对print()函数的end参数进行赋值。

```
>>> a=2
>>> print(a,end=".")
2.>>> print(a)
2
>>>
```

- 例：
- 在程序方式下运行下列语句：
- **print(10,20,sep=',',end='*')**
- **print(30,40,end="#")**
- **print(50,60,end="")**
- **print(70,80,90,sep=":",end="Goodbye!")**
- 输出结果：
- **10,20*30 40#50 6070:80:90Goodbye!**

- #每次执行的print()函数并没有产生换行

2.4.2 格式化输出

- 例：
- **>>>7.80**
- **7.8**
- 末尾的0没有输出，在很多情况下没有太大问题，但有时就必须输出。
- 例：在财务系统中，表示七元八角不应显示成**7.8**，而应显示为**7.80**，甚至在前面还要加货币符号，即**¥7.80**。
- 为了解决这个问题，可以采用格式化输出。

Python的格式化输出方法有三种：

- （1）利用字符串格式化运算符%。（不建议使用，这种旧式的格式化方法最终会从Python中移除）
- （2）利用format()内置函数。
- （3）利用字符串的format()方法。

- 例：
- >>>print('t=¥%.2f'%7.8)
- t=¥7.80
- >>>print('t=¥',format(7.8,'.2f'))
- t=¥7.80
- >>>print('t=¥{0:.2f}'.format(7.8))
- t=¥7.80

1. 字符串格式化运算符%

（不建议使用，这种旧式的格式化方法最终会从Python中移除）

- 用运算符%分隔格式字符串与输出项，一般格式为：
- 格式字符串%(输出项1,输出项2,……,输出项n)
- 格式字符串由普通字符和格式说明符组成。
- 普通字符原样输出，格式说明符决定所对应输出项的输出格式。
- 格式说明符以百分号%开头，后接格式标志符。

- 例：
- >>>'V=%s,%s,%s'%(1,2.3,['AA','BB','CC'])
- "V=1,2.3,['AA','BB','CC']"
- >>>print('V=%s,%s,%s'%(1,2.3,['AA','BB','CC']))
- V=1,2.3,['AA','BB','CC']

- 在格式化运算符%后面的括号内有三个输出项，即1、2.3和['AA','BB','CC']，都使用格式说明符%s将值转换为字符串。
- 一般情况下，如果没有什么特殊要求，不管输出项的类型如何，都可使用格式符%s。

常用格式说明符

格式说明符	格式化结果
%%	百分号
%c	字符
%s	字符串
%d	带符号整数（十进制）
%o	带符号整数（八进制）
%x或%X	带符号整数（十六进制，用小写字母或大写字母）
%e或%E	浮点数字（科学计数法，用小写e或大写E）
%f或%F	浮点数字（用小数点符号）
%g或%G	浮点数字（根据值的大小，采用%e、%f或%E、%F）

- 例：
- >>>'%6.2f'%1.235
- ' 1.24'
- 总共输出的长度为6个字符，其中小数部分占2位。
- 例：
- >>>'%06.2f'%1.235
- '001.24'
- 如果输出的位数不足6位就用0补足6位。（小数点也占用1位）
- 类似于这里0的标记还有-、+。其中，-表示左对齐，+表示在正数前面也标上+号（默认是不加的）。

- 例：
- >>>'%(name)s:%(score)06.1f'%{'score':9.5,'name':'Lucy'}
- 'Lucy:0009.5'
- 这种形式只用在要输出的内容为字典类型时。
- 每个格式说明符对应哪个输出项由圆括号中的键来指定。

- 例：
- >>>'%0*.*f'%(6,2,2.345)
- '002.35‘
- #相当于'%06.2f'%1.235

- 例：
- >>>print("%+3d,%0.2f"%(25,123.567))
- +25,123.57
- >>>print("N:%-10s A:%-8d S:%-.2e"%("Aviad",25,1839.8))
- N:Aviad A:25 S:1.84e+03
- >>>nH=0xFF
- >>>print("nH=%x,nD=%d,nO=%o"%(nH,nH,nH))
- nH=ff,nD=255,nO=377

2. format()内置函数

- format()内置函数可以将一个输出项单独进行格式化，一般格式为:
- format(输出项[,格式字符串])
- 当省略格式字符串时，该函数等价于函数“str(输出项)”的功能。
- format()函数解释格式字符串是根据输出项的类型来决定的，不同的类型有不同的格式化解释。

