CBCS SCHEME

USN

MBABA313

Third Semester MBA Degree Examination, Dec.2025/Jan.2026
Introduction to Python Data and Control Systems

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FOUR full questions from Q.No.1 to Q.No.7.
2. Question No. 8 is compulsory.
3. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
Q.1	a.	Define variables and data types.	3	L1	CO1
	b.	Explain conditional statements with syntax.	7	L2	CO1
	c.	Write a program to check if a number is prime.	10	L3	CO1
Q.2	a.	What is a function?	3	L1	CO1
	b.	Distinguish between lists and tuples.	7	L2	CO1
	c.	Write a function to return highest-priced item from a dictionary.	10	L3	CO1
Q.3	a.	Define data cleaning.	3	L1	CO2
	b.	Explain handling of missing values.	7	L2	CO2
	c.	Describe the steps in Python required to load a CSV file containing monthly sales data and generate a bar chart to visualize the sales for each month.	10	L3	CO2
Q.4	a.	What is NumPy?	3	L1	CO2
	b.	Explain aggregation in Pandas.	7	L2	CO2
	c.	Calculate mean, median & standard deviation in Python.	10	L4	CO2
Q.5	a.	What is a class?	3	L1	CO1
	b.	Explain inheritance with an example.	7	L2	CO1
	c.	Write a Python program that accepts two numbers from the user and uses a try–except block to handle a division by zero error, ensuring that the program does not crash when the denominator is zero.	10	L3	CO4

Q.6	**a.**	Define control system.	**3**	**L1**	**CO3**
	b.	Compare open-loop and closed-loop control systems by explaining how each system operates and highlighting the key differences in terms of feedback and accuracy.	**7**	**L2**	**CO3**
	c.	Describe role of sensors, actuators & controllers.	**10**	**L3**	**CO3**
Q.7	**a.**	What is feedback?	**3**	**L1**	**CO3**
	b.	Explain how accuracy is achieved in a closed-loop control system, focusing on the role of continuous monitoring and corrective adjustments.	**7**	**L2**	**CO3**
	c.	State the standard PID controller formula used in closed-loop control systems and explain the function of each component in adjusting the system output.	**10**	**L3**	**CO3**
Q.8	**CASE STUDY : (Compulsory)** Modern organisations increasingly rely on automated decision systems for monitoring data trends, predicting outcomes, and regulating operational parameters in real time. Explain how Python can be used to integrate data acquisition, processing, and automatic control adjustments within such systems. In your answer, discuss: **Questions :**				
	a.	How Python handles data collection and cleaning?	**3**	**L5**	**CO4**
	b.	How processed data supports decision and control logic?	**7**	**L5**	**CO4**
	c.	How the control output is implemented to maintain system targets?	**10**	**L5**	**CO4**

* * * * *

CBCS SCHEME

Third Semester MBA Degree Examination, Dec. 2025 / Jan. 2026
Introduction to Python Data and Control Systems (MBABA313)

Answer Key

1a. Define variables and data types. (3 Marks)

Variables:
A variable is a named storage location in a program used to hold data that can change during execution. For example, x = 10 stores the value 10 in variable x.

Data Types:
Data types specify the kind of data a variable can store and the operations that can be performed on it. Common data types include:

- **Integer (int):** Whole numbers (e.g., 5, -3)
- **Float:** Decimal numbers (e.g., 3.14)
- **String (str):** Sequence of characters (e.g., "Hello")
- **Boolean (bool):** Logical values (True or False)

1b. Explain conditional statements with syntax. (7 Marks)

Conditional Statements:
Conditional statements are used in programming to make decisions based on certain conditions. They allow the program to execute different blocks of code depending on whether a condition is **True** or **False**.

Types of Conditional Statements:

1. **if Statement:**
   Executes a block of code if the condition is true.
   Syntax:

```
if condition:
  statements
```

2. **if-else Statement:**
   Executes one block if the condition is true, otherwise another block.
   Syntax:

```
if condition:
  statements1
else:
  statements2
```

3. **if-elif-else Statement:**
   Used to check multiple conditions.
   Syntax:

```
if condition1:
  statements1
```

```
elif condition2:
   statements2
else:
   statements3
```

4. **Nested if Statement:**
   An if statement inside another if statement.
   Syntax:

```
if condition1:
   if condition2:
      statements
```

Example:

```
num = 10

if num > 0:
   print("Positive number")
elif num == 0:
   print("Zero")
else:
   print("Negative number")
```

1c. Write a program to check if a number is prime. (10 Marks)

Program to Check if a Number is Prime (Python):

A prime number is a number greater than 1 that has no divisors other than 1 and itself.

