ADVANCED PYTHON PROGRAMMING

BRANCH:-CSE

SEMESTER – THIRD

These important questions have been prepared using your previous exam papers (PYQs), verified concepts, and additional reference from trusted online academic sources.
For deeper understanding, please refer to your class notes as well.

📘 ***For more study materials, notes, important questions, and updates, visit – DiplomaWallah.in***
📱 ***To join our official WhatsApp group for free updates, contact: 9508550281***

Unit I: Python Data Structures (Dictionary, Tuple, Set)

1. Differentiate between **List, Tuple, and Set** based on their core properties (mutability, ordering, and efficiency). Explain the utility of the **Dictionary** built-in methods, such as items(), keys(), and get().
    - Write a Python program to perform **union, intersection, and difference** operations on two sets.
2. **Short Program:** Write a Python snippet to sort a list of tuples based on the second element of each tuple (e.g., sorting student records by marks).

Unit II: Modules and Packages

3. Explain the concept of **Modular Programming**. Clearly distinguish between a **Module** and a **Package**. Describe the purpose of the **__init__.py** file.
    - Demonstrate the three ways to **import** a function from a module (Normal, from import, and from import *) using a custom module example.
4. **Concept:** What is **PIP**? Explain its role in managing Python libraries. Write the commands to **install** and **uninstall** a package named matplotlib.

Unit III: Exception Handling

5. Define **Exceptions**. Provide a detailed explanation of the **try-except-else-finally** block structure, detailing when each part of the block is executed.
    - Write a Python program to handle both a **ZeroDivisionError** and a **ValueError** (e.g., inputting text when expecting a number) within a single try block.
6. Explain the need for **User-Defined Exceptions**.
    - Write a short program that defines and **raises** a custom exception, such as InsufficientBalanceError, when a withdrawal amount exceeds the current account balance.

Unit IV: Files Handling

7. Differentiate between **Text Files** and **Binary Files**. Explain the concept of **Object Serialization** and how the **pickle module** is used for this purpose.
   - Write a program that uses the **pickle.dump()** function to store a dictionary into a binary file, and then uses **pickle.load()** to read and print the dictionary back to the console.
8. Explain the different **File Access Modes** in Python (e.g., 'r', 'w', 'a', 'r+'). What is the role of **File Pointers**?
   - Write a code snippet to demonstrate the usage of the **seek()** and **tell()** methods to navigate a file pointer.

Unit V: Graphics with Turtle

9. Write a complete Python program using the **turtle module** to draw a **Colorful Spiral Design** (such as a square spiral or an equiangular spiral). Your program must utilize **loops** (like for or while) and include commands to **change the pen color and size** dynamically within the loop.
10. List and explain the function of any six common **Turtle Methods** (e.g., forward(), right(), circle(), penup(), pendown(), speed()).
11.
    - Write a snippet that uses the **Turtle module** to draw a green circle if a variable status is True and a red square if status is False, demonstrating the use of a simple **conditional statement (if/else)**.

🐍 Classic Python Programming Questions

1. Number Property Checks (Conditions & Loops)

These programs usually involve taking a single integer input from the user and checking a specific property.

Concept	**Question Format**
Armstrong Number	Write a Python program to check if a given integer is an **Armstrong number** (a number that is equal to the sum of cubes of its digits).
Prime Number	Write a Python program to check if a given positive integer is a **Prime number**. (The number is divisible only by 1 and itself).
Palindrome Number	Write a Python program to check if a given number is a **Palindrome** (reads the same backward as forward, e.g., 121).

Concept	Question Format
Perfect Number	Write a Python program to check if a given number is a **Perfect number** (a positive integer that is equal to the sum of its proper positive divisors, excluding the number itself, e.g., 6).
Odd/Even	Write a Python program to check if a number is **Odd or Even** using the modulus operator.

2. Series & Sequence Generation (Loops)

These programs require generating a series of numbers based on a pattern.

Concept	Question Format
Fibonacci Series	Write a Python program to generate the **Fibonacci series** up to *n* terms, where *n* is provided by the user (e.g., 0, 1, 1, 2, 3, 5, 8...).
Factorial	Write a Python program to find the **Factorial** of a number entered by the user (e.g., Factorial of 5 is $5 \times 4 \times 3 \times 2 \times 1$).
Sum of Digits	Write a Python program to calculate the **sum of the digits** of a given number (e.g., for 123, the sum is $1+2+3=6$).
Tables/Multiples	Write a Python program to print the multiplication table (of a number) up to 10 using a loop.

