

Python程序语言入门与应用



python

Life is short, use Python
人生苦短，我用Python



课程回顾



Python语言开发和运行环境的配置方法

- Python解释器安装
- Python代码 交互式/**文件式** 运行
- Python IDLE
- Python IDE: VS Code

Python语言的特点

- 语法简洁、跨平台、粘性扩展、开源理念、通用灵活、强制可读、支持中文、模式多样、类库丰富

Python语言版本更迭



上周作业



1.1 字符串拼接

如：[某人]想去[某地]看看。

接受用户输入的2个字符串，将它们组合后输出。

1.2 整数序列求和

用户输入一个正整数N，计算从1到N相加之和。

1.3 九九乘法表输出

工整打印输出常用的九九乘法表，格式不限。

1×1=1									
1×2=2	2×2=4								
1×3=3	2×3=6	3×3=9							
1×4=4	2×4=8	3×4=12	4×4=16						
1×5=5	2×5=10	3×5=15	4×5=20	5×5=25					
1×6=6	2×6=12	3×6=18	4×6=24	5×6=30	6×6=36				
1×7=7	2×7=14	3×7=21	4×7=28	5×7=35	6×7=42	7×7=49			
1×8=8	2×8=16	3×8=24	4×8=32	5×8=40	6×8=48	7×8=56	8×8=64		
1×9=9	2×9=18	3×9=27	4×9=36	5×9=45	6×9=54	7×9=63	8×9=72	9×9=81	

1.4 计算 $1!+2!+3!+\dots+10!$

1.5 五角星的绘制

绘制一个红色的五角星。





新乡医学院

Python程序语言入门与应用

初识Python

第二章 Python程序实例解析



郭长江

changjiangguo@xxmu.edu.cn

生命科学技术学院

新乡医学院





Talk is cheap. Show me the code.

— *Linus Torvalds* —

代码胜于雄辩。

——林纳斯·托瓦兹（Linux之父）




学习目标




基本要求

掌握

-  Python语言的基本语法，包括缩进、变量、命名等

理解

-  Python标准库的导入与使用

了解

-  Python语言turtle库绘制图形的一般方法



本课概要



第2章 Python程序实例解析

2.1 实例1: 温度转换

2.2 Python程序语法元素分析

2.3 实例2: Python蟒蛇绘制

2.4 turtle库语法元素分析



2.1 实例1: 温度转换



2.1 实例1: 温度转换



❖ 温度转换问题

❖ 摄氏度 (Celsius)

结冰点为0度, 沸点100度, 区间100等分确定为1度

❖ 华氏度 (Fahrenheit)

结冰点为32度, 沸点为212度, 区间180等分为1度

❖ 不同国家使用不同的温度表示方法

❖ 中国: 摄氏度 美国: 华氏度

❖ 温度转换需求 (如何通过计算机程序解决?)



2.1 实例1: 温度转换



✚ 计算机程序解决温度转换问题

6个步骤（程序编写基本方法）

✚ 1. 分析问题

温度转换问题计算部分：利用程序进行温度转换，由用户输入温度值，程序计算并给出结果。

✚ 2. 划分边界

程序接受华氏温度和摄氏温度并互相转换，其功能的IPO如下：

✚ 输入：带华氏（F）或摄氏（C）标志的温度值

✚ 处理：根据温度标志选择适当的温度转换算法

✚ 输出：带摄氏（C）或华氏（F）标志的温度值



2.1 实例1: 温度转换



❖ 计算机程序解决温度转换问题

6个步骤（程序编写基本方法）

❖ 3. 设计算法

根据华氏温度和摄氏温度定义，其转换算法如下：

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

$$F = C * 1.8 + 32$$

其中，C表示摄氏温度值，F表示华氏温度值

❖ 4. 编写程序

根据IPO描述和算法设计，编写温度转换的Python程序代码



2.1 实例1: 温度转换



❖ 计算机程序解决温度转换问题

6个步骤 (程序编写基本方法)

❖ 4. 编写程序

#2.1TempConvert.py

```
TempStr = input("请输入带有符号的温度值: ")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```



2.1 实例1: 温度转换



❖ 计算机程序解决温度转换问题

6个步骤（程序编写基本方法）

❖ 5. 调试测试

将上述程序保存为文件：2.1TempConvert.py，使用文件式运行程序：**python** 2.1TempConvert.py

