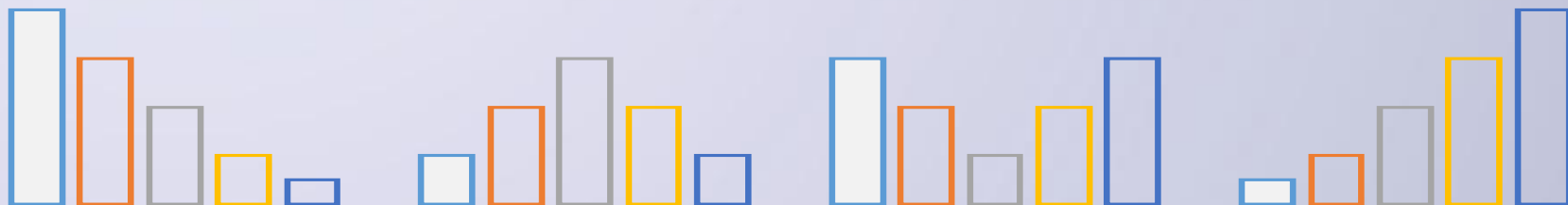
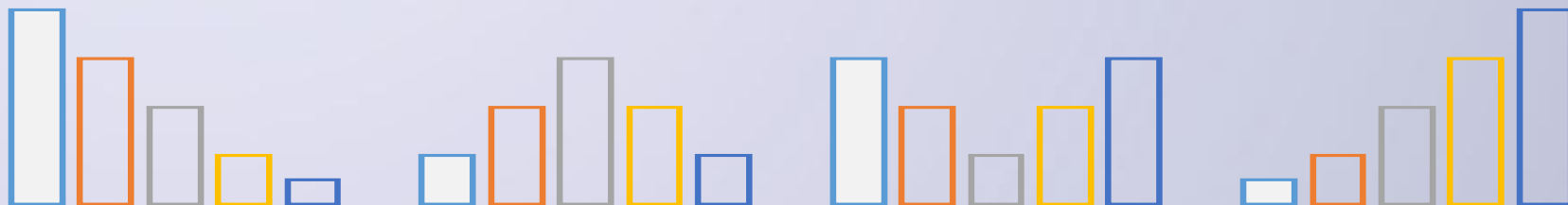


# Python语言程序设计

北京理工大学 嵩天



# 第2章 Python程序实例解析





# 温度转换程序实例



# 温度体系


温度刻画存在不同体系，摄氏度以1标准大气压下水的结冰点为0度，沸点为100度，将温度进行等分刻画。华氏度以1标准大气压下水的结冰点为32度，沸点为212度，将温度进行等分刻画。



# 温度转换实例

问题：如何利用Python程序进行摄氏度和华氏度之间的转换

■ 步骤1：分析问题的计算部分：采用公式转换方式解决计算问题



# 温度转换实例

## ■ 步骤2：确定功能


输入：华氏或者摄氏温度值、温度标识

处理：温度转化算法

输出：华氏或者摄氏温度值、温度标识

F表示华氏度，82F表示华氏82度

C表示摄氏度，28C表示摄氏28度



# 温度转换实例

## ■ 步骤3：设计算法

根据华氏和摄氏温度定义，转换公式如下：

$$C = (F - 32) / 1.8$$

$$F = C * 1.8 + 32$$

其中，C表示摄氏温度，F表示华氏温度

# 温度转换实例

## ■ 步骤4：编写程序

```
#TempConvert.py
val = input("请输入带温度表示符号的温度值(例如: 32C): ")
if val[-1] in ['C', 'c']:
    f = 1.8 * float(val[0:-1]) + 32
    print("转换后的温度为: %.2fF"%f)
elif val[-1] in ['F', 'f']:
    c = (float(val[0:-1]) - 32) / 1.8
    print("转换后的温度为: %.2fC"%c)
else:
    print("输入有误")
```





# 温度转换实例

## ■ 步骤5：调试、运行程序

在系统命令行上运行如下命令执行程序：

```
C:\>python TempConvert.py
```

或者：使用IDLE打开上述文件，按F5运行（推荐）

输入数值，观察输出



# Python语法元素分析



# 程序的格式框架

Python语言采用严格的“缩进”来表明程序的格式框架。缩进指每一行代码开始前的空白区域，用来表示代码之间的包含和层次关系。

1个缩进 = 4个空格

- 用以在Python中标明代码的层次关系
- 缩进是Python语言中表明程序框架的唯一手段



# 程序的格式框架

## 单层缩进

```
#e1.1TempConvert.py
TempStr = input("请输入带有符号
if TempStr[-1] in ['F','f']:
    ↪ C = (eval(TempStr[0:-1]) -
        print("转换后的温度是{:.2f}C
elif TempStr[-1] in ['C','c']:
    ↪ F = 1.8*eval(TempStr[0:-1])
        print("转换后的温度是{:.2f}F
else:
    ↪ print("输入格式错误")
```

