Python Data Types & Data Structures

Objective: Research Python's built-in data types and explain key concepts about data structures.

Tasks:

1. **Built-in Data Types:**
    - List Python's **primitives** and **collections**.
    - Provide **examples** of each.
2. **Primitive vs Collection:**
    - Explain the difference.
    - Why are primitives considered "simple" and collections "compound"?
3. **Data Structure:**
    - Define "data structure".
    - Explain how collections are examples of data structures.
4. **Mutable vs Immutable:**
    - Explain the difference.
    - Give examples of mutable vs immutable types.
    - Why does immutability matter?

Comments, Spacing and brackets

Objective: Research Python comments and formatting rules to write readable code.

Tasks:

1. **Comments:**
    - Single-line comments using #
    - Multi-line comments using triple quotes """ or '''
2. **Why Comments Matter:**
    - Explain how comments help maintain and understand code.
3. **Spacing & Indentation:**
    - Explain Python's use of indentation for blocks.
    - Show examples of correct and incorrect indentation.
    - Show how indentation may affect execution
4. **Explain the different use of brackets:**
    - []
    - ()
    - {}

Functions

Objective: Research Python functions and explain how they work.

Tasks:

1. **Definition:**
    - What is a function? Why do we use functions?
2. **Defining Functions:**
    - How do you define a function in Python?
    - Syntax rules (name, parameters, indentation).
3. **The return keyword:**
    - Explain what return does.
    - Show a function with and without return.
4. **Parameters and Arguments:**
    - What are they?
    - Difference between positional and keyword arguments.
5. **Scope in Functions:**
    - Explain local vs global variables.

Loops

Objective: Research Python loops and explain how iteration works.

Tasks:

1. **Types of Loops:**
    - Difference between for and while loops.
    - Show syntax and example of each.
2. **range() in Loops:**
    - Explain how range() works with for.
    - Show examples with start, stop, and step.
3. **Loop Control Keywords:**
    - break:
    - continue:
    - pass:
4. **Nested Loops:**
    - Explain and show an example.

Branching / Conditionals

Objective: Research Python branching and explain how programs make decisions.

Tasks:

1. **Basic Conditionals:**
    - if, if else, if elif else — explain syntax and flow.
    - Show examples for each.
2. **Logical Operators:**
    - and, or, not, !=, ==, <, >, <=, >=
    - Explain the difference between = (assignment) and == (comparison).
3. **Combining Conditions:**
    - Show how to combine multiple logical expressions.
4. **Explain Truthy and Falsy**
5. **Additional Exploration:**
    - Boolean context in Python (if []: vs if [1]:)
    - Ternary operator: x = a if condition else b