- 基本的格式控制符有：
- d，b，o，x或X：分别按十进制、二进制、八进制、十六进制输出一个整数；
- f或F、e或E、g或G：按小数形式或科学计数形式输出一个整数或浮点数；
- c：输出以整数为编码的字符；%输出百分号。

- 例：
- >>>print(format(15,'X'),format(65,'c'),format(3.145,'f'))
- F A 3.145000
- 例：
- >>>print(format(3.145,'6.2f'))
- 3.15

- 格式字符串还可以指定填充字符、对齐方式（其中，<表示左对齐、>表示右对齐、^表示居中对齐、=表示填充字符位于符号和数字之间）、符号（其中，+表示正号，-表示负号）。
- 例：
- **>>>print(format(3.145,'0=+10'),format(3.14159,'05.3'))**
- **+00003.145 03.14**
- **>>>print(format('test','<20')**
- **test**
- **>>>print(format('test','^20'))**
- **test**

3. 字符串的format()方法

- 调用格式为：
- 格式字符串.format(输出项1,输出项2,……,输出项n)
- 格式字符串中可以包括普通字符和格式说明符。普通字符原样输出，格式说明符决定所对应输出项的转换格式。

- >>>a,b=2,3
- >>>print(“a={},b={},a*b={}”.format(a,b,a*b))
- a=2,b=3,a*b=6
- 大括号{}表示一个槽位置，大括号{}中的内容由后面紧跟的format()方法中的参数按从左到右的次序顺序填充（也可以按照序号或键指定）。

- 格式说明符使用大括号括起来，一般形式如下：
- {[序号或键]:格式说明符}
- 可选的序号对应于要格式化的输出项的位置，从0开始。0表示第一个输出项，1表示第二个输出项，以后依此类推。序号全部省略则按输出项的自然顺序输出；
- 可选的键对应于要格式化的输出项的名字或字典的键值；
- 格式说明符同format()内置函数。格式说明符用冒号(:)开头。

- 例：
- >>>'{0:.2f},{1}'.format(3.145,500)
- '3.15,500'
- 格式说明符“{0:.2f}”包含了两方面的含义：“0”表示该输出项是format括号中的第一个输出项（序号为0）；格式符“:.2f”用于描述该输出项如何被格式化，即小数部分占2位，按输出项实际位数输出。
- “{1}”表示该输出项是format括号中的第二个输出项（序号为1）。它采用默认格式输出。

- （1）使用大括号“{}”格式说明符，大括号及其里面的字符（称为格式化字符）将会被format()中的参数替换。
- 例:
- >>>print('a={},b={}'.format(1,2))
- a=1,b=2
- >>>import math
- print("PI={}.".format(math.pi))
- PI=3.141592653589793.

- （2）使用“{序号}”形式的格式说明符，在大括号中的数字用于指向输出对象在format()函数中的位置。
- 例：
- >>>print('a={0}, b={1},c={2}'.\
- format(11,22,33))
- a=11,b=22,c=33
- 例：
- >>>print('a={1},b={0},c={2}'.\
- format(11,22,33))
- a=22,b=11,c=33

- （3）使用“{键}”形式的格式说明符，大括号中是一个标识符，该标识符会指向使用该名字的参数。
- 例:
- >>>print('a={k},b={m},c={n}'.format(m='AA',n='BB',k='CC'))
- a=CC,b=CC,c=BB
- （4）混合使用“{序号}”、“{键}”形式的格式说明符。
- 例:
- >>>print('a={1},b={n},c={0},d={m}'.\
- format('AA','BB',m='CC',n='DD'))
- a=BB,b=DD,c=AA,d=CC

- （5）输出项的格式控制
- 在{}中序号或键后面可跟一个冒号:和格式符。
- 例：
- {0:8}表示format中的第一个参数（序号为0）占8个字符宽度，如果输出位数大于该宽度，就按实际位数输出；如果输出位数小于此宽度，默认右对齐，左边补空格，补足8位。
- {1:.3}表示第二个参数（序号为1）除小数点外的输出位数是3位。
- {1:.3f}表示浮点数的小数位保留3位，其中f表示浮点型（ d表示整型）。