## 多层缩进

```
DARTS = 1000
hits = 0.0
clock()
for i in range(1, DARTS):
    ↪ x, y = random(), random()
        ↪ dist = sqrt(x ** 2 + y ** 2)
            if dist <= 1.0:
                ↪ hits = hits + 1
pi = 4 * (hits/DARTS)
print("Pi的值是{:.2f}".format(pi))
```



# 注释

注释：程序员在代码中加入的说明信息，不被计算机执行

注释的两种方法：

- 单行注释以#开头

```
#Here are the comments
```

- 多行注释以''' 开头和结尾

```
'''
```

```
This is a multiline comment  
used in Python
```

```
'''
```



# 命名与保留字

- 常量：程序中值不发生改变的元素
- 变量：程序中值发生改变或者可以发生改变的元素

Python语言允许采用大写字母、小写字母、数字、下划线(\_)和汉字等字符及其组合给变量命名，但名字的首字符不能是数字，中间不能出现空格，长度没有限制

注意：标识符对大小写敏感，python和Python是两个不同的名字



# 命名与保留字

- ✓ 保留字，也称为关键字，指被编程语言内部定义并保留使用的标识符。
- ✓ 程序员编写程序不能定义与保留字相同的标识符。
- ✓ 每种程序设计语言都有一套保留字，保留字一般用来构成程序整体框架、表达关键值和具有结构性的复杂语义等。
- ✓ 掌握一门编程语言首先要熟记其所对应的保留字。



# 命名与保留字

✓ Python 3.x保留字列表 (33个)

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



# 字符串

- Python语言中，字符串是用两个双引号 “ ” 或者单引号 ‘ ’ 括起来的一个或多个字符。
- Python字符串的两种序号体系





# 赋值语句

■ Python语言中，`=` 表示“赋值”，即将等号右侧的值计算后将结果值赋给左侧变量，包含等号

(`=`) 的语句称为“赋值语句”

■ 同步赋值语句：同时给多个变量赋值

`<变量1>, ..., <变量N> = <表达式1>, ..., <表达式N>`

# 赋值语句

例：将变量x和y交换

■采用单个赋值，需要3行语句：


即通过一个临时变量t缓存x的原始值，然后将y值赋给x，再将x的原始值通过t赋值给y。

■采用同步赋值语句，仅需要一行代码：

```
>>>t=x  
>>>x=y  
>>>y=t
```

等价于

```
>>>x, y=y, x
```



# input()函数

- 获得用户输入之前，input()函数可以包含一些提示性文字  
`<变量> = input(<提示性文字>)`

```
>>>input("请输入: ")
```

```
请输入: python
```

```
'python'
```

```
>>> input("请输入: ")
```

```
请输入: 1024.256
```


```
'1024.256'
```



# 分支语句

- 分支语句是控制程序运行的一类重要语句，它的作用是根据判断条件选择程序执行路径，使用方式如下：

```
if <条件1>:  
    <语句块1>  
elif <条件2>:  
    <语句块2>  
...  
else:  
    <语句块N>
```



# eval ( ) 函数

- `eval(<字符串>)` 函数是 Python 语言中一个十分重要的函数，它能够以 Python 表达式的方式解析并执行字符串，将返回结果输出

```
>>>x = 1
>>>eval("x + 1")
2
>>>eval("1.1 + 2.2")
3.3
```

# 赋值语句

例：将变量x和y交换

■采用单个赋值，需要3行语句：

即通过一个临时变量t缓存x的原始值，然后将y值赋给x，再将x的原始值通过t赋值给y。

■采用同步赋值语句，仅需要一行代码：

```
>>>t=x  
>>>x=y  
>>>y=t
```

等价于


```
>>>x, y=y, x
```



# 输出函数

- `print()`函数用来输出字符信息，或以字符形式输出变量。
- `print()`函数可以输出各种类型变量的值。
- `print()`函数通过%来选择要输出的变量。





# 实例

- 用户输入两个数字，计算它们的平均数，并输出平均数

```
num1 = input("The first number is")
num2 = input("The second number is")
avg_num = (float(num1) + float(num2)) / 2
print("The average number is %f" % avg_num)
```



