Python语言程序设计

第二章 Python程序基础

Part One

第一讲 一个简单的Python程序实例

一个简单的实例

- 问题：计算圆的面积。
- 分析
  - （1）从用户处得到圆的半径
  - （2）利用公式计算圆面积
  - （3）显示结果

```
#2-1 Computing Area
radius = 2
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

```
The area for the circle of radius 2 is 12.56636
```

Python注释

```
'''A Simple Program
Compute Area'''
radius = 2  #Assign a value to radius
area = 3.14159 * radius * radius
#Display Results
print("The area for the circle of radius", radius, "is", area)
```

Python使用#开始一行单行注释。

Python可以使用一对三引号表示多行注释（段注释）。

注释不是程序语句，被解释器忽略。

Python标识符

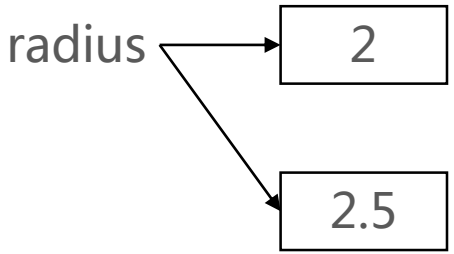
```
radius = 2
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

- 标识符用于命名程序中变量和函数这样的元素。
- 标识符的命名规则：
  - 允许用大写小写字母、数字、下划线及它们的各种组合，长度不限
  - 首字符不能是数字
  - 不能是关键字（保留字）
  - 命名尽量做到“见名知义”
- 标识符区分大小写

```
radius = 2
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)

radius = 2.5
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

Python中，变量是指向存储在内存中某个值的名字。



赋值

```
radius = 2
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

- Python采用“=”表示赋值，其含义为把等号右侧的表达式计算结果（或函数的返回值）赋值给等号左边的变量。

```
x = 1
y = 5 * (3 / 2) + 1
y = y + 2
```

```
1 = x    #wrong
```

赋值

- 一个值被赋给多个变量

```
x = y = z = 1
```

```
z = 1
y = z
x = y
```

- 同时赋值（等号右边的逗号隔开的表达式的值分别赋给等号左边的对应变量）

```
x, y = 1, 2
```

```
x = 1
y = 2
```

赋值

- 如何交换两个变量的值

其它语言的常规方法

```
x = 1
y = 2
temp = x
x = y
y = temp
```

Python的方法

```
x, y = 1, 2
x, y = y, x
```

变量赋值

- 所有的变量在使用前必须被创建

错误

```
x = x + 1
```

NameError: name 'x' is not defined

正确

```
x = 1
x = x + 1
```

Part Two

第二讲　改进的Python程序实例（一）

```
#2-1 Computing Area
radius = 2
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

如果需要修改半径的值，怎么办？
如何让程序更具通用性？

使用input函数接受用户输入。

输入

Python使用input函数从控制台获取用户输入。

获得用户输入之前，可将提示性文本以参数的形式传递给input函数。

<变量> = input(<提示性文本>)

无论用户输入什么，input函数均以字符串形式返回结果。

可以使用eval函数对字符串求值并转换为一个数值。

```
#2-2 Computing Area
radius = eval(input(“Please enter a value for radius:”))
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

```
#2-2 Computing Area
radius = eval(input(“Please enter a value for radius:”))
area = 3.14159 * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

程序中，圆周率π是一个常量。
可以使用一个描述性的名字PI来代表3.14159这个值。

```
#2-3 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
area = PI * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

可以简单地创建一个变量来表示常量，通常采用大写字母来表示。

使用常量的好处：

- 如果一个值需多次使用，不必重复输入。
- 如果需要修改该值，只需修改一处。
- 提高程序的可读性

Part Three

第三讲　改进的Python程序实例（二）

```
#2-3 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
area = PI * radius * radius
print("The area for the circle of radius", radius, "is", area)
```

如果用户输入的不是正数，怎么办？

```
#2-4 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
if radius > 0:
    area = PI * radius * radius
    print("The area for the circle of radius", radius, "is", area)
else:
    print (“Input error“)
```

使用分支结构，判定输入的radius是否是正数，根据判断结果执行对应的分支。

```
#2-4 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
if radius > 0:
    area = PI * radius * radius
    print("The area for the circle of radius", radius, "is", area)
else:
    print (“Input error“)
```

