Introduction to Python Programming

Dr. Navaneeth Bhaskar
Dept. of AI&DS
NMAM Institute of Technology, NITTE University

Part -3

Error

- An error is a problem that stops a program from running.
- Errors can be in software or hardware.

Types of Errors

a) Syntax Errors

- These happen when the rules of a programming language are broken.
- Example: Missing : or) in Python.

```
def main()
    a = 10
    b = 3
    print("Sum is", a + b
```

b) Semantic Errors

- These happen when the statement is correct in form but has no proper meaning.

Example:

- "Guitar plays Rama" – syntax is fine, but meaning is wrong.
- X * Y = Z – Invalid because expressions can't be on the left side of assignment.

c) Run Time Errors

- These occur when the program is running.
- Caused by illegal operations.

Examples:

- Opening a file that doesn't exist.
- Dividing a number by zero.

d) Logical Errors

- The program runs but gives the wrong output.

Example:

```
ctr = 1
while(ctr < 10):
    print(n * ctr)
```

- If n is not defined or logic is wrong, it gives unexpected output.

Exception

- Exceptions are errors that happen while the program is running.
- They are different from syntax errors.
- Even if code is written correctly (no syntax errors), something can still go wrong during execution.
- Example: Dividing a number by zero.

Difference Between Errors and Exceptions

Errors	Exceptions
Caused by mistakes in code (like syntax).	Caused by events during program execution.
Stops the program immediately.	Can be handled to continue the program.
Example: `if x ==` (syntax error)	Example: `10 / 0` (runtime error)

Common Python Exceptions

Exception	Cause
`SyntaxError`	Invalid syntax (e.g., missing colon, wrong indentation).
`TypeError`	Wrong type used in operation (e.g., adding int and string).
`NameError`	Variable/function name not defined.
`IndexError`	Index out of range in list/tuple.
`KeyError`	Key not found in dictionary.
`ValueError`	Wrong value for a function (e.g., converting 'abc' to int).
`AttributeError`	Accessing an undefined attribute/method.
`IOError`	Input/output operation failed (e.g., file read error).
`ZeroDivisionError`	Dividing a number by zero.
`ImportError`	Module cannot be found or imported.
`KeyboardInterrupt`	Program interrupted by the user (Ctrl+C).
`EOFError`	Input() reaches end of file without reading any data.

Methods to Handle Exceptions in Python

1. try-except Block

- Most basic way to handle exceptions.
- Code that may raise an error is placed in the try block.
- If an error occurs, the except block is executed.

Example:

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

2. Multiple except Blocks

- Used to handle different types of exceptions separately.

Example:

```
try:
    num = int(input("Enter number: "))
    print(10 / num)
except ValueError:
    print("Invalid input")
except ZeroDivisionError:
    print("Cannot divide by zero")
```

3. try-except with No Exception Name

- Catches all exceptions, regardless of type.
- Not recommended because it hides specific errors.

Example:

```
try:
    risky_code()
except:
    print("An error occurred")
```

4. else Block

- Optional block after all except blocks.
- Runs only if no exception occurs in the try block.

Example:

```
try:
    print("No error")
except:
    print("Error occurred")
else:
    print("This runs if no error")
```

5. finally Block

- Always runs, whether an exception occurs or not.
- Useful for clean-up tasks (like closing a file).

Example:

```
try:
    f = open("file.txt")
except:
    print("File error")
finally:
    f.close()
    print("File closed")
```

6. raise Statement

- Used to manually raise an exception.
- Useful for custom validation.

Example:

```
age = int(input("Enter age: "))
if age < 0:
    raise ValueError("Age cannot be negative")
```

Pandas

- Pandas is a Python library used for working with structured data.
- Name "Pandas" comes from "Panel Data" (used in statistics).
- Mainly used to handle tabular data (like Excel or databases).

Features of Pandas

- Read/write data from CSV, Excel, SQL.
- Organize data in rows and columns.
- Clean messy data.
- Filter, sort, and group data.
- Perform statistical operations (mean, max, min).
- Prepare data for machine learning or visualization.

Main Data Structures in Pandas

Structure	Description
`Series`	1D labeled array (like a single column of values)
`DataFrame`	2D structure (like a full Excel table) — can store text, numbers, etc.

```python
# Series Example
import pandas as pd
s = pd.Series([10, 20, 30, 40])
print(s)
```

```python
# DataFrame Example
data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}
df = pd.DataFrame(data)
print(df)
```

Data Cleaning and Preprocessing

Task	Code
Fill missing values	`df.fillna(0)`
Drop missing rows	`df.dropna()`
Filter records	`df[df['Age'] > 30]`
Add new column	`df['Salary'] = [50000, 60000, 70000]`

NumPy

- NumPy is a Python library for numerical computing.
- Works with arrays (faster and uses less memory than lists).
- Supports multi-dimensional arrays (1D, 2D, 3D).

Features of NumPy

- Perform math directly on arrays (add, subtract, multiply).
- Built-in functions like sum, mean, sqrt.
- Create arrays with zeros, ones, or random values.
- Used in machine learning, data science, and image processing.

Matplotlib

- A Python library for creating graphs and charts.
- Helps visualize data clearly.
- Works well with NumPy and Pandas.
- Types of Charts in Matplotlib
    - Line plot
    - Bar chart
    - Pie chart
    - Histogram
    - Scatter plot

Uses of Matplotlib

- Visualize trends, patterns, relationships in data.
- Explore large datasets interactively.
- Customize plots (title, labels, colors, gridlines, legends).
- Export plots to PNG, JPG, PDF.

Basic Plotting Examples

- Line Chart:

```
# Line Plot
import matplotlib.pyplot as plt
x = [1, 2, 3]
y = [2, 4, 1]
plt.plot(x, y)
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.title("Line Plot")
plt.show()
```

- Bar Chart:

```
# Bar Chart
categories = ['A', 'B', 'C', 'D']
values = [10, 20, 15, 25]
plt.bar(categories, values, color='blue')
plt.title("Bar Chart")
plt.show()
```