PYTHON程序设计

张树兵

20230319改

励志

[清] 郝懿行

五日不读书，筋力欲衰老。
十日不作文，心思已枯槁。
常恐名不立，青青去人早。
铅椠无虚陈，经训恣蒐讨。
况偕素心人，疑义相与考。
偶然窥新境，郎若睹清昊。
孜孜入编记，日久成初稿。
敢云希古人，亦曰敦宿好。

补充知识点：零散概念，一共九个，常考

补充知识点1：list（）创建一个新列表

Python List list()方法

Python 列表

描述

list() 方法用于将元组转换为列表。

注：元组与列表是非常类似的，区别在于元组的元素值不能修改，元组是放在括号中，列表是放于方括号中。

语法

list()方法语法：

```
list( tup )
```

参数

- tup -- 要转换为列表的元组。

返回值

返回列表。

补充知识点1：list（）

```
1  a=list(range(9))
2  print(a)
3
4  b=list("123456789")
5  print(b)
6
```

Shell

```
>>> %Run 'list add.py'
  [0, 1, 2, 3, 4, 5, 6, 7, 8]
  ['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

补充知识点2： 作用域

变量仅在创建区域内可用。这称为作用域。

补充知识点2：局部作用域

在函数内部创建的变量属于该函数的局部作用域，并且只能在该函数内部使用。

在函数内部创建的变量在该函数内部可用：

```
def myfunc():
  x = 100
  print(x)

myfunc()
```

如上例中所示，变量 x 在函数外部不可用，但对于函数内部的任何函数均可用：

补充知识点3：全局作用域

在 Python 代码主体中创建的变量是全局变量，属于全局作用域。

全局变量在任何范围（全局和局部）中可用。

补充知识点3： 全局作用域

实例

在函数外部创建的变量是全局变量，任何人都可以使用：如 **x变量**

```
x = 100

def myfunc():
  print(x)

myfunc()

print(x)
```

补充知识点3： 同名变量 怎么办?

- 如果在函数内部和外部操作同名变量，Python 会将它们视为两个单独的变量，一个在全局范围内可用（在函数外部），而一个在局部范围内可用（在函数内部）：
- **x = 100**
- **def myfunc():**
- ** x = 200**
- ** print(x)**

- **myfunc()**
- **print(x)**

补充知识点4： Global 关键字

- 如果您需要创建一个全局变量，但被卡在本地作用域内，则可以使用 global 关键字。

- global 关键字使变量成为**全局变量**。

```
def myfunc():
    global x
   x = 100

myfunc()
print(x)
```

补充知识点5：复数

复数类型

与数学中复数的概念一致

如果$x^2 = -1$，那么x的值什么？

- **定义 $j = \sqrt{-1}$ ，以此为基础，构建数学体系**
- **a+b*j* 被称为复数，其中，a是实部，b是虚部**

补充知识点5：复数

复数类型

复数实例

z = 1.23e-4+5.6e+89*j*

- 实部是什么？ z.real 获得实部
- 虚部是什么？ z.imag 获得虚部

补充知识点5:: 复数

- 复数是由一个实数和一个虚数组合构成，表示为：x+yj
- 一个复数时一对有序浮点数 (x,y)，其中 x 是实数部分，y 是虚数部分。
- Python 语言中有关复数的概念：
- 1、虚数不能单独存在，它们总是和一个值为 0.0 的实数部分一起构成一个复数
- 2、复数由实数部分和虚数部分构成
- 3、表示虚数的语法：real+imagej
- 4、实数部分和虚数部分都是浮点数
- 5、**虚数部分必须有后缀j或J**

补充知识点5:: 复数

- aa=123-12j
- print aa.real # output 实数部分 123.0
- print aa.imag # output虚数部分 -12.0

补充知识点6：类型自动转换

数字类型的关系

类型间可进行混合运算，生成结果为"最宽"类型

- **三种类型存在一种逐渐"扩展"或"变宽"的关系：**

整数 -> 浮点数 -> 复数

- **例如：123 + 4.0 = 127.0 (整数+浮点数 = 浮点数)**

CC BY NC SA 4.0 许可

补充知识点7： 数值函数

数值运算函数

一些以函数形式提供的数值运算功能

函数及使用	描述
abs(x)	绝对值，x的绝对值 abs(-10.01) 结果为 10.01
divmod(x,y)	商余，(x//y, x%y)，同时输出商和余数 divmod(10, 3) 结果为 (3, 1)
pow(x, y[, z])	幂余，(x**y)%z，[..]表示参数z可省略 pow(3, pow(3, 99), 10000) 结果为 4587

补充知识点7：数值函数

数值运算函数

一些以函数形式提供的数值运算功能

函数及使用	描述
round(x[, d])	四舍五入，d是保留小数位数，默认值为0 round(-10.123, 2) 结果为 -10.12
max(x_1,x_2, ... ,x_n)	最大值，返回x_1,x_2, ... ,x_n中的最大值，n不限 max(1, 9, 5, 4 3) 结果为 9
min(x_1,x_2, ... ,x_n)	最小值，返回x_1,x_2, ... ,x_n中的最小值，n不限 min(1, 9, 5, 4 3) 结果为 1

补充知识点7： 数值函数

数值运算函数

一些以函数形式提供的数值运算功能

函数及使用	描述
int(x)	将x变成整数，舍弃小数部分 int(123.45) 结果为123； int("123") 结果为123
float(x)	将x变成浮点数，增加小数部分 float(12) 结果为12.0； float("1.23") 结果为1.23
complex(x)	将x变成复数，增加虚数部分 complex(4) 结果为 4 + 0j

补充知识点8： Python 日期

Python 中的日期不是其自身的数据类型，但是我们可以导入名为 datetime 的模块，把日期视作日期对象进行处理。

补充知识点8： Python 日期

导入 **datetime** 模块并显示当前日期：

```
import datetime

x = datetime.datetime.now()
print(x)
```

补充知识点8： 日期

日期包含年、月、日、小时、分钟、秒和微秒。

datetime 模块有许多方法可以返回有关日期对象的信息。

补充知识点8： weekday

返回 weekday 的名称和年份：

```
import datetime

x = datetime.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

补充知识点8：创建日期对象

如需创建日期，我们可以使用 datetime 模块的 datetime() 类（构造函数）。

datetime() 类需要三个参数来创建日期：年、月、日。

补充知识点8： datetime

创建日期对象：

```
import datetime

x = datetime.datetime(2020, 5, 17)

print(x)
```

补充知识点8： strftime() 方法

datetime 对象拥有把日期对象格式化为可读字符串的方法。

该方法称为 strftime()，并使用一个 format 参数来指定返回字符串的格式：

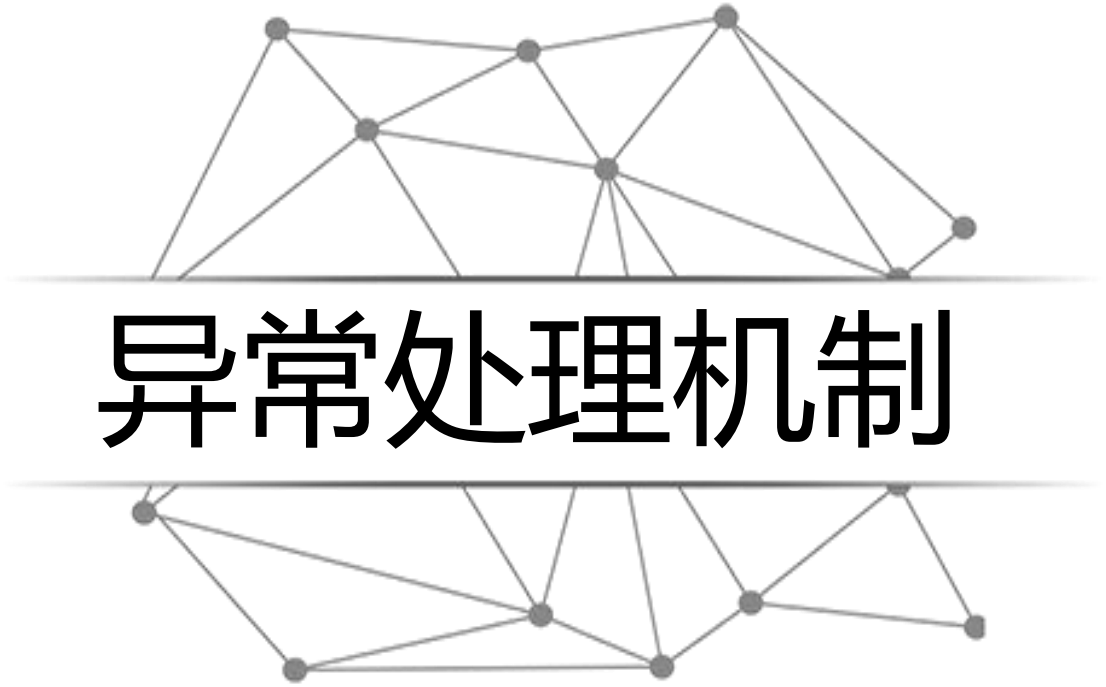
```
import datetime

x = datetime.datetime(2019, 10, 1)

print(x.strftime("%B")) #输出控制，见下表
```

所有合法格式代码的参考：

指令	描述	实例
%a	Weekday，短版本	Wed
%A	Weekday，完整版本	Wednesday
%w	Weekday，数字 0-6，0 为周日	3
%d	日，数字 01-31	31
%b	月名称，短版本	Dec
%B	月名称，完整版本	December
%m	月，数字01-12	12
%y	年，短版本，无世纪	18
%Y	年，完整版本	2018
%H	小时，00-23	17
%I	小时，00-12	05
%p	AM/PM	PM
%M	分，00-59	41
%S	秒，00-59	08
%f	微妙，000000-999999	548513
%z	UTC 偏移	+0100
%Z	时区	CST

%M	分，00-59	41
%S	秒，00-59	08
%f	微妙，000000-999999	548513
%z	UTC 偏移	+0100
%Z	时区	CST
%j	天数，001-366	365
%U	周数，每周的第一天是周日，00-53	52
%W	周数，每周的第一天是周一，00-53	52
%c	日期和时间的本地版本	Mon Dec 31 17:41:00 2018
%x	日期的本地版本	12/31/18
%X	时间的本地版本	17:41:00
%%	A % character	%

补充知识点9：



异常处理机制

补充知识点9:

异常处理: try-except语句

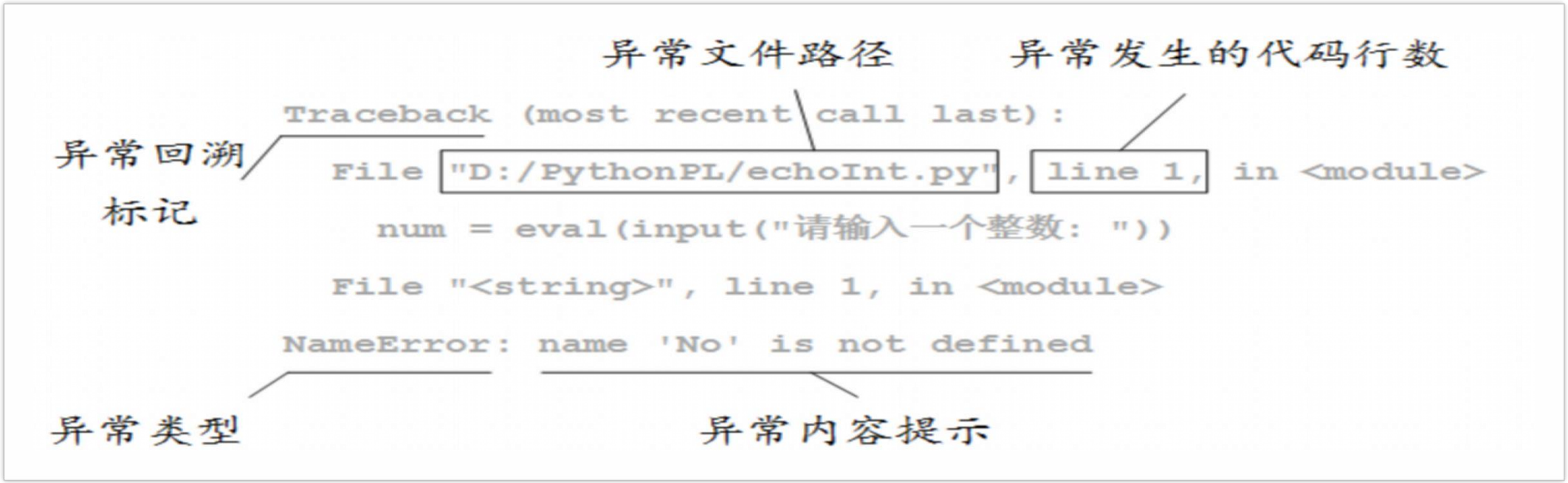
```
num = eval(input("请输入一个整数: "))

print(num**2)
```

```
Shell
请输入一个整数：5
5

>>> %Run '异常.py'

请输入一个整数：no
Traceback (most recent call last):
  File "D:\zsb\python2021\2021python\0527第十二次课\example\异常.py", line 1, in <module>
    num=eval(input("请输入一个整数："))
  File "<string>", line 1, in <module>
NameError: name 'no' is not defined

>>>
```

补充知识点9:

异常处理: try-except语句

Python解释器返回了异常信息，同时程序退出



补充知识点9:

异常处理: try-except语句

- Python异常信息中最重要的部分是异常类型，它表明了发生异常的原因，也是程序处理异常的依据。
- Python使用try-except语句实现异常处理，基本的语法格式如下:

```
try:
        <语句块1>
except <异常类型>:
        <语句块2>
```

补充知识点9:

异常处理: try-except语句

```
try:
num = eval(input("请输入一个整数: "))
print(num**2)
except NameError:
      print("输入错误，请输入一个整数!")
```

该程序执行效果如下:

```
>>> %Run '异常2.py'
 请输入一个整数: 5
 25
>>> %Run '异常2.py'
 请输入一个整数: no
 输入错误，请输入一个整数!
>>>
```

案例1：try except

```
try:
    num = eval(input("请输入一个整数: "))

    print(num**2)

except NameError:
    print("输入错误，请输入一个整数!")
```

补充知识点9:

异常处理: try-except语句

try-except语句可以支持多个except语句，语法格式如下：

```
try:
        <语句块1>
except <异常类型1>:
        <语句块2>
....
except <异常类型N>:
        <语句块N+1>
except:
        <语句块N+2>
```

异常的高级用法

- **最后一个except语句没有指定任何类型**，表示它对应的语句块可以处理所有其他异常。这个过程与if-elif-else语句类似，是分支结构的一种表达方式，一段代码如下

```
try:
    alp = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    idx = eval(input("请输入一个整数: "))
    print(alp[idx])
except NameError:
    print("输入错误，请输入一个整数!")
except:
    print("其他错误")
```

案例2：try except 高级

```
try:
    alp = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

    idx = eval(input("请输入一个整数: "))

    print(alp[idx])
except NameError:
    print("输入错误，请输入一个整数!")

except:
    print("其他错误")
```

异常的高级用法

该程序将用户输入的数字作为索引从字符串alp中返回一个字符，当用户输入非整数字符时，except NameError异常被捕获到，提示升用户输入类型错误，当用户输入数字不在01到256之间时，异常被except捕获，程序打印其他错误信息，执行过程和结果如 下：