Algorithm:

1. Input a number.
2. Check if the number is less than or equal to 1 → Not prime.
3. Otherwise, check divisibility from 2 to √n.
4. If divisible by any number → Not prime.
5. Else → Prime number.

Program:

```
num = int(input("Enter a number: "))

if num <= 1:
   print("Not a Prime Number")
else:
   for i in range(2, int(num**0.5) + 1):
      if num % i == 0:
         print("Not a Prime Number")
         break
   else:
      print("Prime Number")
```

Sample Output:

```
Enter a number: 7
Prime Number
```

2a. What is a function? (3 Marks)

Function:
A function is a reusable block of code that performs a specific task. It helps in organizing programs, reducing repetition, and improving readability.
Example (Python):

```
def greet():
  print("Hello, World!")
```

2b. Distinguish between lists and tuples. (7 Marks)

Difference between Lists and Tuples (Python):

Feature	**List**	**Tuple**
Definition	A collection of items that is **mutable** (can be changed)	A collection of items that is **immutable** (cannot be changed)
Syntax	Defined using square brackets []	Defined using parentheses ()
Modification	Elements can be added, removed, or updated	Elements cannot be modified once created
Performance	Slower due to mutability	Faster due to immutability
Methods	Many built-in methods (append, remove, sort, etc.)	Fewer methods (count, index)
Use Case	Used when data needs to be changed frequently	Used when data should remain constant
Example	list = [1, 2, 3]	tuple = (1, 2, 3)

Lists are flexible and changeable, while tuples are fixed and more efficient for storing constant data.

2c. Write a function to return highest-priced item from a dictionary. (10 Marks)

A dictionary stores items as **key-value pairs**, where the key is the item name and the value is its price.
Approach:

1. Take a dictionary as input.
2. Use a loop or built-in function to find the maximum value.
3. Return the item (key) with the highest price.

Program:

```
def highest_priced_item(items):
  max_item = max(items, key=items.get)
  return max_item, items[max_item]

# Example dictionary
products = {
  "Laptop": 75000,
  "Mobile": 30000,
```

```
  "Tablet": 25000,
  "Headphones": 5000
}

item, price = highest_priced_item(products)
print("Highest priced item:", item)
print("Price:", price)
```

Output:

Highest priced item: Laptop
Price: 75000

This function uses the max() function with key=items.get to efficiently find the item with the highest price.

3a. Define data cleaning. (3 Marks)

Data Cleaning:
Data cleaning is the process of identifying and correcting errors, inconsistencies, and missing values in a dataset to improve its quality and accuracy.
Example:
If a dataset contains missing values, duplicate records, or incorrect entries (e.g., age = -5), data cleaning involves removing duplicates, filling missing values, and correcting invalid data.
This ensures the data is accurate and suitable for analysis.

3b. Explain handling of missing values. (7 Marks)

Missing values occur when no data is stored for a variable in a dataset. Handling them is important to ensure accurate analysis and reliable results.

Methods to Handle Missing Values:

1. **Deletion Method:**
    - Remove rows or columns with missing values.
    - Used when missing data is very small.
2. **Mean/Median/Mode Imputation:**
    - Replace missing values with:
        - **Mean** (for numerical data)
        - **Median** (for skewed data)
        - **Mode** (for categorical data)
3. **Forward Fill / Backward Fill:**
    - Replace missing values with previous or next available value.
    - Common in time-series data.
4. **Interpolation:**
    - Estimate missing values based on nearby data points.
5. **Using Constant Value:**
    - Replace missing values with a fixed value (e.g., 0 or “Unknown”).
6. **Prediction Methods:**

- Use machine learning models to predict missing values based on other data.

Example:

```
import pandas as pd

data = {'Marks': [80, 90, None, 70]}
df = pd.DataFrame(data)

# Fill missing value with mean
df['Marks'].fillna(df['Marks'].mean(), inplace=True)
```

3c. Describe the steps in Python required to load a CSV file containing monthly sales data and generate a bar chart to visualize the sales for each month. (10 Marks)

Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
```

Load the CSV File

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

View Data

```
print(data)
```

Sample CSV Data (sales.csv):