```
Windows PowerShell
PS F:\xxmu\教学\选修课申请\Python程序语言入门与应用\2019课件\材料\源代码\第02章> python .\2.1TempConvert.py
请输入带有符号的温度值：82F
转换后的温度是27.78C
PS F:\xxmu\教学\选修课申请\Python程序语言入门与应用\2019课件\材料\源代码\第02章> python .\2.1TempConvert.py
请输入带有符号的温度值：28C
转换后的温度是82.40F
PS F:\xxmu\教学\选修课申请\Python程序语言入门与应用\2019课件\材料\源代码\第02章> python .\2.1TempConvert.py
请输入带有符号的温度值：-30C
转换后的温度是-22.00F
PS F:\xxmu\教学\选修课申请\Python程序语言入门与应用\2019课件\材料\源代码\第02章> python .\2.1TempConvert.py
请输入带有符号的温度值：-10F
转换后的温度是-23.33C
PS F:\xxmu\教学\选修课申请\Python程序语言入门与应用\2019课件\材料\源代码\第02章> 
```



2.1 实例1: 温度转换



❖ 计算机程序解决温度转换问题

6个步骤 (程序编写基本方法)

❖ 6. 升级维护

❖ 平台更换

❖ 使用方式变化 (图形界面、web)

❖ 算法改进



2.2 Python程序语法元素分析



2.2 Python程序语法元素分析



Python程序语法元素

格式框架

注释

变量

表达式

分支语句

循环语句

函数

#2.1TempConvert.py

```
TempStr = input("请输入带有符号的温度值: ")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```




2.2 Python程序语法元素分析



❧ 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：每一行代码开始前的空白区域，表示代码之间的包含和层次关系。
- ❧ 缩进使用Tab键或者多个空格实现，但不能混用。
- ❧ 采用不合理的代码缩进将抛出SyntaxError异常。
- ❧ 按照**PEP 8**建议使用**4个空格**表示单层缩进。

<https://www.python.org/dev/peps/pep-0008/>



2.2 Python程序语法元素分析



格式框架

单层缩进&多层缩进

缩进表达从属关系（判断、循环、函数、类等）

以关键字开始，以冒号(:)结束

```
#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")
```

(a) 单层缩进

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


(b) 多层缩进



2.2 Python程序语法元素分析




注释

-  注释是代码中用来对语句、函数、数据结构或方法等进行说明的信息，以提升代码的可读性

#2.1TempConvert.py

-  Python语言有2种注释方法

-  单行注释：以 **#** 开头

-  多行注释：以 **'''** (3个单引号) 或 **"""** (3个双引号) 开头和结尾



2.2 Python程序语法元素分析



☞ 注释

```
# 这是单行注释, 单行注释可以独占一行
# 单行注释可以连续使用, 相当于多行注释
print(pow(2,10))    # 计算2的10次幂, 单行注释可以放在行尾
'''
print(pow(2,10))    此行是注释, 不被执行
此行也是注释
此行还是注释^_^
'''
```

- ☞ 非注释语句按顺序执行, 注释语句被解释器过滤掉, 不被执行
- ☞ 注释主要用途: 标明作者和版权信息、解释代码原理和用途、辅助程序调试



2.2 Python程序语法元素分析



命名与保留字

```
TempStr = input("请输入带有符号的温度值：")
```

- Python采用“**变量**”来保存和表示具体的数据值。
- 命名：为变量等程序元素关联一个**标识符**，保证程序元素的唯一性。
- 命名规则：大写字母、小写字母、数字、下划线_和汉字等字符及其组合，但**首字母不能是数字**，且**中间不能出现空格**，长度一般没有限制。如：
 - python_is_good, Python_is_not_good, _is_it_a_question_
- 标识符对**大小写敏感**，python≠Python

<https://namingconvention.org/python/>



2.2 Python程序语法元素分析



命名与保留字

程序元素的标识符不能与Python保留字（关键字：编程语言内部定义并保留使用的标识符）相同。

Python 3.x版本共有33个保留字

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

```
if TempStr[-1] in ['F', 'f']:
elif TempStr[-1] in ['C', 'c']:
else:
```

不建议使用中文等非英语字符进行命名



2.2 Python程序语法元素分析



字符串

- 文本在程序用字符串 (string) 类型来表示。
- Python使用""或者"括起来的若干字符表示字符串。

```
TempStr = input("请输入带有符号的温度值：")
```

- 字符串是字符的序列，可以按照单个字符或者字符片段进行索引。





2.2 Python程序语法元素分析



字符串

TempStr[-1]

TempStr[0:-1]

