UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 1 Examination in Engineering: March 2026

Module Number: EE1102 **Module Name: Programming Fundamentals**

Part-B

[Part-A + Part-B = 2 hours]

Programming Language: **Python**

Instructions for candidates

- Answer all questions
- Write the index number very clearly in each answer sheet.
- For Q1 answer in the seperate sheet Provided.
- For Q2 answer in the booklet provided.

Q1. Complete the following Python program by filling in the blanks in the sheet provided. Choose the correct term for each blank from the word list given.
(Note: Words may be used once, more than once, or not at all. Each correct blank is 0.5 marks.)

a) The following Python program simulates a control system for a tea leaf drying machine. The system continuously reads temperature inputs until the operator types "stop". It includes validation to ignore bad data and triggers different cooling or heating actions based on the temperature.

break	continue	elif	else	float	for	if
input	int	pass	print	range	str	while

```
print("Tea Leaf Drying Monitor Initialized.")
print("Enter 'stop' to shut down the system.")

..... (1) True:
    temp_str = ..... (2) ("Enter current temperature (C): ")

    if temp_str == "stop":
        print("Shutting down...")
        ..... (3)
```

```
if not temp_str.isdigit():
    print("Sensor error: Invalid data format.")
    (4) .....

temp = (5) .....(temp_str)

(6) ..... temp > 35:
    print("Alert: Too hot! Increasing fan speed.")
(7) ..... temp < 25:
    print("Alert: Too cold! Activating heater.")
(8) .....:
    print("Temperature is optimal.")
```

b) The following Python program simulates a secure login system for a digital vault. The user is allowed a maximum of three attempts to enter the correct 4-digit PIN (8080). The program includes input validation to check for non-numeric characters and provides feedback based on the user's input.

break	continue	elif	else	float	for	if
input	int	pass	print	range	str	while

```
max_attempts = 3
print("Vault Security System Initialized.")

for attempt in (9) .....(1, max_attempts + 1):
    pin_str = (10) .....("Enter 4-digit PIN: ")

    if not pin_str.isdigit():
        print("Error: PIN must contain only numbers.")
        (11) .....

    pin = (12) .....(pin_str)

    (13) ..... pin == 8080:
        print("Access Granted! Vault opening...")
        (14) .....
    (15) ..... attempt == max_attempts:
        print("Maximum attempts reached. Account Locked!")
    (16) .....:
```

```
        print(f"Incorrect PIN. You have {max_attempts - attempt}
    tries left.")
```

c) The Python program below uses the `turtle` library to draw a symmetrical fractal tree. It defines a recursive function called `branch` that draws a line, turns, and then calls itself to draw progressively smaller sub-branches until a base case is reached.

backward	branch	def	done	for	forward	if
import	left	range	return	right	turtle	while

```
(17)..... turtle

(18)..... branch(length):
    # Base case to stop recursion
    (19)..... length < 10:
        (20).....

    turtle.(21).....(length)
    turtle.left(30)

    # Recursive call for the left sub-branch
    (22).....(length * 0.7)

    turtle.right(60)

    # Recursive call for the right sub-branch
    branch(length * 0.7)

    # Return to the original branch split position
    turtle.left(30)
    turtle.(23).....(length)

# Setup turtle and draw the fractal
turtle.speed(0)
turtle.left(90)
branch(100)
turtle.done()
```

d) The Python program below uses the `numpy` and `matplotlib.pyplot` libraries to generate and display a continuous sine wave. The x-axis values are mathematically generated to span from -2π to 2π.

arange	cos	figure	import	legend	linspace	plot
print	scatter	show	sin	title	xlabel	ylabel

```
(24) matplotlib.pyplot as plt
import numpy as np

# Generate 100 evenly spaced x values from -2*pi to 2*pi
x = np.(25)(-2 * np.pi, 2 * np.pi, 100)

# Calculate the corresponding y values using the sine function
y = np.(26)(x)

# Create a continuous line graph of x and y
plt.(27)(x, y, color='blue', label='Waveform')

# Label the horizontal axis
plt.(28)("Angle in Radians")
plt.ylabel("Amplitude")

# Add a header to the top of the graph
plt.(29)("Sine Wave Plot")

# Activate the legend to show the label
plt.legend()

# Render and display the generated plot on the screen
plt.(30)()
```

Q2. a) Write a Python program that prints a formatted data table for the mechanical properties of materials given below. You must use **f-strings** to create a header row and align the data.

- The `Material` column should be 12 characters wide and left-aligned.
- The `Force (N)` column should be 10 characters wide and right-aligned.
- The `Strain` column should be 10 characters wide and right-aligned, with the value shown to 4 decimal places.

