Ch16-Exceptions

September 10, 2025

1 Exceptions

http://openbookproject.net/thinkcs/python/english3e/exceptions.html - dealing with bugs is normal part of programming - debugging is a very handy programming skill

1.1 category of bugs

- syntax errors
- logical/semantic errors
- runtime errors/exceptions

1.2 exceptions

- when runtime error occurs, it creates and throws an exception object
- program halts; Python prints out the traceback with exception name and message
- https://docs.python.org/3/tutorial/errors.html

[1]:
```
print(55/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-86210e37274a> in <module>
----> 1 print(55/0)

ZeroDivisionError: division by zero
```

[2]:
```
alist = []
print(alist[0])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-00bfde7948e0> in <module>
      1 alist = []
----> 2 print(alist[0])

IndexError: list index out of range
```

[3]:
```
atup = ('a', 'b', 'c')
atup[0] = 'A'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-92001c61f3b5> in <module>
      1 atup = ('a', 'b', 'c')
----> 2 atup[0] = 'A'

TypeError: 'tuple' object does not support item assignment
```

- each exception has two parts- Name: description

1.3 catching exceptions

- use try and except blocks
- try statement has several separate clauses/parts
- [ ] optional

1.3.1 example 1

[12]:
```
try:
    x = int(input("Enter dividend: "))
    y = int(input("Enter divisor: "))
    quotient = x/y
    remainder = x%y
except ZeroDivisionError as ex:
    print('Exception occured:', ex)
    print('arguments:', ex.args)
except ValueError as ex:
    print(ex)
except Exception as ex1:
    print('Some exception occured...', ex1)
except:
    print('some exception flew by...')
else:
    print("quotient=", quotient)
    print("remainder=", remainder)
finally:
    print("executing finally clause")
```

```
Enter dividend: 10
Enter divisor: 2
quotient= 5.0
remainder= 0
executing finally clause
```

[9]:
```
int(input('enter a number'))
```

```
enter a numberadsf
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-8f4f55f39064> in <module>
----> 1 int(input('enter a number'))

ValueError: invalid literal for int() with base 10: 'adsf'
```

1.3.2 example 2

- input validation

[13]:
```
while True:
    try:
        x = int(input("Please enter a integer: "))
        break
    except ValueError:
        print("Oops! That was not a valid number. Try again...")
```

```
Please enter a number: adf
Oops! That was not a valid number. Try again…
Please enter a number: asdf
Oops! That was not a valid number. Try again…
Please enter a number: asdf
Oops! That was not a valid number. Try again…
Please enter a number: sdasf
Oops! That was not a valid number. Try again…
Please enter a number: 2434.3534
Oops! That was not a valid number. Try again…
Please enter a number: 3534
```

[14]:
```
x
```

[14]:
```
3534
```

1.4 raising exceptions

- raise statement allows programer to throw their own exceptions
- Python provides several built-in exceptions
  - e.g.: NameError, ModuleNotFoundError, MemoryError, etc.
  - for details: https://docs.python.org/3/library/exceptions.html

1.4.1 example 1

[15]:
```
raise NameError("MyException")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-64e59e30969a> in <module>
----> 1 raise NameError("MyException")

NameError: MyException
```

[16]:
```
# except and raise execption
try:
    raise NameError('My Exception')
except NameError:
    print('An exception flew by...')
    raise
```

```
An exception flew by…
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-06305128be25> in <module>
      1 # except and raise execption
      2 try:
----> 3     raise NameError('My Exception')
      4 except NameError:
      5     print('An exception flew by…')

NameError: My Exception
```

1.5 user-defined exceptions

- one can define their own exceptions and raise them as needed
- should typically derive from the Exception class, either directly or indirectly

1.5.1 example 1

[ ]:
```
class InputError(Exception):
    """
    Exception raised for errors in the input.

    Attributes:
    expression -- input expression in which the error occured
    message -- explaination of the error
    """
```

```
    def __init__(self, expression, message):
        self.expression = expression
        self.message = message
```

[ ]:
```
help(InputError)
```

[ ]:
```
def getInteger():
    x = input('Enter an integer number: ')
    if not x.isdigit():
        raise InputError(x, 'That is not an integer!')
    return int(x)
```

[ ]:
```
x = getInteger()
print(x)
```

1.6 catch user-defined exception

[ ]:
```
try:
    x = getInteger() #may throw InputError
except InputError as ie:
    print('Exception:', ie)
    # can throw ie again
else:
    print('{}^2 = {}'.format(x, x**2))
```

[ ]:
```

```