Palindrome Checker in Python :

Source Code:

```
#Here we are taking input on which we have to check
# Are used for comments in python s= input("Enter the
Value : \n ") #Reverse function #There are alot of
methods for reversing #but we are using slicing
method reverse = s[::-1] if (s==reverse):


    print("Yes it is a palindrome")
else :
    print("No it is not a palindrome")
```

Notes:

- **What is a Palindrome:**

A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward (ignoring spaces, punctuation, and capitalization). Examples include "madam," "racecar," and "level."

- **Code Explanation:**

1. **Taking User Input:**

 - `s = input("Enter the value: \n")` prompts the user to enter a string and stores the input in the variable `s`.

2. **Reversing the String:**

SKILLBHAVAN GET YOUR STUDYROOM!!

- `reverse = s[::-1]` creates a reversed version of the string `s` using slicing. The slicing notation `[::-1]` means "start at the end of the string and end at position 0, moving with the step -1," effectively reversing the string.

3. **Checking for Palindrome:**

   - `if s == reverse:` compares the original string `s` with its reversed version `reverse`.

     - If they are equal, the string is a palindrome.

     - If they are not equal, the string is not a palindrome.

4. **Printing the Result:**

   - If the original string and the reversed string are equal, the program prints "Yes, it is a Palindrome."

   - Otherwise, the program prints "No, it is not a Palindrome."

- **Result:**

  This program checks if a user-provided string is a palindrome and informs the user of the result.

This script is a simple and effective way to demonstrate string manipulation and comparison in Python, providing an interactive way to check for palindromes.

SKILLBHAVAN
GET YOUR STUDYROOM!!

Guess the Number Game in Python :

Source Code:

```
import random

computer=random.randrange(1,100)

#randrange is use to define a range for computer to choose a random
number
user=int(input("Enter your number : \n"))



#Comperision


if user>computer:
	print("computer : ",computer)
	print("Your Guess number is too High")

elif computer>user:
	print("computer number : " , computer )
	print("Your Guess number is too Low ")

else:
	print ("Computer number : " ,computer)
	print("Your Guess number is equal ! You win !!")



#we can make this game more difficult ! by increasing range
```

Notes:

What is the `random` Library in Python:

The `random` library in Python is a powerful tool for generating randomness and simulating various chance-based scenarios. It provides functions for

SKILLBHAVAN
GET YOUR STUDYROOM!!

generating random numbers, selecting random elements from sequences, shuffling sequences, and more.

Code Explanation:

1. Importing the Random Library:

   - The code starts by importing the `random` module, which allows the program to generate random numbers.

2. Generating a Random Number:

   - `computer = random.randrange(1, 10)` generates a random integer between 1 and 9 (inclusive of 1 and exclusive of 10).

   - This random number is stored in the variable``computer`. This is the number the user will try to guess.

3. Taking User Input:

   - `user = int(input("Enter your number : \n"))` prompts the user to input a number.

   - The entered value, which is initially a string, is converted to an integer using `int()`, and then stored in the variable `user`.

4. Comparing the User's Guess with the Computer's Number:

   - First condition (`if user > computer`):

     - If the user's guess is greater than the computer's number, the program executes this block.

     - It prints the computer's number and informs the user that their guess is too high.

   - Second condition (`elif computer > user`):

SKILLBHAVAN
CRAFT YOUR STUDYROOM!!

- If the user's guess is less than the computer's number, the program executes this block.

    - It prints the computer's number and informs the user that their guess is too low.

  - Else block (`else`):

    - If the user's guess is equal to the computer's number, the program executes this block.

    - It prints the computer's number and congratulates the user for guessing correctly.

- Result:

  The program compares the user's guess with the randomly generated number and provides feedback:

  - If the guess is too high, it informs the user.

  - If the guess is too low, it informs the user.

  - If the guess is correct, it congratulates the user.

SKILLBHAVAN GET YOUR STUDYROOM!!

This script is a simple interactive game that leverages the `random` module and basic control flow statements (`if`, `elif`, `else`) to create an engaging user experience.

Snake-Water-Gun Game in Python :

Source Code:

```
import random
def check (computer,user):

    if (computer==user) :
        return 0
    if (computer==0 and user ==1 ) :
        return -1
    if(computer == 1 and user ==2) :
        return -1
    if(computer == 2 and user ==0) :
        return -1
    return 1



user = int(input("0 for snake , 1 for water and 2 for gun "))
computer = print(random.randint(0,2))
score = check(computer , user )
print("computer : ",computer)
print("user : ", user)
if (score==0 ):
    print ("Its a Draw !")

elif score==-1:
    print ("You Lose")
else :
    print("You Win")
```

Notes:

� Random Number Generation:

SKILLBHAVAN GET YOUR STUDYROOM!!

