PYTHON CONCEPTS

FULL STACK SKILLS BOOTCAMP

PYTHON CONCEPTS

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. Modularising Functions
2. Classes
3. Data Structures: Lists, Sets, Tuples, and Dictionaries
4. Using the File System

MODULARISING FUNCTIONS BY IMPORTING

- **Why Modularise?**
  - Improves code readability and reusability
  - Facilitates easier debugging and maintenance

- **Creating a Module**

```
# my_module.py
def greet(name):
    return f"Hello, {name}!"
```

MODULARISING FUNCTIONS BY IMPORTING

- **Importing a Module**

```
import my_module
print(my_module.greet("Alice"))
```

- **Using an import**

```
from my_module import greet
print(greet("Bob"))
```

demo...

CLASSES IN PYTHON

- **What is a Class?**
- Blueprint for creating objects
- Contains attributes (variables)
- and methods (functions)
- **Defining a Class**

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def introduce(self):
        return f"My name is {self.name} and I am {self.age} years old."
```

CLASS INSTANCES

- **Creating an instance of a class**

```
person1 = Person("John", 30)
print(person1.introduce())
```

demo

DATA STRUCTURES IN PYTHON

- **Lists**
- Ordered, mutable collection
- Example:

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
```

- **Real-World Use Case:**
- Managing a list of tasks in a to-do list application
- Storing a series of user inputs in a web form

DATA STRUCTURES IN PYTHON

- **Tuples**
- Ordered, immutable collection
- Example:

```
coordinates = (10, 20)
```

- **Real-World Use Case:**
- Representing latitude and longitude coordinates for mapping applications
- Storing fixed configuration values like RGB colour codes

DATA STRUCTURES IN PYTHON

- **Sets**

- Key-value pairs
- Example:

```
student = {"name": "Alice", "age": 25}
```

- **Real-World Use Case:**
- Removing duplicate entries in a list of email addresses
- Keeping track of unique tags in a blog post system

DATA STRUCTURES IN PYTHON

- **Dictionaries**

- Unordered, unique elements
- Example:

```
unique_numbers = {1, 2, 3, 4}
```

- **Real-World Use Case:**
- Storing user profile information in a web application
- Mapping product IDs to product details in an e-commerce platform....demo

USING THE FILE SYSTEM

- **Reading from a File**

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

- **Writing to a File**

```
with open("example.txt", "w") as file:
    file.write("Hello, World!")
```

USING THE FILE SYSTEM

- **Appending to a File**

```
with open("example.txt", "a") as file:
    file.write("\nNew Line!")
```

- **Checking if a File Exists**

```
import os
if os.path.exists("example.txt"):
    print("File exists!")
```

CONCLUSION

- Modularising functions improves code structure and reusability.
- Classes and object-oriented create structured and reusable code.
- Data structures like lists, tuples, sets, and dictionaries, are flexible ways to store and manage data
- file handling allows us to work with external files for reading, writing, and managing data.

Step8Up Academy

QUESTIONS?