```
Month,Sales
Jan,100
Feb,150
Mar,200
Apr,180
May,220
```

Output:

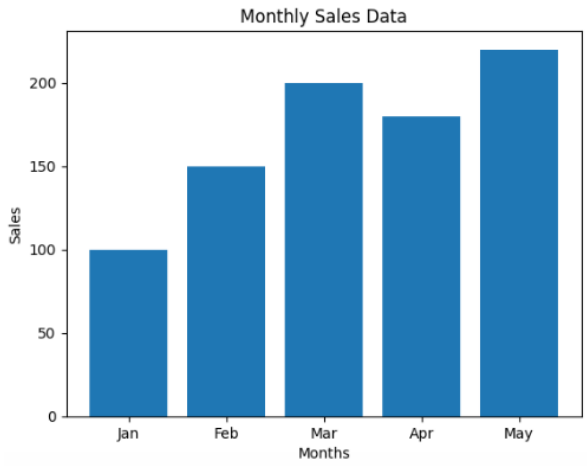
```
  Month  Sales
0   Jan    100
1   Feb    150
2   Mar    200
3   Apr    180
4   May    220
```

Extract Columns

```
months = data['Month']
sales = data['Sales']
```

Plot Bar Chart

```
plt.bar(months, sales)
plt.xlabel("Months")
plt.ylabel("Sales")
plt.title("Monthly Sales Data")
plt.show()
```



4a. What is NumPy? (3 Marks)

NumPy:
NumPy (Numerical Python) is a Python library used for numerical computing. It provides powerful support for multi-dimensional arrays and mathematical operations on them.
Example:

```
import numpy as np

arr = np.array([1, 2, 3, 4])
print(arr * 2)
```

Output:

```
[2 4 6 8]
```

4b. Explain aggregation in Pandas. (7 Marks)

Aggregation in Pandas refers to the process of applying statistical or summary functions on data to obtain a single or reduced set of values. It is mainly used to summarize large datasets.
Common Aggregation Functions:

- **sum()** – Returns total of values
- **mean()** – Returns average
- **min() / max()** – Returns minimum/maximum value
- **count()** – Counts non-null values
- **median()** – Returns middle value
- **std()** – Standard deviation

Aggregation on DataFrame:
Aggregation can be applied on columns or entire DataFrame.

```
import pandas as pd

data = {'Marks': [80, 90, 70, 60]}
df = pd.DataFrame(data)

print(df['Marks'].mean())
```

Group-wise Aggregation (groupby):
Used to perform aggregation on grouped data.

```
data = {
   'Dept': ['A', 'A', 'B', 'B'],
   'Marks': [80, 90, 70, 60]
}
df = pd.DataFrame(data)

result = df.groupby('Dept')['Marks'].mean()
print(result)
```

Output:

```
Dept
A   85.0
B   65.0
```

4c. Calculate mean, median & standard deviation in Python. (10 Marks)

Calculation of Mean, Median & Standard Deviation in Python:
These are basic statistical measures used to analyze data.
Definitions:

- **Mean:** Average of all values
- **Median:** Middle value after sorting
- **Standard Deviation:** Measure of data spread

Using Python (statistics module):

```
import statistics

data = [10, 20, 30, 40, 50]

# Mean
mean_value = statistics.mean(data)

# Median
median_value = statistics.median(data)

# Standard Deviation
std_dev = statistics.stdev(data)

print("Mean:", mean_value)
print("Median:", median_value)
print("Standard Deviation:", std_dev)
```

Output:

```
Mean: 30
Median: 30
Standard Deviation: 15.811388300841896
```

Using NumPy:

```
import numpy as np

data = np.array([10, 20, 30, 40, 50])

print("Mean:", np.mean(data))
print("Median:", np.median(data))
print("Standard Deviation:", np.std(data))
```

Output:

Mean: 30.0
Median: 30.0
Standard Deviation: 14.142135623730951

5a. What is a class? (3 Marks)

Class:

A class is a blueprint or template used to create objects in object-oriented programming. It defines properties (variables) and behaviors (methods) that the objects created from it will have.

Example:

```
class Student:
    def __init__(self, name):
        self.name = name
```

A class helps in organizing data and functions together in a structured way.

5b. Explain inheritance with an example. (7 Marks)

Inheritance:

Inheritance is a feature of Object-Oriented Programming (OOP) that allows one class (child/subclass) to acquire the properties and methods of another class (parent/superclass). It promotes code reuse and hierarchical classification.

Types (brief):

- Single inheritance
- Multiple inheritance
- Multilevel inheritance

Syntax:

```
class Parent:
    # parent properties and methods

class Child(Parent):
    # child inherits from Parent
```

Example:

```
class Animal:
    def speak(self):
        print("Animal makes a sound")

class Dog(Animal):   # Inheriting from Animal
    def bark(self):
        print("Dog barks")

