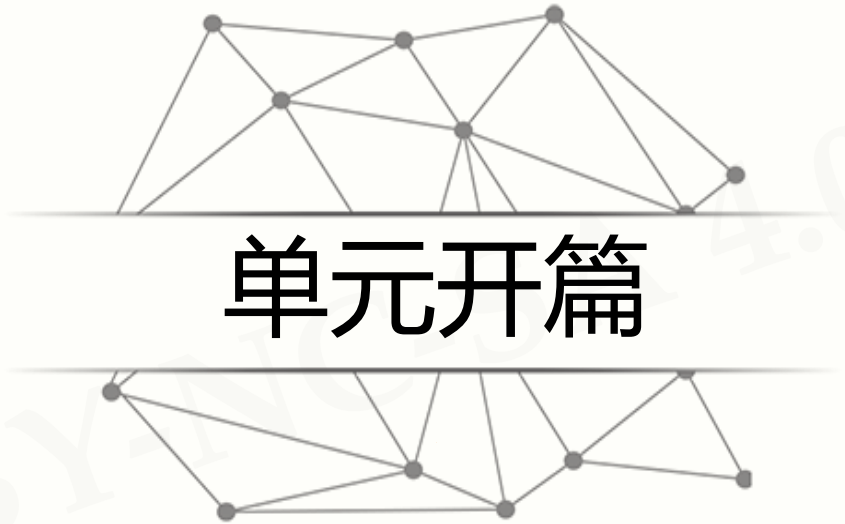

python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for 1 in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40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

Python语言程序设计

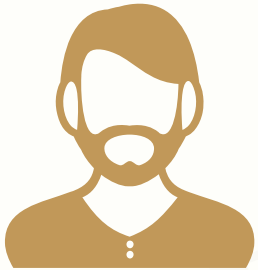
Python程序语法元素分析

嵩 天
北京理工大学

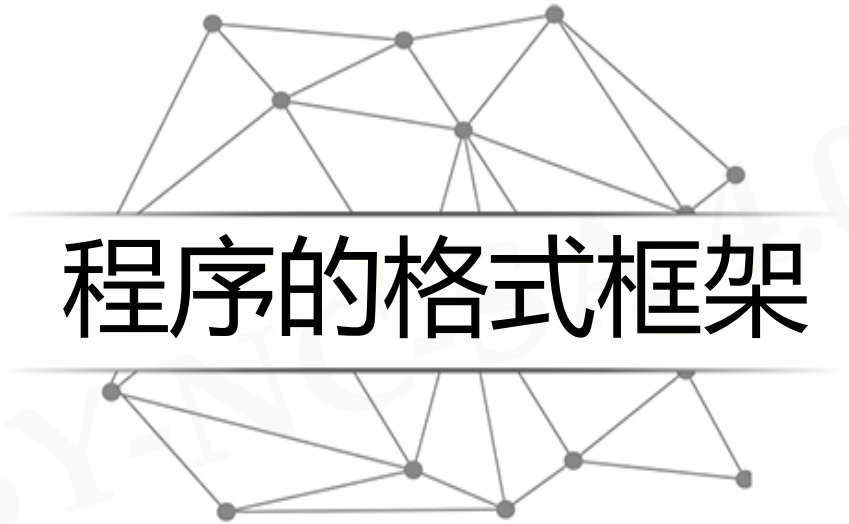


单元开篇

Python程序语法元素分析



- 程序的格式框架
- 命名与保留字
- 数据类型
- 语句与函数
- Python程序的输入输出
- "温度转换"代码分析



程序的格式框架

```
#TempConvert.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")
```

代码高亮：编程的色彩辅助体系，不是语法要求

```python
#TempConvert.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")
```

缩进：一行代码开始前的空白区域，表达程序的格式框架

```
#TempConvert.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")
```

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**2)
    if dist <= 1.0:
        hits = hits + 1
pi = 4 * (hits/DARTS)
print("Pi的值是 {:.2f}F".format(pi))
```

多层缩进

缩进

缩进表达程序的格式框架

- 严格明确：缩进是语法的一部分，缩进不正确程序运行错误
- 所属关系：表达代码间包含和层次关系的唯一手段
- 长度一致：程序内一致即可，一般用4个空格或1个TAB

```
#TempConvert.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")
```

注释：用于提高代码可读性的辅助性文字，不被执行

注释

不被程序执行的辅助性说明信息

- **单行注释：以#开头，其后内容为注释**

```
# 这里是单行注释
```

- **多行注释：以'''开头和结尾**

```
'''     这是多行注释第一行
        这是多行注释第二行     '''
```

```
#TempConvert.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")
```

缩进 注释

命名与保留字

```
#TempConvert.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")
```

变量：程序中用于保存和表示数据的占位符号

变量

用来保存和表示数据的占位符号

- **变量采用标识符(名字) 来表示，关联标识符的过程叫命名**

