Question 1
List the steps to create a GUI application using Tkinter (2024 may)

```
import tkinter as tk

# Create main window
root = tk.Tk()
root.title("Simple App")

# Create a label and an entry box
label = tk.Label(root, text="Enter your name:")
label.pack()

entry = tk.Entry(root)
entry.pack()

# Function to update the label
def say_hello():
    label.config(text=f"Hello, {entry.get()}!")

# Create a button
button = tk.Button(root, text="Click Me", command=say_hello)
button.pack()

# Run the app
root.mainloop()
```

Question 2

Describe the methods and Python modules commonly used for converting a color image to a grayscale image.

1. Using OpenCV (`cv2`)

Method: OpenCV provides a built-in function to convert a color image to grayscale using the `cv2.cvtColor()` function.

```
import cv2



# Read the color image

image = cv2.imread("image.jpg")



# Convert to grayscale

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Save and display the grayscale image

cv2.imwrite("gray_image.jpg", gray_image)

cv2.imshow("Grayscale Image", gray_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

2. Using PIL (Pillow)

Method: The `convert('L')` method of the `PIL.Image` module is used to convert an image to grayscale.

```
from PIL import Image


# Open the image

image = Image.open("image.jpg")


# Convert to grayscale

gray_image = image.convert("L")


# Save and show the image

gray_image.save("gray_image.jpg")

gray_image.show()
```

Question 3

How do you display an image with a caption using Python in a graphical interface? (2024 may)

To display an image with a caption in a graphical interface using Python, you can use GUI libraries like **Tkinter** or **PyQt**.

1. **Using Tkinter:**
    - Create a main application window.
    - Load the image using the **Pillow (PIL) library** and convert it to a format compatible with Tkinter.
    - Display the image using a **Label widget**.

- Add another **Label widget** below the image to show the caption.
- Use `mainloop()` to keep the window running.

2. **Using PyQt:**
   - Create a **QWidget** as the main application window.
   - Load the image using **QPixmap**, which is PyQt's way of handling images.
   - Display the image inside a **QLabel**.
   - Add another **QLabel** below it to show the caption.
   - Arrange everything using a **vertical layout (QVBoxLayout)** and show the window.

Both methods allow you to display an image with a caption in a simple graphical interface, with **Tkinter** being easier for beginners and **PyQt** offering more flexibility and modern UI elements.

Question 4

Comment on event driven programming.(2024 jan)

Event-driven programming is a programming paradigm where the flow of execution is determined by events such as user inputs (clicks, key presses), sensor outputs, or messages from other programs. Instead of executing code sequentially, the program responds to events as they occur.

Key Features:

1. **Event Handlers:**
   - Functions or methods that execute in response to an event (e.g., a button click triggering a function).
2. **Event Loop:**
   - A continuous loop that listens for events and dispatches them to the appropriate handler.
3. **Callbacks:**
   - Functions that are registered to be executed when an event occurs.
4. **Asynchronous Execution:**
   - Events may happen at unpredictable times, making this approach useful for interactive applications.

Question 5
Discuss on the types of window components and their functions.

1. Window (Main Container)

- The **main application window** that serves as the primary container for all other components.
- Example: The main frame in Tkinter or `QMainWindow` in PyQt.

2. Labels

- **Function:** Display static text or images.
- Used for providing instructions, titles, or messages to users.
- Example: `Label` in Tkinter, `QLabel` in PyQt.

3. Entry (Text Box)

- **Function:** Allows users to input text.
- Often used for forms, search bars, or user authentication fields.
- Example: `Entry` in Tkinter, `QLineEdit` in PyQt.

4. Buttons

- **Function:** Execute a function when clicked.
- Used for submitting forms, opening dialogs, or triggering actions.
- Example: `Button` in Tkinter, `QPushButton` in PyQt.

5. Checkboxes

- **Function:** Allow users to select multiple options from a list.
- Example: Used in preference settings or selecting multiple items.
- Example: `Checkbutton` in Tkinter, `QCheckBox` in PyQt.

6. Radio Buttons

- **Function:** Allow users to select **only one** option from multiple choices.
- Example: Selecting gender (Male/Female/Other) in a form.
- Example: `Radiobutton` in Tkinter, `QRadioButton` in PyQt.

Question 6
List any three image processing Python libraries.(2022 June)

1. OpenCV (`cv2`)

- **Features:**
  1.Supports a wide range of image processing functions (filtering, transformations, edge detection).
  2.Used in computer vision tasks like face recognition, object detection, and motion tracking.
  3.Highly optimized and supports real-time processing.
- **Installation:** `pip install opencv-python`

2. PIL (Pillow - Python Imaging Library)

- **Features:**
- 1. Simple and lightweight for basic image manipulation.
  2.Supports image opening, resizing, cropping, filtering, and conversion.
  3.Works well for web-based applications that require image processing.
- **Installation:** `pip install pillow`

3. Scikit-Image (`skimage`)

- **Features:1**
- 1. Advanced image processing functionalities (feature extraction, segmentation, and transformations).
- 2.Built on top of NumPy and SciPy, making it efficient for scientific computing.
  3. Used in research and medical imaging applications.
- **Installation:** `pip install scikit-image`

Question 7

Explain basic image processing with inbuilt functions in Python(2022 June)

1. Reading and Displaying an Image

- **Library:** `PIL (Pillow)`
- **Function Used:** `Image.open()`, `show()`

Example :

from PIL import Image

image = Image.open("example.jpg") # Load image
image.show() # Display image

2. Converting an Image to Grayscale

- **Library:** `OpenCV`
- **Function Used:** `cv2.cvtColor()`

Example:

```
import cv2

image = cv2.imread("example.jpg")  # Load image
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
cv2.imshow("Grayscale Image", gray_image)  # Display image
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3. Resizing an Image

- **Library:** `PIL (Pillow)`
- **Function Used:** `resize()`

Example:

```
from PIL import Image

image = Image.open("example.jpg")  # Load image
```

```
resized_image = image.resize((200, 200))  # Resize to 200x200
pixels
resized_image.show()  # Display resized image
```

4. Cropping an Image

- **Library:** `PIL (Pillow)`
- **Function Used:** `crop()`

`Example:`

```
image = Image.open("example.jpg")
cropped_image = image.crop((50, 50, 200, 200))  # Crop from
(x1, y1, x2, y2)
cropped_image.show()
```

Question 5

What are the advantages of GUI based programs over terminal based programs.(

1. Ease of Use

1.**Intuitive and Visual**

- GUI applications use icons, buttons, and menus, making them easier to navigate.
- No need to remember commands like in terminal-based programs.

2. User-Friendly for Non-Technical Users

- Ideal for beginners who are unfamiliar with command-line syntax.

2. Better User Experience (UX)

1. Interactive & Engaging

- GUI programs provide better interaction using buttons, sliders, text boxes, and images.
- Users get real-time feedback (e.g., progress bars, tooltips, animations).

2. Multi-Tasking & Accessibility

- Multiple windows can be opened simultaneously for better workflow.
- Accessibility features like screen readers, zoom, and color contrast can be implemented.

3. No Need to Remember Commands

1. Point-and-Click Interface

- Users can execute tasks with clicks instead of typing long commands.

- Dropdown menus and buttons provide options without needing to memorize commands.

4. More Attractive & Professional Appearance

1.**Customizable UI**

- GUI apps can have different themes, colors, and layouts, making them visually appealing.
- Used in most commercial software to provide a professional look.

5. Supports Multimedia & Graphics

1.**Image, Video, and Graph Support**

- Unlike terminal-based programs, GUI apps can handle images, graphs, and videos efficiently.
- Example: Photoshop, video players, and web browsers rely on GUI interfaces.

6. Easier Data Visualization

1.**Graphs, Charts, and Tables**

- GUI apps can display complex data in charts, graphs, and tables, making it easier to analyze.
- Example: Excel, stock market applications, and dashboard applications.

Question 6

What is meant by abstraction mechanism in programming? Give one example abstraction mechanism in Python(2023)

Abstraction is a fundamental concept in programming that **hides complex implementation details** and exposes only the necessary functionality to the user. It helps in reducing complexity, improving code readability, and enhancing maintainability.

Types of Abstraction Mechanisms:

1. Procedural Abstraction – Using functions to hide implementation details.
2. Data Abstraction – Using classes and objects to encapsulate data.
3. Control Abstraction – Using loops and conditionals instead of repetitive code.

Abstraction in Python helps in **hiding unnecessary details** while exposing only the required functionality. Abstract classes ensure **a structured approach** to designing programs
Example:

```
from abc import ABC, abstractmethod

# Abstract Class
class Shape(ABC):
    @abstractmethod
    def area(self):
        pass  # Abstract method
```

```
# Concrete Class (Implements Abstract Class)
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius  # Implements the abstract method

# Using the concrete class
circle = Circle(5)
print("Area of Circle:", circle.area())  # Output: Area of Circle: 78.5
```

Questions 7

Describe two fundamental differences between terminal-based user interfaces and GUIs.

1. Input & Interaction

- **TUI:** Users interact through **text-based commands** typed in a terminal (e.g., `cd Documents/` to change directories).
- **GUI:** Users interact through **visual elements** like buttons, icons, and menus, usually with a mouse and keyboard.

2. Efficiency & Usability

- **TUI:** Often **faster** for experienced users because commands can automate tasks efficiently. However, it has a **steeper learning curve**.
- **GUI:** Easier for beginners since it provides **intuitive, visual navigation**, but it may be **slower** for repetitive tasks.

Question 8

What are the attributes of a window? how the attribute value is changed?(may 2023)

1. **Title** – Sets the text displayed in the title bar of the window.
2. **Size (Geometry)** – Defines the width and height of the window.
3. **Resizable** – Determines whether the user can resize the window (True or False).
4. **Background Color** – Changes the background color of the window.
5. **Transparency (Alpha)** – Controls the transparency level of the window, with values between 0.0 (fully transparent) and 1.0 (fully opaque).
6. **Fullscreen Mode** – Makes the window take up the entire screen.
7. **Topmost (Always on Top)** – Keeps the window always on top of other windows.
8. **Minimized or Maximized State** – Sets whether the window starts minimized or maximized.
9. **Window Position** – Defines where the window appears on the screen by setting its x and y coordinates.