

# Python-Module-3-Important-Topics

🔗 For more notes visit

<https://rtpnotes.vercel.app>

- Python-Module-3-Important-Topics
  - 1. Graphics
  - 2. Turtle graphics
    - Turtle Examples
  - 3. Image Processing
    - Image formats
    - The images Module
    - Examples
      - Converting an image to black and white
  - 4. Graphical User Interfaces
    - What are GUIs
    - Event driven programming
    - Template for GUI
    - Example Program - Calculating Square root
- Python-Module-3-University-Questions-Part-A
  - 1. List any three image processing Python libraries.
  - 2. List the steps to create a GUI application using Tkinter
    - Hello world program
  - 3. Illustrate the function of following methods in turtle
    - setheading
    - Forward
    - Left
  - 4. Describe two fundamental differences between terminal-based user interfaces and GUI
- Python-Module-3-University-Questions-Part-B

- 1. How to draw a star shape using turtle in Python.
- 2. Explain basic image processing with inbuilt functions in Python.
- 3. Write Python GUI program to take the birth date and output the age when a button is pressed
- 4. How do you display an image in Python GUI?.
- 5. What are the attributes of a turtle object
- 6. Write a GUI-based program that allows the user to convert amount in Indian Rupees to amount in Euro
- 7. What are the attributes of a window? How the attributes value can be changed?
- 8. Comment on event driven programming.
- 9. Write a Python program to convert a color image to a grayscale image
- 10. Write Python GUI program to input two strings and output a concatenated string when a button is pressed.
- 11. Discuss on the types of window components and their functions.
- 12. Write a GUI-based program that allows the user to convert temperature values between degrees Fahrenheit and degrees Celsius.

## 1. Graphics

- The representation and display of geometric shapes in two-and three- dimensional space, as well as image processing.



## 2. Turtle graphics

### Tip

You can try the turtle code online here <https://pythonsandbox.com/turtle>

- A Turtle graphics toolkit provides a simple and enjoyable way to draw pictures in a window and gives you an opportunity to run several methods with an object.

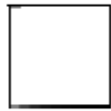
## Turtle Examples

**Lets draw a square using turtle**

```
import turtle
t = turtle.Turtle()
for i in range(4):
    t.forward(50)
    t.right(90)
t.hideturtle()
```

- First we import turtle module
- We need to store the turtle in a variable t
- This `t` will now represent the turtle what we are going to move
- The loop does these things 4 times
  - Move the turtle forward
  - Rotate the turtle right
- At last, we will hide the turtle

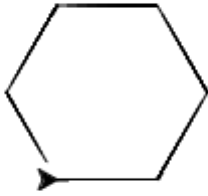
## Output



## Lets draw a hexagon using turtle

### Program

```
import turtle
t = turtle.Turtle()
for i in range(6):
    t.forward(50)
    t.left(60)
```



- Here i have not hidden the turtle, so you can see what it looks like



## 3. Image Processing

### Image formats

- Jpeg
  - Joint Potographic Expert Group
  - Some of the image information is lost on decompression
  - Support 16 million colors and mostly used in photographs
- GIF
  - Use looseless compression algorithm
  - No image information is lost on decompression
  - Support only 256 colors
  - Support Animations

### The images Module

- The images module is an open-source Python tool.
- The images module includes a class named Image.
- The Image class represents an image as a two-dimensional grid of RGB values (Red, Green, Blue).

### Examples

#### Converting an image to black and white

#### Algorithm

- For each pixel, the algorithm computes the average of the red, green, and blue values.
- The algorithm then resets the pixel's color values to 0 (black) if the average is closer to 0.
- To 255 (white) if the average is closer to 255

## Program


```
from images import Image
def blackAndWhite(image):
    blackPixel = (0,0,0)
    whitePixel = (255,255,255)
    for y in range(image.getHeight()):
        for x in range(image.getWidth()):
            (r,g,b) = image.getPixel(x,y)
            average = (r+g+b)
            if average < 128:
                image.setPixel(x,y,blackPixel)
            else:
                image.setPixel(x,y,whitePixel)

def main(filename = "cat.png"):
    image = Image(filename)
    print("Close the image window to continue.")
    image.draw()
    blackAndWhite(image)
    print("Close the image window to quit")
    image.draw()
    image.save("cat.png")

if __name__ == "__main__":
    main()
```



