**QUESTION 1**

**What is Python?**

Python is a high-level, interpreted programming language known for its simplicity and readability.

**Key Features:**

* Readability and Simplicity: Easy-to-read syntax.
* Versatility: Used in web development, data science, automation, etc.
* Large Standard Library: Extensive modules for various tasks.
* Dynamic Typing: No need to declare variable types.
* Interpreted Language: Executes code line by line.
* Community Support and Libraries: Rich ecosystem of libraries and frameworks.

**Use Cases:**

* Web Development: Popular frameworks include Django and Flask.
* Data Science and Machine Learning: Libraries like pandas, NumPy, scikit-learn, and TensorFlow.
* Automation and Scripting: Useful for automating repetitive tasks.
* Scientific Computing: Libraries such as SciPy and SymPy.
* Game Development: Tools like Pygame.
* Networking: Built-in support for network programming.

Python's readability, extensive library support, and versatility make it a popular choice among developers for a wide range of applications.

**QUESTION 2**

**Steps to Install Python on Windows**

1. **Download Python Installer:**
   * Go to the Python <https://www.python.org/downloads>
   * Download the latest Python installer for Windows.
2. **Run the Installer:**
   * Double-click the downloaded installer file.
   * Ensure you check the box “Add Python to PATH.”
   * Click “Install Now.”
3. **Verify Installation:**
   * Open Command Prompt.
   * Type python --version and press Enter.
   * You should see the installed Python version.
4. **Set Up a Virtual Environment**
5. **Open Command Prompt:**
   1. Navigate to your project directory using cd path\to\your\project.
6. **Create a Virtual Environment:**
   1. Run python -m venv myenv (replace myenv with your preferred name for the virtual environment).
7. **Activate the Virtual Environment:**
   1. Run myenv\Scripts\activate.
8. **Verify the Virtual Environment:**
   1. You should see (myenv) at the beginning of the command prompt line.
   2. To check the Python version within the virtual environment, type python --version.

**QUESTION 3**

**Python Syntax and Semantics:**

print("Hello, World!")

**Explanation of Basic Syntax Elements**

* **print()**: A built-in function in Python used to output text to the console.
* **"Hello, World!"**: A string literal enclosed in double quotes, representing the text to be printed.

This program uses the print() function to display the string "Hello, World!" on the console.

**QUESTION 4**

Data Types And Variables

**Basic Data Types in Python**

1. **Integer (int)**: Whole numbers, positive or negative.
2. **Float (float)**: Numbers with decimal points.
3. **String (str)**: Sequence of characters.
4. **Boolean (bool)**: Represents True or False.
5. **List (list)**: Ordered collection of items.
6. **Tuple (tuple)**: Ordered, immutable collection of items.
7. **Dictionary (dict)**: Collection of key-value pairs.

Script Demonstrating Variables of Different Data Types

**# Integer**

x = 10

print("Integer:", x)

**# Float**

y = 3.14

print("Float:", y)

**# String**

name = "Alice"

print("String:", name)

**# Boolean**

is\_active = True

print("Boolean:", is\_active)

**# List**

fruits = ["apple", "banana", "cherry"]

print("List:", fruits)

**# Tuple**

coordinates = (10.0, 20.0)

print("Tuple:", coordinates)

**# Dictionary**

person = {"name": "Alice", "age": 30}

print("Dictionary:", person)

**QUESTION 5**

Control Structures:

**Conditional Statements in Python**

**Conditional statements** allow you to execute code based on certain conditions.

**Example: if-else Statement**

python

Copy code

x = 10

if x > 5:

print("x is greater than 5")

else:

print("x is 5 or less")

* **if**: Checks if a condition is true.
* **else**: Executes if the preceding if condition is false.

**Loops in Python**

**Loops** allow you to execute a block of code multiple times.

**Example: for Loop**

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

* **for**: Iterates over a sequence (like a list).
* **fruit in fruits**: Iterates through each item in the list fruits.

**QUESTION 6**

**Functions In Python:**

Functions are reusable blocks of code that perform a specific task. They are useful for organizing code, reducing redundancy, and improving readability.

**Example Function to Return Sum of Two Arguments**

def add(a, b):

return a + b

**How to Call the Function**

result = add(5, 3)

print(result) # Output: 8

**Note:**

- Functions help organize and reuse code.

- The `add` function takes two arguments and returns their sum.