- **Random Module**: Provides functions to generate random numbers and choices (random.randint).
- Used here to simulate the computer's choice in the game, ensuring unpredictability and fairness.

Conditional Statements:

- **If Statements**: Used to evaluate different conditions (computer == user, specific combinations of choices) and determine the game outcome (check function).

Game Logic:

- **Rock-Paper-Scissors**: A simple hand game typically played between two people where each player simultaneously forms one of three shapes with an outstretched hand:
  - Rock (beats scissors)
  - Paper (covers rock)
  - Scissors (cut paper)
- This code adapts the concept with different choices: snake, water, and gun, applying similar logic for determining the winner.

SKILLBHAVAN GET YOUR STUDYROOM!!

Bike Rental System in Python :

Source Code:

```
class bikeshop:

    def __init__(self,stock):
        self.stock=stock
    def displaybike(self):
        print ("Total Bikes",self.stock)
    def rentforbike(self,q):


        if q<=0:
            print ("Enter the positive value or greater then zero")
        elif q>self.stock:
            print("Enter the value (Less then stock )")

        else :
            print("Total Prices ", q*100)
            print("Total bikes ", self.stock)


while True:
    obj=bikeshop(100)
    uc=int(input('''
1-Display Total Number of bikeshop
2-Rent a Bike
3-Exit
    '''))
    if uc==1:
        obj.displaybike()
    elif uc==2:
        n=int(input("Enter Number of Bikes you want : " ))
        obj.rentforbike(n)
    else:
        break
```

SKILLBHAVAN GET YOUR STUDYROOM!!

Notes:

- **Object-Oriented Programming (OOP)**:
  - o **Classes and Objects**: Fundamental concepts in OOP where classes define objects (instances) with attributes (data) and methods (functions).
  - o **Encapsulation**: Bundling data (attributes) and methods that operate on the data into a single unit (class).
  - o **Constructor (__init__ Method)**: Special method in Python classes used to initialize objects with default or provided values.
- **Class Methods**:
  - o **Instance Methods**: Functions defined within a class that operate on instance data (self).
  - o **Method Invocation**: Calling methods on class instances (obj.method() syntax).
- **User Input Handling**:
  - o **input Function**: Retrieves user input as a string, which can be converted to integers or other data types for processing.
  - o **Conditional Statements (if, elif, else):** Used to control program flow based on user choices and conditions (uc variable).

Summary

- The code exemplifies OOP principles with the bikeshop class, encapsulating bike rental functionality.
- Demonstrates how classes encapsulate attributes and methods, promoting code organization and reusability.
- Illustrates interactive programming with user input handling and conditional logic, providing a structured approach to managing bike rentals within the simulated shop environment.

SKILLBHAVAN GET YOUR STUDYROOM!!

Calculator:

Source Code:

```
print("Calculator")

num1=int(input("Enter First Number: "))

num2=int(input("Enter second Number:"))


#Operations !


print ("press + for addition \n press - for substraction \n press * for
Multiplication \n press / for Division")

Choice=(input ("Enter Your operation from above "))


if Choice == '+' :

        print(num1+num2)

elif Choice == '-' :

        print(num1-num2)


elif Choice == '*' :

        print(num1*num2)

elif Choice == '/' :

        print(num1/num2)

else:

        print("Enter a Valid Operator")
```

Notes:

- **User Input Handling**:
    - **input Function**: Retrieves user input as a string.

SKILLBHAVAN
GET YOUR STUDYROOM!!

- **Type Conversion (int)**: Converts the user input from string to integer for numerical operations.
- **Conditional Statements**:
  - **if, elif, else**: Used to conditionally execute blocks of code based on the user's choice of operation (Choice).
- **Arithmetic Operations**:
  - **Addition (+)**: Combines two numbers.
  - **Subtraction (-)**: Finds the difference between two numbers.
  - **Multiplication (*)**: Repeats addition of a number.
  - **Division (/)**: Splits one number into multiple parts.

Summary

- The code snippet implements a basic calculator that allows users to perform arithmetic operations on two numbers.
- Demonstrates interactive programming by prompting users for input and providing clear instructions for operation selection.
- Utilizes conditional statements to determine the operation to perform and outputs the result accordingly.
- Provides a foundational example of user interaction, basic arithmetic operations, and conditional logic in Python programming.