- 例：
- >>>print('PI={0:.3f}.'.format(math.pi))
- PI=3.142.
- >>>print('PI={0:.3}.'.format(math.pi))
- PI=3.14.
- {0:.3}表示第一个参数（序号为0）除小数点外的输出位数是3位。
- {0:.3f}表示浮点数的小数位保留3位，其中f表示浮点型。

- 例：
- print('{0:<15}'.format(1234567890)) #左对齐
- print('{0:>15}'.format(1234567890)) #右对齐
- print('{0:*^15}'.format(1234567890)) #居中，用*填充
- print('{0:10b}'.format(65)) #二进制，默认右对齐
- print('{0:10o}'.format(65)) #八进制
- print('{0:10x}'.format(65)) #十六进制，字母小写

```
1234567890
     1234567890
**1234567890***
   1000001
       101
        41
>>>
```

2.5 顺序结构程序举例

- 一个Python程序不需要变量定义，可直接描述程序功能。
- 程序结构：
- （1）输入（I, input）： 输入原始数据。
- （2）计算（P, process）：对原始数据进行处理。
- （3）输出（O, output）：输出处理结果。

例　已知 $x=5+3i$，　$y=e^{\frac{\sqrt{\pi}}{2}}$，求 $z=\dfrac{2\sin 56^\circ}{x+\cos|x+y|}$ 的值。

- import math
- x=5+3J;　　#x是一个复数
- y=math.exp(math.sqrt(math.pi)/2);
- z=2*math.sin(math.radians(56))　#z的分子
- z/=(x+math.cos(abs(x+y)))　　#求z
- print("z=",z)

- 例：从键盘输入一个3位整数n，输出其逆序数m。例如，输入n=127，则m=721。
- n=int(input("n="))
- a=n%10; #求n的个位数字
- b=n//10%10; #求n的十位数字
- c=n//100; #求n的百位数字
- m=a*100+b*10+c
- print("{0:3}的逆序数是{1:3}".format(n,m))

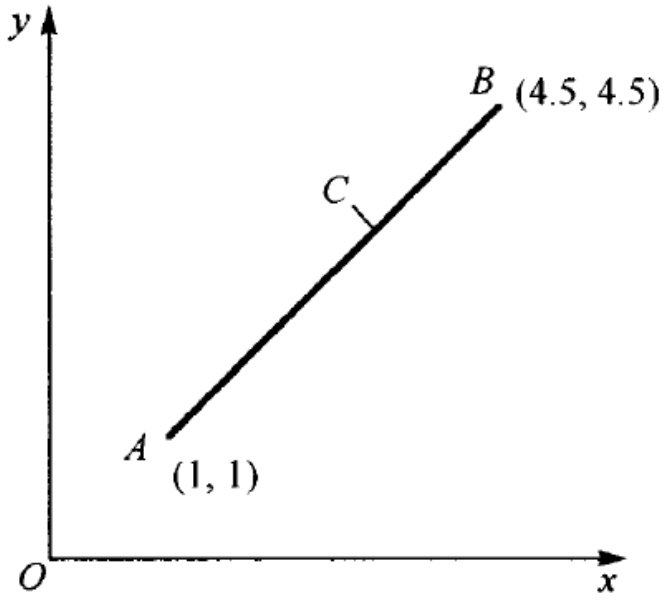
例：求一元二次方程 $ax^2+bx+c=0$ 的根。

- from cmath import sqrt
- a=float(input('a='))
- b=float(input('b='))
- c=float(input('c='))
- d=b*b-4*a*c
- x1=(-b+sqrt(d))/(2*a)
- x2=(-b-sqrt(d))/(2*a)
- print("x1={0:.5f}, x2={1:.5f}".format(x1,x2))

例：

有一线段 AB，A 的坐标为(1，1)，B 的坐标为(4.5，4.5)。求 AB 的长度，以及黄金分隔点 C 的坐标。

黄金分割点在线段的 0.618 处。



- a=complex(input("a="))
- b=complex(input("b="))
- c=a+0.618*(b-a)
- s=abs(a-b)
- print("长度：",s)
- print("黄金分割点：",c)

```
a=1+j
b=4.5+4.5j
长度：  4.949747468305833
黄金分割点：  (3.163+3.163j)
>>>
```

