CMR INSTITUTE OF TECHNOLOGY, BENGALURU	USN										

Internal Assesment Test – II

Sub:	Introduction to Python, Data and Control Systems							Code:	MBABA313
Date:	23-01-2026	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MBA

SET- I

	ANSWER KEY/SOLUTION	Marks	OBE CO	OBE RBT
	Part A - Answer Any Two Full Questions (2* 20 = 40 marks)			
1 (a)	**Differentiate between mutable and immutable data types in Python with suitable examples.** **Mutable Data Types** Mutable objects can be changed after they are created. You can modify their content without creating a new object. • Examples of mutable data types: list dict set bytearray **Example:** numbers = [1, 2, 3] numbers[0] = 10 print(numbers) Output: [10, 2, 3] **Immutable Data Types** Immutable objects cannot be changed once they are created. Any modification creates a new object. • Examples of immutable data types: int float str tuple frozenset **Example:** name = 'Python' name[0] = 'J' TypeError: 'str' object does not support item assignment **Key Differences** Mutable — Immutable Can be modified — Cannot be modified List, Dict, Set — Int, String, Tuple	[03]	CO3	L2
(b)	**Evaluate the significance of local & global scope of variables with examples.** Variable scope defines where a variable can be accessed in a program. In Python, the most important scopes are local and global. **Local Scope:** A local variable is defined inside a function and can only be accessed within that function. **Example:** def calculate_sum():	[07]	CO3	L5

```
    a = 10
    b = 20
    print(a + b)

calculate_sum()
```

Significance of Local Scope:

- Prevents accidental modification of data
- Makes functions independent and reusable
- Improves readability and security
- Avoids name conflicts

Global Scope:

A global variable is defined outside all functions and can be accessed anywhere in the program.

Example:

```
x = 50

def display():
    print(x)

display()
```

Modifying Global Variable:

```
count = 0

def increment():
    global count
    count += 1

increment()
print(count)
```

Comparison:

Local variables are accessible only within functions and exist during function execution.

Global variables are accessible throughout the program and exist during the entire execution.

(c)	**Create a list with the contents 1,2,3,4,5,6,7,8,9 and perform the following:** **a) Print first 3 elements from the list** **b) Print from 5th element to end of the list** **c) Print any three elements from the middle.** **d) Add the element '10' to the list.**	[10]	CO3	L3

```
# Create the list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# a) Print first 3 elements
print("First 3 elements:", numbers[:3])

# b) Print from 5th element to end
print("From 5th element to end:", numbers[4:])

# c) Print any three elements from the middle
print("Three elements from the middle:", numbers[3:6])

# d) Add the element '10' to the list
```

	numbers.append(10)
	print("List after adding 10:", numbers)
	
	Output:
	`First 3 elements: [1, 2, 3]`
	`From 5th element to end: [5, 6, 7, 8, 9]`
	`Three elements from the middle: [4, 5, 6]`
	`List after adding 10: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`			

2 (a)	**Elaborate on different string handling functions with examples.** String handling functions are built-in methods used to **manipulate and process strings** in Python. Some commonly used string handling functions are: • `len()` – Returns the length of the string Example: `len("Python")` • `lower()` – Converts string to lowercase Example: `"HELLO".lower()` • `upper()` – Converts string to uppercase Example: `"hello".upper()` • `strip()` – Removes leading and trailing spaces Example: `" hi ".strip()` • `replace()` – Replaces one substring with another Example: `"I like Python".replace("Python","Java")` • `split()` – Splits a string into a list Example: `"Python is easy".split()` • `find()` – Returns index of a character or substring Example: `"Python".find("t")` • `isalpha()` – Checks if string contains only alphabets Example: `"Hello".isalpha()` These functions are widely used in **data validation, text processing, and input handling**.	[03]	CO3	L2

(b)	**What is a function? Analyze define and call a function in a Python program.**
	What is a Function?
	A **function** is a block of reusable code that performs a specific task. Functions help in reducing code repetition, improving readability, and making programs modular.
	Defining a Function in Python
	In Python, a function is defined using the **`def`** keyword followed by the function name, parentheses, and a colon.
	