## 4. Graphical User Interfaces

 [To try the programs](#)

Install breezypythongui

<https://lambertk.academic.wlu.edu/files/breezypythongui/breezypythongui.zip>

Put the python file in the same location as your file

## What are GUIs

- Displays text/images
- A GUI Program is event driven
- Its inactive until user interacts GUI components
- Enter inputs in any order

## Event driven programming

- Type of programming used to create user-generated events, processing them, displaying results is called event driven programming
  - GUI based programs are almost always object based
1. Define a new class to represent the main application window
  2. Instantiate the classes of window components such as labels, fields and command buttons
  3. Position components in the window
  4. Register a method with each window component in which an event relevant to application might occur
  5. Define methods to handle the events
  6. Define a main function that instantiates the window class and runs

## Template for GUI

```
from breezypythongui import EasyFrame

class ApplicationName(EasyFrame):
    __init__ method definition

def main():
    ApplicationName().mainloop()
```

```
if __name__ == "__main__":  
    main()
```

## Example Program - Calculating Square root

```
from breezypythongui import EasyFrame  
import math  
  
class NumberFieldDemo(EasyFrame):  
    """  
    This class creates a simple graphical user interface (GUI) application  
    that allows users to enter an integer and calculate its square root.  
    """  
  
    def __init__(self):  
        """  
        This method initializes the GUI elements of the application.  
        """  
        EasyFrame.__init__(self, title="Number Field Demo") # Set window title  
  
        # Create a label to instruct the user  
        self.addLabel(text="An integer", row=0, column=0)  
  
        # Create an input field for the user to enter an integer  
        self.inputField = self.addIntegerField(value=0, row=0, column=1,  
width=10)  
  
        # Create a label to display "Square root"  
        self.addLabel(text="Square root", row=1, column=0)  
  
        # Create an output field to display the calculated square root  
        self.outputField = self.addFloatField(value=0.0, row=1, column=1,  
width=8, precision=2)  
  
        # Create a button to trigger the square root calculation  
        self.addButton(text="Compute", row=2, column=0, columnspan=2,  
command=self.computeSqrt)  
  
    def computeSqrt(self):  
        """  
        This method is called when the "Compute" button is clicked.
```

It retrieves the entered number, calculates its square root, and displays the result in the output field.

"""

```
number = self.inputField.getNumber() # Get the number from the input
field
result = math.sqrt(number) # Calculate the square root of the number
self.outputField.setNumber(result) # Display the result in the output
field

# Create an instance of the NumberFieldDemo class and run the GUI
application
NumberFieldDemo().mainloop()
```



## Python-Module-3-University-Questions-Part-A

### 1. List any three image processing Python libraries.

- OpenCV
- Scikit-image
- Pillow



### 2. List the steps to create a GUI application using Tkinter

```
from breezypythongui import EasyFrame

class ApplicationName(EasyFrame):
    __init__ method definition

def main():
    ApplicationName().mainloop()

if __name__ == "__main__":
    main()
```



# Hello world program

```
from breezypythongui import EasyFrame

class LabelDemo(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self)
        self.addLabel(text = "Hello World!", row = 0, column = 0)

def main():
    LabelDemo().mainloop()

if __name__ == "__main__":
    main()
```



## 3. Illustrate the function of following methods in turtle

i) turtle.setheading(0) ii) turtle.forward(50) iii) turtle.left(90)

### setheading

- This method sets the orientation of the turtle to the specified angle in degrees.
- An angle of 0 degrees means the turtle will face "east" or to the right.
- This is useful for resetting the direction of the turtle to a known reference point.