- 单个字符索引采用[N]格式：如 hw[0], hw[-11]为'H'；
hw[10], hw[-1]为'd'
- 字符片段索引采用[N:M]格式，表示从N到M（不包含M）的子字符串，如hw[0:10], hw[0:-1], hw[-11:10],
hw[:-1]为'Hello Worl'； hw[9:], hw[-2:]为'ld'； hw[:]为
'Hello World'； hw[::-1]为'dlroW olleH' “切片”





2.2 Python程序语法元素分析



字符串

```
>>> a = 'abc'
>>> a[3]
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    a[3]
IndexError: string index out of range
```

索引超出范围 (IndexError)



2.2 Python程序语法元素分析



赋值语句

```
TempStr = input("请输入带有符号的温度值：")  
C = (eval(TempStr[0:-1]) - 32) / 1.8  
F = 1.8 * eval(TempStr[0:-1]) + 32
```

- 表达式：产生或者计算新数据值的代码（运算）
- 运算过程和结果由操作符或运算符所决定
- 赋值运算符： =
- 赋值语句：将等号右侧的计算结果赋给左侧变量
- 同步赋值语句：多个变量同时赋值
 - <变量1>, ..., <变量N> = <表达式1>, ..., <表达式N>
 - 应用：互换变量值 ($t = x, x = y, y = t \rightarrow x, y = y, x$)




2.2 Python程序语法元素分析



input()函数

```
TempStr = input("请输入带有符号的温度值：")
```

 input()函数从控制台获得用户输入，且都以字符串类型返回结果。

用法

<变量> = input(<提示性文字>)

```
>>> input("请输入：")
请输入：python
'python'
>>> input("请输入：")
请输入：3.1415
'3.1415'
```



2.2 Python程序语法元素分析



分支语句

```
if TempStr[-1] in ['F', 'f']:
elif TempStr[-1] in ['C', 'c']:
else:
```

根据判断条件选择程序执行路径（程序控制）

使用方式：if-elif-else

条件表达式

产生布尔值（真True/假False）

判断TempStr最后一个字符串在由'F'和'f'组成的集合中。如果在返回True，不在则返回False。

```
if TempStr[-1] in ['F', 'f']
```

```
if <条件1>:
    <语句块1>
elif <条件2>:
    <语句块3>
else:
    <语句块3>
```




2.2 Python程序语法元素分析



eval()函数

```
C = (eval(TempStr[0:-1]) - 32)/1.8  
F = 1.8*eval(TempStr[0:-1]) + 32
```

 以Python表达式的方式解析并执行字符串，即：将字符串部分 (TempStr[0:-1]) 转换为数字再进行运算。

```
C = (float(TempStr[0:-1]) - 32)/1.8  
F = 1.8*float(TempStr[0:-1]) + 32
```

```
>>> x = 1  
>>> eval("x + 1")  
2  
>>> eval("1.1 + 2.2")  
3.3000000000000003  
>>> TempStr = "102C"  
>>> eval(TempStr[:-1])  
102  
>>> value = eval(input("请输入数值: "))  
请输入数值: 3.1415  
>>> print(value*2)  
6.283
```

注意：此处x和hello作为变量名使用。

```
>>> eval("hello")  
Traceback (most recent call last):  
  File "<pyshell#15>", line 1, in <module>  
    eval("hello")  
  File "<string>", line 1, in <module>  
NameError: name 'hello' is not defined  
>>> eval("'hello'")  
'hello'
```



2.2 Python程序语法元素分析



print()函数

- 输出字符串信息和以字符串形式输出变量
- 输出变量值时需要采用格式化输出方式
 - 格式化符号%
 - 槽格式 ({}) 和format()方法
 - 格式化字符串常量f-string
- {:.2f}表示C或F的输出格式，具体表示输出数值保留2位小数。

```
print("转换后的温度是{:.2f}C".format(C))  
print("转换后的温度是{:.2f}F".format(F))  
print("输入格式错误")
```



2.2 Python程序语法元素分析



🔗 循环语句

- 🔗 根据判断条件确定一段程序是否再次执行一次或多次
(程序控制)
- 🔗 温度转换程序连续运行并接受用户输入，只有当用户输入的最后一个字符串为"N"或"n"时才退出程序。

```
#2.1TempConvert.v2.py
TempStr = input("请输入带有符号的温度值：")
while TempStr[-1] not in ["N", "n"]:
    if TempStr[-1] in ['F', 'f']:
        C = (eval(TempStr[0:-1]) - 32)/1.8
        print("转换后的温度是{:.2f}C".format(C))
    elif TempStr[-1] in ['C', 'c']:
        F = 1.8*eval(TempStr[0:-1]) + 32
        print("转换后的温度是{:.2f}F".format(F))
    else:
        print("输入格式错误")
    TempStr = input("请输入带有符号的温度值：")
```

```
请输入带有符号的温度值：62C
转换后的温度是143.60F
请输入带有符号的温度值：62F
转换后的温度是16.67C
请输入带有符号的温度值：30F
转换后的温度是-1.11C
请输入带有符号的温度值：100C
转换后的温度是212.00F
请输入带有符号的温度值：N
```



