Day 23

Exception Handling in Python:

What is an Exception?

An exception is an error that occurs while the program is running, which stops the normal flow of execution.

Example:

```
x = 10 / 0   # ZeroDivisionError
```

Output:

ZeroDivisionError: division by zero

Why Exception Handling?

Without handling exceptions:

- The program crashes
- Remaining code does not execute

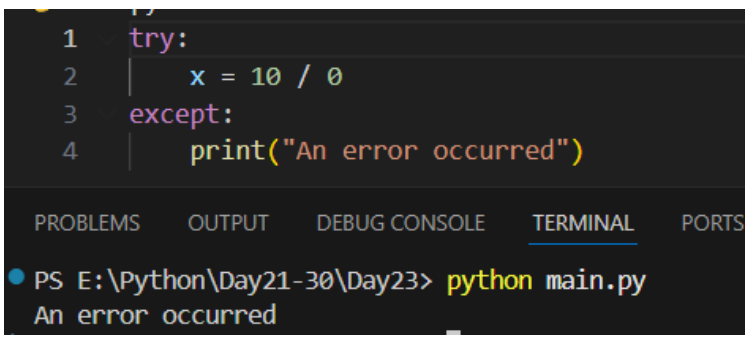
With exception handling:

- Program continues gracefully
- Errors are managed properly

Basic try–except: Use try to write risky code and except to handle errors.

```
try:
   x = 10 / 0
except:
   print("An error occurred")
```

Example:



Example: code to print the multiplication table.

```
6   a = input("Enter the number:")
7   print(f"Multiplication table of {a} is: ")
8
9   for i in range(1,11):
10      print(f"{int(a)} X {i} = {int(a)*i}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day21-30\Day23> python main.py
Enter the number:2
Multiplication table of 2 is:
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20
```

Example: what if we give string as the input in the above example? The program will show an error.

```
6   a = input("Enter the number:")
7   print(f"Multiplication table of {a} is: ")
8
9   for i in range(1,11):
10      print(f"{int(a)} X {i} = {int(a)*i}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day21-30\Day23> python main.py
Enter the number:Aditya
Multiplication table of Aditya is:
Traceback (most recent call last):
  File "E:\Python\Day21-30\Day23\main.py", line 10, in <module>
    print(f"{int(a)} X {i} = {int(a)*i}")
          ~~~^^^
ValueError: invalid literal for int() with base 10: 'Aditya'
```

Example: using try…except to handle the error.

```
6   a = input("Enter the number:")
7   try:
8       print(f"Multiplication table of {int(a)} is: ")
9       for i in range(1,11):
10          print(f"{int(a)} X {i} = {int(a)*i}")
11  except Exception as e:
12      print("Sorry some error occured.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day21-30\Day23> python main.py
Enter the number:Aditya
Sorry some error occured.
```

Example: we can omit the “exception as e”/

```
6   a = input("Enter the number:")
7   try:
8       print(f"Multiplication table of {int(a)} is: ")
9       for i in range(1,11):
10          print(f"{int(a)} X {i} = {int(a)*i}")
11  except:
12      print("Sorry some error occured.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day21-30\Day23> python main.py
Enter the number:k
Sorry some error occured.
```

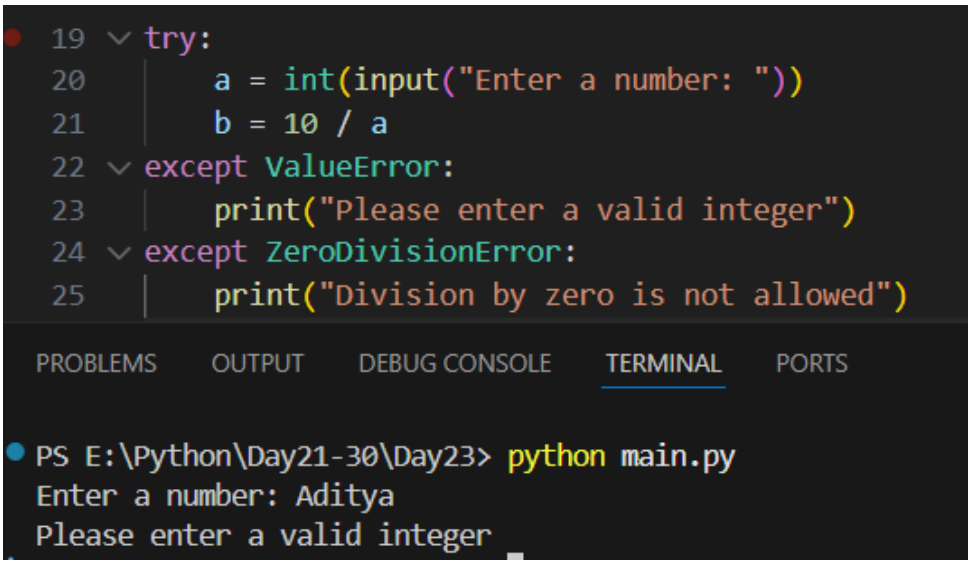
Example: catching specific exceptions

```
14  try:
15      x = 10 / 0
16  except ZeroDivisionError:
17      print("Cannot divide by zero")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR

PS E:\Python\Day21-30\Day23> python main.py
Cannot divide by zero
```

Example: we can add multiple “except”.



