Python程序实例解析

学习目标：

1 掌握解决计算问题的一般方法；

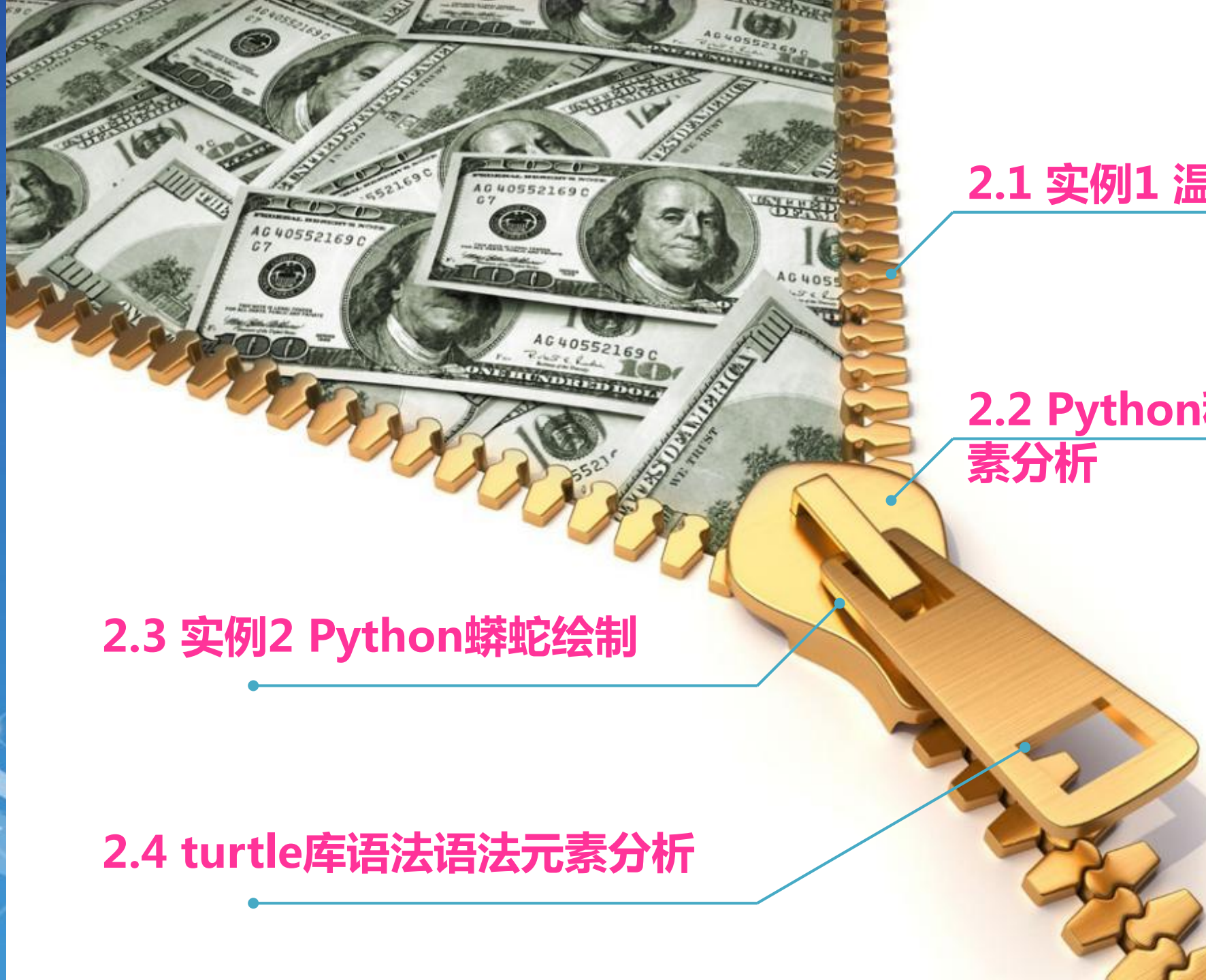
2 掌握Python语言的基本语法，包括缩进、变量、命名等

3 掌握Python语言绘制图形的一般方法

4 了解Python标准库的导入和使用

目录页

Contents Page



2.1 实例1 温度转换

一．分析问题

二．划分边界：确定输入、处理、输出

三．设计算法：

C=(F-32)/1.0

F=C*1.8+32

四．编写程序：

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [‘F’,’f’]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [‘C’ , ‘c’] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.          print(“转换后的温度是{:.2f}F”.format(F))
9.  else
10.         print(“输入格式错误”)
```

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [‘F’,’f’]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [‘C’ , ‘c’] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.           print(“转换后的温度是{:.2f}F”.format(F))
9.  else :
10.         print(“输入格式错误”)
```

程序结构框架：

- 为使得程序结构清晰，子语句采用缩进的格式进行编写，如④⑤⑦⑧⑩行语句，如果有多层，采用继续缩进的方式；
- 当父语句结束需要包含子语句时，在父语句末尾用分号分割；如②⑧行；
- 为增加程序的可读性，可在程序的任何地方增加注释，单行注释采用#，表示#后所有语句均不参加程序的翻译，多行注释采用’’’开始和结尾

2. 2 Python程序语法元素分析

