INTRODUCTION TO PYTHON

WEEK 3

[3:56 PM] Hombisa Dyasi

# List Methods  
- Lists in Python are versatile data structures, and they come with various built-in methods for easy manipulation.  
- Common list methods include `append()`, `extend()`, `insert()`, `remove()`, `pop()`, `index()`, `count()`, `sort()`, and `reverse()`.

# Using Lists as Stacks and Queues  
- \*\*Stacks:\*\* Last In, First Out (LIFO) structure. Use `append()` to push, and `pop()` to pop elements.  
- \*\*Queues:\*\* First In, First Out (FIFO) structure. Use `append()` to enqueue, and `pop(0)` to dequeue.

# List Comprehensions  
- A concise way to create lists in Python.  
- Syntax: `[expression for item in iterable if condition]`.  
- Example: `squared\_numbers = [x\*\*2 for x in range(10) if x % 2 == 0]`.

# The Delete Statement  
- The `del` statement is used to delete items from a list or variables.  
- Example: `del my\_list[2]` deletes the item at index 2 in `my\_list`.

# Tuples and Sequences  
- Tuples are immutable sequences, often used for heterogeneous data.  
- Sequences in Python include lists, tuples, and strings.

# Sets  
- Unordered collections of unique elements.  
- Created using curly braces `{}` or the `set()` constructor.

# Looping Techniques  
- `for` loops: Iterate over a sequence.  
- `while` loops: Repeatedly execute a block of code while a condition is true.

# Comparing Sequences and Other Types  
- Comparisons involve comparing elements in sequences or other data types.  
- Common comparison operators: `<`, `<=`, `>`, `>=`, `==`, `!=`.

# How to Create a List  
- Use square brackets `[]` to define a list.  
- Example: `my\_list = [1, 2, 3, 'hello']`.

# Access List Elements  
- Use indexing (`my\_list[0]`) to access elements in a list.

# Negative Indexing  
- Access elements from the end using negative indices (`my\_list[-1]`).

# How to Slice Lists in Python  
- Use slicing (`my\_list[start:stop:step]`) to extract a portion of the list.

# Add/Change List Elements  
- Use `append()`, `insert()`, or simple assignment to add or modify elements.

# Delete/Remove List Elements  
- Use `remove()`, `pop()`, or `del` to delete elements.

# Python List Methods  
- Built-in methods like `sort()`, `reverse()`, and others to manipulate lists.

# List Comprehension: Elegant Way to Create Lists  
- Concise syntax to create lists based on existing ones.

# Other List Operations in Python  
- Additional operations like copying lists, finding the length (`len()`), or checking membership.

# Iterating Through a List  
- Use `for item in my\_list:` to iterate through elements in a list.

# Errors  
- Mistakes in code that prevent it from running successfully.

# Exceptions  
- Events that occur during program execution that disrupt the normal flow.

# Handling Exceptions  
- Use `try` and `except` blocks to handle exceptions gracefully.

# Raising an Exception  
- Use `raise` to create and raise custom exceptions.

# The AssertionError Exception  
- Raised when an `assert` statement fails.

# The try and except Block  
- Used to catch and handle exceptions in Python code.

# User-Defined Exceptions  
- Define custom exceptions for specific error conditions.

# Defining Clean-Up Actions  
- Use `finally` block to define clean-up actions that must be executed, regardless of whether an exception occurred or not.

# Classes  
- Blueprint for creating objects with attributes and methods.

# Names and Objects  
- Objects are instances of classes, and variables hold references to objects.

# Python Scopes and Namespaces  
- Scope defines the visibility of a variable. Namespaces hold variable names.

# Class Definition Syntax  
- `class ClassName:` is used to define a class.

# Class Objects  
- Instances of a class that encapsulate data and behavior.

# Instance Objects  
- Objects created from a class.

# Method Objects  
- Functions defined within a class that operate on instances of that class.

# Class and Instance Variables  
- Variables shared among all instances (`class` variables) or unique to each instance (`instance` variables).

# Random Remarks  
- Miscellaneous comments or observations related to the discussed topics.