SKILLBHAVAN GET YOUR STUDYROOM!!

Face Detection :

Source Code:

```
#pip install opencv-python


import cv2

# we will use haarcascade for detecting face

# open cv has many classifiers for detecting faces, smiles etc

face_cascade                                                                                =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

img= cv2.imread('profile.png')

# this method work only for grey scale

# we have to convert our img to gray scale

gray= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces= face_cascade.detectMultiScale(gray, 1.1,4)

# rectangle box on faces

for (x,y,w,h) in faces:

cv2.rectangle (img, (x,y), (x+w, y+h), (255,0,0),2)

 cv2.imshow('img', img)

cv2.waitKey()
```

SKILLBHAVAN GET YOUR STUDYROOM!!

Notes:

- **Haar Cascade Classifiers**:
  - o **Cascade Classifiers**: Machine learning-based approach where a cascade function is trained from a lot of positive and negative images to detect objects like faces, eyes, etc.
  - o **Haar Cascade**: A machine learning object detection algorithm used to identify objects in images or video frames based on their features.

- **Image Processing with OpenCV**:
  - o **cv2.imread**: Loads an image from the specified file path.
  - o **cv2.cvtColor**: Converts an image from one color space to another (e.g., BGR to grayscale).

- **cv2.rectangle**: Draws rectangles on images or video frames.
- **cv2.imshow**: Displays an image in a window.
- **cv2.waitKey**: Waits for a key event from the user for a specified amount of time.

 Computer Vision Applications:

- **Face Detection**: A fundamental task in computer vision that involves locating and identifying human faces in digital images or video streams.

- **Real-time Processing**: Techniques and algorithms optimized for efficient processing of image data in real-time applications.

Summary

 The code demonstrates how to perform face detection using OpenCV and a pre-trained Haar Cascade classifier.
 Utilizes image loading, grayscale conversion, face detection using a cascade classifier, and drawing rectangles around detected faces.
 Provides a practical example of using computer vision techniques for detecting and highlighting specific objects (faces) within images, showcasing the capabilities of OpenCV in image processing applications.

SKILLBHAVAN
GET YOUR STUDY.COM

Password Generator :

Source Code:

```
import string

import random

if __name__ =="__main__":

      s1 = string.ascii_uppercase

      s2 = string.ascii_lowercase

      s3 = string.digits

      s4 = string.punctuation

     # print(s1,s2,s3,s4)

      #Now we will creat a Empty list & Extend all values in one variable

 #remember there is a difference in append & Extend


passwordlength = int(input("Enter the length of the password : \n"))

s= []

s.extend(list(s1))

s.extend(list(s2))

s.extend(list(s3))

s.extend(list(s4))

#print (s)


random.shuffle(s)

print("".join(s[0:passwordlength]))
```

Notes:

- **String Module (string)**:

SKILLBHAVAN GET YOUR STUDYROOM!!

  - Provides constants and classes for working with string data in Python.
  - Includes predefined sets like ascii_uppercase, ascii_lowercase, digits, and punctuation for easy access to common character sets.
- **Random Module (random)**:
  - Provides functions for generating random numbers and selecting random elements from sequences.
  - **random.shuffle()**: Randomly rearranges the elements of a list.
- **List Operations**:
  - **.extend() vs .append()**:
    - **.extend()**: Adds elements from an iterable (like a list) to the end of the list.
    - **.append()**: Adds a single element to the end of the list.
- **Password Generation**:
  - A common application in cybersecurity and user authentication.
  - Importance of randomness and length in generating secure passwords.
  - Techniques to combine character sets to increase password complexity and strength.

Summary

- The code exemplifies how to generate a random password in Python using characters from different sets (uppercase, lowercase, digits, punctuation).
- Demonstrates the use of the string and random modules to efficiently manage and manipulate strings and generate randomness.
- Provides practical insights into password generation techniques for enhancing security by combining diverse character sets and ensuring randomness.

SKILLBHAVAN GET YOUR STUDYROOM!