习题课

- 一、选择题
- 1. 流程图中表示判断框的是（　）。
- A.矩形框　B.菱形框　C.平行四边形框　D.椭圆形框
- 2. 下面不属于程序的基本控制结构的是（　）。
- A.顺序结构　B.选择结构　C.循环结构　D.输入/输出结构
- 3. 以下关于Python语句的叙述中，正确的是（　）。
- A.同一层次的Pymon语句必须对齐
- B.Python语句可以从一行的任意一列开始
- C.在执行Python语句时，可发现注释中的拼写错误
- D.Python程序的每行只能写一条语句

- 一、选择题
- 1．流程图中表示判断框的是（　　）。B
- A．矩形框　　B．菱形框
- C．平行四边形框　D．椭圆形框
- 2．下面不属于程序的基本控制结构的是（　　）。D
- A．顺序结构　B．选择结构
- C．循环结构　D．输入输出结构
- 3．以下关于Python语句的叙述中，正确的是（　　）。A
- A．同一层次的Python语句必须对齐
- B．Python语句可以从一行的任意一列开始
- C．在执行Python语句时，可发现注释中的拼写错误
- D．Python程序的每行只能写一条语句

- 4.下列语句中，在Python中非法的是（　）。
- A.x=y=z=1　B.x,y=y,x　C.x=(y=z+1)　D.x+=y
- 5.已知x=2，语句x*=x+1执行后，x的值是（　）。
- A.2　B.3　C.5　D.6
- 6.在Python中，正确的赋值语句为（　）。
- A.x+y=10 B.x=2y C.x=y=30 D.3y=x+1
- 7.为了给整型变量x，y，z赋初值10，下面正确的Python赋值语句是（　）。
- A.xyz=10　　B.x=10 y=10 z=10
- C.x=y=z=10　D.x=10,y=10,z=10

- 4．下列语句中，在Python中非法的是（　　）。C
- A．x=y=z=1　　B．x,y=y,x
- C．x=(y=z+1)　　D．x+=y
- 5．已知x=2，语句x*=x+1执行后，x的值是（　　）。D
- A．2　B．3　C．5　D．6
- 6．在Python中，正确的赋值语句为（　　）。C
- A．x+y=10　B．x=2y　C．x=y=30　D．3y=x+1
- 7．为了给整型变量x，y，z赋初值10，下面正确的Python赋值语句是（　　）。C
- A．xyz=10　　B．x=10 y=10 z=10
- C．x=y=z=10　　D．x=10,y=10,z=10

- 8.语句x=input()执行时，如果从键盘输入12并按回车键，则x的值是（　）。
- A.12　B.12.0　C.1e2　D.'12'
- 9.语句x,y=eval(input())执行时，输入数据格式错误的是（　）。
- A.3 4　B.(3,4)　C.3,4　D.[3,4]
- 10.语句print('x=${:7.2f}'.format(123.5678))执行后的输出结果是（　）。选项中的□代表空格。
- A.x=□123.56　B.$□123.57
- C.x=$□123.57　D.x=$□123.56

- 8．语句x=input()执行时，如果从键盘输入12并按回车键，则x的值是（　　）。D
- A．12　B．12.0　C．1e2　D．'12'
- 9．语句x,y=eval(input())执行时，输入数据格式错误的是（　　）。A
- A．3 4　B．(3,4)　C．3,4　D．[3,4]
- 10．语句print('x=${:7.2f}'.format(123.5678))执行后的输出结果是（　　）。选项中的□代表空格。C
- A．x=□123.56　B．$□123.57
- C．x=$□123.57　D．x=$□123.56

- 11.print('{:7.2f}{:2d}'.format(101/7,101%8))的运行结果是（　）。(□代表空格)
- A.{:7.2f}{:2d}　　B.□□14.43□5
- C.□14.43□□5　D.□□101/7□101%8
- 12.下列程序的运行结果是（　）。
- x=y=10
- x,y,z=6,x+1,x+2
- print(x,y,z)
- A.10 10 6　B.6 10 10
- C.6 7 8　　D.6 11 12

- 11．print('{:7.2f}{:2d}'.format(101/7,101%8))的运行结果是（　　）。B
- A．{:7.2f}{:2d}
- B．□□14.43□5（□代表空格）
- C．□14.43□□5（□代表空格）
- D．□□101/7□101%8（□代表空格）
- 12．下列程序的运行结果是（　　）。D
- x=y=10
- x,y,z=6,x+1,x+2
- print(x,y,z)
- A．10 10 6　B．6 10 10　C．6 7 8　D．6 11 12