2.2 Python程序语法元素分析



❧ 函数

- ❧ 目前的程序代码由一个序列表达式组成，从头到尾依次顺序执行。
- ❧ 实际编程中，将**特定功能代码**写在一个函数中，便于阅读和代码复用，有利于**程序模块化**。
- ❧ 函数是对一组表达特定功能表达式的封装，可接受变量并**输出或返回结果**。
- ❧ `input()`、`print()`、`eval()`都是Python解释器内置函数。



2.2 Python程序语法元素分析



函数

温度转换程序的函数化改造

```
#2.1TempConvert.v3.py
def tempConvert(TempStr):
    # def保留字定义名为tempConvert()的函数, 参数为TempStr
    if TempStr[-1] in ['F', 'f']:
        C = (eval(TempStr[0:-1]) - 32)/1.8
        print("转换后的温度是{:.2f}C".format(C))
    elif TempStr[-1] in ['C', 'c']:
        F = 1.8*eval(TempStr[0:-1]) + 32
        print("转换后的温度是{:.2f}F".format(F))
    else:
        print("输入格式错误")

TempStr = input("请输入带有符号的温度值: ")
while TempStr[-1] not in ["N", "n"]:
    tempConvert(TempStr)
    TempStr = input("请输入带有符号的温度值: ")
```



2.3 实例2: Python蟒蛇绘制



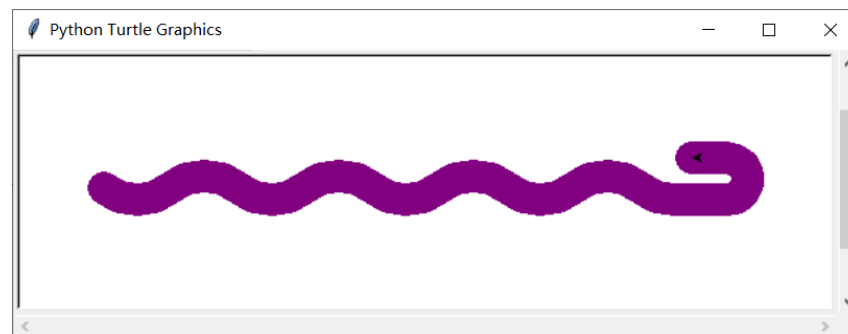
2.3 实例2: Python蟒蛇绘制



turtle库绘制python

```
#2.2DrawPython.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)
turtle.done()
```

Python
模块化编程
思想





2.3 实例2: Python蟒蛇绘制



❏ 代码特点

```
turtle.penup()  
turtle.fd(-250)
```

- ❏ 没有显式的用户输入输出
- ❏ 以 `<a>.()` 为主要代码形式，无赋值语句
 - ❏ `<a>.()` 是Python编程的一种典型表达形式，表示调用一个**对象** `<a>` 的 `()` 方法，也表示一个函数库 `<a>` 中的函数 `()`。
 - ❏ Python中**一切皆对象**，对象有**属性**和**方法**。
 - ❏ 属性是对象中的变量，方法是对象能够完成的操作。
 - ❏ `Car.color` 汽车的颜色； `Car.forward()` 汽车前进
 - ❏ 面向对象编程OOP（模块编程）



2.3 实例2: Python蟒蛇绘制



❏ turtle标准库(模块)的导入

```
import turtle
```

❏ import引入(函数)库的方法

❏ `import <库名>` 调用函数库中的所有函数, 使用格式:

`<库名>.<函数名>(<函数参数>)`

```
turtle.penup()
```

❏ `from <库名> import <函数名, 函数名, ...>` 调用函数库中的指定函数, 使用格式:

`<函数名>(<函数参数>)`

```
penup()
```

❏ `from <库名> import *` 调用函数库的所有函数, 使用格式:

`<函数名>(<函数参数>)`

```
penup()
```



2.3 实例2: Python蟒蛇绘制



库引用方式比较

#2.2DrawPython.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)
turtle.done()
```

表明函数来源, 可读性高

#2.2DrawPython.v2.py

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

代码更简洁



2.4 turtle库语法元素分析



2.4 turtle库语法元素分析

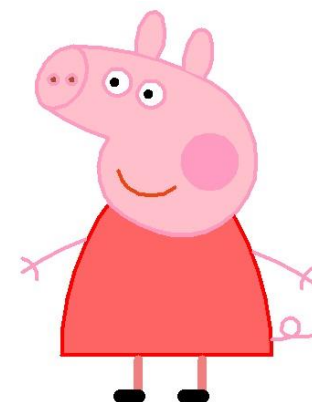
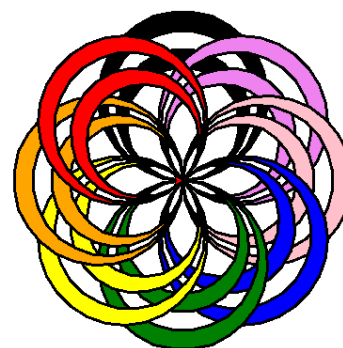
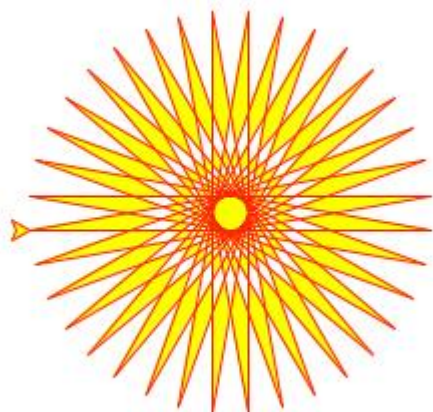


🐢 turtle (海龟) 库

🔗 一个直观而有趣的入门级图形绘制函数库

🔗 Python标准库之一

🔗 说明文档: <https://docs.python.org/zh-cn/3/library/turtle.html>




turtle的魅力



2.4 turtle库语法元素分析



 turtle (海龟) 库



turtle的魅力



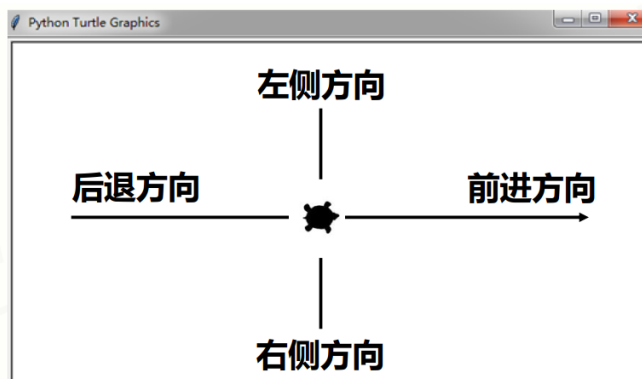
2.4 turtle库语法元素分析



绘图坐标体系

基本绘图框架

- 一个海龟在坐标系中爬行，其轨迹形成绘制图形
- 海龟由程序控制，可以变换颜色、改变宽度等
- 海龟有“前进”、“后退”、“旋转”等爬行行为
- 开始绘制时海龟位于画布中央，坐标为(0,0)，行进方向为水平向右





2.4 turtle库语法元素分析



绘图坐标体系