```
TempStr是变量名字
```

- **可以使用等号(=)向变量赋值或修改值，=被称为赋值符号**

```
TempStr = "82F"   #向变量TempStr赋值"82F"
```

命名

关联标识符的过程

- **命名规则: 大小写字母、数字、下划线和中文等字符及组合**

  如: `TempStr`，`Python_Great`，这是门Python好课

- **注意事项: 大小写敏感、首字符不能是数字、不与保留字相同**

  `Python`和`python`是不同变量，`123Python`是不合法的

保留字

被编程语言内部定义并保留使用的标识符

- **Python语言有35个保留字(也叫关键字)**

  if, *elif*, *else*, *in*

- **保留字是编程语言的基本单词，大小写敏感**

  if **是保留字，** `If` **是变量**

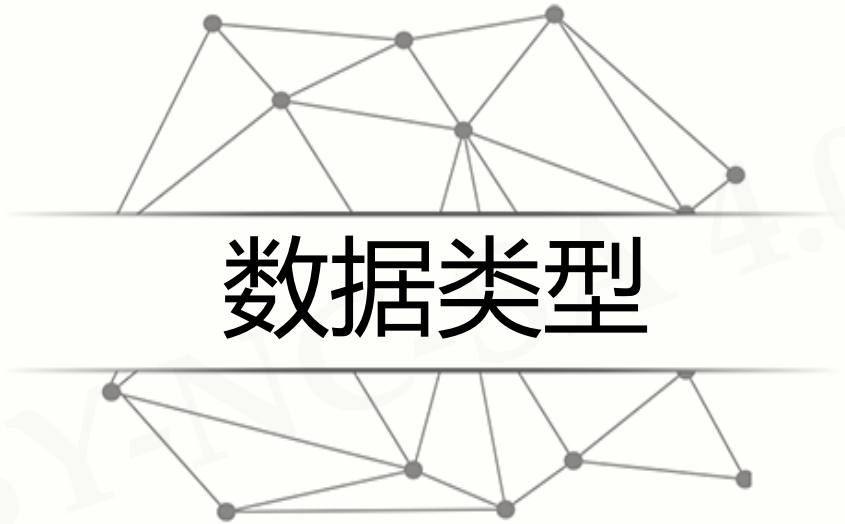
保留字

(26/35)

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

```
#TempConvert.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")
```

变量 命名 保留字



数据类型

```python
#TempConvert.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")
```

数据类型：字符串、整数、浮点数、列表

数据类型

10,011,101 该如何解释呢?

- **这是一个二进制数字 或者 十进制数字**

  作为二进制数字，10,011,101的值是十进制157

- **这是一段文本 或者 用逗号,分隔的3个数字**

  作为一段文本，逗号是文本中的一部分，一共包含10个字符

数据类型

供计算机程序理解的数据形式

- 程序设计语言不允许存在语法歧义，需要定义数据的形式

  需要给10,011,101关联一种计算机可以理解的形式

- 程序设计语言通过一定方式向计算机表达数据的形式

  "123"表示文本字符串123，123则表示数字123

数据类型

10,011,101

- **整数类型：** `10011101`
- **字符串类型：** `"10,011,101"`
- **列表类型：** `[10, 011, 101]`

```
#TempConvert.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")
```

字符串：由0个或多个字符组成的有序字符序列

字符串

由0个或多个字符组成的有序字符序列

- **字符串由一对单引号或一对双引号表示**

  "请输入带有符号的温度值: "**或者** 'C'

- **字符串是字符的有序序列，可以对其中的字符进行索引**

  "请" **是** "请输入带有符号的温度值: " **的第0个字符**

字符串的序号

正向递增序号 和 反向递减序号

字符串的使用

使用[ ]获取字符串中一个或多个字符

- **索引：返回字符串中单个字符　　<字符串>[M]**

  "请输入带有符号的温度值: "[0]　**或者**　TempStr[-1]

- **切片：返回字符串中一段字符子串　　<字符串>[M: N]**

  "请输入带有符号的温度值: "[1:3] **或者**　TempStr[0:-1]

```
#TempConvert.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")
```

数字类型：整数和浮点数

数字类型

整数和浮点数都是数字类型

- **整数：数学中的整数**

  32　或者　-89

- **浮点数：数学中的实数，带有小数部分**

  1.8　或者　-1.8　或者　-1.0

```
#TempConvert.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")
```

列表类型：由0个或多个数据组成的有序序列

列表类型

由0个或多个数据组成的有序序列

- **列表使用[ ]表示，采用逗号(,)分隔各元素**

  ['F','f']表示两个元素'F'和'f'

- **使用保留字 in 判断一个元素是否在列表中**

  TempStr[-1] *in* ['C','c']判断前者是否与列表中某个元素相同

```python
#TempConvert.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")
```

字符串 整数 浮点数 列表

语句与函数

```
#TempConvert.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")
```

赋值语句：由赋值符号构成的一行代码

赋值语句

由赋值符号构成的一行代码

- 赋值语句用来给变量赋予新的数据值

