Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions

Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

Exceptions

Lect. PhD. Arthur Molnar

Babes-Bolyai University

arthur@cs.ubbcluj.ro

Overview

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

1 Exceptions
- Exception handling
- Specifications and exceptions
- Using exceptions in your programs

Exceptions

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions

Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

An **exception** is an event that disrupts the normal flow of a program's code

- Exceptions are present and used in many programming languages
- They are raised by code to signal an exceptional situation
- Your code will both raise (create) exception as well as "treat" them

NB!

The presence of an exception does not automatically mean that there's an error in the code

Exceptions

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

Most programming languages that support exceptions[1] use a
common terminology and syntax

- Raising or throwing exceptions
- Catching or treating an exception
- Exception propagation
- **try** / **raise** (throw) / **except** (catch) keywords

---

[1]https://docs.python.org/3/tutorial/errors.html#exceptions

Exception handling is the process of handling error conditions in a program systematically by taking the necessary action.

```
try:
    # code that may raise exceptions
except ValueError:
    # code that handles the situation
```

A few points from the Python syntax above[2]:

- If you want to catch exceptions, the code has to be in a **try** - **except** block
- Exceptions are caught using their type
- One try block can catch **one**, **several** or **all** exception types
- Creating exceptions in your code is done using the **raise** keyword
- You can provide additional arguments (e.g. an error message) to any Exception you raise

---

[2]https:www.tutorialspoint.com/python/python_exceptions.htm

An exception can be **handled** by:

- The function where the exception was raised
- Any function that called the raising function
- The Python runtime - **this will crash your program**.

Discussion

If the phrase *"unhandled exception has occured in you application..."* sounds familiar, now you understand what happened!

Demo

Exceptions example, **ex13_exceptions.py**

Exception handling

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

When to use exceptions?

- Signal an exceptional situation - the function is unable to fulfil its contract (e.g. function preconditions are violated, or the function encountered a situation in which it cannot progress - a required file was not found, is not accessible, etc.)
- Enforce function preconditions
- Generally speaking, you should **not use** exceptions to control program flow!

Function specification

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
**Specifications
and exceptions**
Using exceptions
in your programs

Is a way for abstracting **functions** that will only work if we provide:

- Meaningful name for the function
- Short description of the function (the problem solved by the function)
- Type and meaning of each input parameter
- Conditions imposed over the input parameters (**preconditions**)
- Type and meaning of each output parameter
- Relation between the input and output parameters (**post condition**)
- **Exceptions** that the function may raise

Function specification

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

- **Precondition** - a condition that must be true just prior to the execution of some section of code.
- **Post condition** - a condition that must be true just after the execution of some section of code.

```
def gcd(a, b):
    '''
    Return the greatest common divisor of two
        positive integers
    a,b - integers
    Return the greatest common divisor of a and b
    Raise ValueError if a <= 0 or b <= 0
    '''
```

How do we integrate exceptions into our test cases?

- Sometimes, a function works correctly if it raises an exception, and this must be tested

Demo

Test cases for functions throwing exceptions,
08-TestDrivenDevelopment-2.py, test function
testFindGoldbachPrimes()

Exceptions and layered architecture

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

How do we integrate exceptions into layered architecture programs?

NB!

- UI module(s) should not do a lot of processing
- Non-UI modules should not have any UI input/output

Our solution:

- We create exception with an argument or error message
- We catch them in the UI and display the corresponding message

Exceptions and layered architecture

Lecture 07

Lect. PhD.
Arthur Molnar

Exceptions
Exception
handling
Specifications
and exceptions
Using exceptions
in your programs

Demo

Let's take a look at **12-CalculatorModular.zip**, the feature for adding numbers