3. String & List Operations (Iteration & Methods)

These programs test your ability to handle basic Python data structures.

Concept	Question Format
Reverse String	Write a Python program to **reverse a given string** without using built-in slicing functions.
Count Vowels	Write a Python program to count the number of **vowels** in a given string.
List Manipulation	Write a Python program to find the **largest or smallest element** in a list without using built-in functions like max() or

Concept	Question Format
	min().
List Sorting	Write a Python program to **sort a list of numbers** in ascending or descending order (often testing bubble sort logic).

---

4. Advanced Logic & Output (Nested Loops)

These questions test more complex logic, often involving nested loops.

Concept	Question Format
Patterns	Write a Python program to print **star patterns** (e.g., a right-angled triangle, pyramid, or diamond) using nested loops.
Prime Numbers in Range	Write a Python program to **find all Prime numbers** within a specified range (e.g., between 100 and 200).
GCD/LCM	Write a Python program to find the **GCD (Greatest Common Divisor)** or **LCM (Least Common Multiple)** of two numbers.

QUICK REVISE

Unit I: Data Structures (Dict, Tuple, Set)

1. **Tuple: Immutable** (cannot be changed after creation), ordered, fast for read operations.
   - **Program Focus:** Accessing elements, **tuple packing/unpacking**, and iterating using loops.
2. **Set: Mutable**, unordered collection of **unique** items. Used for fast membership testing and mathematical set operations.
   - **Program Focus:** Using methods like **.add(), .remove(), union(), intersection()**.
3. **Dictionary: Mutable** and unordered collection of **key-value** pairs. Keys must be unique and immutable.
   - **Program Focus:** Accessing values by key, using methods like **.keys(), .values(), .items(), .get(), and .update()**.
   - **Quick Program Snippet:** Frequency counting of list elements using a dictionary.

Unit II: Modules and Packages

4. **Module:** A single file (.py) containing Python definitions (functions, classes, variables). Promotes **modularity** and reusability.
5. **Package:** A collection of related modules organized in a directory structure. Must contain an **__init__.py** file (can be empty).
6. **Import Methods:**
   - **import math**: Access via math.sqrt().
   - **from math import sqrt**: Access via sqrt().
   - **from math import** *: Imports all names; generally discouraged due to naming conflicts.
7. **PIP: Package Installer for Python**. Used to download, install, and manage third-party libraries (e.g., pip install pandas).
   - **Program Focus:** Creating a simple module/package structure and successfully importing its components.

Unit III: Exception Handling

8. **Error vs. Exception: Errors** (e.g., SyntaxError) prevent code from running. **Exceptions** (e.g., ZeroDivisionError) occur during execution but can be handled.
9. **Handling Blocks:** The core structure is **try** (code that might fail), **except** (code to run if an exception occurs), **else** (runs only if try succeeds), and **finally** (always runs, regardless of success or failure).
10. **Raising Exceptions:** Use the **raise** keyword to trigger an exception manually. Essential for **User-Defined Exceptions**.
    - **Program Focus:** Implementing the full **try-except-finally** block to manage multiple exception types (e.g., ZeroDivisionError, ValueError).

Unit IV: Files Handling

11. **File Types: Text Files** (store text, handle encoding, uses newline character \n). **Binary Files** (store raw bytes—images, executables, pickled objects).
12. **File Access Modes:** Know the common modes:
    - **'r'** (Read), **'w'** (Write, overwrites), **'a'** (Append).
    - **'rb'** (Read Binary), **'wb'** (Write Binary).
13. **File Pointers:** Control the position within the file.
    - **seek(offset)**: Moves the pointer to a specified position (offset).
    - **tell()**: Returns the current position of the pointer.
14. **Object Serialization (Pickle):** The process of converting a Python object (like a list or dictionary) into a stream of bytes for storage or transmission.

Unit V: Graphics with Turtle

15. **Turtle Module:** Provides an environment for drawing shapes and patterns using an on-screen **"turtle"** pen. Excellent for teaching loops and control flow.
16. **Key Methods:**
    - **Pen Movement: forward(), backward(), left(), right()**.
    - **Pen Control: penup(), pendown(), pensize(), color()**.
    - **Screen/State Control: turtle.done()** or **turtle.mainloop()** (to keep the window open).

DIPLOMA WALLAH (SWANGAM 💗)