Python Notes – Exception Handling

Prepared by: **Dhananjay Kumar**
Trainer | Java Full-Stack & Python | Contact: 7023577968
LinkedIn: Dhananjay Kumar

Exception Handling in Python

---

1 Introduction to Exceptions

- An **exception** is an *unexpected error* that occurs during program execution.
- When Python encounters an error, it **stops** the program and throws an exception.
- Example of common exceptions:
  - `ZeroDivisionError`
  - `ValueError`
  - `IndexError`
  - `TypeError`
  - `FileNotFoundError`

Why Exception Handling?

To prevent your program from crashing and instead **handle errors gracefully**.

---

2 Exception Handling Basics

Python handles exceptions using:

try:

code that may raise an exception

except:

code that runs if exception occurs

---

3 The try–except Block

Basic Example

```
try:
    x = int(input("Enter a number: "))
    print(10 / x)
except ZeroDivisionError:
    print("You cannot divide by zero!")
```

In [2]:
```
## Catching Multiple Exceptions

try:
    a = int(input("Enter number: "))
    b = int(input("Enter number: "))
    print(a / b)
except ZeroDivisionError:
    print("Error: Division by zero.")
except ValueError:
    print("Error: Invalid number.")
```

```
1.0
```

4 . Using else Clause

-> Runs only if no exception occurs.

In [ ]:
```
try:
    num = int(input("Enter number: "))
```

```
except ValueError:
    print("Invalid input.")
else:
    print("Square is:", num*num)
```

5.The finally Clause

Code inside finally always executes

Used for cleanup activities (closing files, database connections, etc.)

In [ ]:
```
try:
    f = open("data.txt", "r")
    print(f.read())
except FileNotFoundError:
    print("File not found.")
finally:
    print("Closing file...")
```

6. Using except without Exception Type

Not recommended (but allowed)

In [ ]:
```
try:
    print(10 / 0)
except:
    print("An error occurred!")
```

7.Raising Exceptions Manually (raise)

In [ ]:
```
age = int(input("Enter age: "))
if age < 18:
    raise Exception("Age must be 18+")
print("Eligible.")
```

8. User Defined Exceptions

You can create your own exception by extending the Exception class.

In [ ]:
```
# Example: Custom Exception

class NegativeNumberError(Exception):
    pass

num = int(input("Enter positive number: "))

try:
    if num < 0:
        raise NegativeNumberError("Negative numbers not allowed!")
    print("Number:", num)
except NegativeNumberError as e:
    print("Custom Exception:", e)
```

9.Real-Life Example

In [ ]:
```
# Example: ATM Withdrawal Program

class InsufficientBalanceError(Exception):
    pass

balance = 5000

try:
    amount = int(input("Enter amount to withdraw: "))

    if amount > balance:
        raise InsufficientBalanceError("Insufficient balance in your account!")

    balance -= amount
    print("Withdrawal successful. Remaining balance:", balance)

except InsufficientBalanceError as e:
    print("Transaction Failed:", e)
except ValueError:
    print("Please enter a valid amount.")
finally:
    print("Thank you for using our ATM.")
```

Classroom Tasks –

---

1. Handle `IndexError` When Accessing a List

Question:

Write a Python program that safely accesses an element in a list and handles `IndexError`.

Explanation:

`IndexError` occurs when you try to access a list index that does not exist.
Using `try–except`, we can prevent the program from crashing.

Solution:

```
numbers = [10, 20, 30, 40]

try:
    index = int(input("Enter index (0–3): "))
    print("Value at index:", numbers[index])
except IndexError:
    print("Error: Index out of range! Please enter a valid index.")
except ValueError:
    print("Please enter a valid number for the index.")
```

2. User-Defined Exception – InvalidMarksError

Question:

Create a custom exception InvalidMarksError that triggers when marks > 100 or marks < 0.

Explanation:

Custom exceptions help us enforce rules in our program (like valid marks between 0 and 100). We create a class that extends the Exception class and use raise.

