# Python Fundamentals for Data Analysis

ADELGACODE

"Data May Reveal Patterns, But It All Starts With Simple Lines Of Python Written By Curious Minds."

# WHAT IS PYTHON

**Python** is a programming language that helps people communicate instructions to computers in a simple and readable way.
It is used to analyze data, automate tasks, build applications, create websites, and even power artificial intelligence.

**Data analysis** means using tools like Python to work with data — cleaning it, organizing it, finding patterns, and visualizing it so it makes sense.

# Why Python for Data Analysis?

- Easy to learn and highly readable
- Powerful libraries(NumPy, Pandas, Matplotlib, etc.)
- Widely used in Business Intelligence and AI
- Ideal for data cleaning, analysis, and visualization

# Variables & Data Types

1. **int (integer)**

   Used to store **whole numbers** (no decimal point).

   eg.

```
age = 24
print(age)        # Output: 24
print(type(age))  # Output: <class 'int'>
```

   Integers are often used for counting, indexing, or arithmetic operations.

# Variables & Data Types

2. Float (Floating Point Number)

   eg.

```
temperature = 36.7
print(temperature)        # Output: 36.7
print(type(temperature))  # Output: <class 'float'>
```

*Floats are useful when precision matters, like in scientific or financial calculations.*

# Variables & Data Types

3. str (String)

Used to store **text** enclosed in quotes.

```
name = "Inusah"
print(name)        # Output: Inusah
print(type(name))   # Output: <class 'str'>
```

Strings are for letters, words, and sentences — anything made up of characters.

# Variables & Data Types

4. bool (Boolean)

   Represents **True** or **False** values.

```
is_student = True
print(is_student)        # Output: True
print(type(is_student))  # Output: <class 'bool'>
```

Booleans are mostly used in conditions and decision-making (e.g., if statements).

# Variables & Data Types

5. list :

An **ordered**, **changeable** collection of items.

```
fruits = ["apple", "banana", "mango"]
print(fruits)        # Output: ['apple', 'banana', 'mango']
print(type(fruits))  # Output: <class 'list'>
```

Lists can store different data types and can be modified (you can add, remove, or change items).

# Variables & Data Types

6. tuple:

An **ordered**, but **unchangeable** collection of items.

```
coordinates = (10.5, 20.8)
print(coordinates)        # Output: (10.5, 20.8)
print(type(coordinates))  # Output: <class 'tuple'>
```

Tuples are useful when you want to store data that shouldn’t be modified.

# Variables & Data Types

## 7. dict (Dictionary)

- Stores data in **key–value pairs** — like a real dictionary with "word: meaning.

```
student = {"name": "Adelga", "age": 24, "course": "ICT Education"}
print(student["name"])        # Output: Adelga
print(type(student))          # Output: <class 'dict'>
```

Dictionaries are perfect for structured data — quick lookup by key!

# Variables & Data Types

## 8. **set:**

**Unordered** collection of **unique** items — meaning it **automatically removes duplicates**.

```
numbers = {1, 2, 3, 3, 2, 4}
print(numbers)        # Output: {1, 2, 3, 4}
print(type(numbers))  # Output: <class 'set'>
```

Sets are great when you need to store data without duplicates or perform mathematical operations like union, intersection, and difference.

for, while

# Control Flow

- Conditional statements: if , elif, else
- Loops: (for, while)
- Eg.

```
for i in range(5):
print("Python is fun!")
```

Control flow gives logic to your data operations.

# Functions

A **function** is a reusable block of code that performs a specific task.

You can think of it like a **machine**:

**You give it input → it processes it → and gives you output.**

**Syntax**

```
def function_name(parameters):
    # code block
    return result
```

# A simple function

```
def greet(name):
    return f"Hello, {name}! Welcome to Python."
message = greet("Inusah")
print(message)
```

**Output:**

Hello, Inusah! Welcome to Python.

# Function without parameters

```
def say_hello():
    print("Hello there! Keep learning.")
say_hello()
```

**Output:**

```
Hello there! Keep learning.
```

# Function with multiple parameters

```
def add_numbers(a, b):
    return a + b
result = add_numbers(5, 7)
print(result)
```

**Output:**

12

# Function with default parameter

```
def greet_student(name="bro"):
    ptrint (f"Hi {name}, ready to explore STEM?")
greet_student()         # Uses default
greet_student("Adelga")  # Custom input
```

**Output**

Hi bro, ready to explore STEM?

Hi Adelga, ready to explore STEM?

# Why Functions Matter

1. Reduce repetition
2. Make code easier to read and debug
3. Help in teamwork and modular programming
4. Encourage reusability and logical organization

# File Handling in Python

File handling allows you to create, read, write, and delete files from your program,  just like how you open and edit files on your computer.

## In Python, you handle files in three main steps:

**Open** the file

**Read/Write** data

**Close** the file

# Syntax

```
file = open("filename", "mode")

# perform operations

file.close()
```

# Common File Modes

1. r Read (error if file doesn't exist)
2. w Write (creates a new file or overwrites existing one)
3. a Append (adds new data to the end of the file)
4. r+ Read and write
5. x Create a new file (error if file already exists)

# Writing to a File

```
file = open("student.txt", "w")

file.write("Name: Inusah\nCourse: ICT Education\nLevel: 300")

file.close()

print("Data written successfully!")
```

This creates (or overwrites) a file named **student.txt** and stores your data.

# Reading from a File

```
file = open("student.txt", "r")
content = file.read()
print(content)
file.close()
```

**Output:**

Name: Inusah

Course: ICT Education

Level: 300

# Appending Data to a File

```
file = open("student.txt", "a")
file.write("\nDepartment: Mathematics and ICT Education")
file.close()
```

This adds new content **without deleting** the previous one.

# Using the with statement (Best Practice)

```
with open("student.txt", "r") as file:
    data = file.read()
print(data)
```

The with statement automatically closes the file. No need to manually close().

# Reading Line by Line

```
with open("student.txt", "r") as file:
    for line in file:
print(line.strip())
```

## Why File Handling is Important

- Allows you to **process reports, logs, user info, or datasets**.
- Useful in **data analysis**, **automation scripts**, and **applications** that save user input.
- Used to **store data permanently** (unlike variables in memory).

# Mini Practice Task

Create a program that:

- Stores names and marks of 5 students in a list
- Prints the average mark
- Shows the highest and lowest scorer

# Key Takeaways

- Python is beginner-friendly yet powerful
- Understand logic before using libraries
- Lists and dictionaries are foundational for Pandas
- Build habits of writing clean, modular code

# THANK YOU

# References


Python Software Foundation. *(2025). Python Documentation. Retrieved from https://www.python.org*

NumPy Developers. (2025). *NumPy Documentation. Retrieved from https://numpy.org*

Pandas Community. (2025). *Pandas Documentation. Retrieved from https://pandas.pydata.org*

Matplotlib Developers. (2025*). Matplotlib Documentation. Retrieved from https://matplotlib.org*