- Call the function with `add(5, 3)` and store the result in a variable.

**QUESTION 7**

**Lists And Dictionaries:**

**Differences Between Lists and Dictionaries**

* **List**: An ordered collection of items. Accessed by index.
* **Dictionary**: An unordered collection of key-value pairs. Accessed by key.

**Script Demonstrating Lists and Dictionaries**

**# Creating a list of numbers**

numbers = [1, 2, 3, 4, 5]

**# Basic operations on a list**

numbers.append(6) # Adding an element

print(numbers) # Output: [1, 2, 3, 4, 5, 6]

print(numbers[2]) # Accessing an element by index, Output: 3

**# Creating a dictionary with key-value pairs**

person = {"name": "Alice", "age": 30, "city": "New York"}

**# Basic operations on a dictionary**

person["email"] = "alice@example.com" # Adding a key-value pair

print(person) # Output: {'name': 'Alice', 'age': 30, 'city': 'New York', 'email': 'alice@example.com'}

print(person["name"]) # Accessing a value by key, Output: Alice

**QUESTION 8**

**Exception Handling in Python**

**Exception handling** allows you to gracefully manage and respond to errors that occur during program execution.

**Example Using try, except, and finally Blocks**

try:

x = 10 / 0 # Attempting division by zero

except ZeroDivisionError:

print("Error: Division by zero!")

finally:

print("This block always executes, regardless of whether an exception occurred or not.")

* **try**: Encloses code that might raise an exception.
* **except**: Handles specific exceptions that occur within the try block.
* **finally**: Executes cleanup code, always runs regardless of exceptions.

**QUESTION 9**

**Modules and Packages in Python**

* **Modules**: Python files that contain reusable code, including functions, classes, and variables.
* **Packages**: Collection of modules grouped together. Packages are directories containing multiple module files.

**Importing and Using Modules**

**Example Using the math Module**

**# Importing the entire math module**

import math

**# Using functions from the math module**

print(math.sqrt(16)) # Output: 4.0

print(math.pi) # Output: 3.141592653589793

**# Importing specific functions from the math module**

from math import ceil, floor

print(ceil(3.7)) # Output: 4

print(floor(3.7)) # Output: 3

 **import math**: Imports the entire math module.

 **math.sqrt(16)**: Uses the sqrt() function from the math module to compute the square root of 16.

 **from math import ceil, floor**: Imports specific functions ceil() and floor() from the math module directly.

**QUESTION 10**

**File I/O:**

**Reading from a File in Python**

**Script to Read and Print File Content;**

**# Open and read a file**

with open('example.txt', 'r') as file:

content = file.read()

print(content)

* **open('example.txt', 'r')**: Opens example.txt file in read mode ('r').
* **file.read()**: Reads the entire content of the file as a string.

**Writing to a File in Python**

**Script to Write a List of Strings to a File**

**# List of strings**

lines = ['First line\n', 'Second line\n', 'Third line\n']

**# Open and write to a file**

with open('output.txt', 'w') as file:

file.writelines(lines)

**SUMMARY**

 **Python Basics:**

* Python is a high-level, interpreted language known for readability and simplicity.

 **Installing Python on Windows:**

* Download from official site: <https://www.python.org> , run installer with "Add Python to PATH" checked, verify with python --version, set up virtual environments.

 **Syntax and Semantics:**

* Simple, clear syntax; uses whitespace for code structure.

 **Data Types and Variables:**

* Types include int, float, str, bool, list, tuple, dict; dynamically typed.

 **Control Structures:**

* Conditional statements (if-else), loops (for, while).

 **Functions:**

* Defined with def, reusable blocks of code.

 **Lists and Dictionaries:**

* Lists are ordered, accessed by index; dictionaries are unordered, accessed by key-value pairs.

 **Exception Handling:**

* try, except, finally blocks manage errors gracefully.

 **Modules and Packages:**

* Modules are Python files containing functions, classes; packages are directories of modules.

 **File I/O:**

* Read with open('file.txt', 'r'), write with open('file.txt', 'w').

**REFERENCES**

**Bing search engine:** [www.bing.com](http://www.bing.com)

**Google:** [www.google.com](http://www.google.com)

**w3schools:** <https://www.w3schools.com/python/python_intro.asp#:~:text=Python%20has%20a%20simple%20syntax,prototyping%20can%20be%20very%20quick>.