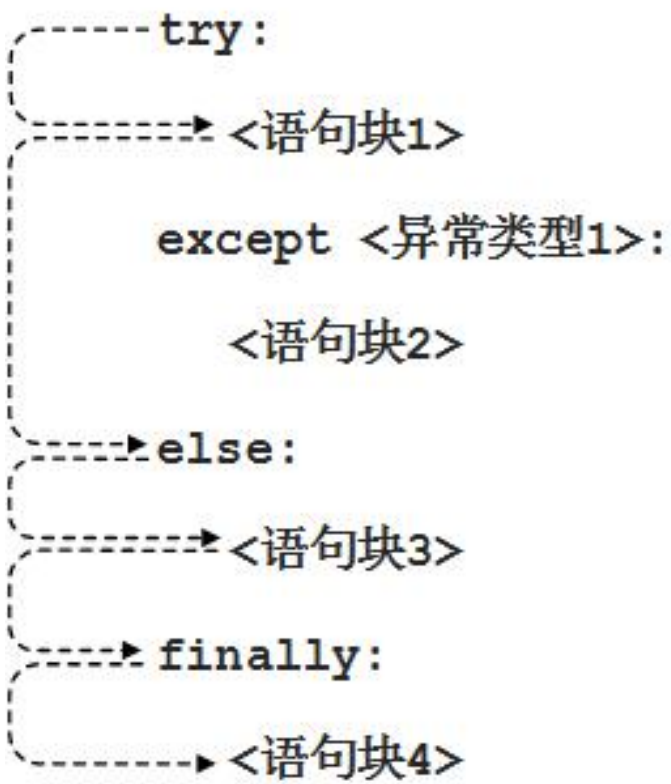
```
>>>
请输入一个整数： NO
输入错误，请输入一个整数!
>>>
请输入一个整数： 100
其他错误
```

异常的高级用法

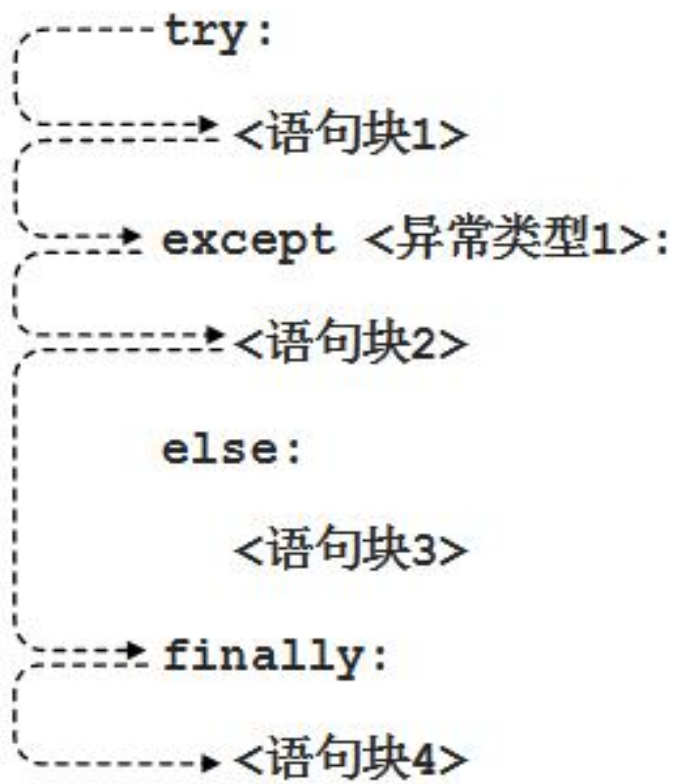
除了try和except保留字外，异常语句还可以与else和finally保留字配合使用，语法格式如下：

```
try:
        <语句块1>
except <异常类型1>:
        <语句块2>
else:
        <语句块3>
finally:
        <语句块4>
```

异常的高级用法



正常处理流程



异常处理流程

异常的高级用法

采用else和finally修改代码如下：

```
try:
    alp = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    idx = eval(input("请输入一个整数： "))
    print(alp[idx])
except NameError:
    print("输入错误，请输入一个整数!")
else:
print("没有发生异常")
finally:
    print("程序执行完毕，不知道是否发生了异常")
```

执行过程和结果如下：

```
>>>
请输入一个整数： 5
F
没有发生异常
程序执行完毕，不知道是否发生了异常
>>>
请输入一个整数： NO
输入错误，请输入一个整数!
程序执行完毕，不知道是否发生了异常
```

课程 结束语：

"人生苦短，我学Python"