Summary:

- try block contains code that may cause an error
- except block handles the error if it occurs
- Program does not crash when exception is handled
- Always catch specific exceptions, not generic ones
- Multiple except blocks can handle different errors
- else runs only if no exception occurs
- finally runs whether an exception occurs or not
- Use Exception as e to get error details
- raise is used to create custom errors
- Keep try blocks short and focused

Finally Keyword in Python:

What is finally?

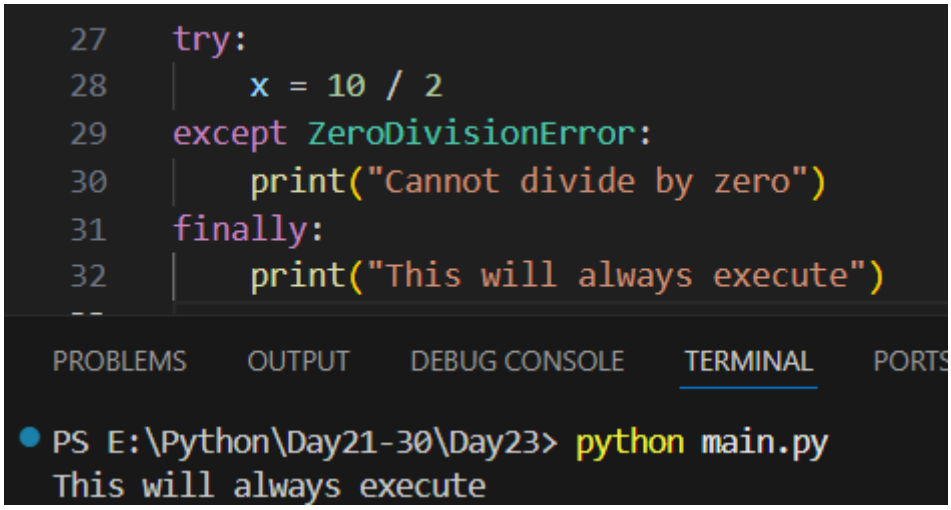
The finally block is used with try–except and always executes, whether:

- an exception occurs
- no exception occurs
- an exception is handled or not

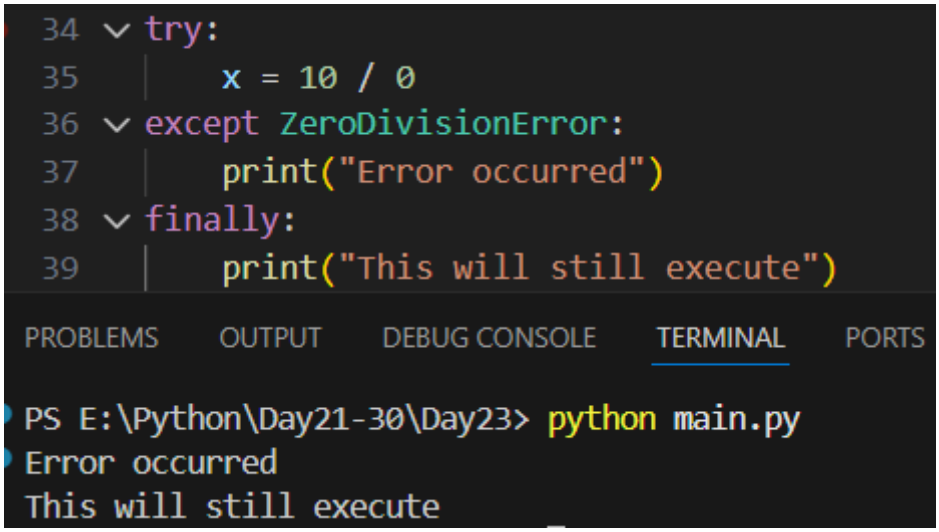
Basic Syntax

```
try:
    # risky code
except:
    # error handling
finally:
    # cleanup code (always runs)
```

Example: a basic example.



Example: finally is always executed even if try or except executed.



Why Use finally?

- To close files
- To release resources
- To disconnect databases
- To ensure cleanup code always runs

Important Points

- finally executes even if return is used
- finally executes even if an exception is not handled
- Only skipped if the program forcibly terminates
- finally is used to run cleanup code that must execute no matter what happens in try or except.

--The End--