实验 7 参考解答

2022 年 10 月 18 日

1 作业二

[1]:
```
import sys, math
```

对照教材中例子可以看出我们使用 sys.argv 直接解析命令行参数，这也是不需要考虑输入边数超过 3 个的原因：直接取用前三条边，不需要报错。

对于输入边数是否不足 3 条，只需要用一条 if 语句判断，当少于 3 条时配合 `raise` 报错即可。

这里直接赋值给 sys.argv 模拟命令行输入，并演示报错的效果：

[2]:
```
sys.argv = 'Demo.py 114514 1919810'.split(' ')
```

[4]:
```
if len(sys.argv) < 4:
    raise OSError("输入边数少于 3 条")

triangle = [sys.argv[1], sys.argv[2], sys.argv[3]]
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Input In [4], in <cell line: 1>()
      1 if len(sys.argv) < 4:
----> 2     raise OSError("输入边数少于 3 条")
      4 triangle = [sys.argv[1], sys.argv[2], sys.argv[3]]

OSError: 输入边数少于 3 条
```

非法的浮点数输入包含两种：一种是作为参数传入的字符串本身不是浮点数格式（如 “1a2”），另一种是传入的浮点数为非正数（如 “-2.3”，显然这无法构成三角形的一条边）。

我们先前已经将三条边构成了一个列表，此时只需要对列表每个元素进行检查。值得注意的是，如

果我们使用 `try-except` 结构来捕获异常，可以直接用 `float()` 尝试进行类型转换，失败时直接进入 `exception`（处理错误类型一）；而对于小于等于 0 的情况，用 `if` 检测出来后可以直接 `raise`（相当于手动“报错”），从而进入 `exception`（处理错误类型二）。两种 `exception` 可以用同一个 `raise` 进行报错。

这里手动设置 triangle 列表进行两种错误类型的演示：

[5]:
```
triangle = ['114514', 'homo', ' 谢谢茄子']
```

[6]:
```
try:
    for i in range(len(triangle)):
        triangle[i] = float(triangle[i])
        if triangle[i] <= 0:
            raise
except Exception as e:
    raise ValueError("不是合法的正浮点数: " + str(triangle[i]))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
      2 for i in range(len(triangle)):
----> 3     triangle[i] = float(triangle[i])
      4     if triangle[i] <= 0:

ValueError: could not convert string to float: 'homo'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Input In [6], in <cell line: 1>()
      5             raise
      6 except Exception as e:
----> 7     raise ValueError("不是合法的正浮点数: " + str(triangle[i]))

ValueError: 不是合法的正浮点数: homo
```

[7]:
```
triangle = ['114514', '-4.2', ' 谢谢茄子']
```

[8]:
```
try:
    for i in range(len(triangle)):
        triangle[i] = float(triangle[i])
        if triangle[i] <= 0:
            raise
except Exception as e:
    raise ValueError("不是合法的正浮点数: " + str(triangle[i]))
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Input In [8], in <cell line: 1>()
      4         if triangle[i] <= 0:
----> 5             raise
      6 except Exception as e:

RuntimeError: No active exception to reraise

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Input In [8], in <cell line: 1>()
      5             raise
      6 except Exception as e:
----> 7     raise ValueError("不是合法的正浮点数: " + str(triangle[i]))

ValueError: 不是合法的正浮点数: -4.2
```

判断三条边是否构成三角形有很多种方法，此处不予赘述；报错仍然用 `if` + `raise` 方式，自己定义的报错类型直接抄教材即可。

这里手动设置 triangle 列表进行错误演示：

[9]:
```
triangle = [1, 14, 514]
```

[10]:
```
class InvalidTriangleError(Exception):
    def __init__(self, message):
        self.message = message
```

```
for edge in triangle:
    if edge == max(triangle) or edge == min(triangle):
        continue
    if max(triangle) - min(triangle) >= edge:
        raise InvalidTriangleError("三条边不构成三角形")
```

```
---------------------------------------------------------------------------
InvalidTriangleError                      Traceback (most recent call last)
Input In [10], in <cell line: 5>()
      7     continue
      8 if max(triangle) - min(triangle) >= edge:
----> 9     raise InvalidTriangleError("三条边不构成三角形")

InvalidTriangleError: 三条边不构成三角形
```