# 循环语句

- 循环语句：控制程序运行，根据判断条件或计数条件确定一段程序的运行次数


- 遍历循环，基本过程如下

```
for i in range (<计数值>):  
    <表达式1>
```

- 例如，使某一段程序连续运行10次

```
for i in range (10):  
    <源代码>
```

- 其中，变量*i*用于计数



# turtle库和蟒蛇绘制程序



# python



```
import turtle
def drawSnake(rad, angle, len)
for i in range(len):
    turtle.circle(rad, angle)
    turtle.circle(-rad, angle)
    turtle.fd(rad*2/3)
    turtle.circle(neckrad+1, 180)
    turtle.circle(rad*3/5)
pythonsize = 100, 800, 0, 0)
turtle.pensize(pythonsize)
turtle.pencolor("#3B99D9")
turtle.seth(-40)
drawSnake(40, 80, 5, pythonsize)
if __name__ == '__main__':
    main()
```

The Python logo, consisting of two interlocking snakes, is positioned to the left of the title.

# Python小蛇

- Python英文是蟒蛇的意思
- 通过下面的例子，来实践用Python语言输出图形效果。



# Python蟒蛇绘制实例



```
import turtle

def drawSnake(rad, angle, len, neckrad):
    for i in range(len):
        turtle.circle(rad, angle)
        turtle.circle(-rad, angle)
        turtle.circle(rad, angle/2)
        turtle.fd(rad)
        turtle.circle(neckrad+1, 180)
        turtle.fd(rad*2/3)

def main():
    turtle.setup(1300, 800, 0, 0)
    pythonsize = 30
    turtle.pensize(pythonsize)
    turtle.pencolor("blue")
    turtle.seth(-40)
    drawSnake(40, 80, 5, pythonsize/2)

main()
```



# Python语法元素

■ `import turtle`

■ `import`是一个关键字，用来引入一些外部库，这里的含义是引入一个名字叫turtle的函数库



# Turtle库

- Turtle库是Python语言中一个很流行的绘制图像的函数库
- 使用turtle库，同学们头脑里需要有这样一个概念：
  - 想象一个小乌龟，在一个横轴为x、纵轴为y的坐标系原点，(0,0)位置开始
  - 它根据一组函数指令的控制，在这个平面坐标系中移动，从而在它爬行的路径上绘制了图形





# def定义函数

- **def** 用于定义函数，这段程序中，共出现两次def关键词，包含两个函数drawSnake和main。
- 函数是一组代码的集合，用于表达一个功能，或者说，函数表示一组代码的归属，函数名称是这段代码的名字。
- def所定义的函数在程序中未经调用不能直接执行，需要通过函数名调用才能够执行。



