



CMPE 30032

Module 3

Exception Handling

TABLE OF CONTENTS

- 01 What are Exceptions?
- 02 Example of Exceptions Cases
- 03 Types of Exceptions
- 04 Handling an Unchecked Exceptions
- 05 Examples



What is an Exception?

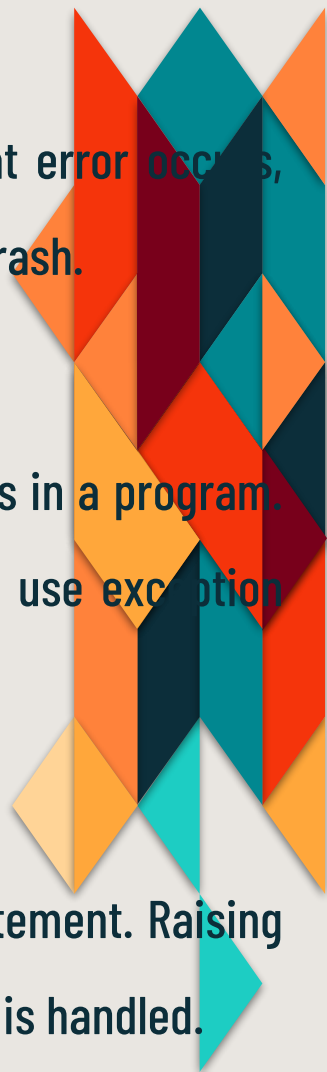
An exception is an error that happens during execution of a program. When that error occurs, Python generate an exception that can be handled, which avoids your program to crash.

Why use Exceptions?

Exceptions are convenient in many ways for handling errors and special conditions in a program. When you think that you have a code which can produce an error then you can use exception handling.

Raising an Exception

You can raise an exception in your own program by using the raise exception statement. Raising an exception breaks current code execution and returns the exception back until it is handled.



Example for Exception Cases:

IOError

- If the file cannot be opened.

ImportError

- If python cannot find the module

ValueError

- Raised when a function receives an argument that has the right type but an inappropriate value



Types of Exceptions

- Checked (Compile-time)

- Checked exceptions are checked at compile-time.
- Exceptions are “checked” because they are subject to the “**catch**” or “**specify requirement**” otherwise, the program code will not compile.

Examples: Invalid Syntax, Incorrect statements

- Unchecked (Runtime)

- Unchecked exceptions are not checked at compile time.
- Errors are not subject to the “**catch**” or “**specify requirement**”.
- It occurs during Runtime

Examples: invalid input, invalid arithmetic operations, number divided by zero



Python Unchecked Exceptions

SyntaxError

```
a = 1  
if a < 0  
    print("Negative")
```

ZeroDivisionError

```
num = 30  
result = num/0  
print(result)
```

NameError

```
print(result)
```

TypeError

```
num = "five"  
result = num+5  
print(result)
```

ValueError

```
num = int(input("Enter a number: "))  
result = num+2  
print(result)
```

Sample input: two



Handling Unchecked Exception

- ✓ **try** is used to test a block of code for errors
- ✓ **except** is used to handle errors
- ✓ **finally** is used to execute block of code regardless of
the result of the try and except blocks





Exception Handling - Structure

try:

Try block of Statements

except <<Exception Class>>:

Exception handling statements

else:

Else block statements

finally:

Finally block of statements



Using try...catch

```
print(age)
```

Output:

Traceback (most recent call last):

File "F:\DICTPython\Exception.py", line 1, in
<module>

```
    print(age)
```

NameError: name 'age' is not defined

```
try:
```

```
    print(age)
```

```
except:
```

```
    print("Age variable is not defined")
```

```
    print("You must assign a value first or  
        declare it")
```


Output:

Age variable is not defined

You must assign a value first or declare it

Print a specific error message if the try block raises a **NameError**

```
try:  
    print(age)  
except NameError:  
    print("Variable age is not defined")  
except:  
    print("Something else went wrong")
```



