Python Fundamentals 1

Introduction PF1

ADD SOMETHING HERE. Testing a change.

Learning Overview

| Concepts | Synergy | Syntax (Functions & Methods) | Objectives |

Input Output Operations

- Understand how to use the `print()` and `input()` functions for basic **input/output**
- Learn what **functions** are, and how **arguments** are passed into them
- Work with **strings**, including **escape sequences** and **f-strings** for formatting output
- Use **comments** to annotate code for clarity
- Create and use **variables** to store and manipulate data
- Recognise common built-in **data types** in Python (`int`, `float`, `str`, `bool`, *etc.*)
- Convert between types using **typecasting** functions such as `int()`, `float()`, `str()`, and `bool()`
- Explore **Boolean values** and how Python treats different values as `True` or `False`
- Apply built-in **mathematical operators** (`+`, `-`, , `/`, `//`, `%`, *)
- Use the `abs()` function to calculate absolute values when only magnitude matters

Logical Operations:

- Understand the role of logical operations in programming
- Learn the basic comparison operators (==, `!=`, <, >, <=, >=)
- Use Boolean values (`True` and `False`) in expressions
- Combine conditions with logical operators (`and`, `or`, `not`)
- Evaluate and predict the outcomes of logical expressions

Conditional Statements:

- Understand the purpose of conditional statements in controlling program flow
- Learn the syntax of `if`, `elif`, and `else` statements
- Correctly apply colons and indentation in conditional blocks
- Use comparison operators to create conditions
- Write simple decision-making programs that respond to user input
- Explore how multiple conditions can be structured and ordered logically
- Recognise and correct common errors in conditional statements

Errors

- Understand what **errors** are in Python
- Differentiate between **syntax errors** and **runtime errors**
- Read and interpret Python error messages
- Recognise common error types (e.g. `NameError`, `TypeError`, `IndexError`)
- Practise strategies for identifying and fixing errors
- Learn how **exceptions** work in Python
- Use `try` and `except` to handle errors
- Apply best practices for writing robust, error-tolerant code

Next