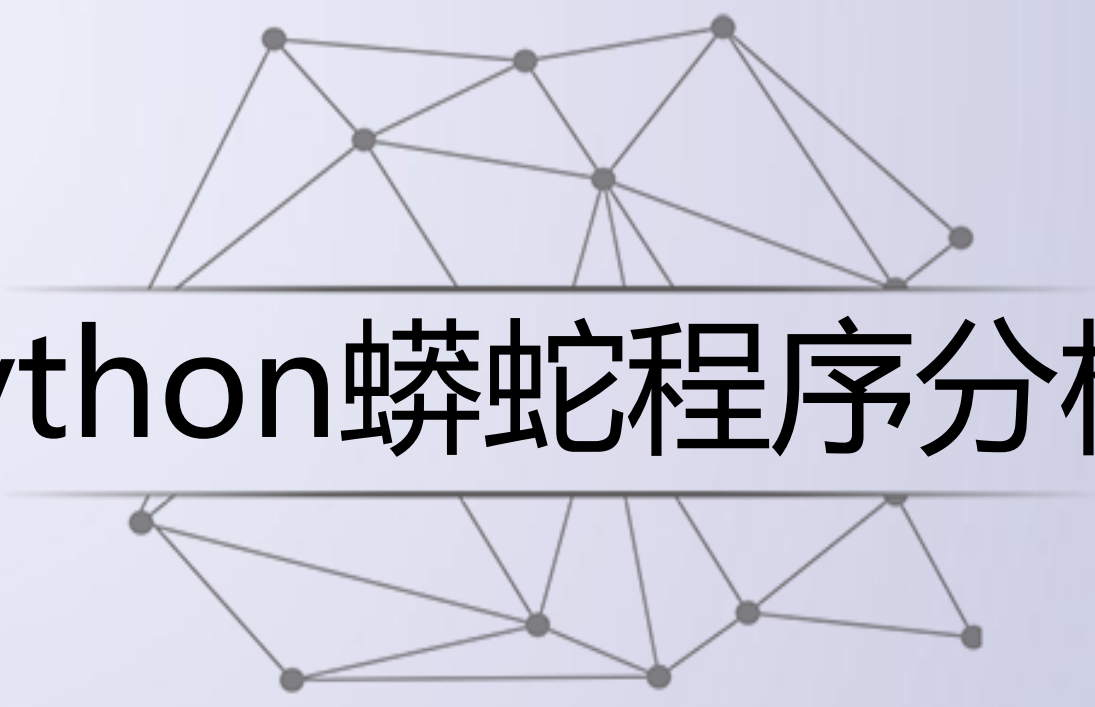
# 程序运行

- 两个def语句定义的函数所包含语句与def行存在缩进关系，def后连续的缩进语句都是这个函数的一部分。
- 由于def定义的函数在程序中未经调用不会被执行，整个程序第一条执行的语句是main()，它表示执行名字为main()的函数。



# 程序运行

- 从而，该程序跳转到main()函数定义的一组语句中执行，即开始执行 turtle.setup()语句
- 同样的，main()函数的最后一条语句调用了drawSnake()函数，当执行到这条语句时，程序跳转到drawSnake()函数中运行。



# Python蟒蛇程序分析




# Python小蛇实例

```
import turtle

def drawSnake(rad, angle, len, neckrad):
    for i in range(len):
        turtle.circle(rad, angle)
        turtle.circle(-rad, angle)
    turtle.circle(rad, angle/2)
    turtle.fd(rad)
    turtle.circle(neckrad+1, 180)
    turtle.fd(rad*2/3)

def main():
    turtle.setup(1300, 800, 0, 0)
    pythonsize = 30
    turtle.pensize(pythonsize)
    turtle.pencolor("blue")
    turtle.seth(-40)
    drawSnake(40, 80, 5, pythonsize/2)

main()
```

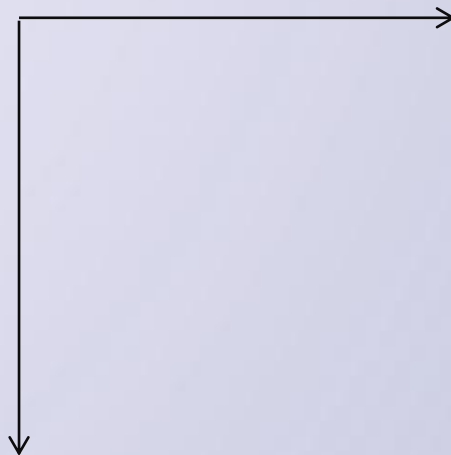



# 程序

- 程序运行main()函数中语句，遇到setup函数
- Turtle中的turtle.setup()函数用于启动一个图形窗口，它有四个参数
  - turtle.setup(width, height, startx, starty)
- 分别是：启动窗口的宽度和高度
- 表示窗口启动时，窗口左上角在屏幕中的坐标位置。

# 程序

- 我们所使用的显示屏幕也是一个坐标系，该坐标系以左上角为原点，向左和向下分别是x轴和y轴。
- 蟒蛇程序代码启动一个1300像素宽、800像素高的窗口，该窗口的左上角是屏幕的左上角。






# 程序

- Turtle中的turtle.pensize()函数表示小乌龟运动轨迹的宽度。
- 它包含一个输入参数，这里我们把它设为30像素，用pythonsize变量表示。



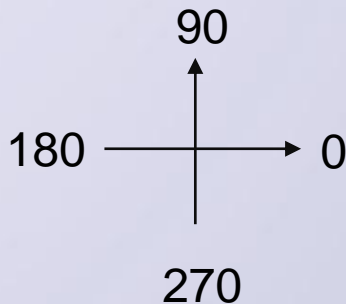


# 程序


- Turtle中的turtle.pencolor()函数表示小乌龟运动轨迹的颜色。
- 它包含一个输入参数，这里我们把它设为蓝色，blue，其他颜色单词也可以使用。Turtle采用RGB方式来定义颜色，如果希望获得和图片中颜色一致的小蛇，请输入turtle.pencolor( "#3B9909" )