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [‘F’,’f’]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [‘C’ , ‘c’] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.          print(“转换后的温度是{:.2f}F”.format(F))
9.  else :
10.         print(“输入格式错误”)
```

命名和保留字：

Python用变量来保存和表示具体的数据，为方便起见，给每个变量以标识符命名，称为变量名，标识符的命名规则：以字母、数字、下划线和汉字构成，并且首字符不能是数字,如④⑤⑦⑧行中的C和F

注：Python中的标识符大小写敏感；
不能与Python的保留字重名；

Python3.X的保留字：False、def、if、raise、None、del、import、return、True、elif、in、try、and、else、is、 while、as、except、Lambda、with、assert、finally、nonlocal、yield、Break、for、not、class、from、or、continue、global、pass (P.39)

2. 2 Python程序语法元素分析

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [ ‘F’,’ f’ ]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [ ‘C’ , ‘c’ ] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.          print(“转换后的温度是{:.2f}F”.format(F))
9.  else :
10.         print(“输入格式错误”)
```

字符串：

字符串是一种重要的数据形式，Python中的字符串是用双引号或者单引号括起来的零到多个字符组成，如②③⑤⑥⑧⑩行中都包含字符串；

- 字符串可通过下标进行索引，正向递增索引（0，1，2...）：反向递减索引（...,-4,-3,-2,-1）
- 有时可以使用区间索引的方式（N,M）表示字符串的从第N到M-1个字符，
- 并且采用区间索引时可以正向和反向混合使用,如⑦行