In [3]:
```
class InvalidMarksError(Exception):
    pass

try:
    marks = float(input("Enter marks (0–100): "))

    if marks < 0 or marks > 100:
        raise InvalidMarksError("Marks must be between 0 and 100.")

    print("Marks accepted:", marks)

except InvalidMarksError as e:
    print("Custom Exception:", e)
except ValueError:
    print("Please enter a valid numeric value for marks.")
```

```
Marks accepted: 2.0
```

3. Build a Simple Calculator with Exception Handling

Question:

Create a calculator that performs +, -, *, / operations and handles exceptions.

Explanation:

Errors can occur due to:

Invalid numeric input → ValueError

Division by zero → ZeroDivisionError

Invalid operator → custom logic handling

Using try–except ensures the calculator runs smoothly.

In [4]:
```
try:
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    op = input("Enter operator (+, -, *, /): ")

    if op == "+":
        print("Result:", a + b)

    elif op == "-":
        print("Result:", a - b)

    elif op == "*":
        print("Result:", a * b)

    elif op == "/":
        try:
            print("Result:", a / b)
        except ZeroDivisionError:
            print("Error: Cannot divide by zero!")

    else:
        print("Invalid operator. Please choose +, -, *, /")

except ValueError:
    print("Invalid input! Please enter numeric values only.")
```

```
Result: 13.0
```

In [ ]:
```
# 1. Practice Question: Convert string to integer safely

# Question:

# Write a program to read a number from the user and safely convert it to an integer using exception handling

try:
    num = int(input("Enter an integer: "))
    print("You entered:", num)
except ValueError:
    print("Invalid input! Please enter a number only.")
```

In [ ]:
```
# 2. Practice Question: File reading with FileNotFoundError

# Question:

# Write a program that tries to open a file. If it does not exist, handle the error gracefully.

try:
    f = open("sample.txt", "r")
    print(f.read())
except FileNotFoundError:
    print("File not found! Please check the filename.")
```

In [ ]:
```
 # 3 Practice Question: Multiple Except Blocks

# Question:

# Write a program to divide two numbers and handle both ValueError and ZeroDivisionError.

try:
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    print("Result:", a / b)
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Please enter valid numbers only.")
```

In [ ]:
```
 # 4. Practice Question: Using else block

# Question:

# Use the else block to print “success” if no exception occurs.

try:
    num = int(input("Enter a number: "))
except ValueError:
    print("Invalid number!")
else:
    print("Success! Number is:", num)
```

In [ ]:
```python
# 5. Practice Question: Using finally block

# Question:

# Demonstrate the use of finally to close a file safely.

try:
    f = open("data.txt", "r")
    print(f.read())
except FileNotFoundError:
    print("File not found!")
finally:
    print("Closing file (if opened)...")
```

In [ ]:
```python
 # 6. Practice Question: Check voting eligibility

# Question:

# Create a program that checks if age >= 18. If not, raise a custom exception NotEligibleError.

class NotEligibleError(Exception):
    pass

try:
    age = int(input("Enter age: "))
    if age < 18:
        raise NotEligibleError("You are not eligible to vote.")
    print("Eligible to vote!")
except NotEligibleError as e:
    print("Error:", e)
except ValueError:
    print("Age must be a number!")
```

In [ ]:
```python
# 7 .Practice Question: List sum with error handling

# Question:

# Add all elements in a list, but handle if the user enters non-numeric values.

values = ["10", "20", "abc", "40"]
total = 0

for v in values:
    try:
        total += int(v)
    except ValueError:
        print(f"Skipping invalid value: {v}")

print("Total =", total)
```

In [ ]:
```python
# 8. Practice Question: Dictionary key handling

# Question:

# Access a dictionary key and handle KeyError.

student = {"name": "Rohit", "age": 21}

try:
    print(student["marks"])
except KeyError:
    print("Key not found in dictionary!")
```

In [ ]:
```python
# 9.Practice Question: Nested try-except

# Question:

# Demonstrate nested try-except to handle multiple levels of errors

try:
    x = int(input("Enter a number: "))
    try:
        print(x / 0)
    except ZeroDivisionError:
        print("Division by zero not allowed!")
except ValueError:
    print("Please enter only integers.")
```