# 程序

- Turtle中的turtle.seth(angle)函数表示小乌龟启动时运动的方向。它包含一个输入参数，是角度值。
- 其中，0表示向东，90度向北，180度向西，270度向南；负值表示相反方向。
- 程序中，我们让小乌龟向-40度启动爬行，即：向东南方向40度。



standard mode	logo mode
0 - east	0 - north
90 - north	90 - east
180 - west	180 - south
270 - south	270 - west



# 程序

- `main()`函数给出了小乌龟爬行的窗体大小，爬行轨迹颜色和宽度以及初始爬行的方位。
- 最后，调用`drawSnake`函数启动绘制蟒蛇功能。
- `drawSnake`函数有四个参数，根据调用时给出的参数，分别将40传递给`rad`、80给`angle`，5给`len`，15给`neckrad`



# turtle.circle()函数功能

- turtle.circle()函数让小乌龟沿着一个圆形爬行

- 参数rad描述圆形轨迹半径的位置

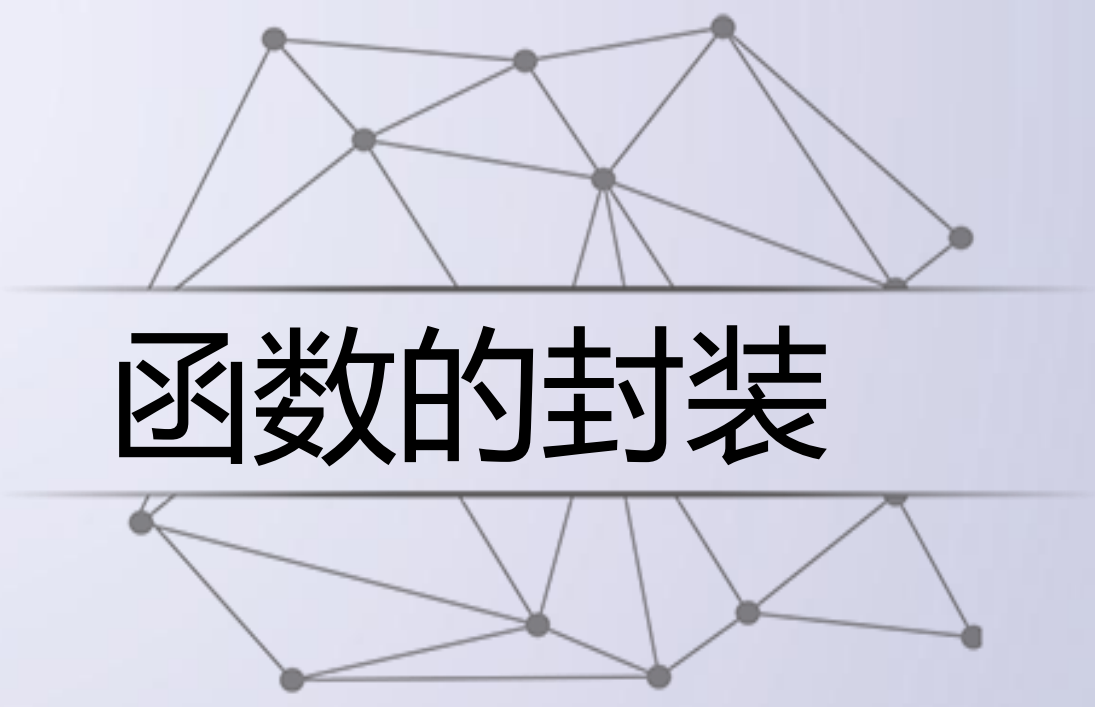
这个半径在小乌龟运行的左侧rad远位置处，如果rad为负值，则半径在小乌龟运行的右侧

- 参数angle表示小乌龟沿着圆形爬行的弧度值



# turtle.fd()函数功能

- turtle.fd()函数也可以用turtle.forward()表示乌龟向前直线爬行移动
- 表示小乌龟向前直线爬行移动，它有一个参数表示爬行的距离



# 函数的封装



# Python的函数封装

蟒蛇程序功能可以分成两类：

- 绘制图形前对画笔的设置，包括颜色、尺寸、初始位置等
- 以及绘制Python蟒蛇的功能。

由于蟒蛇绘制的功能相对独立，可以用函数来封装



# 函数封装

```
#e2.3DrawPython.py
import turtle
def drawSnake(radius, angle, length):
    turtle.seth(-40)
    for i in range(length):
        turtle.circle(radius, angle)
        turtle.circle(-radius, angle)
    turtle.circle(radius, angle/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40* 2/3)
turtle.setup(650, 350, 200, 200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
drawSnake(40, 80, 4)
turtle.done()
```