



## 2.3 语法元素分析





## 2.2 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





# 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")
```



# 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")
```



# 程序的格式框架

```
#TempConvert.  
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，缩进不正确程序将运行错误
- **所属关系** 表达代码间包含和层次关系的唯一手段
- **长度一直** 程序内一致即可，一般用4个空格和1个TAB





# 注释

## #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

## 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

'''这是注释的第一行

这是注释的最后一行'''







# 变量

```
#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 是变量的名字

- 可以使用（=）向变量赋值或修改值，=被称为是赋值符号

TempStr="82F"#向变量TempStr赋值"82F"





# 命名

## 给变量关联标识符的过程

- 标识符规则：大小写字母、数字、下划线和汉字等字符及组合  
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、不能与保留字相同  
Python与python是不同的变量，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33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





# 保留字

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	
def	if	pass	del	

(26/33)



```
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

```
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

## 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157

- 这是一段文本或者用逗号分开的3个数字







# 数据类型

## 供计算机理解的数据形式

- 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的方式向计算机表达数据的形式

“123”表示文本字符串123, 123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]





# 数据类型

```
#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

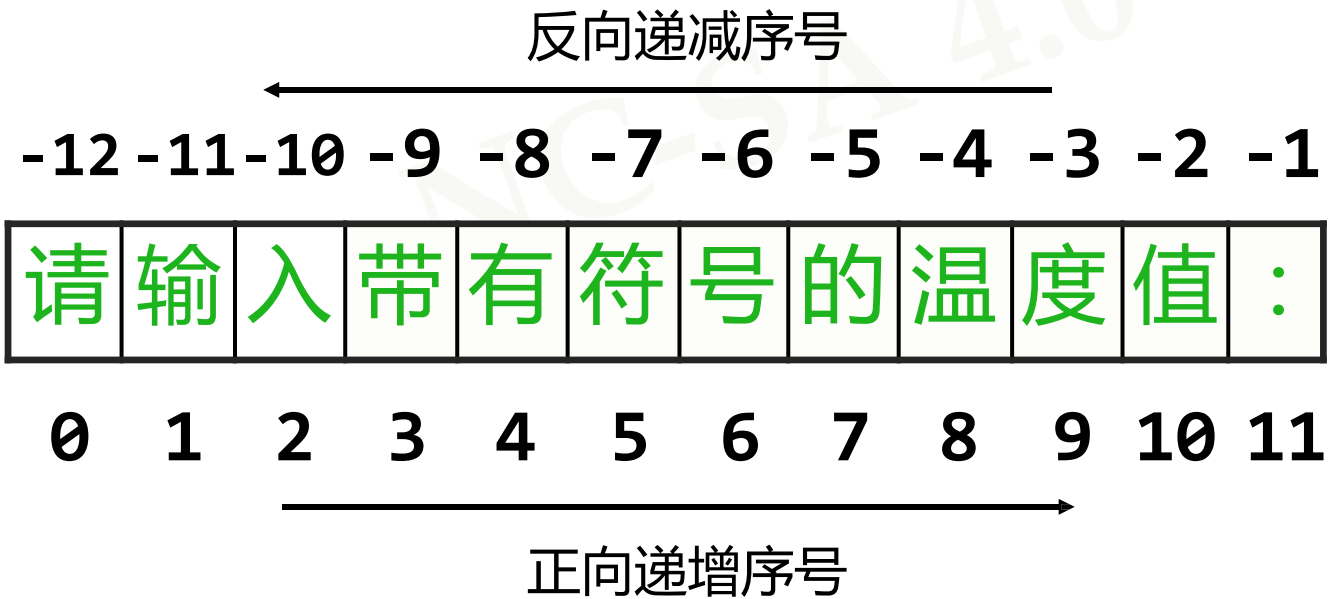
“请”是 "请输入带有符号的温度值:" 的第0个字符





# 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串     <字符串>[M: N]

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





# 数据类型

```
#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







# 数据类型

```
#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





# 数据类型

```
#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")
```



# 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** 由赋值符号构成的单行代码



# 赋值语句

## 由赋值符号构成的单行代码

- 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





# 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")
```