```
C=(eval(TempStr[0:-1])-32)/1.8 #右侧运算结果赋给变量C
```

- 赋值语句右侧的数据类型同时作用于变量

```
TempStr=input("") #input()返回一个字符串，TempStr也是字符串
```

```
#TempConvert.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")
```

分支语句：由判断条件决定程序运行方向的语句

分支语句

由判断条件决定程序运行方向的语句

- **使用保留字*if elif else*构成条件判断的分支结构**

```
if TempStr[-1] in ['F','f']:#如果条件为True则执行冒号后语句
```

- **每个保留字所在行最后存在一个冒号(:)，语法的一部分**

  冒号及后续缩进用来表示后续语句与条件的所属关系

```
#TempConvert.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")
```

函数：根据输入参数产生不同输出的功能过程

函数

根据输入参数产生不同输出的功能过程

- **类似数学中的函数， y = f(x)**

```
print("输入格式错误") #打印输出 "输入格式错误"
```

- **函数采用 <函数名>(<参数>) 方式使用**

```
eval(TempStr[0:-1]) # TempStr[0:-1]是参数
```

```
#TempConvert.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")
```

赋值语句 分支语句 函数

Python程序的输入输出

```
#TempConvert.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")
```

input()：从控制台获得用户输入的函数

输入函数 input()

从控制台获得用户输入的函数

- **input()函数的使用格式：**

```
<变量> = input(<提示信息字符串>)
```

- **用户输入的信息以字符串类型保存在<变量>中**

```
TempStr = input("请输入") # TempStr保存用户输入的信息
```

```
#TempConvert.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")
```

print()：以字符形式向控制台输出结果的函数

输出函数 print()

以字符形式向控制台输出结果的函数

- **print()函数的基本使用格式：**

```
print(<拟输出字符串或字符串变量>)
```

- **字符串类型的一对引号仅在程序内部使用，输出无引号**

```
print("输入格式错误")  # 向控制台输出 输入格式错误
```

输出函数 print()

以字符形式向控制台输出结果的函数

- **print()函数的格式化：**

```
print("转换后的温度是{:.2f}C".format(C))
```

{ }表示槽，后续变量填充到槽中

{ :.2f }表示将变量C填充到这个位置时取小数点后2位

输出函数 print()

以字符形式向控制台输出结果的函数

```
print("转换后的温度是{:.2f}C".format(C))
```

如果C的值是 123.456789，则输出结果为：

转换后的温度是123.45C

```
#TempConvert.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")
```

eval()：去掉参数最外侧引号并执行余下语句的函数

评估函数 eval()

去掉参数最外侧引号并执行余下语句的函数

- **eval()函数的基本使用格式：**

eval(<字符串或字符串变量>)

```
>>> eval("1")
1
>>> eval("1+2")
3
```

```
>>> eval('"1+2"')
'1+2'
>>> eval('print("Hello")')
Hello
```

评估函数 eval()

去掉参数最外侧引号并执行余下语句的函数

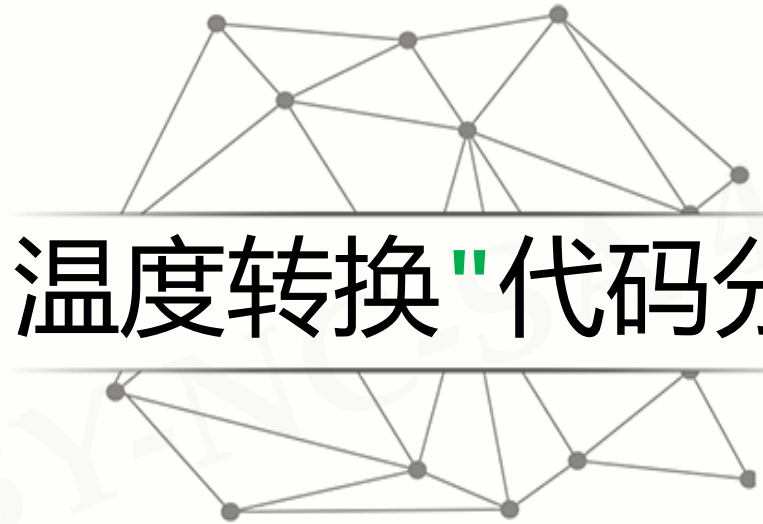
```
eval(TempStr[0:-1])
```

如果TempStr[0:-1]值是"12.3"，输出是：

```
12.3
```

```
#TempConvert.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")
```

input()　print()　eval()



"温度转换"代码分析

```
#TempConvert.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")
```

"温度转换"实例代码逐行分析

单元小结

Python程序语法元素分析

- 缩进、注释、命名、变量、保留字
- 数据类型、字符串、 整数、浮点数、列表
- 赋值语句、分支语句、函数
- input()、print()、eval()、 print()格式化

小花絮

版权说明

- 本课程所有教学资料(课件)受CC BY-NC-SA 4.0知识产权协议保护
  - 未经授权不能商业使用、不能修改使用、不能格式转换
  - 非商业使用时(如教学)必须以恰当且明确方式声明原作者信息
  - 进行使用和传播时必须保持原有形式和内容
- 保护知识产权、分享发展成果、建立良性创新机制