- 二、填空题
- 1. 流程图是描述（　）的常用工具。
- 2. 在Python语句行中使用多条语句，语句之间使用（　）分隔；如果语句太长，可以使用
- （　）作为续行符。
- 3. Python语言通过（　）来区分不同的语句块。
- 4. 在Python中，赋值的含义是使变量（　）一个数据对象，该变量是该数据对象的（　）。

- 二、填空题
- 1．流程图是描述（　）的常用工具。算法
- 2．在Python语句行中使用多条语句，语句之间使用（　）分隔；如果语句太长，可以使用（　）作为续行符。分号，反斜杠\
- 3．Python语言通过（　）来区分不同的语句块。缩进对齐
- 4．在Python中，赋值的含义是使变量（　）一个数据对象，该变量是该数据对象的（　）。指向，别名

- 5. 和x/=x*y+z等价的语句是（　）。
- 6. 语句print(‘AAA’,"BBB“,sep='-‘,end='!')执行的结果是（　）。
- 7. 下列Python语句的输出结果是（　）。
- print("数量{0},单价{1} ".format(100,285.6))
- print(str.format("数量{0},单价{1:3.2f} ",100,285.6))
- print("数量%4d,单价%3.3f "%(100,285.6))

- 5．和x/=x*y+z等价的语句是（　）。x=x/(x*y+z)
- 6．语句print(‘AAA’,“BBB”,sep=‘-’,end=‘!’)执行的结果是（　）。AAA-BBB!
- 7．下列Python语句的输出结果是（　）。
- print("数量{0},单价{1} ".format(100,285.6))
- print(str.format("数量{0},单价{1:3.2f} ",100,285.6))
- print("数量%4d,单价%3.3f "%(100,285.6))
- 数量100,单价285.6
- 数量100,单价285.60
- 数量 100,单价285.600

- 8. 下列Python语句的输出结果是（　）。
- print(format("121",">20"))
- print(format("12321",">20"))

- 8. 下列Python语句的输出结果是（ ）。
- print("1".rjust(20," ")) #右缩进函数
- print(format("121",">20"))
- print(format("12321",">20"))

```
                   1
                 121
               12321
>>>
```

- 三、问答题
- 1. 用Python语句完成下列操作：
- （1）将变量i的值增加1。
- （2）i的立方加上j，并将其结果保存到i中。
- （3）将2^32-1的值存放到g中。
- （4）将2位自然数的个位与十位互换，得到一个新的数（不考虑个位为0的情况）。

- 2. 设a=10，分别独立执行下列语句后a的值是多少?
- （1）a+=a
- （2）a*=2
- （3）a<<2
- （4）a,a=5,2*a
- （5）a*=1<<1
- （6）x=a;a+=x

自测题

- 一、选择题
- 1.以下哪个选项不是Python语言的保留字？（　）
- A.False　B.and　C.true　D.if
- 2.x=2,y=3,执行x,y=y,x之后,x和y的值分别是什么？（　）
- A.2,3　B.3,2　C.2,2　D.3,3
- 3.与0xf2值相等的是（　）
- A.240　B.0d242　C.0b11110010　D.0o360
- 4.以下变量名不合法的是（　）
- A.j　B.i　C.$　D.o

- 一、选择题
- 1.以下哪个选项不是Python语言的保留字？（ ）C
- A.False B.and C.true D.if
- 2.x=2,y=3,执行x,y=y,x之后,x和y的值分别是什么？（ ）B
- A.2,3 B.3,2 C.2,2 D.3,3
- 3.与0xf2值相等的是（ ）C
- A.240 B.0d242 C.0b11110010 D.0o360
- 4.以下变量名不合法的是（ ）。C
- A.j B.i C.$ D.o

- 5.print(len("hello world!"))的输出结果为（　）
- A.10　B.11　C.9　D.12
- 6.以下赋值语句中合法的是（　）
- A.x=2,y=3　B.x=y=3　C.x=2y=3　D.x=(y=3)
- 7.'12'+'34'的输出结果是（　）
- A.'46'　B.46　C.'1234'　D.1234
- 8.表达式“4>3>2”的输出结果是（　）
- A.False　B.True　C.0　D.出错

