WEEK 2: DATABASE ASSIGNMENT

Python Basics:

Python is a high-level, interpreted programming language known for its readability and simplicity. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python's extensive standard library and its dynamic typing and memory management capabilities make it a popular choice among developers.

Key Features of Python:

1. **Readability**: Python's syntax is designed to be readable and straightforward, which reduces the cost of program maintenance.
2. Interpreted Language: Python code is executed line by line, which makes debugging easier.
3. Dynamic Typing: Data types are determined at runtime, allowing for more flexible coding.
4. Extensive Standard Library: Python's standard library supports many common programming tasks like file I/O, system calls, and internet protocols.
5. Community and Ecosystem: Python has a large and active community, which contributes to a wealth of third-party packages and libraries.

Use Cases:

- Web Development: Frameworks like Django and Flask.
- Data Science and Machine Learning: Libraries such as Pandas, NumPy, and Scikit-learn.
- Automation and Scripting: Automating repetitive tasks and system administration.
- Software Development: Building desktop applications with frameworks like Tkinter or PyQt.

Installing Python:

Steps to Install Python:

Windows:

- Download the Python installer from the official Python website.
- Run the installer and ensure you check the box that says "Add Python to PATH".
- Verify the installation by opening Command Prompt and typing python --version.

Setting Up a Virtual Environment:

1. Create a virtual environment:

```
python3 -m venv myenv
```

2. Activate the virtual environment:

WEEK 2: DATABASE ASSIGNMENT

- Windows: myenv\Scripts\activate

3. Deactivate the virtual environment:

```
deactivate
```

Python Syntax and Semantics:

Simple Python Program:

```
print("Hello, World!")
```

Explanation:

- print: A built-in function that outputs a message to the console.
- "Hello, World!": A string literal enclosed in double quotes.

Data Types and Variables:

- Basic Data Types in Python:
- int: Integer numbers, e.g., 42.
- float: Floating-point numbers, e.g., 3.14.
- str: String literals, e.g., "hello".
- bool: Boolean values, True or False.
- list: Ordered collection of items, e.g., [1, 2, 3].
- dict: Key-value pairs, e.g., {"key": "value"}.

WEEK 2: DATABASE ASSIGNMENT

Script Example:

```
# Integer
x = 5
print(x, type(x))


# Float
y = 3.14
print(y, type(y))


# String
name = "Python"
print(name, type(name))


# Boolean
is_python_easy = True
print(is_python_easy, type(is_python_easy))


# List
numbers = [1, 2, 3, 4, 5]
print(numbers, type(numbers))


# Dictionary
student = {"name": "John", "age": 21}
print(student, type(student))
```

WEEK 2: DATABASE ASSIGNMENT

Control Structures:

Conditional Statements:

```
x = 10
if x > 5:
    print("x is greater than 5")
```

Lo

```
# For loop
for i in range(5):
    print(i)


# While loop
count = 0
while count < 5:
    print(count)
    count += 1
```

Functions in Python:

Definition and Example:

Functions are blocks of reusable code that perform a specific task. They help in organizing code into manageable sections.

```
def add(a, b):
    return a + b


# Calling the function
result = add(5, 3)
print(result)
```

WEEK 2: DATABASE ASSIGNMENT

Lists and Dictionaries:

Differences:

- Lists are ordered collections of items, which can be accessed by their index.
- Dictionaries are collections of key-value pairs, which are accessed by key.

Script Example:

```
# List
numbers = [1, 2, 3, 4, 5]
print(numbers)
numbers.append(6)
print(numbers)

# Dictionary
student = {"name": "Alice", "age": 22}
print(student)
student["major"] = "Computer Science"
print(student)
```

Exception Handling:

Example:

```
try:
    number = int(input("Enter a number: "))
    print(f"The number is {number}")
except ValueError:
    print("That's not a valid number!")
finally:
    print("This block always executes.")
```

WEEK 2: DATABASE ASSIGNMENT

Modules and Packages:

Explanation and Example:

Modules are files containing Python code, and packages are collections of modules.

```
import math


print(math.sqrt(16))  # Using the math module
```

Reading and Writing Files:

```
# Reading a file
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)


# Writing to a file
lines = ["Hello", "World"]
with open('output.txt', 'w') as file:
    for line in lines:
        file.write(line + "\n")
```