```
turtle.setup(650, 350, 200, 200)
```

设置函数 `turtle.setup()`

使用方法：

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

作用：设置主窗体的大小和位置

width：窗口宽度； **height**：窗口高度 整数为像素值，小数为窗口宽/高度与屏幕比例

startx：窗口左侧与屏幕左侧的像素距离，None为屏幕水平中央

starty：窗口顶部与屏幕顶部的像素距离，None为垂直水平中央

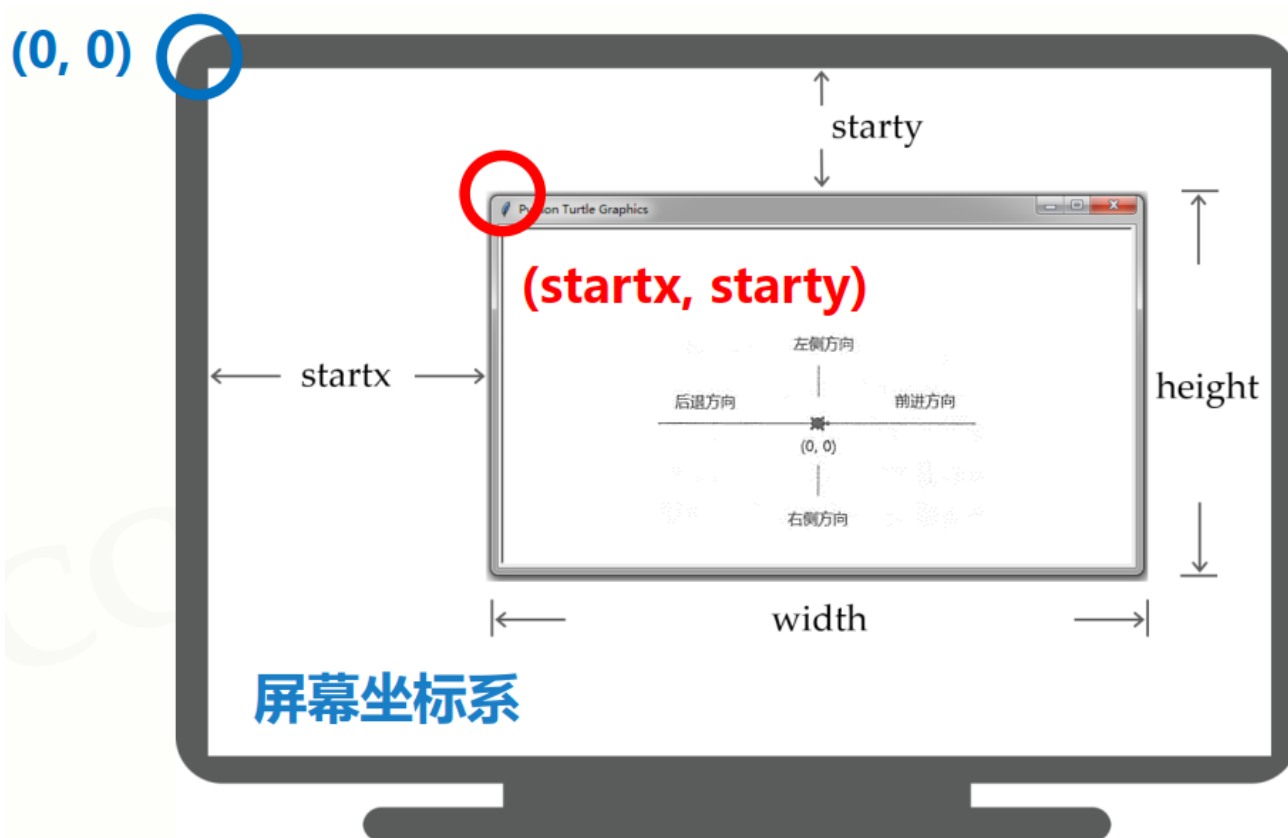


2.4 turtle库语法元素分析



绘图坐标体系

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





2.4 turtle库语法元素分析



画笔控制函数

turtle.penup()

作用：抬起画笔，之后移动画笔不绘制形状

参数：无

turtle.pendown()

作用：落下画笔，之后移动画笔将绘制形状

参数：无

turtle.pensize(width)

作用：设置画笔宽度，无参数时返回当前画笔宽度

参数：**width**画笔线条宽度（像素数）



2.4 turtle库语法元素分析

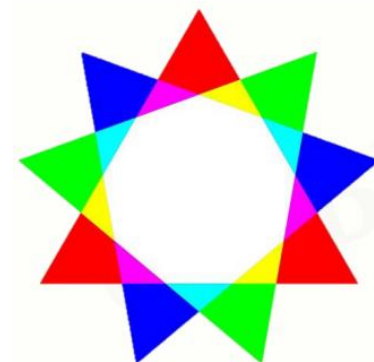


画笔控制函数

`turtle.pencolor("purple")`

作用：设置画笔颜色，无参数时返回当前画笔颜色

参数：colorstring 或 (r,g,b)



RGB色彩模式

英文名称	RGB整数值	RGB小数值	中文名称
white	255, 255, 255	1, 1, 1	白色
yellow	255, 255, 0	1, 1, 0	黄色
magenta	255, 0, 255	1, 0, 1	洋红
cyan	0, 255, 255	0, 1, 1	青色
blue	0, 0, 255	0, 0, 1	蓝色
black	0, 0, 0	0, 0, 0	黑色



2.4 turtle库语法元素分析



形状绘制函数

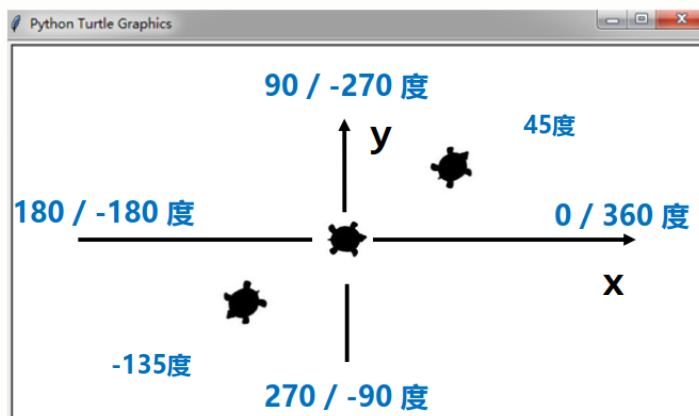
```
turtle.fd(-250)  
turtle.seth(-40)
```

turtle.fd(distance)

- 作用：向当前行进方向前进distance距离
- 参数：distance行进距离的像素值（负值向相反方向前进）

turtle.seth(to_angle)

- 作用：设置当前行进方向为to_angle（绝对方向角度值）