### Example

```
import turtle

# Create a screen and a turtle
screen = turtle.Screen()
t = turtle.Turtle()

# Step 1: Set heading to 0 degrees (face right)
t.setheading(0)
```

## Output



## Example

Lets set the heading to 90

```
t.setheading(90)
```

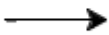
## Output



## Forward

- This method moves the turtle forward by the specified number of units (pixels).
- If the turtle is facing right (0 degrees), it will move to the right by the given distance.

```
turtle.forward(50)
```



## Left

- This method turns the turtle to the left by the specified number of degrees.
- Turning left by 90 degrees will make the turtle face "north" or up from its current position.

```
t.left(90)
```



## 4. Describe two fundamental differences between terminal-based user interfaces and GUI

### 1. Interaction Style and Event Handling:

- **Terminal-Based:** Sequential input prompts; users must follow a strict order and can't easily change inputs once entered.
- **GUI:** Event-driven; users interact in any order, triggering responses from the program.

### 2. User Experience and Convenience:

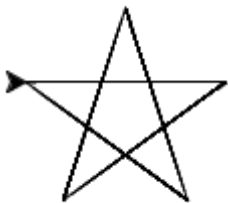
- **Terminal-Based:** Rigid input sequence; requires restarting to change inputs.
- **GUI:** Intuitive, graphical interaction; allows easy modification of inputs without restarting.



## Python-Module-3-University-Questions-Part-B

### 1. How to draw a star shape using turtle in Python.

```
import turtle
star_turtle = turtle.Turtle()
for _ in range(5):
    star_turtle.forward(100) # Move forward by 100 units
    star_turtle.right(144)   # Turn right by 144 degrees
```



### 2. Explain basic image processing with inbuilt functions in Python.

Image Method	What It Does
<code>i = Image(filename)</code>	Loads and returns an image from a file with the given filename. Raises an error if the filename is not found or the file is not a GIF file.
<code>i = Image(width, height)</code>	Creates and returns a blank image with the given dimensions. The color of each pixel is transparent, and the filename is the empty string.
<code>i.getWidth()</code>	Returns the width of <code>i</code> in pixels.
<code>i.getHeight()</code>	Returns the height of <code>i</code> in pixels.
<code>i.getPixel(x, y)</code>	Returns a tuple of integers representing the RGB values of the pixel at position (x, y).
<code>i.setPixel(x, y, (r, g, b))</code>	Replaces the RGB value at the position (x, y) with the RGB value given by the tuple (r, g, b).
<code>i.draw()</code>	Displays <code>i</code> in a window. The user must close the window to return control to the method's caller.
<code>i.clone()</code>	Returns a copy of <code>i</code> .
<code>i.save()</code>	Saves <code>i</code> under its current filename. If <code>i</code> does not yet have a filename, <code>save</code> does nothing.
<code>i.save(filename)</code>	Saves <code>i</code> under <code>filename</code> . Automatically adds a <code>.gif</code> extension if <code>filename</code> does not contain it.



### 3. Write Python GUI program to take the birth date and output the age when a button is pressed

```

from breezypythongui import EasyFrame
from datetime import date

class AgeCalculator(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self, title="Age Calculator")
        self.addLabel(text="Enter your birth date (YYYY-MM-DD):", row=0,
column=0)
        self.dateField = self.addTextField(text="", row=0, column=1, width=10)
        self.addButton(text="Calculate Age", row=1, column=0, columnspan=2,
command=self.calculateAge)
        self.ageLabel = self.addLabel(text="Age:", row=2, column=0)
        self.ageValueField = self.addTextField(text="", row=2, column=1)

    def calculateAge(self):
        date_text = self.dateField.getText()

        try:

```

```

        birth_date = date.fromisoformat(date_text)
        self.calculateAgeFromDate(birth_date)
    except ValueError:
        # Handle invalid date format
        self.ageLabel.setText("Error:")
        self.ageValueLabel.setText("Invalid date format (YYYY-MM-DD)")

    def calculateAgeFromDate(self, birth_date):
        today = date.today()
        years = today.year - birth_date.year

        if today.month < birth_date.month or (today.month == birth_date.month
and today.day < birth_date.day):
            years -= 1
        self.ageValueField.setText(str(years))

AgeCalculator().mainloop()

