Ch16-Exceptions

August 7, 2020

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 errros
- runtime errors/exceptions

1.2 exceptions

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

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

```
        ␣
↪---------------------------------------------------------------------------

        ZeroDivisionError                         Traceback (most recent call␣
↪last)

        <ipython-input-1-ef255b193978> in <module>()
    ----> 1 print(55/0)


        ZeroDivisionError: division by zero
```

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

```
        ␣
↪---------------------------------------------------------------------------
```

```
        IndexError                                Traceback (most recent call␣
↪last)

        <ipython-input-2-d994a3a20fe2> 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-3-8aeda75553d6> 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

[6]:
```
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:
    print('Some exception occured...')
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
```

[ ]:

```

```

1.3.2 example 2

- input validation

[7]:

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

```
Please enter a number: f
Oops! That was not a valid number. Try again…
Please enter a number: dsaf
Oops! That was not a valid number. Try again…
Please enter a number: adsf
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: 10
```

1.4 raising exceptions

- raise statement allows programer to throw their own exceptions

1.4.1 example 1

[8]:

```
raise NameError("MyException")
```

```
      ␣
↪---------------------------------------------------------------------------
```

```
        NameError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-8-290333e3086c> in <module>()
    ----> 1 raise NameError("MyException")


        NameError: MyException
```

```
[9]: 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-9-9b6ca7775e88> in <module>()
          1 try:
    ----> 2     raise NameError('My Exception')
          3 except NameError:
          4     print('An exception flew by…')
          5     raise


        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

```
[12]: 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
```

[13]:
```
help(InputError)
```

```
Help on class InputError in module __main__:

class InputError(builtins.Exception)
 |  Exception raised for errors in the input.
 |
 |  Attributes:
 |  expression -- input expression in which the error occured
 |  message -- explaination of the error
 |
 |  Method resolution order:
 |      InputError
 |      builtins.Exception
 |      builtins.BaseException
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, expression, message)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from builtins.Exception:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from builtins.BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(…)
 |      helper for pickle
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(…)
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  with_traceback(…)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from builtins.BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

[1]:
```
def getInteger():
    x = input('Enter an integer number: ')
    if not x.isdigit():
        raise InputError(x, 'That is not an integer!')
    return int(x)
```

[15]:
```
x = getInteger()
print(x)
```

```
Enter an integer number: dsaf


       ␣
↪---------------------------------------------------------------------------

        InputError                                Traceback (most recent call␣
↪last)

        <ipython-input-15-f90a077ee9cc> in <module>()
    ----> 1 x = getInteger()
          2 print(x)


        <ipython-input-14-6a80b90df6da> in getInteger()
          2     x = input('Enter an integer number: ')
          3     if not x.isdigit():
    ----> 4         raise InputError(x, 'That is not an integer!')
          5     return int(x)


        InputError: ('dsaf', 'That is not an integer!')
```

1.6 catch user-defined exception

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

```
Enter an integer number: 10
10^2 = 100
```

[ ]:
```

```