This statement will be executed since the **try** block encountered a **NameError**

Output:
Variable age is not defined



Else

else is used to define a block of code to be executed if no errors were raised

```
try:  
    print("Hello World")  
except:  
    print("An error occurred")  
else:  
    print("No error")
```

This statement will be executed since there's no error inside the **try** block

Output:
Hello World
No error



Finally

finally block will be executed regardless if the **try** block raises an error or not.

```
try:  
    print("Hello World")  
except:  
    print("An error occurred")  
else:  
    print("No error")  
finally:  
    print("Finished")
```

Output:

Hello World
No error
Finished

```
try:  
    print(age)  
except:  
    print("An error occurred")  
else:  
    print("No error")  
finally:  
    print("Finished")
```

Output:

An error occurred
Finished



Raise an Exception

raise is used to throw an exception if a condition occurs.

```
age = -1  
if age < 0:  
    raise Exception("Sorry, age is out of range")
```

```
age = "30"  
if not type(age) is int:  
    raise TypeError("Can accept integer value only")
```





Exception Handling – Example 01

```
x = 10
```

```
y = 5
```

```
try:
```

```
    result = x // y
```

```
    print("Yeah ! Your answer is :", result)
```

```
except ZeroDivisionError:
```

```
    print("Sorry ! You are dividing by zero ")
```

```
finally:
```

```
    print("End of Program")
```





Exception Handling - Example 02

```
x = -1
```

```
if x < 0:
```

```
    raise Exception("Sorry, no numbers below zero")
```

```
C:\Users\My Name>python demo_keyword_raise.py
```

```
Traceback (most recent call last):
```

```
  File "demo_ref_keyword_raise.py", line 4, in <module>
```

```
    raise Exception("Sorry, no numbers below zero")
```

```
Exception: Sorry, no numbers below zero
```





Exception Handling – Example 03

```
x = "hello"
```

```
if not type(x) is int:
```

```
    raise TypeError("Only integers are allowed")
```

```
C:\Users\My Name>python demo_keyword_raise2.py
```

```
Traceback (most recent call last):
```

```
  File "demo_ref_keyword_raise2.py", line 4, in <module>
```

```
    raise TypeError("Only integers are allowed")
```

```
TypeError: Only integers are allowed
```




Exception Handling - Example 04

try:

```
x = int(input("Enter the value of X : "))
y = int(input("Enter the value of Y : "))
result = x / y
print(" Answer : ", result)
```

except ZeroDivisionError:

```
print("Sorry ! You are dividing by zero ")
```

except ValueError:

```
print("Sorry ! Wrong input. ")
```

finally:

```
print("End of Program")
```



Exception Handling – Example 05



try:

```
f = open("demo.txt")  
f.write("Python Programming")
```

except:

```
print("Something went wrong when writing to the file")
```

finally:

```
f.close()
```





Exception Handling - Example 06

try:

```
a = 3
```

```
    if a < 4 :
```

```
        b = a / (a-3)
```

```
    print( "Value of b = ", b)
```

except (ZeroDivisionError, NameError):

```
    print( "Error Occurred and Handled" )
```

except:

```
    print ( "Unknown Exception" )
```





Exception Handling – Example 07

try:

```
x = int(input("Enter the value of X : "))  
y = int(input("Enter the value of Y : "))  
    print("Answer :", x / y)
```

except ZeroDivisionError:

```
    print("Division by Zero Error")
```

else:

```
    print("Else Block")
```

finally:

```
    print("End of Program")
```





Programming Exercise:

Create a Simple App Calculator

1. The application will ask the user to choose one of the four math operations (Addition, Subtraction, Multiplication and Division)
2. The application will ask the user for two numbers
3. Display the result
4. The application will ask if the user wants to try again or not.
5. If yes, repeat Step 1.
6. If no, Display "Thank you!" and the program will exit
7. Use Python Function and appropriate Exceptions to capture errors during runtime.