```



## 4. How do you display an image in Python GUI?.

```

from breezypythongui import EasyFrame
from tkinter import PhotoImage
from tkinter.font import Font

class ImageDemo(EasyFrame):
    """Displays an image and a caption"""
    def __init__(self):
        """Sets up the window and the widgets"""
        EasyFrame.__init__(self, title="Image Demo")
        self.setResizable(False)
        imageLabel = self.addLabel(text="", row=0, column=0)
        textLabel = self.addLabel(text="Cat", row=1, column=0)

        # Load the image and associate it with the image label
        self.image = PhotoImage(file="cat.png")
        imageLabel["image"] = self.image

```



## 5. What are the attributes of a turtle object

Turtle Method	What It Does
<code>t = Turtle()</code>	Creates a new Turtle object and opens its window.
<code>t.home()</code>	Moves <code>t</code> to the center of the window and then points <code>t</code> east.
<code>t.up()</code>	Raises <code>t</code> 's pen from the drawing surface.
<code>t.down()</code>	Lowest <code>t</code> 's pen to the drawing surface.
<code>t.setheading(degrees)</code>	Points <code>t</code> in the indicated direction, which is specified in degrees. East is 0 degrees, north is 90 degrees, west is 180 degrees, and south is 270 degrees.
<code>t.left(degrees)</code> <code>t.right(degrees)</code>	Rotates <code>t</code> to the left or the right by the specified degrees.
<code>t.goto(x, y)</code>	Moves <code>t</code> to the specified position.
<code>t.forward(distance)</code>	Moves <code>t</code> the specified distance in the current direction.
<code>t.pencolor(r, g, b)</code> <code>t.pencolor(string)</code>	Changes the pen color of <code>t</code> to the specified RGB value or to the specified string, such as "red". Returns the current color of <code>t</code> when the arguments are omitted.
<code>t.fillcolor(r, g, b)</code> <code>t.fillcolor(string)</code>	Changes the fill color of <code>t</code> to the specified RGB value or to the specified string, such as "red". Returns the current fill color of <code>t</code> when the arguments are omitted.
<code>t.begin_fill()</code> <code>t.end_fill()</code>	Enclose a set of turtle commands that will draw a filled shape using the current fill color.
<code>t.clear()</code>	Erases all of the turtle's drawings, without changing the turtle's state.
<code>t.width(pixels)</code>	Changes the width of <code>t</code> to the specified number of pixels. Returns <code>t</code> 's current width when the argument is omitted.
<code>t.hideturtle()</code> <code>t.showturtle()</code>	Makes the turtle invisible or visible.
<code>t.position()</code>	Returns the current position ( <code>x, y</code> ) of <code>t</code> .
<code>t.heading()</code>	Returns the current direction of <code>t</code> .
<code>t.isdown()</code>	Returns True if <code>t</code> 's pen is down or False otherwise.



## 6. Write a GUI-based program that allows the user to convert amount in Indian Rupees to amount in Euro

The interface should have labeled entry fields for these two values. These components should be arranged in a grid where the labels occupy the first row and the corresponding fields occupy the second row. At start-up, the Rupees field should contain 0.0, and the Euro field should contain 0.0. The third row in the window contains two command buttons, labeled R->E and E->R. When the user presses the first button, the program should use the data in the Rupee field to compute the amount in Euro, which should then be output to the Euro field. The second button should perform the inverse function

```
from breezypythongui import EasyFrame
```

```
# Constant exchange rate (replace with your desired rate)
EXCHANGE_RATE = 0.012 # This is roughly 1 INR = 0.012 EUR (adjust as
needed)

class CurrencyConverter(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self, title="Currency Converter")

        # Labels and Entry fields for Rupees and Euros
        self.addLabel(text="Indian Rupees (INR)", row=0, column=0)
        self.rupee_field = self.addFloatField(value=0.0, row=1, column=0,
width=10, precision=2)
        self.addLabel(text="Euros (EUR)", row=0, column=1)
        self.euro_field = self.addFloatField(value=0.0, row=1, column=1,
width=10, precision=2)

        # Conversion buttons
        self.addButton(text="INR to EUR", row=2, column=0,
command=self.inr_to_eur) # ボタン is button in Japanese for better
compatibility with EasyFrame
        self.addButton(text="EUR to INR", row=2, column=1,
command=self.eur_to_inr)

        # Convert INR to EUR
        def inr_to_eur(self):
            inr_amount = self.rupee_field.getNumber()
            eur_amount = inr_amount * EXCHANGE_RATE
            self.euro_field.setNumber(eur_amount)

        # Convert EUR to INR
        def eur_to_inr(self):
            eur_amount = self.euro_field.getNumber()
            inr_amount = eur_amount / EXCHANGE_RATE
            self.rupee_field.setNumber(inr_amount)

# Run the program
CurrencyConverter().mainloop()
```