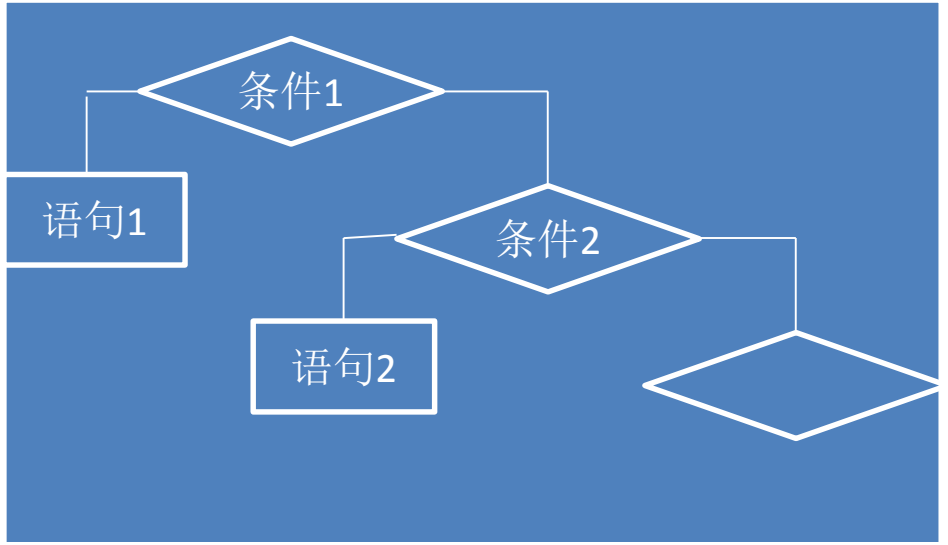
2.2 Python程序语法元素分析

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [‘F’,’f’]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [‘C’ , ‘c’] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.          print(“转换后的温度是{:.2f}F”.format(F))
9.  else :
10.         print(“输入格式错误”)
```

Python中用=进行赋值；
还可以同时对多个变量进行赋值
　如：x,y,z=2.5
同步赋值：
　x,y=y,x等同于
　t=x;
　x=y;
　y=t;

2.2 Python程序语法元素分析

```
1. #e1.1 TempConvert.py
2. TempStr=input(“请输入带符号的温度值”)
3. if TempStr[-1] in [‘F’,’f’]:
4.         C=(eval(TempStr[0:-1])-32)/1.8
5.         print(“转换后的温度是{:.2f}C”.format(C))
6. elif TempStr[-1] in [‘C’ , ‘c’] :
7.         F=1.8*eval(TempStr[0:-1])+32
8.         print(“转换后的温度是{:.2f}F”.format(F))
9. else :
10.        print(“输入格式错误”)
```




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

```
1.  #e1.1 TempConvert.py
2.  TempStr=input(“请输入带符号的温度值”)
3.  if TempStr[-1] in [ ‘F’,’ f’ ]:
4.          C=(eval(TempStr[0:-1])-32)/1.8
5.          print(“转换后的温度是{:.2f}C”.format(C))
6.  elif TempStr[-1] in [ ‘C’ , ‘c’ ] :
7.          F=1.8*eval(TempStr[0:-1])+32
8.          print(“转换后的温度是{:.2f}F”.format(F))
9.  else :
10.         print(“输入格式错误”)
```

print函数：实例中用print函数输出信息，当输出纯字符时，可直接将待输出内容传递给print函数，当输出变量值时，需要采用format()方法格式化输出，
Python中采用槽位置的方式结合format()完成变量的输出，如第⑤行中{:.2f}表示一个槽位置，{}中的内容对应format(C).

eval函数：能以Python表达式方式解析执行字符串，如：

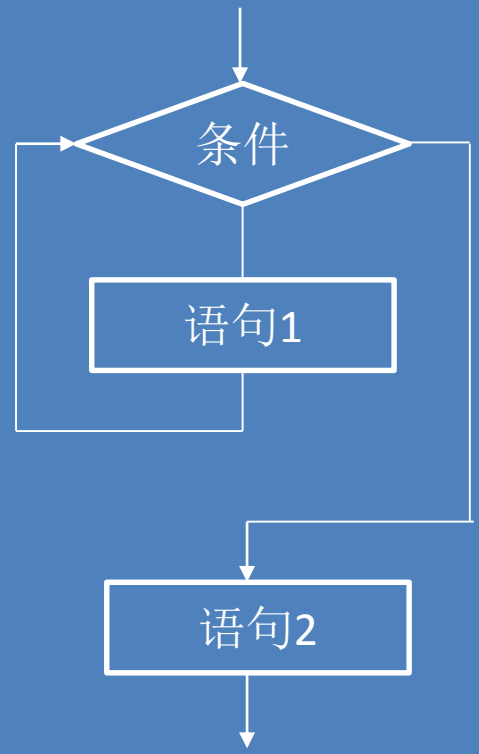
```
x=1
eval(“x+1”)          #结果为2
```

本例中，eval(TempStr[0:-1])表示将键盘输入的字符串（如“102C”）中第1到第3的字符串转化成数字102，然后参与表达式的计算。