30 MCQs on Exception Handling in Python

---

1. What is an exception?

a) Syntax error
b) Logical error
c) Runtime error
d) Typing error
Answer: c

---

2. Which keyword is used to handle exceptions?

a) catch
b) try
c) except
d) handle
Answer: c

---

3. Code that may raise an exception is written inside which block?

a) except
b) try
c) final
d) throw
Answer: b

---

4. Which block always executes?

a) try
b) except
c) finally
d) else
Answer: c

---

5. Which keyword is used to raise an exception manually?

a) rise
b) throw
c) except
d) raise
Answer: d

---

6. What happens if exception is not handled?

a) Program continues
b) Program crashes
c) OS fixes it
d) Python ignores
Answer: b

---

7. What type of error is ZeroDivisionError?

a) Syntax
b) Logical
c) Runtime
d) Type
Answer: c

---

8. Which block runs only if no exception occurs?

a) finally
b) else
c) try
d) except
Answer: b

---

9. What exception occurs when converting "abc" to int?

a) TypeError
b) ValueError
c) NameError
d) IndexError
Answer: b

---

10. What exception occurs when accessing out-of-range list index?

a) KeyError
b) ValueError
c) IndexError
d) OverflowError
Answer: c

---

11. Which of the following catches all exceptions?

a) except Exception
b) except BaseException
c) except:
d) All of these
Answer: d

---

12. Which is the correct syntax?

a) except: try
b) try: except
c) try: ... except: ...
d) try except:
Answer: c

---

13. What exception occurs when a dictionary key is missing?

a) IndexError
b) KeyError
c) NameError
d) TypeError
Answer: b

---

14. Custom exceptions must inherit from:

a) Error
b) Exception
c) Base
d) Throwable
Answer: b

---

15. What will happen?

```
try:
    print(10 / 0)
```

```
except:
    print("Error")
```

```
a) Crash
b) Error
c) Prints "Error"
d) None
Answer: c
```

16. Which block is used for cleanup code?

a) clean
b) close
c) finally
d) end
Answer: c

---

17. What exception is raised if file not found?

a) IOError
b) FileError
c) FileNotFoundError
d) MissingFileError
Answer: c

---

18. What exception occurs when using an undefined variable?

a) NameError
b) ValueError
c) TypeError
d) SyntaxError
Answer: a

---

19. What exception occurs when dividing 10/3 normally?

a) DivisionError
b) Nothing
c) ZeroDivisionError
d) ValueError
Answer: b

---

20. Which of these is NOT an exception?

a) KeyError
b) FileNotFoundError
c) DivideByError
d) ValueError
Answer: c

---

21. Which keyword is used to define a custom exception class?

a) define
b) new
c) class
d) error
Answer: c

---

22. What exception occurs when adding int + str?

a) TypeError
b) ValueError
c) SyntaxError
d) KeyError
Answer: a

---

23. What does the else block do?

a) Always runs
b) Runs only if exception occurs
c) Runs only if no exception occurs
d) Never runs
Answer: c

---

24. What is the parent class of all exceptions?

a) Exception
b) BaseException
c) Error
d) PythonError
Answer: b

---

25. What happens here?

```python
try:
    x = int("12")
except:
    print("Error")
```

```
a) Error
b) Prints "Error"
c) Nothing
d) Converts and executes normally
Answer: d
```

```
## 26. Which exception is raised during invalid type conversion?
a) ValueError
b) TypeError
c) IndexError
d) NameError
**Answer: a**

---

## 27. What will this line do?
```python
raise ValueError("Invalid")
```

```
a) Ignore
b) Print nothing
c) Raise an error
d) Run normally
Answer: c
```

```
## 28. Can try block exist without except?

a) Yes
b) No
c) Yes, if finally exists
d) Only in Java
Answer: c

## 29. What exception occurs when list.remove(x) fails?

a) RemoveError
b) ValueError
```
```

c) KeyError
d) IndexError
Answer: b

```
## 30. What is the output?
try:
    print("A")
    1/0
except:
    print("B")
finally:
    print("C")
```

a) A
b) A B
c) A B C
d) B C
Answer: c

In [ ]: