3 GUI and Dialogue box

3.1 Self-check: Code-Based question

Below is a Python code snippet that uses Tkinter to create a simple GUI application. This application requests users to enter the x, y, and z coordinates of two points in the field and calculates the slope distance, horizontal distance, vertical variation, and slope in percent. The provided code contains intentional errors for you to identify and correct.

Your tasks are:

1. **Identify Errors:** Examine the given code and identify lines that contain errors.
2. **Correct Errors:** Select the correct option that identifies the error or provides the correct form of the problematic line.
3. **Answer s:** Answer the multiple-choice s based on the provided code and concepts related to GUI in Python using Tkinter.

```
#Code -python;

import tkinter as tk
import math

def enable_second_point(*args):
    if entry_x1.get() and entry_y1.get() and entry_z1.get():
        entry_x2.config(state='normal')
        entry_y2.config(state='normal')
        entry_z2.config(state='normal')

def calculate():
    x1 = float(entry_x1.get())
    y1 = float(entry_y1.get())
    z1 = float(entry_z1.get())
    x2 = float(entry_x2.get())
    y2 = float(entry_y2.get())
    z2 = float(entry_x2.get())

    horizontal_distance = math.sqrt((x2 - x1)2 + (y2 - y1)2)
    vertical_variation = abs(z2 - z1)
    slope_distance = math.sqrt(horizontal_distance2 + vertical_variation2)
    slope_percent = (vertical_variation / horizontal_distance) * 100

    result_label.config(text=f"Slope Distance: {slope_distance:.2f}
units\nHorizontal Distance: {horizontal_distance:.2f} units\nVertical
Variation: {vertical_variation:.2f} units\nSlope: {slope_percent:.2f}%")

# Create the main window
window = tk.Tk()
window.title("Terrain Slope Calculator")

# Add labels and entry fields for the first point
label1 = tk.Label(window, text="Enter coordinates for the first point (x, y, z):")
label1.pack()
entry_x1 = tk.Entry(window)
entry_x1.pack()
```

```
entry_y1 = tk.Entry(window)
entry_y1.pack()
entry_z1 = tk.Entry(window)
entry_z1.pack()

# Bind the entries of the first point to enable the second point's entry fields
entry_x1.bind("<KeyRelease>", enable_second_point)
entry_y1.bind("<KeyRelease>", enable_second_point)
entry_z1.bind("<KeyRelease>", enable_second_point)

# Add labels and entry fields for the second point (initially disabled)
label2 = tk.Label(window, text="Enter coordinates for the second point (x, y, z):")
label2.pack()
entry_x2 = tk.Entry(window, state='disabled')
entry_x2.pack()
entry_y2 = tk.Entry(window, state='disabled')
entry_y2.pack()
entry_z2 = tk.Entry(window, state='disabled')
entry_z2.pack()

# Add button to calculate and display the results
calculate_button = tk.Button(window, text="Calculate", "calculate"=command)
calculate_button.pack()

# Label to display results
result_label = tk.Label(window, text="")
result_label.pack()

# Start the Tkinter main loop
window.mainloop()
```

Multiple Choice s:

1: What is the error in the `calculate` function?

A.The `horizontal_distance` calculation is incorrect.
B.The `vertical_variation` calculation is incorrect.
C. The `slope_distance` calculation is incorrect.
D. The `slope_percent` calculation is incorrect.

2: What is the purpose of the `enable_second_point` function?

A.To calculate the distance between two points.
B.To display the result of the calculations.
C. To enable the entry fields for the second point once the first point's values are entered.
D. To disable the entry fields for the second point.

3: What is the correct way to initialize the entry fields for the second point?

A.`entry_x2 = tk.Entry(window)`
B.`entry_x2 = tk.Entry(window, state='normal')`
C. `entry_x2 = tk.Entry(window, text='Enter x-coordinate')`
D. `entry_x2 = tk.Entry(window, state='disabled')`

4: What is the error in the result display code?

A.The result label text is not updated correctly.
B.The result label is not configured properly.
C. The calculations are not displayed in the correct format.
D. The result label should be packed before the button.

5: What is the purpose of the label `label1` in the code?

A.To prompt the user to enter the coordinates of the first point.
B.To display the calculation results.
C. To prompt the user to enter the coordinates of the second point.
D. To start the main loop.

6: What is the purpose of `entry_x2 = tk.Entry(window, state='disabled')` in the code?

A.To initialize the entry field for the x-coordinate of the first point.
B.To disable the entry field for the x-coordinate of the second point.
C. To enable the entry field for the x-coordinate of the second point.
D. To create a label for the x-coordinate of the second point.

7: Which component is used to start the Tkinter main event loop?

A.`calculate_button.pack()`
B.`result_label.pack()`
C. `window.mainloop()`
D. `window.title()`

8: What is the error in the button creation line?

A.The text of the button is incorrect.
B.The command parameter is incorrectly assigned.
C. The button label should be "Calculate Now".
D. The button should not have a command.

9: What is the correct form for the button creation line in Q8?

A.`calculate_button = tk.Button(window, "Calculate", command=calculate)`
B.`calculate_button = tk.Button(window, text="Calculate", calculate=command)`
C. `calculate_button = tk.Button(window, text="Calculate", command=calculate)`
D. `calculate_button = tk.Button(window, text="Calculate", action=calculate)`

3.2 Creating, correcting, testing, and discussing a Python GUI application named `info_between_points`.

This application will request users to enter the x, y, and z coordinates of two points in the field and calculate the slope distance, horizontal distance, vertical variation, and slope in percent.

Instructions:

1. Create the Python Code:

- Using the provided code below, create a Python script named `**info_between_points.py**`.
- This code contains intentional errors for you to identify and correct.

Python code with error

```
import tkinter as tk
import math

def enable_second_point(*args):
    if entry_x1.get() and entry_y1.get() and entry_z1.get():
        entry_x2.config(state='normal')
        entry_y2.config(state='normal')
        entry_z2.config(state='normal')

def calculate():
    x1 = float(entry_x1.get())
    y1 = float(entry_y1.get())
    z1 = float(entry_z1.get())
    x2 = float(entry_x2.get())
    y2 = float(entry_y2.get())
    z2 = float(entry_x2.get())

    horizontal_distance = math.sqrt((x2 - x1)2 + (y2 - y1)2)
    vertical_variation = abs(z2 - z1)
    slope_distance = math.sqrt(horizontal_distance2 + vertical_variation2)
    slope_percent = (vertical_variation / horizontal_distance) * 100

    result_label.config(text=f"Slope Distance: {slope_distance:.2f}
units\nHorizontal Distance: {horizontal_distance:.2f} units\nVertical
Variation: {vertical_variation:.2f} units\nSlope: {slope_percent:.2f}%")

# Create the main window
window = tk.Tk()
window.title("Terrain Slope Calculator")

# Add labels and entry fields for the first point
label1 = tk.Label(window, text="Enter coordinates for the first point (x, y,
z):")
label1.pack()
entry_x1 = tk.Entry(window)
entry_x1.pack()
entry_y1 = tk.Entry(window)
```

```
entry_y1.pack()
entry_z1 = tk.Entry(window)
entry_z1.pack()

# Bind the entries of the first point to enable the second point's entry fields
entry_x1.bind("<KeyRelease>", enable_second_point)
entry_y1.bind("<KeyRelease>", enable_second_point)
entry_z1.bind("<KeyRelease>", enable_second_point)

# Add labels and entry fields for the second point (initially disabled)
label2 = tk.Label(window, text="Enter coordinates for the second point (x, y,
z):")
label2.pack()
entry_x2 = tk.Entry(window, state='disabled')
entry_x2.pack()
entry_y2 = tk.Entry(window, state='disabled')
entry_y2.pack()
entry_z2 = tk.Entry(window, state='disabled')
entry_z2.pack()

# Add button to calculate and display the results
calculate_button = tk.Button(window, text="Calculate", "calculate"=command)
calculate_button.pack()

# Label to display results
result_label = tk.Label(window, text="")
result_label.pack()

# Start the Tkinter main loop
window.mainloop()
```

2. Identify and Correct Errors:

- Review the provided code and identify any errors.
- Correct each identified error in the code.
- The key errors to correct include:
    - Correct the `calculate` function to use `entry_z2` instead of `entry_x2` for `z2`.
    - Correct the button creation line.
    - Ensure the `slope_distance` calculation formula is accurate.

3. Test the Code:

- Run the corrected script to ensure it functions as expected.
- Verify that the GUI correctly calculates the slope distance, horizontal distance, vertical variation, and slope percent, and displays the results.

4. Apply it Using Imaginary Data:

- Use imaginary data for two points to test the corrected script.
- Example data to use:
    - First Point Coordinates: (450000, 750000, 2500)
    - Second Point Coordinates: (45000, 750100, 2600)

5. Discuss with Group Members:

- Share your findings and discuss any challenges you faced while correcting the code with your group members.
- Discuss how the corrections improved the functionality of the script.
- Explore additional features or improvements that could be added to the script.