**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





# 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")
```





# 函数

根据输入参数产生不同输出功能的过程

- 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





# 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

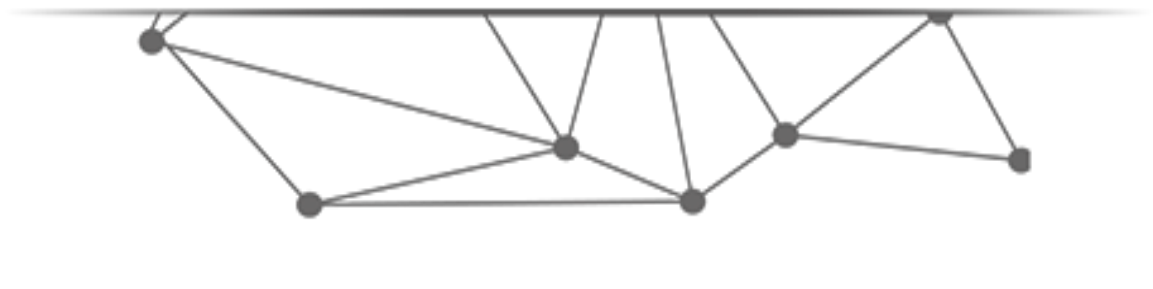
print("转换后的温度是{:.2f}F".format(F))

*else*:

print("输入格式错误")

A geometric diagram consisting of a horizontal line with several points above it. Lines connect these points to form a series of triangles and quadrilaterals, creating a complex, interconnected shape.

# Python程序的输入与输出

A geometric diagram consisting of a horizontal line with several points below it. Lines connect these points to form a series of triangles and quadrilaterals, creating a complex, interconnected shape.



# Python程序的输入与输出

```
#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")
```



# 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
```





# Python程序的输入与输出

```
#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")
```

`print()` 以字符串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





# Print函数用法总结

<https://www.runoob.com/w3cnote/python3-print-func-b.html>

RUNOOB.COM

搜索.....

首页 笔记首页 ANDROID ES6 教程 排序算法 程序员人生 程序员笑话 编程技术 登录 云服务器

## Python3 print 函数用法总结

分类 编程技术

### 1. 输出字符串和数字

```
>>>print("runoob") # 输出字符串
runoob
>>> print(100)      # 输出数字
100
>>> str = 'runoob'
>>> print(str)      # 输出变量
runoob
>>> L = [1,2,'a']    # 列表
>>> print(L)
[1, 2, 'a']
>>> t = (1,2,'a')    # 元组
>>> print(t)
(1, 2, 'a')
>>> d = {'a':1, 'b':2} # 字典
>>> print(d)
{'a': 1, 'b': 2}
```

### 2. 格式化输出整数

支持参数格式化，与 C 语言的 printf 类似

### 教程列表

ADO 教程	Ajax 教程	Android 教程
Angular2 教程	AngularJS 教程	AppML 教程
ASP 教程	ASP.NET 教程	Bootstrap 教程
Bootstrap4 教程	C 教程	C# 教程
C++ 教程	CSS 参考手册	CSS 教程
CSS3 教程	Django 教程	Docker 教程
DTD 教程	ECharts 教程	Eclipse 教程
Firebug 教程	Font Awesome 图	Foundation 教
Git 教程	Go 语言教程	Google 地图 AP
Highcharts 教程	HTML DOM 教程	HTML 参考手
HTML 字符集	HTML 教程	HTTP 教程
ionic 教程	iOS 教程	Java 教程
JavaScript 参考手	Javascript 教程	jQuery EasyUI
jQuery Mobile 教程	jQuery UI 教程	jQuery 教程
JSON 教程	JSP 教程	Kotlin 教程
Linux 教程	Lua 教程	Markdown 教
Maven 教程	Memcached 教程	MonnockR 教



python





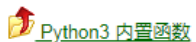
# input函数用法总结

<https://www.runoob.com/python3/python3-func-input.html>

← Python3 日期和时间

Python MongoDB →

## Python3 input() 函数



Python3.x 中 input() 函数接受一个标准输入数据，返回为 string 类型。

注意：在 Python3.x 中 raw\_input() 和 input() 进行了整合，去除了 raw\_input()，仅保留了 input() 函数，其接收任意任性输入，将所有输入默认为字符串处理，并返回字符串类型。

### 函数语法

```
input([prompt])
```

参数说明：

- prompt: 提示信息

### 实例

input() 需要输入 python 表达式

```
>>>a = input("input:")
input:123          # 输入整数
>>> type(a)
<class 'str'>      # 字符串
>>> a = input("input:")
input:runoob       # 正确，字符串表达式
>>> type(a)
<class 'str'>      # 字符串
```

更多内容





# 评估函数eval ()

```
#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





# 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")
```

**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")
```