Weight Conversion :

Source Code:

```
weight = float(input("Enter your weight: "))

unit = input("Kilograms or Pounds?(K or L): ")


if unit =="K":

    weight = weight * 2.205

    unit = "Lbs."

    print(f"Your weight is:{round(weight,1)}{unit}")

elif unit =="L":

    weight = weight / 2.205

    unit = "Kgs."

    print(f"Your weight is:{round(weight,1)}{unit}")

else:

    print(f"{unit}was not valid")
```

Notes:

- **User Input and Data Types**:
  - Uses input() function to receive user input as a string and float() function to convert input into a numerical (floating-point) format for calculations.
- **Conditional Statements (if-elif-else)**:
  - **if unit == "K":** checks if the input unit is kilograms.
  - **elif unit == "L":** checks if the input unit is pounds.
  - **else:** handles any other input that is not K or L.
- **Unit Conversion**:
  - Demonstrates the concept of unit conversion between different systems (kilograms to pounds and vice versa) using multiplication and division by conversion factors.
- **Formatted Output (f-strings)**:
  - Utilizes formatted strings (f"{...}") to output the converted weight and unit neatly formatted for display.

SKILLBHAVAN GET YOUR STUDYROOM!!

Summary

- The code allows users to convert their weight between kilograms and pounds based on user input.
- Highlights the use of conditional statements to determine which conversion formula to apply based on the user's choice (K or L).
- Provides a practical example of how to handle user input, perform calculations, and display formatted output in Python.

SKILLBHAVAN GET YOUR STUDYROOM!!

Text to Handwriting assignment :

Source Code:

```
#import Library and give a alias name

import pywhatkit as pw


#add your custom text

#we use triple code because we have to add whole para

txt ="""" Python is an interpreted high-level general-purpose programming language.

Its design philosophy emphasizes code readability with its use of
significant indentation. """


#we will use inbuilt function


#if we didn't give any name it will save by pywhatkit.png

#we have to give three parameters : text , file name , color

#by default color is blue


pw.text_to_handwriting(txt)

print ("Code is running")
```

SKILLBHAVAN GET YOUR STUDYROOM!!

Notes:

- **Library Importing with Alias**
    - o Allows importing external libraries in Python using import statement.
    - o **Alias (as pw)**: Provides a shorter or more descriptive name (pw) to refer to the imported library (pywhatkit) throughout the code.
- **Multi-line String (""" ... """)**:
    - o Used to define a string that spans multiple lines.
    - o Ideal for embedding large blocks of text or paragraphs within Python code.

- **Function Call (pw.text_to_handwriting(txt))**:
  - o Invokes a function (text_to_handwriting()) from the imported library (pywhatkit) to perform a specific task (text to handwriting conversion).

  - o Demonstrates the use of library functions to leverage pre-built functionalities in applications such as text processing and visualization.
- **Output and Display**
  - o The converted handwritten text is typically saved as an image file (png format) unless specified otherwise.
  - o Users can modify parameters such as text content, filename, and handwriting color (if supported by the library function) to customize the output.

Summary

- The code snippet showcases how to use the pywhatkit library in Python to convert text into a handwritten format.

 Highlights the simplicity of calling functions from imported libraries to perform complex tasks such as generating graphical outputs from textual inputs.
Illustrates practical applications of text processing and visualization using external libraries, enhancing the functionality and versatility of Python programming.

SKILLBHAWAN GET YOUR STUDYROOM!!

Dice Rollar :

Source Code:

```
import random

#remember this codes ! to make your dice more attractive


 # ● ┌ ─ ┐  └ ┘


" ┌─────────────┐ "
"|              |"
"|              |"
"|              |"
" └─────────────┘ "


#Create dictionary
#Make all boxes
Dice_art = {
     1:(" ┌─────────────┐ ",
        "|              |",
        "|      ●       |",
        "|              |",
        " └─────────────┘ "),
     2:(" ┌─────────────┐ ",
        "| ●            |",
        "|              |",
        "|           ● |",
        " └─────────────┘ "),
```

SKILLBHAVAN GET YOUR STUDYROOM!!

```
        3:(" ┌─────────────┐ ",
            "| ●           |",
            "|      ●      |",
            "|           ● |",
            " └─────────────┘ "),
        4:(" ┌─────────────┐ ",
            "| ●         ● |",
            "|             |",
            "| ●         ● |",
            " └─────────────┘ "),
        5:(" ┌─────────────┐ ",
            "| ●         ● |",
            "|      ●      |",
            "| ●         ● |",
            " └─────────────┘ "),
        6:(" ┌─────────────┐ ",
            "| ●         ● |",
            "| ●         ● |",
            "| ●         ● |",
            " └─────────────┘ ")
}

#Create a list of numbers

dice=[]
total =0
number_of_dice = int(input("How many dices ? : \n "))
```

SKILLBHAVAN GET YOUR STUDYROOM!!

```
for die in range(number_of_dice):
 dice.append(random.randint(1,6))


print(dice)


for die in range(number_of_dice):
   for line in Dice_art.get(dice[die]):
      print(line)


for die in dice:
    total+=die
print(f"total: {total}")


#In python we have to give proper space between code functions
```