当所有报错检查都通过后，计算三角形面积即可。

这里手动设置 triangle 列表进行正确结果的演示：

[11]:
```
triangle = [1919, 810, 1145]
```

[12]:
```
def triangle_area(a, b, c):
    s = (a + b + c) / 2
    return math.sqrt(s * (s - a) * (s - b) * (s - c))

print(' 面积: ', triangle_area(triangle[0], triangle[1], triangle[2]))
```

面积： 176410.97965829677

2 作业三

先检查我们要处理的字符串格式：用户的所有输入都是 “In[i]: ” 的格式，且位于同一行内。

[ ]:
```
In[1]: import numpy as np; a = np.arange(1, 16, 2)**2; a
Out[1]: array([  1,   9,  25,  49,  81, 121, 169, 225], dtype=int32)
In[2]: b = a.reshape(2, 4); b
Out[2]:
array([[  1,   9,  25,  49],
```

```
       [ 81, 121, 169, 225]], dtype=int32)
In[3]: np.savetxt('D:/Python/dat/b.txt', b)
In[4]: c = np.loadtxt('D:/Python/dat/b.txt'); c
Out[4]:
array([[  1.,   9.,  25.,  49.],
       [ 81., 121., 169., 225.]])
In[5]: np.save('D:/Python/dat/b.npy', b)
In[6]: c = np.load('D:/Python/dat/b.npy'); c
Out[6]:
array([[  1,   9,  25,  49],
       [ 81, 121, 169, 225]])
In[7]: np.savez('D:/Python/dat/ab.npz', a, b)
In[8]: cd = np.load('D:/Python/dat/ab.npz')
In[9]: c = cd['arr_0']; c
Out[9]: array([  1,   9,  25,  49,  81, 121, 169, 225])
In[10]: d = cd['arr_1']; d
Out[10]:
array([[  1,   9,  25,  49],
       [ 81, 121, 169, 225]])
```

接下来我们用 `with open() as f` 的方式读入该文件（不妨命名为 code.txt）。事实上，我们可以将整个文件作为字符串读入，读入结束后直接在 `with` 外侧来对其进行具体处理。

读入后建议先进行输出，以检查你自己保存的编码格式是否正确；如果是乱码，则需要增加 `encoding=`参数（完整格式：`open("fname", 'r', encoding='')`，对于输出时同理），如 gbk 编码为 `encoding='gbk'`，而 utf-8 编码为 `encoding='utf8'`。

[13]:
```
with open('code.txt', 'r') as f:
    in_str = f.read()

in_strs = in_str.split('\n')
```

代码的检测可以用是否以 “In” 开头来判断，而用户真正输入代码的起始位置判断可以用 “]:” 这一具有特征的子串。

[14]:
```
out_str = []
for line in in_strs:
    if line[0:2] == 'In':
```

```
        pos = line.find(']:')  # 找到输入代码起始位置所具备的特征子串
        out_str.append(line[(pos + 3):])  # 需要+3移动至输入代码真正的起始位置

out_str = '\n'.join(out_str)
```

输出没什么好讲的。

这里我们打印一下 out_str 来看看输出效果：

[15]:
```
with open('user_input.txt', 'w') as f:
    f.write(out_str)
```

[16]:
```
print(out_str)
```

```
import numpy as np; a = np.arange(1, 16, 2)**2; a
b = a.reshape(2, 4); b
np.savetxt('D:/Python/dat/b.txt', b)
c = np.loadtxt('D:/Python/dat/b.txt'); c
np.save('D:/Python/dat/b.npy', b)
c = np.load('D:/Python/dat/b.npy'); c
np.savez('D:/Python/dat/ab.npz', a, b)
cd = np.load('D:/Python/dat/ab.npz')
c = cd['arr_0']; c
d = cd['arr_1']; d
```

思考：

1. 我使用的是 `f.read()` 方法来一次性读入整个文本文件。能否用教材提供的 `f.readline()` 方法来逐行读入处理？
2. 事实上，直接搜索子串的方法并不是很严谨。判断一个字符串开头是否是 “In[i]” 的最严格方法是使用**正则表达式**（即 Python 的 re 模块）中的 `re.match()` 函数，这一方法该怎么实现？