## 7. What are the attributes of a window? How the attributes value can be changed?

A window has several attributes. The most important ones are its

- title (an empty string by default)
- width and height in pixels
- resizable (true by default)
- background color (white by default)

EasyFrame Method	What It Does
<code>setBackground(color)</code>	Sets the window's background color to <code>color</code> .
<code>setResizable(aBoolean)</code>	Makes the window resizable ( <code>True</code> ) or not ( <code>False</code> ).
<code>setSize(width, height)</code>	Sets the window's width and height in pixels.
<code>setTitle(title)</code>	Sets the window's title to <code>title</code> .



## 8. Comment on event driven programming.

**Event-driven nature of GUIs:**

- GUIs typically open a window and wait for user interaction.
- User actions like mouse clicks generate events.
- These events trigger program responses involving input processing and output display.

**EDP principles in action:**

- The program execution flow is dictated by user-generated events.
- Event handlers manage specific events, decoupling components.
- This design promotes responsiveness to user interactions.

**Analysis and Design:**

- GUI design involves choosing or creating appropriate window components.
- Object-oriented programming (OOP) is often used for GUI development.





## 9. Write a Python program to convert a color image to a grayscale image

```
from images import Image # Assuming this imports an Image class
representing an image

def grayscale(image):
    """
    Converts an image to grayscale.

    Args:
        image: An Image object representing the image to be converted.

    Returns:
        None (modifies the image object in-place).
    """

    # Iterate over each pixel in the image
    for y in range(image.getHeight()):
        for x in range(image.getWidth()):
            # Get the original red, green, and blue components (assuming 0-255
            range)
            (r, g, b) = image.getPixel(x, y)

            # Apply grayscale conversion formula (weighted sum)
            # - 0.299, 0.587, and 0.114 are standard weighting factors for human
            perception of luminance
            gray_value = int(r * 0.299) + int(g * 0.587) + int(b * 0.114)

            # Set all three color components (R, G, B) to the calculated gray
            value
            # This effectively makes the pixel appear grayscale
            image.setPixel(x, y, (gray_value, gray_value, gray_value))

def main(filename="cat.jpeg"):
    """
    Loads an image, converts it to grayscale, and displays it.

    Args:
        filename: The filename of the image to process (default: "cat.jpeg").
    """
```

```

# Load the image from the specified file
image = Image(filename)

# Optionally, draw the original image (depends on your Image class
implementation)
image.draw()

# Convert the image to grayscale
grayscale(image)

# Optionally, draw the grayscale image (depends on your Image class
implementation)
image.draw()

if __name__ == "__main__":
    main()

```



## ***10. Write Python GUI program to input two strings and output a concatenated string when a button is pressed.***

```

from breezypythongui import EasyFrame

class StringConcatenator(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self, title="String Concatenator")

        # Input labels and fields for strings
        self.addLabel(text="String 1:", row=0, column=0)
        self.stringField1 = self.addTextField(text="", row=0, column=1,
width=20)
        self.addLabel(text="String 2:", row=1, column=0)
        self.stringField2 = self.addTextField(text="", row=1, column=1,
width=20)

        # Output label and field
        self.addLabel(text="Concatenated String:", row=2, column=0)
        self.outputField = self.addTextField(text="", row=2, column=1, width=40)

```

```
# Button to concatenate strings
self.addButton(text="Concatenate", row=3, column=0, columnspan=2,
command=self.concatenateStrings)

def concatenateStrings(self):
    # Get strings from input fields
    string1 = self.stringField1.getText()
    string2 = self.stringField2.getText()

    # Concatenate strings
    concatenated_string = string1 + string2

    # Set output field text
    self.outputField.setText(concatenated_string)

# Run the program
StringConcatenator().mainloop()
```