Notes:

- **Random Number Generation (random.randint(1, 6))**:
   - Utilizes random.randint() function to generate random integers within a specified range (1 to 6 in this case), mimicking the roll of a six-sided die.
- **Data Structures (Dictionary and List)**:
   - **Dictionary (Dice_art)**: Stores ASCII art representations as values mapped to dice face numbers as keys, allowing efficient lookup and retrieval of dice faces.
   - **List (dice)**: Holds randomly generated dice roll results, facilitating storage and manipulation of multiple dice outcomes.
- **ASCII Art Representation**:
   - Demonstrates how to use multiline strings to create visual representations of dice faces using ASCII characters.
   - Enhances user experience by visually presenting the results of each dice roll in a readable and appealing format.
- **User Input Handling**:

SKILLBHAVAN GET YOUR STUDYROOM!!

o Prompts the user to input the number of dice to roll, demonstrating interactive program design where user input influences program behavior.

Summary

 The code showcases how to simulate rolling multiple dice in Python using random number generation and display the results using ASCII art representations of dice faces.
Illustrates the use of dictionaries for efficient data storage and retrieval of ASCII art, enhancing program readability and maintainability.
 Provides a practical example of integrating user input, random number generation, and visual output to create a engaging and interactive simulation of dice rolling.

SKILLBHAVAN
GET YOUR STUDYROOM!!

Turtle Drawing :

Source Code:

```
import turtle


arrow= turtle.Turtle()

arrow.pencolor("red")


screen = turtle.Screen()

screen.bgcolor("black")


arrow.speed(0)


arrow.penup()

arrow.goto(0,150)

arrow.pendown()


forw = 0

right = 0


while True:

    arrow.forward(forw)

    arrow.right(right)


    forw += 2

    right += 1

    turtle.done()
```

SKILLBHAVAN
GET YOUR STUDYROOM!!

Notes:

- **Turtle Graphics**:
    - o **Concept**: Turtle graphics is a popular way for introducing programming to beginners. It uses a metaphor of a turtle moving around a screen with a pen attached to its tail to draw lines.
    - o **Usage**: In this code, turtle.Turtle() creates a turtle object (arrow), and methods like forward() and right() are used to control its movements.

- **Animation**:
    - o **Concept**: Animation involves creating a sequence of images (or frames) to give the illusion of movement when displayed in rapid succession.
    - o **Usage**: The while True: loop in the code continuously updates the turtle's position and angle, creating a dynamic animation effect as it draws lines on the screen.
- **Screen and Turtle Configuration**:
    - o **Concept**: The turtle.Screen() object manages the appearance and behavior of the graphics window.
    - o **Usage**: Commands like bgcolor() set the background color of the screen, while speed() controls the drawing speed of the turtle.
- **Infinite Loop**:
    - o **Concept**: An infinite loop (while True:) continues executing its block of code indefinitely, or until explicitly terminated. **Usage**: In this
    - ○ code, the infinite loop keeps the turtle graphics running until the user manually closes the window or terminates the program.

SKILLBHAVAN GET YOUR STUDYROOM!!

Summary

- The code demonstrates the use of turtle graphics to create dynamic visual effects in Python.
- It employs fundamental concepts such as turtle movement commands (forward() and right()), animation with incremental updates (forw += 2 and right += 1), and configuration of the turtle and screen environment.
- Provides an interactive and engaging way to explore programming concepts through visual representation and dynamic animation effects.

How to create emoji:

Source Code:

```
import turtle


emo = turtle.Turtle()


emo.up()

emo.goto(0,-100)

emo.down()


emo.begin_fill()

emo.pendown()

emo.fillcolor('yellow')

emo.circle(100)

emo.end_fill()


emo.up()

emo.goto(-67,-40)

emo.setheading(-60)

emo.width(5)

emo.down()

emo.circle(80,120)

emo.fillcolor("black")


for i in range(-35,105,70):

    emo.up()
```

SKILLBHAVAN
GET YOUR STUDYROOM!!

```
    emo.goto(i,35)

    emo.setheading(0)

    emo.down()

    emo.begin_fill()

    emo.circle(10)

    emo.end_fill()


emo.penup()

emo.goto(0,-150)

turtle.mainloop()
```

Notes:

- **Turtle Graphics**:
  - o **Concept**: Turtle graphics is a method of computer graphics programming where a turtle cursor is used to draw shapes on a screen.