# Create object
d = Dog()
d.speak()   # inherited method
d.bark()    # own method
```

Output:

```
Animal makes a sound
Dog barks
```

5c. Write a Python program that accept two numbers from the user and uses a try-except block to handle a division by zero error, ensuring that the program does not crash when the denominator is zero. (10 Marks)

Explanation:

- The program takes two numbers as input.
- It uses a try block to perform division.
- If the denominator is zero, a ZeroDivisionError occurs.
- The except block handles the error and prevents the program from crashing.

Program:

```
try:
    num1 = float(input("Enter numerator: "))
    num2 = float(input("Enter denominator: "))

    result = num1 / num2
    print("Result:", result)

except ZeroDivisionError:
    print("Error: Division by zero is not allowed!")

except ValueError:
    print("Error: Please enter valid numbers.")
```

Sample Output 1 (Valid Input):

```
Enter numerator: 10
Enter denominator: 2
Result: 5.0
```

Sample Output 2 (Division by Zero):
Enter numerator: 10
Enter denominator: 0
Error: Division by zero is not allowed!

6a. Define control system. (3 Marks)

Control System:

A control system is a system of devices or components that manages, commands, directs, or regulates the behavior of other devices or systems to achieve a desired output.

It ensures that the system output follows the desired input by controlling its operation.

6b. Compare open-loop and closed-loop control systems by explaining how each system operates and highlighting the key differences in terms of feedback and accuracy. (7 Marks)

Open-loop vs Closed-loop Control Systems:

Operation:

- **Open-loop Control System:**
  In an open-loop system, the output is not fed back to the input. The system operates only based on the given input without considering the actual output.
  Example: Electric heater, washing machine timer.
- **Closed-loop Control System:**
  In a closed-loop system, the output is continuously monitored and fed back to the input. The system compares actual output with desired output and makes corrections.
  Example: Thermostat-controlled heating system.

Key Differences:

Feature	**Open-loop System**	**Closed-loop System**
Feedback	No feedback	Uses feedback
Accuracy	Less accurate	More accurate
Error Correction	Cannot correct errors	Automatically corrects errors
Complexity	Simple design	More complex
Cost	Low cost	Higher cost
Reliability	Less reliable	More reliable

Feature	Open-loop System	Closed-loop System
Examples	Toaster, timer-based systems	Air conditioner, automatic control systems

Open-loop systems are simple and economical but less accurate, while closed-loop systems use feedback to improve accuracy and performance.

6c. Describe role of sensors, actuators & controllers. (10 Marks)

Role of Sensors, Actuators & Controllers in a Control System:
A control system consists of three main components—**sensors, controllers, and actuators**—that work together to achieve the desired output.

1. Sensors:

- Sensors are devices that **measure physical quantities** such as temperature, pressure, speed, or position.
- They convert these physical parameters into electrical signals.
- Provide **feedback** to the controller.

Example: Temperature sensor, pressure sensor

2. Controllers:

- The controller is the **brain of the system**.
- It receives input from sensors and compares it with the desired value (setpoint).
- Based on the error (difference), it decides the corrective action.

Example: Microcontroller, PID controller

3. Actuators:

- Actuators are devices that **execute the control action**.
- They convert electrical signals from the controller into physical action.

Example: Motor, valve, relay

Working Together:

1. Sensor measures the system output.
2. Controller compares it with the desired value.
3. Actuator takes action to correct the system.

Example:
In a **temperature control system**:

- Sensor measures temperature
- Controller processes the data
- Actuator (heater/cooler) adjusts temperature

Conclusion: Sensors provide data, controllers make decisions, and actuators perform actions, ensuring the system operates efficiently and accurately.

7a. What is feedback? (3 Marks)

Feedback:
Feedback in a control system is the process of taking a portion of the output signal and returning it to the input to compare with the desired value.
It helps in reducing errors and improving the accuracy and stability of the system.

7b. Explain accuracy is achieved in a closed-loop control system, focusing on the role of continuous monitoring and corrective adjustments. (7 Marks)

Accuracy in a Closed-Loop Control System:
In a closed-loop control system, accuracy is achieved through **continuous monitoring and automatic correction** of the output.
Key Points:

1. **Continuous Monitoring:**
   Sensors constantly measure the actual output of the system (e.g., temperature, speed).
2. **Comparison with Setpoint:**
   The controller compares the measured output with the desired value (setpoint).
3. **Error Detection:**
   The difference between actual output and desired output is called **error**.
4. **Corrective Action:**
   The controller generates a control signal based on the error and sends it to the actuator.
5. **Adjustment by Actuator:**
   The actuator makes necessary changes to reduce the error.
6. **Feedback Loop:**
   This process repeats continuously, ensuring the output stays close to the desired value.

Example:
In an air conditioner:

- Sensor measures room temperature
- Controller compares it with set temperature
- AC adjusts cooling to maintain accuracy

Continuous feedback, error detection, and real-time corrections make closed-loop systems highly accurate and reliable.

7c. State the standard PID controller formula used in closed-loop control systems and explain the function of each component in adjusting the system output. (10 Marks)

PID Controller Formula (Closed-Loop System):
The standard PID controller equation is:

$$u(t) = K_p\, e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt}$$

Where:

- (u(t)) = Control output
- (e(t)) = Error (difference between setpoint and actual output)
- (K_p, K_i, K_d) = Proportional, Integral, and Derivative gains

Components and Their Functions:

1. Proportional (P) Control – (K_p e(t)):

- Produces output proportional to the current error.
- Larger error → stronger corrective action.
- **Effect:** Improves response speed but may cause overshoot.

2. Integral (I) Control – (K_i \int e(t)dt):

- Considers accumulation of past errors.
- Eliminates steady-state error (offset).
- **Effect:** Improves accuracy but may slow response.

3. Derivative (D) Control – (K_d \frac{de(t)}{dt}):

- Predicts future error based on rate of change.
- Provides damping and stability.
- **Effect:** Reduces overshoot and improves system stability.

Working in Closed-Loop System:

1. Error is continuously measured using feedback.
2. PID controller calculates control action using P, I, and D terms.
3. Output is adjusted to minimize error.
4. Process repeats continuously for accurate control.

Example:

In a temperature control system:

- **P:** Reacts to current temperature difference
- **I:** Removes long-term temperature offset
- **D:** Prevents sudden temperature fluctuations

The PID controller combines proportional, integral, and derivative actions to provide fast, accurate, and stable control of system output.

Q8 – CASE STUDY (Compulsory) (10 Marks)

Modern organizations increasingly rely on automated decision systems for monitoring data trends, predicting outcomes, and regulating operational parameters in real time. Explain how python can be used to integrate data acquisition, processing and automatic control adjustments within such systems. In your answer discuss

Questions:

a. How Python handles data collection and cleaning?

b. How processed data supports decision and control logic?

c. How the control output is implemented to maintain system targets?

Python in Automated Decision Systems:

Python is widely used in modern organizations to integrate **data acquisition, processing, and control actions** due to its powerful libraries and flexibility.

a. How Python handles data collection and cleaning:

Data Collection:

- Python collects data from multiple sources such as:
  - Sensors (IoT devices using libraries like pyserial, socket)
  - Files (CSV, Excel using pandas)
  - Databases (sqlite3, MySQL)
  - APIs (requests library)

Example:

```
import pandas as pd
data = pd.read_csv("data.csv")
```

Data Cleaning:

- Python ensures data quality using:
  - Handling missing values (fillna(), dropna())
  - Removing duplicates (drop_duplicates())
  - Correcting data types (astype())
  - Filtering invalid data

Example:

```
data.fillna(method='ffill', inplace=True)
data.drop_duplicates(inplace=True)
```

Clean data is essential for accurate analysis and decision-making.

b. How processed data supports decision and control logic:

- After cleaning, data is analyzed using:
  - Statistical methods (mean(), std())
  - Machine learning models (scikit-learn)
  - Rule-based logic (if-else conditions)
- The system identifies patterns, trends, or anomalies.

Example:

```
if data['temperature'].mean() > 30:
   print("High temperature - take action")
```

Processed data helps in:

- Predicting outcomes
- Making real-time decisions
- Generating control signals
-

c. How the control output is implemented to maintain system targets:

- Python sends control signals to actuators using:
  - Microcontrollers (Arduino, Raspberry Pi via GPIO)
  - Communication protocols (MQTT, HTTP)

Example:

```
import RPi.GPIO as GPIO

GPIO.output(18, GPIO.HIGH)  # Turn ON device
```

- In advanced systems:
    - PID control algorithms are implemented in Python
    - Real-time adjustments are made based on feedback

This ensures:

- System stability
- Desired output maintenance
- Automatic correction of errors

Conclusion:

Python integrates **data collection, cleaning, analysis, and control actions** into a single system, enabling intelligent automation and real-time decision-making in modern organizations.

*****