- 5.print(len("hello world!"))的输出结果为（　）D
- A.10　B.11　C.9　D.12
- 6.以下赋值语句中合法的是（　）B
- A.x=2,y=3　B.x=y=3　C.x=2y=3　D.x=(y=3)
- 7.'12'+'34'的输出结果是（　）C
- A.'46'　B.46　C.'1234'　D.1234
- 8.表达式“4>3>2”的输出结果是（　）B
- A.False　B.True　C.0　D.出错

- 9.表达式“3==3==3”的输出结果是（　）
- A.False　B.True　C.0　D.出错
- 10.表达式“3>2==True”的输出结果是（　）
- A.False　B.True　C.0　D.出错
- 11.表达式“(3>2)==True”的输出结果是（　）
- A.False　B.True　C.0　D.出错
- 12.Python程序中，用符号什么表示注释的开始？（　）
- A.%　B.//　C.#　D.$

- 9.表达式“3==3==3”的输出结果是（　）B
- A.False　B.True　C.0　D.出错
- 10.表达式“3>2==True”的输出结果是（　）A
- A.False　B.True　C.0　D.出错
- 11.表达式“(3>2)==True”的输出结果是（　）B
- A.False　B.True　C.0　D.出错
- 12.Python程序中，用符号什么表示注释的开始？（　）C
- A.%　B.//　C.#　D.$

- 13.以下哪个是Python不支持的数据类型？（　）
- A.char　B.int　C.float　D.list
- 14.表达式“True+True”的值是（　）
- A.True　B.False　C.1　D.2
- 15.语句isinstance(123,int)的输出是（　）
- A.0　B.1　C.True　D.Flase
- 16.语句isinstance(5.7//3,float)的输出是（　）
- A.0　B.1　C.True　D.Flase

- 13.以下哪个是Python不支持的数据类型？（　）

  A
- A.char　B.int　C.float　D.list
- 14.表达式“True+True”的值是（　）D
- A.True　B.False　C.1　D.2
- 15.语句isinstance(123,int)的输出是（　）C
- A.0　B.1　C.True　D.Flase
- 16.语句isinstance(5.7//3,float)的输出是（　）C
- A.0　B.1　C.True　D.Flase

- 二、填空题
- 1. Python程序中，用缩进表达层次关系，在缩进的前一行最后，需要使用（　）符号来表示。
- 2. 123和'123'分别表示（　）类型和（　）类型。
- 3. Python程序设计中用于输入和输出的函数分别是（　）和（　）。
- 4. Python中使用（　）保留字引用当前程序以外的功能模块库。

- 二、填空题
- 1. Python程序中，用缩进表达层次关系，在缩进的前一行最后，需要使用（　）符号来表示。冒号:
- 2. 123和'123'分别表示（　）类型和（　）类型。整数 字符串
- 3. Python程序设计中用于输入和输出的函数分别是（　）和（　）。input() print()
- 4. Python中使用（　）保留字引用当前程序以外的功能模块库。import

- 5.hex(10)的结果为（　），oct(10)的结果为（　）。
- 6. type(2+3j)的输出结果是（　），type('2')的输出结果是（　）。
- 7.a=1.2345，将print('{0:10.2f}'.format(a))改写为等效的内置函数format()语句是（　）
- 8.若a=1.2345，则将print('{0:010.2f}{1:*^15}'.format(a,123))改写为等效的内置函数format()语句是（　）

- 5.hex(10)的结果为（　），oct(10)的结果为（　）。

  '0xa' '0o12'

- 6. type(2+3j)的输出结果是（　），type('2')的输出结果是（　）。<class 'complex'> <class 'str'>

- 7.a=1.2345，将print('{0:10.2f}'.format(a))改写为等效的内置函数format()语句是（　）　print(format(a,'10.2f'))

- 8.若a=1.2345，则将print('{0:010.2f}{1:*^15}'.format(a,123))改写为等效的内置函数format()语句是（　）　print(format(a,'010.2f'),format(123,'*^15'))

- 三、编程题
- 1. 读入两个整数、两个浮点数和两个复数，分别求其和与积。
- 2. 输入一个十进制整数，分别输出其二进制、八进制、十六进制字符串。