  - o **Usage**: In this code, the turtle (emo) is used to draw geometric shapes like circles and arcs to create an emoticon.
- **Drawing Commands**:
  - o **Concept**: Commands like circle(), goto(), up(), down(), begin_fill(), and end_fill() are fundamental in turtle graphics.
  - o **Usage**: They control the movement of the turtle, the drawing of shapes, and the filling of shapes with colors.
- **Pen Control**:
  - o **Concept**: The turtle's pen can be lifted (penup()) to move without drawing, and put down (pendown()) to start drawing.
  - o **Usage**: It allows precise control over when and where lines and shapes are drawn on the screen.
- **Event Handling**:
  - o **Concept**: Functions like mainloop() in the turtle module keep the graphics window open and responsive to user interactions until explicitly closed.
  - o **Usage**: Ensures that the program stays active and can respond to events like mouse clicks or key presses.

SKILLBHAVAN GET YOUR STUDYROOM!!

Summary

- The code utilizes turtle graphics to draw an emoticon (smiley face) on the screen using basic geometric shapes.
- It demonstrates essential turtle commands for drawing shapes, positioning the turtle, and controlling the pen.
- Provides an interactive and visual way to learn programming concepts related to graphics and geometric shapes in Python.

SKILLBHAVAN GET YOUR STUDYROOM!!

Text extraction from Image :

Source Code:

```
import pytesseract


from PIL import Image

# Provide the correct path to the Tesseract executable

pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-
OCR\tesseract.exe"


# Open the image

img = Image.open("1.png")


# Use pytesseract to extract text from the image

text = pytesseract.image_to_string(img)


# Print the extracted text

print(text)
```

Notes:

- **Optical Character Recognition (OCR)**:
   - o **Concept**: OCR is a technology that recognizes text within images or scanned documents and converts it into machine-readable text.
   - o **Usage**: The code demonstrates how to use Tesseract OCR through pytesseract to extract text from an image (1.png).
- **pytesseract Library**:
   - o **Concept**: pytesseract is a Python wrapper for Tesseract OCR engine. It simplifies the process of integrating OCR capabilities into Python applications.
   - o **Usage**: In the code, pytesseract.image_to_string() is used to perform OCR and extract text from the image.
- **PIL (Python Imaging Library)**:

SKILLBHAVAN
GET YOUR STUDYROOM!!

o **Concept**: PIL (now known as Pillow) is a library in Python used for image processing tasks such as opening, manipulating, and saving images.

o **Usage**: The Image.open() method from PIL is used to open and load the image file (1.png) for OCR processing.

- **Setting Tesseract Path**:
   - **Concept**: It's essential to specify the path to the Tesseract executable (tesseract.exe) using pytesseract.pytesseract.tesseract_cmd to ensure proper functioning of the OCR engine within Python.
   - **Usage**: Ensures that Python can locate and execute the Tesseract OCR engine to perform text extraction from images.

Summary

- The code leverages the pytesseract library along with PIL to extract text from an image (1.png) using Tesseract OCR.
- Demonstrates the integration of OCR capabilities into Python applications for automating text extraction tasks from images.
- Illustrates the importance of configuring the Tesseract executable path (tesseract_cmd) to ensure correct operation of the OCR engine.

SKILLBHAVAN GET YOUR STUDYROOM!

Basic Black drawing in turtle:

Source Code:

```
import turtle,math


def draw_spirograph(R,r,d):

    t=turtle.Turtle()

    t.speed(0)

    t.width(2)

    for theta in range(0,720):

        theta_rad=math.radians(theta)

        x=(R-r)*math.cos(theta_rad)+d*math.cos((R-r)/r*theta_rad)

        y=(R-r)*math.sin(theta_rad)-d*math.sin((R-r)/r*theta_rad)

        t.goto(x,y)

    t.hideturtle()



draw_spirograph(100,20,50)

turtle.done()
```

Notes:

- **Spirograph**:
    - o **Concept**: A spirograph is a geometric drawing tool that produces mathematical curves known as hypotrochoids and epitrochoids.
    - o **Usage**: The code uses mathematical formulas (x and y coordinates) derived from the parametric equations of spirographs to plot points and create the intricate patterns.
- **Parametric Equations**:
    - o **Concept**: Parametric equations describe a set of equations that express a set of quantities as explicit functions of one or more independent variables, known as parameters.
    - o **Usage**: In the code, x and y are calculated using parametric equations specific to spirographs, involving trigonometric functions (cos and sin) to determine the positions of the drawing pen (turtle).