Data:

```
test_data = [
    ("Steel", 4500, 0.00215),
    ("Aluminum", 2100, 0.00489),
    ("Copper", 3200, 0.00312)
]
```

Output:

```
Material        Force (N)      Strain
---------------------------------------
Steel                4500      0.0021
Aluminum             2100      0.0049
Copper               3200      0.0031
```

[3 marks]

b) Write a Python function named `print_box(width, height)` that takes two integers as parameters. The function should print a hollow rectangular box of those dimensions using the `#` character, as shown in the example for an input of `(5, 4)`.

```
#####
#   #
#   #
#####
```

[3 marks]

c) Write a Python function named `group_equipment(items)` that takes a list of tuples as a parameter. Each tuple contains two strings: a laboratory name and an equipment name, formatted as `(lab_name, equipment)`.

The function must process this list and **return a dictionary** where each key is a unique `lab_name`, and its corresponding value is a **list** of all equipment assigned to that lab.

Example Input:

```
data = [
    ("Fluids Lab", "Pump"),
    ("Materials Lab", "Tensile Tester"),
    ("Fluids Lab", "Flow Meter"),
    ("Materials Lab", "Hardness Tester")
]
```

Expected Return Value:

```
{
    'Fluids Lab': ['Pump', 'Flow Meter'],
    'Materials Lab': ['Tensile Tester', 'Hardness Tester']
}
```

[3 marks]

d) Write a complete Python program using `numpy` and `matplotlib.pyplot` to generate the exact polynomial plot shown in the Figure Q2.d.

Your code must satisfy the following requirements:

- Generate an array of 100 x values ranging from -3 to 3.
- Calculate the corresponding y values for the cubic polynomial $y = x^3 - 4x$.

- Plot the function as a **solid blue line** and include a label for the legend.
- Add the title **"Cubic Function Plot"**.
- Label the axes as **"X-axis"** and **"Y-axis"**.
- Display the grid and the legend on the plot.

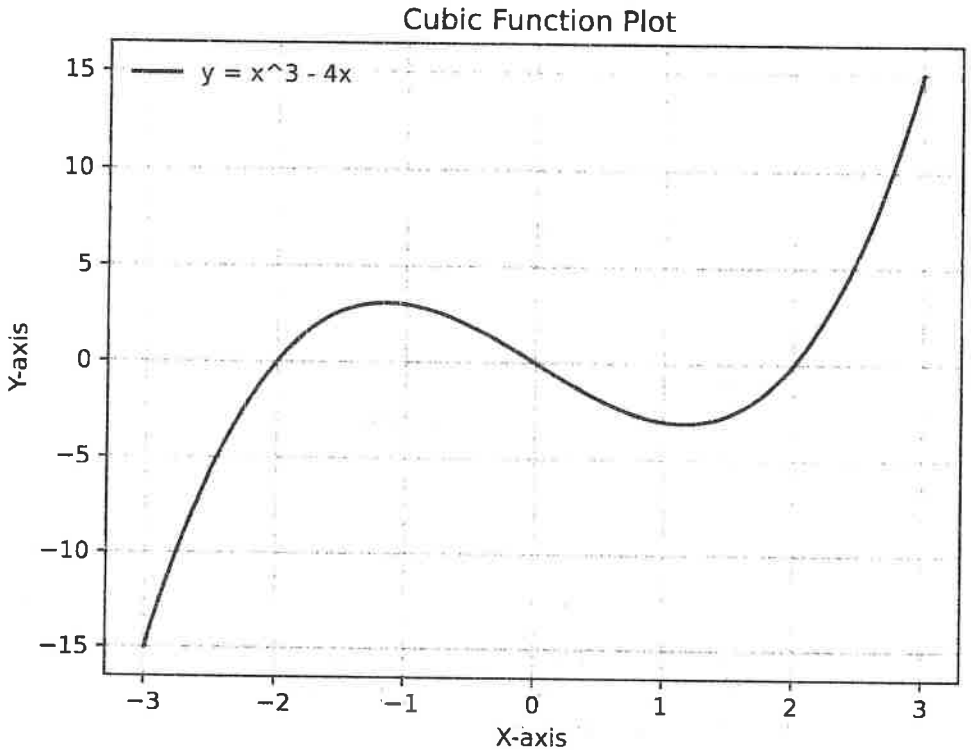

Figure Q2.d: Plot generated by the given Requirements.

[3 marks]

e) The following Python program is intended to calculate the kinetic energy of an object ($KE = \frac{1}{2}mv^2$) but contains **6 errors**.

Identify and clearly describe each of the 6 errors (0.5 marks each). You must state the line number and the error.

```
1 def calc_kinetic_energy(mass, velocity)
2     energy = 0.5 * mass * velocity ^ 2
3   return Energy
4
5 m = input("Enter mass in kg: ")
6 v = float(input("Enter velocity in m/s: "))
7
8 result = calc_kinetic_energy(m, v)
9 print("Kinetic Energy: " + result)
```

[3 marks]