```
turtle.right(to_angle)  
turtle.left(to_angle)
```

相对方向角度值



2.4 turtle库语法元素分析

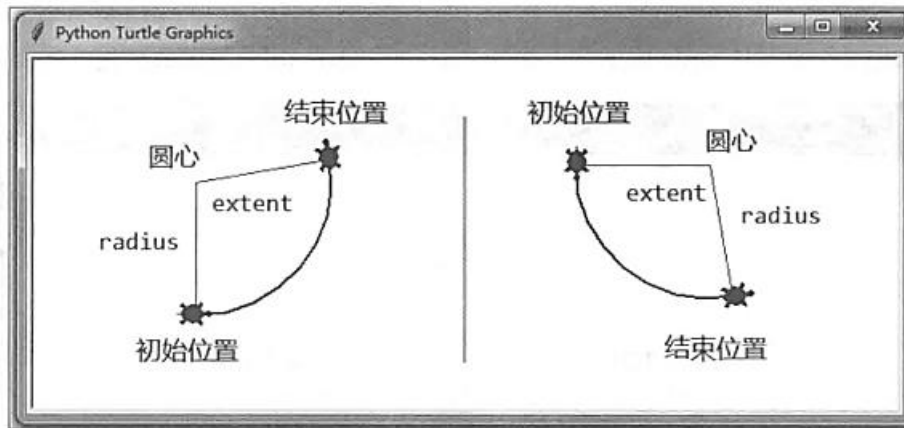


形状绘制函数

```
turtle.circle(40, 80)  
turtle.circle(-40, 80)
```

`turtle.circle(radius, extend=None)`

- 作用：根据半径`radius`绘制`extend`角度的弧形
- 参数1：`radius`弧形半径，正数表示半径在小海龟左侧，负数表示半径在小海龟右侧
- 参数2：`extend`弧形角度，不设或者为None时绘制整个圆形





2.4 turtle库语法元素分析



❏ for循环语句

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

❏ 遍历循环

❏ 格式:

```
for i in range(<循环次数>):  
    <语句块1>
```

❏ range(start, stop[, step])

❏ 从 **start** 开始，默认从 0 开始。如range(5)等价于range(0,5)

❏ 到 **stop** 结束，但不包括 stop。如：range(5)是[0, 1, 2, 3, 4] 没有5

❏ 步长为**step**，默认为1。如：range(0,5)等价于 range(0, 5, 1)

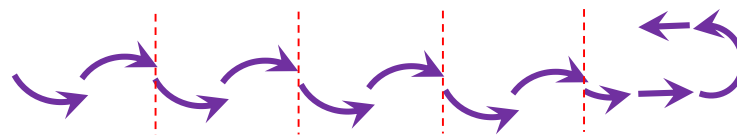


2.4 turtle库语法元素分析



Python蟒蛇绘制代码

```
#2.2DrawPython.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)
turtle.done()
```





2.4 turtle库语法元素分析



Python蟒蛇绘制代码函数封装改造

```
def FuncName(<参数>):
```

```
#2.2DrawPython.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)
turtle.done()
```

```
#2.2DrawPython.v3.py
```

```
import turtle
def drawSnake(radius, angle, length):
    turtle.seth(-angle/2)
    for i in range(length):
        turtle.circle(radius, angle)
        turtle.circle(-radius, angle)
    turtle.circle(radius, angle/2)
    turtle.fd(radius)
    turtle.circle(radius*0.4, 180)
    turtle.fd(radius * 2/3)
    turtle.done()
```