## ***11. Discuss on the types of window components and their functions.***

Here are some common types of window components:

- **Labels:** These show text on the screen, like instructions or titles.
- **Entry fields:** These are boxes where you can type in text.
- **Text areas:** These are bigger boxes where you can type a lot of text.
- **Buttons:** You click these to make something happen in the program.
- **Drop-down menus:** These let you choose an option from a list.
- **Sliding scales:** You can move a slider to choose a value.
- **Scrolling list boxes:** These show a list of things you can scroll through and choose from.

Type of Window Component	Purpose
<code>Label</code>	Displays text or an image in the window.
<code>IntegerField(Entry)</code>	A box for input or output of integers.
<code>FloatField(Entry)</code>	A box for input or output of floating-point numbers.
<code>TextField(Entry)</code>	A box for input or output of a single line of text.
<code>TextArea(Text)</code>	A scrollable box for input or output of multiple lines of text.
<code>EasyListbox(Listbox)</code>	A scrollable box for the display and selection of a list of items.

Type of Window Component	Purpose
<code>Button</code>	A clickable command area.
<code>EasyCheckbutton(Checkbutton)</code>	A labeled checkbox.
<code>Radiobutton</code>	A labeled disc that, when selected, deselects related radio buttons.
<code>EasyRadiobuttonGroup(Frame)</code>	Organizes a set of radio buttons, allowing only one at a time to be selected.
<code>EasyMenuBar(Frame)</code>	Organizes a set of menus.
<code>EasyMenubutton(Menubutton)</code>	A menu of drop-down command options.
<code>EasyMenuItem</code>	An option in a drop-down menu.
<code>Scale</code>	A labeled slider bar for selecting a value from a range of values.
<code>EasyCanvas(Canvas)</code>	A rectangular area for drawing shapes or images.
<code>EasyPanel(Frame)</code>	A rectangular area with its own grid for organizing window components.
<code>EasyDialog(simpleDialog.Dialog)</code>	A resource for defining special-purpose popup windows.



**12. Write a GUI-based program that allows the user to convert temperature values between degrees Fahrenheit and degrees Celsius.**

- The interface should have labeled entry fields for these two values.
- These components should be arranged in a grid where the labels occupy the first row and the corresponding fields occupy the second row.
- At start-up, the Fahrenheit field should contain 32.0, and the Celsius field should contain 0.0.
- The third row in the window contains two command buttons, labeled >>>> and <<<<. When the user presses the first button, the program should use the data in the Fahrenheit field to compute the Celsius value, which should then be output to the Celsius field.
- The second button should perform the inverse function.

```
from breezypythongui import EasyFrame
import math

class TemperatureConverter(EasyFrame):
    def __init__(self):
        EasyFrame.__init__(self, title="Temperature Converter")

        # Labels
        self.addLabel(text="Fahrenheit:", row=0, column=0)
        self.addLabel(text="Celsius:", row=1, column=0)

        # Input Fields
        self.fahrenheit_field = self.addFloatField(
            value=32.0, row=0, column=1, width=8, precision=2
        )
        self.celsius_field = self.addFloatField(value=0.0, row=1, column=1,
width=8, precision=2)

        # Buttons
        self.to_celsius_button = self.addButton(
            text=">>> (F to C)", row=2, column=0,
command=self.convert_to_celsius
        )
        self.to_fahrenheit_button = self.addButton(
            text="<<< (C to F)", row=2, column=1,
command=self.convert_to_fahrenheit
        )
```

```
def convert_to_celsius(self):  
    fahrenheit = self.fahrenheit_field.getNumber()  
    celsius = (fahrenheit - 32) * 5 / 9  
    self.celsius_field.setNumber(celsius)  
  
def convert_to_fahrenheit(self):  
    celsius = self.celsius_field.getNumber()  
    fahrenheit = (celsius * 9 / 5) + 32  
    self.fahrenheit_field.setNumber(fahrenheit)
```

```
TemperatureConverter().mainloop()
```