```
1.  #e1.2 TempConvert.py
2.  TempStr=input("请输入带符号的温度值")
3.  while TempStr[-1] not in ['N','n']:
4.      if TempStr[-1] in ['F','f']:
5.          C=(eval(TempStr[0:-1])-32)/1.8
6.          print("转换后的温度是{:.2f}C".format(C))
7.      elif TempStr[-1] in ['C','c']:
8.          F=1.8*eval(TempStr[0:-1])+32
9.          print("转换后的温度是{:.2f}F".format(F))
10.     else:
11.         print("输入格式错误")
12.     TempStr = input("请输入带符号的温度值")
```

Python中的循环有多种类型，其中while循环的结构如下：

```
while <条件>
    <语句1>
<语句2>
```



```
1.  #e1.3 TempConvert.py
2.  def tempConvert(ValueStr) :
3.      if ValueStr[-1] in ['F','f']:
4.          C=(eval(ValueStr[0:-1])-32)/1.8
5.          print("转换后的温度是{:.2f}C".format(C))
6.      elif ValueStr[-1] in ['C','c']:
7.          F=1.8*eval(ValueStr[0:-1])+32
8.          print("转换后的温度是{:.2f}F".format(F))
9.      else:
10.         print("输入格式错误")
11. TempStr = input("请输入带符号的温度值")
12. tempConvert(TempStr)
```

Python中的函数包括内置函数和自定义函数，如eval()、print()等都是内置函数，

tempConvert()为自定义函数，一旦定义了自定义函数，其引用方法与内置函数相同。
内置函数的定义方式采用：
def 函数名(参数列表)

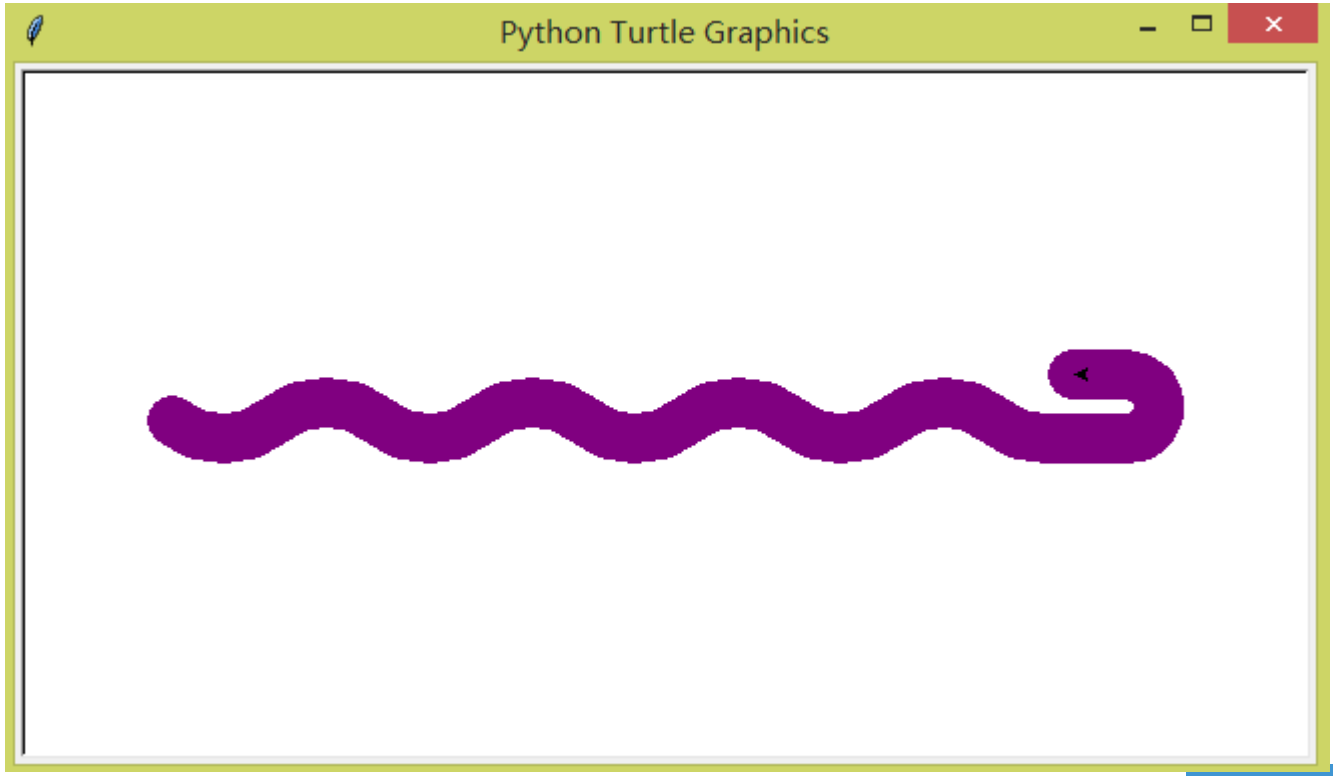
函数的调用方法采用：
函数名(实际参数)

实验2-1汇率兑换程序。按照温度转换程序的设计思路，按照1美元=6人民币的汇率编写一个美元（￥）和人民币（$）的双向兑换程序。

2.3 实例2：Python蟒蛇绘制