缩进

```
#2-4 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
if radius > 0:
    area = PI * radius * radius
    print("The area for the circle of radius", radius, "is", area)
else:
    print (“Input error“)
```

Python采用严格的“缩进”来表明程序的框架！

缩进表达所属关系。

缩进可用TAB键实现，也可用多个空格（通常为4个）。

```
#2-4 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
if radius > 0:
    area = PI * radius * radius
    print("The area for the circle of radius", radius, "is", area)
else:
    print (“Input error“)
```

如果想让用户重复输入新的radius计算结果，如何修改程序？

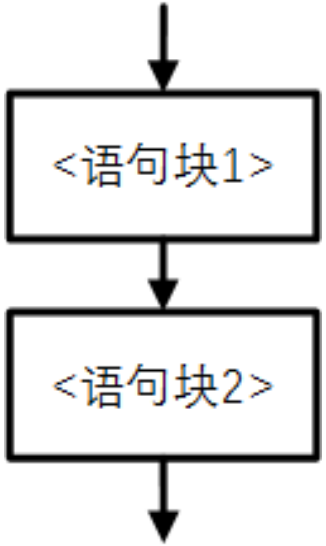
使用循环结构，只要输入的radius是正数就一直计算并重复输入。

```
#2-5 Computing Area
radius = eval(input(“Please enter a value for radius:”))
PI = 3.14159
while radius > 0:
    area = PI * radius * radius
    print("The area for the circle of radius", radius, "is", area)
    radius = eval(input(“Please enter a value for radius:”))
```