SKILLBHAVAN GET YOUR STUDYROOM!!

 Turtle Graphics:

o **Concept**: Turtle graphics is a popular way for introducing programming to beginners. It involves using a cursor (turtle) to draw shapes and patterns on a screen using simple commands. o **Usage**: The code utilizes the turtle module to draw the spirograph, demonstrating basic turtle commands like goto() for moving the turtle and speed() for controlling the drawing speed.

 Mathematical Computation:

o **Concept**: Mathematical computations involving trigonometric functions (cos and sin) are crucial in generating the coordinates required for drawing precise spirograph patterns.

o **Usage**: The code demonstrates how to use mathematical computations within a programming context to create visually appealing geometric designs like spirographs.

Summary

 The code uses the turtle module and mathematical computations (math module) to draw a spirograph.

 Demonstrates the use of parametric equations specific to spirographs to calculate coordinates for drawing intricate patterns.

 Utilizes basic turtle graphics commands to move the turtle and draw lines based on computed coordinates.

 Illustrates the intersection of mathematical concepts (trigonometry, parametric equations) and programming (turtle graphics) to create visually engaging graphical output.

SKILLBHAVAN GET YOUR STUDYROOM!!

Random Spiral Drawing

Source code:

```
import turtle
import random


# Set up screen
screen = turtle.Screen()
screen.setup(width=800, height=800)
screen.bgcolor('black')
# Set up turtle
t = turtle.Turtle()
t.speed(0)
t.width(2)
t.hideturtle()

def draw_spirograph(size, color):
    t.color(color)
    for _ in range(360):
        t.circle(size)
        t.right(5)
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'purple', 'cyan']
for i in range(50):
    color = random.choice(colors)
    size = random.randint(10, 100)
    draw_spirograph(size, color)


t.hideturtle()
screen.mainloop()
```

SKILLBHAVAN GET YOUR STUDYROOM!!

Notes:

□ What is random library in python:

The random library in Python is a powerful tool for generating randomness and simulating various chance-based scenarios. It provides functions for generating random numbers, selecting random elements from sequences, shuffling sequences, and more.

□Code Explanation:

1) Importing the Turtle Library:
The code starts by importing the turtle module, which allows us to create graphics and draw shapes.
The random module is also imported to randomly select colors.

2) Setting Up the Screen and Turtle:

The screen is set up with a black background and dimensions of 800x800 pixels.
A turtle object (t) is created with specific settings:
Speed set to 0 (fastest).
Pen width set to 2.
Initially hidden (using t.hideturtle()).

3) Drawing the Spirograph:

The draw_spirograph function is defined to draw a single spirograph circle.
It takes two parameters: size (radius of the circle) and color.
The turtle's pen color is set to the specified color.
The turtle draws a circle with the given size by moving right 5 degrees at a time for 360 steps.
A list of colors (colors) is defined.
A loop runs 50 times:
A random color is chosen from the list.
A random size (between 10 and 100) is selected.
The draw_spirograph function is called with the chosen color and size.

4) Result:

The code creates a colorful spirograph by drawing multiple circles with varying sizes and colors.

SKILLBHAWAN GET YOUR STUDYROOM!!

Spinograph :

Source code:

```
import turtle
turtle.bgcolor("black")
turtle.pensize(2)
turtle.speed(0)
for i in range(7):
      for colors in
["red ","b lue" ,"gr een" ,'ye llow ','o rang e ','pi nk', 'pur ple' ,'cy an', 'mag enta '
,'white']:
            turtle.color(colors)
            turtle.circle(100)
            turtle.left(10)
turtle.hideturtle()
```

Notes:

The list of colors includes more colors than necessary for the loop iterations, which might be intended for variety or experimentation.
Adjusting the parameters (such as the number of iterations in the outer loop or the rotation angle in turtle.left()) can create different patterns and effects.

Code Explanation:

import turtle
This line imports the Turtle graphics module, which allows us to create drawings using a turtle metaphor.
turtle.bgcolor("black")
Sets the background color of the drawing area to black.
turtle.pensize(2)
Sets the thickness of the pen (turtle's drawing tool) to 2 pixels.
turtle.speed(0)
Sets the drawing speed to the maximum, which is the fastest possible speed (speed of 0).

```
for i in range(7):
for colors in

["red","blue","green",'yellow','orange','pink','purple','cyan','magenta','white']:
      turtle.color(colors)
      turtle.circle(100)
      turtle.left(10)
```