```
1.  #e2.1DrawPython.py
2.  import turtle
3.  turtle.setup(650,350,200,200)
4.  turtle.penup()
5.  turtle.fd(-250)
6.  turtle.pendown()
7.  turtle.pensize(25)
8.  turtle.pencolor("purple")
9.  turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```

Python（大蟒，蟒蛇），是众多Python程序员认为的吉祥物，实例2通过绘制蟒蛇绘制图形程序的基本方法。



2.3 实例2：Python蟒蛇绘制

```
1.  #e2.1DrawPython.py
2.  import turtle
3.  turtle.setup(650,350,200,200)
4.  turtle.penup()
5.  turtle.fd(-250)
6.  turtle.pendown()
7.  turtle.pensize(25)
8.  turtle.pencolor("purple")
9.  turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```

本例与实例1的区别：没有显式的输入输出语句、大多数代码用<a>.<b>的形式

这里a称为对象，b称为函数，即面向对象的程序设计方法。

2. import turtle

本行为引入函数库turtle，库中包含绘制图形的函数，

函数库引用方法：

方法1：import <库名>

此后可以通过<库名>.<函数名(参数)>的方式中调用库的所有函数

方法2：from <库名> import <函数1，函数2，...>

from <库名> import *

其中*为通配符，表示<库名>的所有函数。

此后不再需要用<库名>.<函数名(参数)>调用函数，直接用<函数名(参数)>调用即可。

2.3 实例2：Python蟒蛇绘制

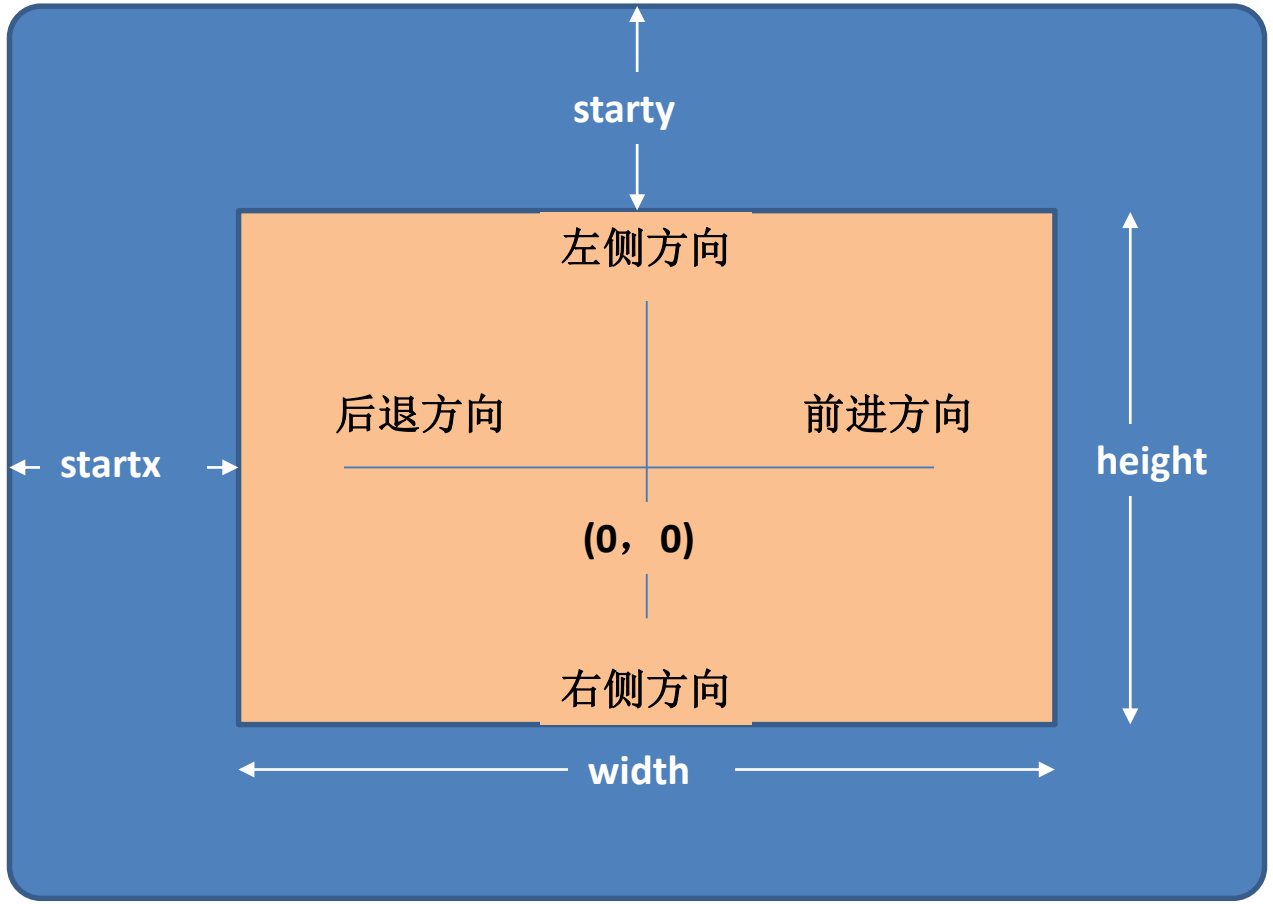
```
1.  #e2.1DrawPython.py
2.  import turtle
3.  turtle.setup(650,350,200,200)
4.  turtle.penup()
5.  turtle.fd(-250)
6.  turtle.pendown()
7.  turtle.pensize(25)
8.  turtle.pencolor("purple")
9.  turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```

```
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
```

是另外一种循环，其中range(4)表示i从0循环到4，每次循环执行circle(40，80)和turtle.circle(-40,80)语句。

2.4 turtle库语法元素分析

```
1. #e2.1DrawPython.py
2. import turtle
3. turtle.setup(650,350,200,200)
4. turtle.penup()
5. turtle.fd(-250)
6. turtle.pendown()
7. turtle.pensize(25)
8. turtle.pencolor("purple")
9. turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```



turtle.setup(width,height,startx,starty)

设置绘制区域大小和位置

2.4 turtle库语法元素分析

```
1. #e2.1DrawPython.py
2. import turtle
3. turtle.setup(650,350,200,200)
4. turtle.penup()
5. turtle.fd(-250)
6. turtle.pendown()
7. turtle.pensize(25)
8. turtle.pencolor("purple")
9. turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```