```
turtle.setup(650, 350, 200, 200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
drawSnake(40, 80, 4)
```



本章要点



- ✚ Python语言的基本语法
 - ✚ 格式框架
 - ✚ 注释
 - ✚ 变量
 - ✚ 表达式
 - ✚ 分支语句
 - ✚ 循环语句
 - ✚ 函数
- ✚ Python标准库的导入与使用
 - ✚ import
 - ✚ 模块编程



程序练习题



2.1 汇率兑换程序

按照1美元=7人民币的汇率编写美元与人民币双向兑换程序。

2.2 绘制彩色蟒蛇

蟒蛇每一小段使用不同绘制颜色。

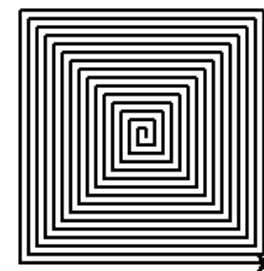


2.3 绘制等边三角形

使用turtle库中turtle.fd()与turtle.seth()绘制等边三角形。

2.4 绘制正方形螺旋线

利用turtle库绘制一个正方形螺旋线。



.py代码文件打包(2.学号+姓名)发送到
python_xxmu@163.com



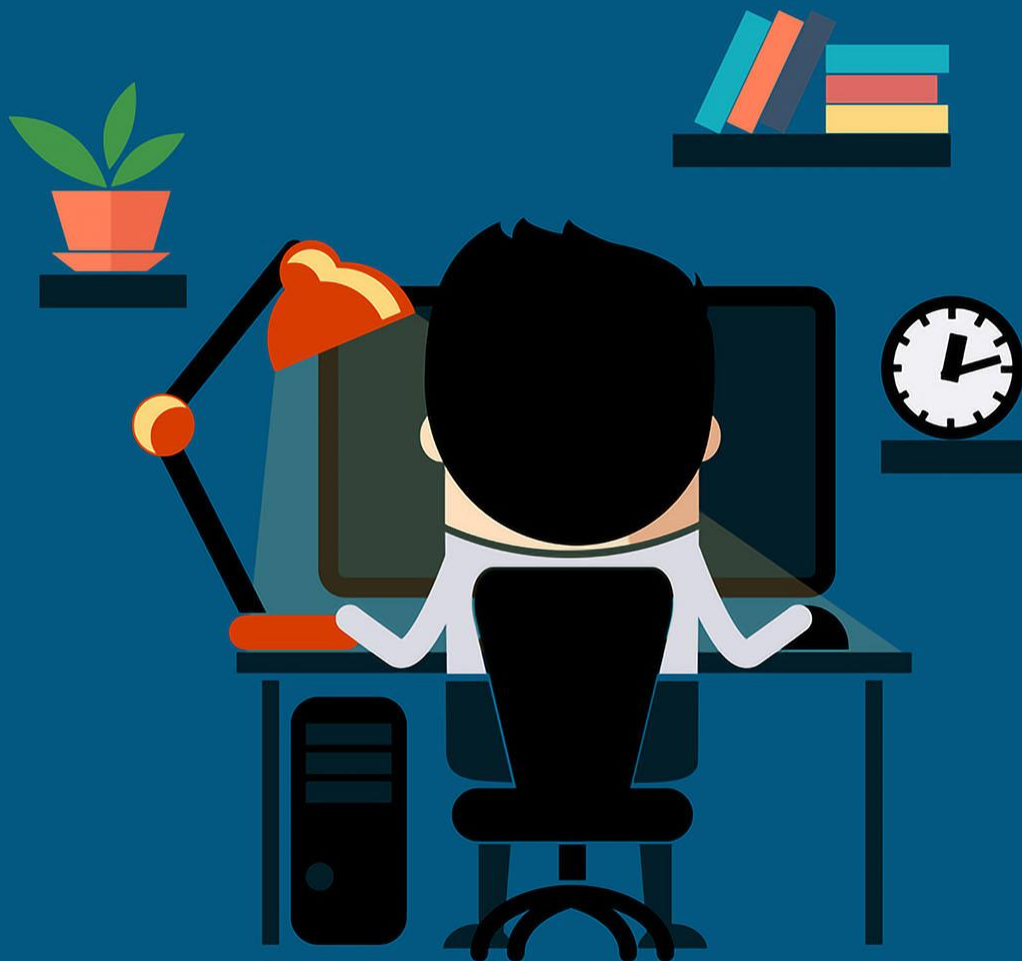
下周课程



第3章 Python基本数据类型

- 数字类型及其操作
- math库
- 实例1：天天向上的力量
- 字符串类型及其操作
- 字符串类型的格式化
- 实例2：文本进度条

编程辣么好，还等什么？开始学习吧！



Programing is an Art