	Syntax:
	
	`def function_name(parameters):`
	`  statements`
	Example (Function Definition):
	
	`def add(a, b):`
	`  return a + b`
	Calling a Function in Python
	A function is called by using its name followed by parentheses and passing arguments.
		[07]	CO3	L4

Syntax:

```
function_name(arguments)
```

Example (Function Call):

```
result = add(10, 20)
print(result)
```

Output:

```
30
```

Complete Python Program

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

result = add(5, 3)
print("Sum =", result)
```

Functions make programs structured, reusable, and easy to understand. They are essential for writing efficient and organized Python programs.

(c)	**Write a Python program using dictionary data structure to store and manage student details such as student ID, Name, Subject, and marks.** **The program should:** **1. Create a dictionary with student details** **2. Display all student records** **3. Update the marks of a student** **Display the updated dictionary.**	[10]	CO4	L3

Python Program Using Dictionary for Student Details

```
# 1. Create a dictionary with student details
student = {
   "Student_ID": 101,
   "Name": "Rahul",
   "Subject": "Python",
   "Marks": 75
}

# 2. Display all student records
print("Student Details:")
for key, value in student.items():
   print(key, ":", value)

# 3. Update the marks of the student
student["Marks"] = 85

# Display updated dictionary
print("\nUpdated Student Details:")
for key, value in student.items():
   print(key, ":", value)
```

Output:

```
Student Details:
Student_ID : 101
Name : Rahul
Subject : Python
Marks : 75
```

```
Updated Student Details:
Student_ID : 101
Name : Rahul
Subject : Python
Marks : 85
```

3 (a) **List the characteristics of Object-Oriented Programming (OOPs).** [03] CO4 L1

Characteristics of Object-Oriented Programming (OOPs)

Class

A class is a blueprint or template used to create objects. It defines properties (data members) and behaviors (member functions).

Object

An object is an instance of a class. It represents a real-world entity and contains actual values for the variables defined in the class.

Encapsulation

Encapsulation is the mechanism of wrapping data and methods into a single unit (class). It helps in data hiding and protects data from unauthorized access.

Abstraction

Abstraction shows only the essential features of an object while hiding the internal implementation details. It reduces complexity and increases efficiency.

(b) **Using Matplotlib, demonstrate how to create following plots with labels, legends and custom colors:** [07] CO4 L4

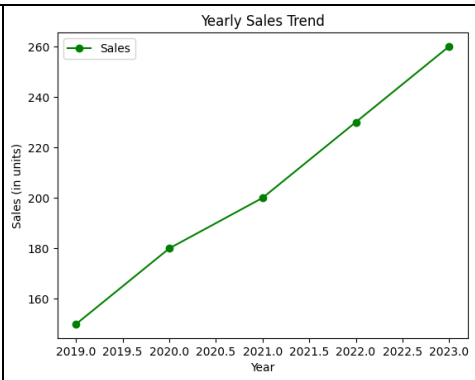
i. **Line Chart**
ii. **Bar Chart**

Line Chart using Matplotlib

Example: Plotting sales over years

```
import matplotlib.pyplot as plt

# Data
years = [2019, 2020, 2021, 2022, 2023]
sales = [150, 180, 200, 230, 260]

# Create line chart
plt.plot(years, sales, color='green', marker='o', label='Sales')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Sales (in units)')
plt.title('Yearly Sales Trend')

# Add legend
plt.legend()

# Display plot
plt.show()
```

Output:



Explanation:

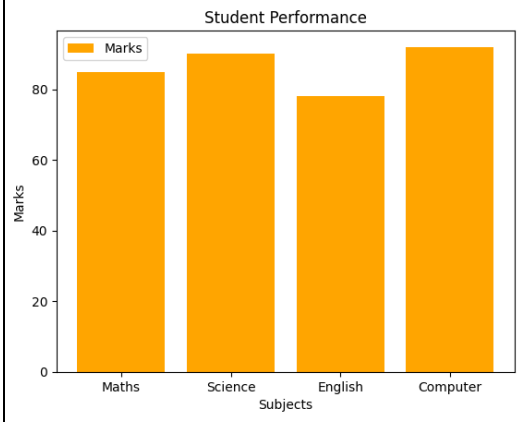
- `plt.plot()` is used to draw the line
- `color`, `marker`, and `label` customize the chart
- `xlabel`, `ylabel`, `title`, and `legend` improve readability

Bar Chart using Matplotlib

Example: Student marks in different subjects

```
import matplotlib.pyplot as plt

# Data
subjects = ['Maths', 'Science', 'English', 'Computer']
marks = [85, 90, 78, 92]

# Create bar chart
plt.bar(subjects, marks, color='orange', label='Marks')

# Add labels and title
plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.title('Student Performance')

# Add legend
plt.legend()

# Display plot
plt.show()
```

Output:



Explanation:

- `plt.bar()` creates a bar chart
- Custom color enhances visual appearance
- Labels, title, and legend make the chart easy to understand

(c)	**Write a Python program using Pandas to:** **i. Import data from a CSV file**	[10]	CO4	L4

ii. Display the first 10 rows

Describe the dataset

Python Program

```
import pandas as pd

# Import data from CSV file
data = pd.read_csv("sample_data.csv")

# Display first 10 rows of the dataset
print("First 10 rows of the dataset:")
print(data.head(10))

# Describe the dataset
print("\nDataset Description:")
print(data.describe())
```

Importing Pandas Library

```
import pandas as pd
```

- Pandas is a powerful Python library used for **data manipulation and analysis**.
- It provides data structures like **DataFrame** for handling tabular data.

Importing Data from a CSV File

