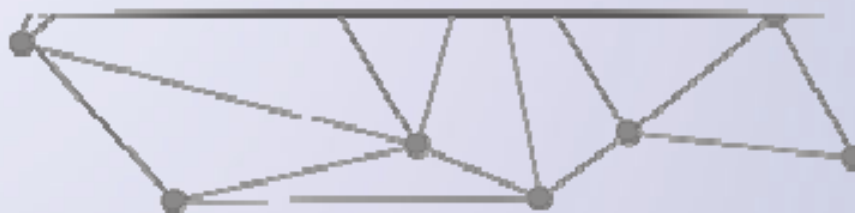




Python语法元素分析



嵩 天

北京理工大学



程序元素

注释

缩进

变量

常量

表达式

输入

输出

分支

循环

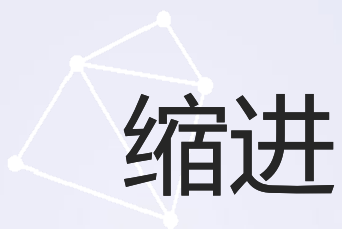




温度转换程序

```
#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")
```



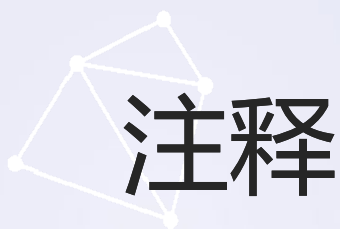


缩进

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





- 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

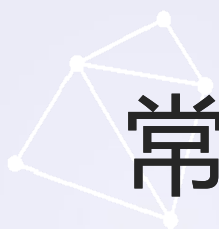
- #Here are the comments

- 多行注释以 ' ' ' 开头和结尾

- ' ' '

- This is a multiline comment
used in Python





常量与变量

- 常量：程序中值不发生改变的元素

- 使用常量的好处：

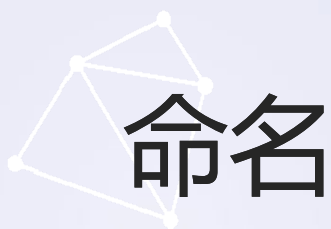
- 例如：程序中含有一个常量， $\pi=3.14$

- 如果程序中多次使用 π ，当我们需要更精确的值时，直接修改常量定义，而不需要每一处使用都修改具体值

- 变量：程序中值发生改变或者可以发生改变的元素

- 在Python语言中，变量和常量使用上基本没有区别





- 命名：给程序元素关联一个标识符，保证唯一性
- 变量和常量都需要一个名字
- 命名规则：
 - 大小写字母、数字和下划线的组合，但首字母只能是大小写字母或下划线，不能使用空格。
 - 中文等非字母符号也可以作为名字。
 - 以下是合法命名的标识符：



python_is_good

python_is_not_good

_is_it_a_question_

python语言



常量、变量与命名

- 标识符对大小写敏感，不能与保留字相同
- 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





表达式

- 表达式：程序中产生或计算新数据值的一行代码
- Python语言的33个保留字或者操作符可以产生符合语法的表达式。例如

```
>>>x=25                #将数字25赋给变量x
```

在使用变量前必须对其赋值，否则编译器报错





表达式中空格的使用

■ 空格的使用：

- 不改变缩进相关的空格数量
- 空格不能将命名分割
- 增加空格增加程序可读性



输入函数

- Input()函数从控制台获得用户输入

〈变量〉= input(〈提示性文字〉)

- 获得的用户输入以字符串形式保存在〈变量〉中

```
>>> input_string = input("请输入: ")
请输入: this is a string
>>>
```



表达式

■字符串操作

- 操作符+可以实现两个字符串的连接操作

```
>>> "python" + " is good"
'python is good'
```

- 字符串可理解为字节序列，若长度为L，第一个字节索引为0或-L，最后一个字节索引为L-1或-1

- 以区间形式获得字符串的子串

```
>>> tIndex[1:-2]
'yth'
>>>
```

```
>>> tIndex = "python"
>>> tIndex[4]
'o'
>>> tIndex[-4]
't'
```





- 如果 `val = "28C"`
- 则 `val[-1]`是最后一个字符串 "C"
- 前两个字符组成的子串可以用 `val[0:2]`表示，它表示一个从 $[0,2)$ 的区间。
- 由于约定用户输入的最后一个字符是C或者F，之前是数字，所以通过 `val[0:-1]`来获取除最后一个字符外的字符串





分支语句

■分支语句：控制程序运行，根据判断条件选择程序执行路径。基本过程如下：

```
if <条件1成立>:  
    <表达式组1>  
elif <条件2成立>:  
    <表达式组2>  
.....  
elif <条件N-1成立>:  
    <表达式组N-1>  
else:  
    <表达式组N>
```





赋值语句

- 赋值语句：使用等号给变量赋值

- `f=1.8*float(input_str[0:-1]) + 32`

- 同步赋值语句：同时给多个变量赋值（先运算右侧N个表达式，然后同时将表达式结果赋给左侧）

- `<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
```




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。



- 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

■ 计数循环，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