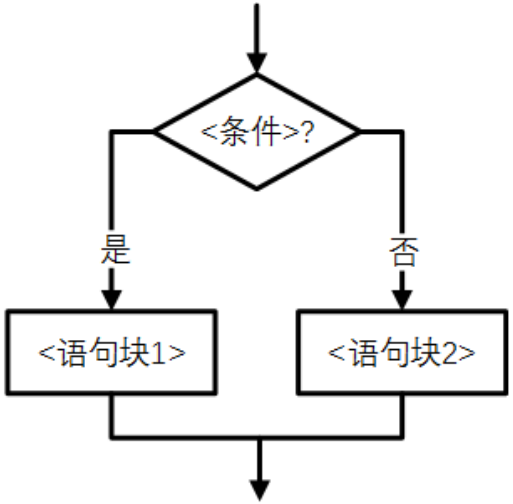
程序结构

- 程序由三种基本结构组成：

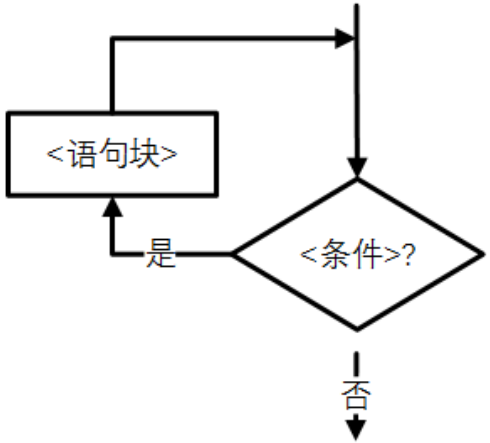
顺序结构



分支结构



循环结构

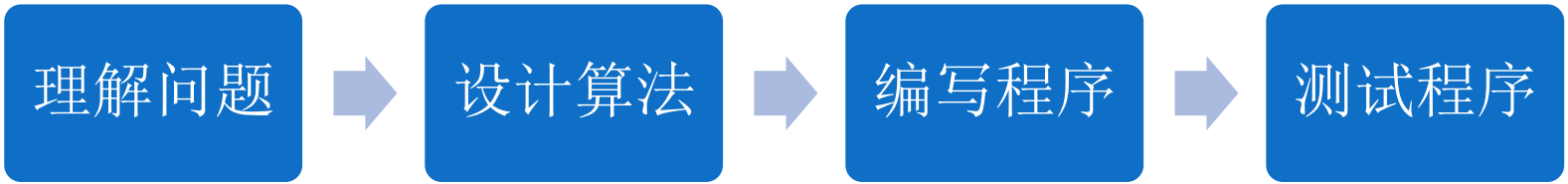


- 任何程序都由这三种基本结构组合而成。

Part Four

第四讲　程序开发流程

程序开发流程



实例

- 问题：将10000元存入银行，年利率固定为每年5%，多少年后可以翻倍？

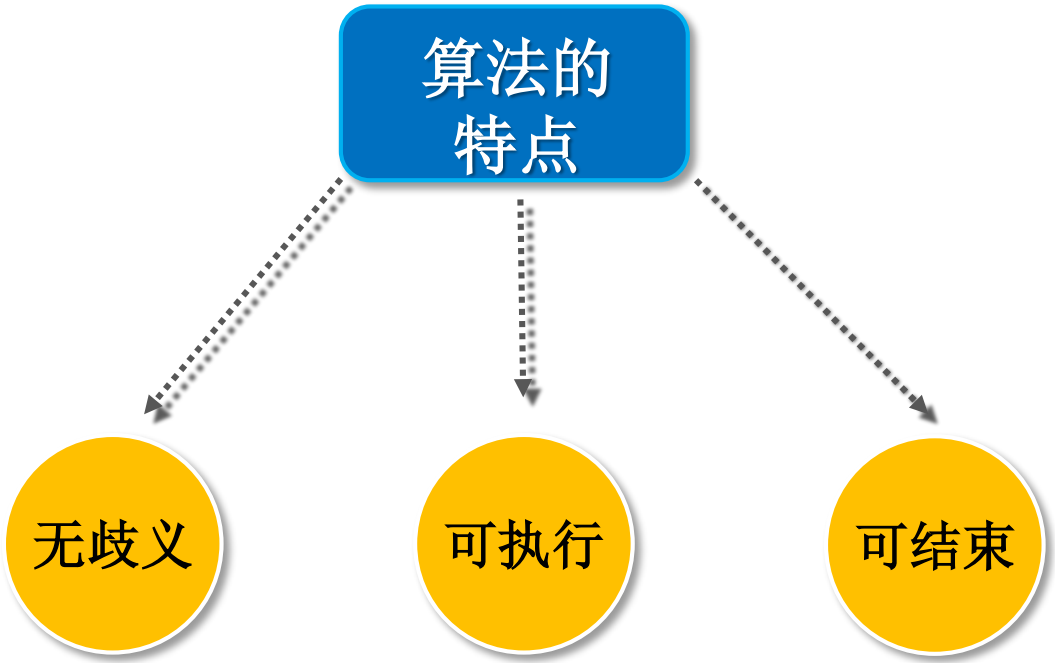
第一步：理解问题

输入：10000元本金，5%的固定年利率
输出：翻倍需要的年数

第二步：设计算法

伪代码描述

1. 设置year=0，balance=10000
2. 当balance小于20000时，重复下面3-5的步骤
3. year值增加1
4. interest=balance*5%
5. 将interest增加到balance上
6. 报告最终的year作为答案


算法的
特点
无歧义
可执行
可结束

- 第三步：编写程序

```
year = 0
balance = 10000
while balance < 20000:
    year = year + 1
    interest = balance * 0.05
    balance = balance + interest
print(year)
```

- 第四步：测试程序

通常利用几组样本输入数据来验证输出是否正确来测试。

Part Five

第五讲　turtle实例——计算两点间距离

计算两点间距离

问题：输入平面上两个点的坐标，输出两点间的距离。使用turtle画出两点间的连线并在图上显示该距离。

分析：

1、计算两点间距离和画图是两个相对独立的任务，可以分开进行。

2、画图时需要知道两个点的坐标和距离，所以后进行。

3、计算两点间距离需要得到用户输入的坐标，并使用公式$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$计算距离。

算术运算符

算数运算符	含义	举例
+	加法	10+3=13
-	减法	10-3=7
*	乘法	10*3=30
/	除法（普通）	10/4=2.5
//	除法（整除）	10//4=2
%	求余	10%3=1
**	指数	2**3=8

算术表达式

- 用Python的表达式表示公式$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

```
((x2 – x1) * (x2 – x1) + (y2 – y1) * (y2 – y1)) ** 0.5
```

```
((x2 – x1) ** 2 + (y2 – y1) ** 2)** 0.5
```

程序演示

```
import turtle
x1, y1 = eval(input("Please input Point1: "))
x2, y2 = eval(input("Please input Point2: "))
distance = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
print("the distance between the two points is", distance)
turtle.penup()
turtle.goto(x1, y1)
turtle.pendown()
turtle.write("Point1")
turtle.goto(x2, y2)
turtle.write("Point2")
turtle.penup()
turtle.goto((x1 + x2) / 2, (y1 + y2) / 2)
turtle.write(distance)
```

Python语言程序设计