画笔控制函数

penup()：抬起画笔，别名up()，pu()

pendown()： 落下画笔，别名pd()，down()

pensize()：设置画笔的尺寸， 别名width()

pencolor()： 设置画笔颜色

可以有两种设置方法：

pencolor(colorstring)或pencolor(r, g, b)

部分颜色对照表如下表：

http://blog.csdn.net/alva_bobo/article/details/78087433

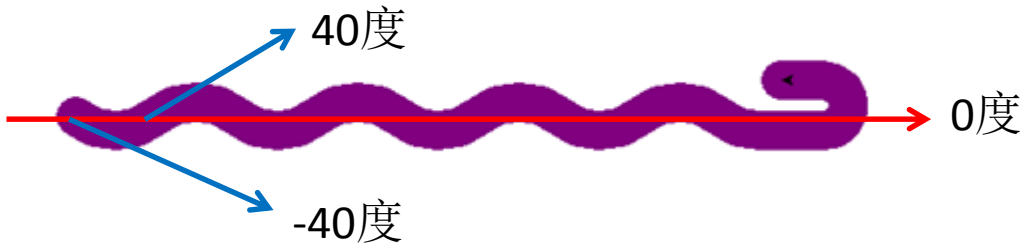
colorstring	R G B	十六进制	中文含义
white	255, 255, 255	#FFFFFF	白色
black	0, 0, 0	#000000	黑色
grey	190, 190, 190	#BEBEBE	灰色
darkgreen	0, 100, 0	#006400	深绿色
gold	255, 215, 0	#FFD700	金色
violet	238, 130, 238	#EE82EE	紫罗兰
purple	160, 32, 240	#A020F0	紫色

2.4 turtle库语法元素分析

```
1.  #e2.1DrawPython.py
2.  import turtle
3.  turtle.setup(650,350,200,200)
4.  turtle.penup()
5.  turtle.fd(-250)
6.  turtle.pendown()
7.  turtle.pensize(25)
8.  turtle.pencolor("purple")
9.  turtle.seth(-40)
10. for i in range(4):
11.     turtle.circle(40,80)
12.     turtle.circle(-40,80)
13. turtle.circle(40,80/2)
14. turtle.fd(40)
15. turtle.circle(16,180)
16. turtle.fd(40*2/3)
```

形状绘制函数

fd(distance)：画笔前进distance距离，别名：forward()
seth(to_angle)：改变绘制方向，别名：setheading()
circle(radius, extent)：绘制弧线，radius为圆弧半径，
　　extent为弧形角度，



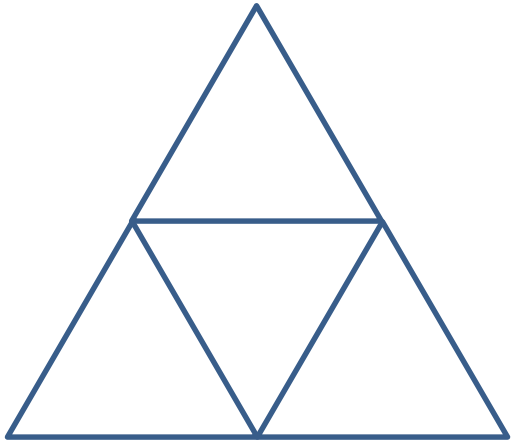
2.4 turtle库语法元素分析

```
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()
```


```
#e2.1DrawPython.py
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for i in range(4):
    turtle.circle(40,80)
    turtle.circle(-40,80)
turtle.circle(40,80/2)
turtle.fd(40)
turtle.circle(16,180)
turtle.fd(40*2/3)
```

实验2-2 实例2的修改。改造实例代码2.1，绘制一条彩色蟒蛇，即在绘制python蟒蛇的每个小段时，画笔的绘制颜色会发生变化。

实验2-3 叠加等边三角形的绘制，使用turtle库中的相关函数绘制一个叠加等边三角形，如图所示。（**注意绘制的起点、终点和三角形大小**）



实验2-4 把绘制一个三角形的功能，编写为一个函数，通过调用完成叠加三角形的功能。