```
data = pd.read_csv("sample_data.csv")
```

- `read_csv()` reads data from a **CSV (Comma Separated Values)** file.
- The data is stored in a **DataFrame**, which is similar to a table with rows and columns.

Displaying the First 10 Rows

```
data.head(10)
```

- `head(10)` displays the **first 10 rows** of the dataset.
- Useful for:
  - Understanding the structure of the data
  - Checking column names and sample values

Describing the Dataset

```
data.describe()
```

- `describe()` provides **statistical summary** of numerical columns:
  - **Count** – number of non-null values
  - **Mean** – average value
  - **Standard Deviation** – data spread
  - **Minimum & Maximum values**
  - **25%, 50%, 75% percentiles**

Importance of `describe()`

- Helps in **data analysis and decision making**
- Detects:
  - Outliers
  - Data distribution
  - Missing or incorrect values

Part B - Compulsory (01*10=10 marks) – CASE STUDY

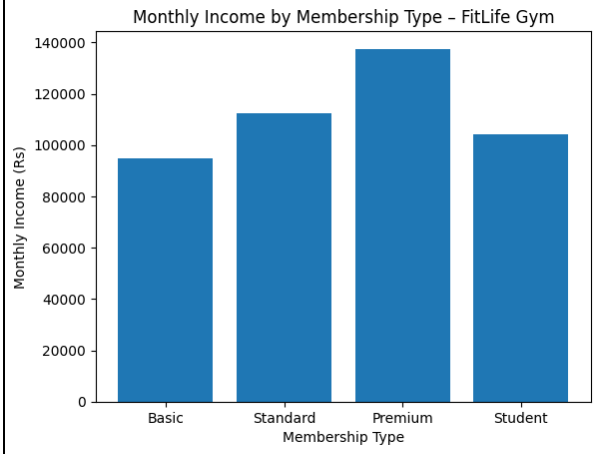
4	**Fitness Center Membership Analysis** **A fitness center "FitLife Gym" wants to analyze its monthly membership data using Python.** **Member Data:** **----------** **Membership Types: ['Basic', 'Standard', 'Premium', 'Student']** **Monthly Fee (Rs): [1000, 1500, 2500, 800]** **Total Members: [120, 90, 60, 150]** **Active Members: [95, 75, 55, 130]** **----------** **Write a Python program to:** **i. Store the membership data using suitable data structures (lists or dictionaries).** **ii. Create a function calculate_income() to compute monthly income from each membership type** ***(Monthly Fee × Active Members)*.** **iii. Create a function low_engagement() to identify membership types where active members are less than 80.** **iv. Using NumPy, calculate the average monthly income.** **v. Using Matplotlib, plot a bar chart showing income by membership type with labels and title.**	[10]	CO4	L4

Fitness Center Membership Analysis – Python Program

```
import numpy as np
import matplotlib.pyplot as plt

# i. Store membership data using dictionaries
membership_data = {
    "Basic":    {"fee": 1000, "total": 120, "active": 95},
    "Standard": {"fee": 1500, "total": 90,  "active": 75},
    "Premium":  {"fee": 2500, "total": 60,  "active": 55},
    "Student":  {"fee": 800,  "total": 150, "active": 130}
}

# ii. Function to calculate monthly income
def calculate_income(data):
    income = {}
    for mtype, details in data.items():
        income[mtype] = details["fee"] * details["active"]
    return income

# iii. Function to identify low engagement memberships
def low_engagement(data):
    low_types = []
    for mtype, details in data.items():
        if details["active"] < 80:
            low_types.append(mtype)
    return low_types

# Calculate income
income_data = calculate_income(membership_data)
```

```
# Identify low engagement membership types
low_engagement_types = low_engagement(membership_data)

# iv. Using NumPy to calculate average monthly income
income_values = np.array(list(income_data.values()))
average_income = np.mean(income_values)

# v. Plot bar chart using Matplotlib
membership_types = list(income_data.keys())
plt.bar(membership_types, income_values)
plt.xlabel("Membership Type")
plt.ylabel("Monthly Income (Rs)")
plt.title("Monthly Income by Membership Type – FitLife Gym")
plt.show()

# Output results

print("Monthly Income by Membership Type:")
for mtype, income in income_data.items():
    print(f"{mtype}: Rs {income}")

print("\nLow Engagement Membership Types (Active < 80):")
print(low_engagement_types)

print(f"\nAverage Monthly Income: Rs {average_income:.2f}")
```

Output:




```
Monthly Income by Membership Type:
Basic: Rs 95000
Standard: Rs 112500
Premium: Rs 137500
Student: Rs 104000

Low Engagement Membership Types (Active < 80):
['Standard', 'Premium']

Average Monthly Income: Rs 112250.00
```

*****