This nested loop structure is where the main drawing happens:
The outer loop for i in range(7): runs 7 times, controlling the number of times the inner loop executes.
The inner loop for colors in ["red","blue","green",'yellow','orange','pink','purple','cyan','magenta','white']: iterates over a list of colors.
turtle.color(colors) sets the turtle's pen color to the current color in the loop.
turtle.circle(100) draws a circle with a radius of 100 units using the current pen

SKILLBHAWAN GET YOUR STUDYROOM!!

color.
turtle.left(10) rotates the turtle 10 degrees to the left after drawing each circle, which creates the overlapping effect.
turtle.hideturtle()
Hides the turtle cursor (the little triangle shape that represents the drawing pen).
Explanation of the Drawing Process:
The code starts by setting up the turtle environment with a black background, a pen size of 2, and maximum drawing speed.
It then enters a loop that repeats 7 times.
Inside this loop, it iterates through a list of colors, drawing a circle for each color and rotating slightly between each circle.
Because of the rotation (turtle.left(10)), each subsequent circle overlaps with the previous ones, creating an intricate pattern.
After drawing all circles, turtle.hideturtle() is called to hide the turtle cursor from the final drawing.

SKILLBHAVAN GET YOUR STUDYROOM!!

ABCD Pattern :

Source code:

```
 import turtle
for   row   in   range(7):

for col in range(6):

    if((row == or row = 6) and (col <5) or (col ==e or col == 5) and (row
       > e and row <6)):

    print("*", end="")

  else:

  print(end = " ")

       print()
```

Notes:

Explanation:

1. **Outer Loop (for row in range(7):)**:
   - o This loop iterates through each row from 0 to 6.
2. **Inner Loop (for col in range(6):)**:
   - o This loop iterates through each column from 0 to 5 for each row.
3. **Condition (if ... else ...)**:
   - o The condition inside the if statement determines when to print an asterisk (*) and when to print a space ( ).
   - o ((row == 0 or row == 6) and (col < 5)): This checks if the current position is on the top or bottom row (row 0 or row 6) and if the column is less than 5 (columns 0 to 4). This condition prints asterisks for the top and bottom edges of the number '7'.
   - o ((col == 5 or col == 0) and (row > 0 and row < 6)): This checks if the current position is on the left or right edge (column 0 or column 5) and if the row is between 1 and 5 (to exclude the top and bottom edges). This condition prints asterisks for the vertical stroke of the number '7'.
   - o In both cases, when the conditions are met, an asterisk (*) is printed using print("*", end="").
4. **Default (else)**:
   - o If none of the conditions are met, a space ( ) is printed using print(end=" ").
5. **New Line (print())**:
   - o After printing each row (either asterisks or spaces), print() without any arguments is called to move to the next line for the next row.

SKIL BHAVAN GET YOUR STUDYROOM!!

Hexogonal Turtle Pattern :

Source code:

```
import turtle

import random

# Set up screen

screen = turtle.Screen()

screen.setup(width=800, height=800)

screen.bgcolor('black')

# Set up turtle

t = turtle.Turtle()

t.speed(0)

t.width(2)

t.hideturtle()

def draw_spirograph(size, color):

    t.color(color)

    for _ in range(360):

        t.circle(size)

        t.right(5)

colors = ['red', 'orange', 'yellow', 'green', 'blue', 'purple', 'cyan']

for i in range(50):

    color = random.choice(colors)

    size = random.randint(10, 100)

    draw_spirograph(size, color)

t.hideturtle()

screen.mainloop()
```

Notes:

- This code sets up a Turtle graphics environment where circles (representing spirograph patterns) of varying sizes and colors are drawn randomly on a black background.

SKILLBHAVAN GET YOUR STUDYROOM!!

- The draw_spirograph function handles the drawing logic for a single spirograph, using a loop to draw 360 circles with a given size and color.
- By using random colors and sizes, each iteration of the loop (for i in range(50)) produces a unique and colorful spirograph pattern.
- The overall effect is a visually dynamic display of spirograph-like patterns that vary in size and color, covering the entire 800x800 pixel screen.

This code demonstrates the creative potential of Turtle graphics and randomness in generating geometric art patterns.

Hexogonal Turtle Pattern :

Source code:

```
import turtle

import random

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed(0)

t.width(2)

def draw_spiral():

   for _ in range(200):

      t.pencolor(random.random(), random.random(), random.random())

      t.circle(100 + _)

      t.left(30)

      screen.bgcolor(random.random(), random.random(), random.random())

t.hideturtle()

draw_spiral()

screen.mainloop()
```

Notes:

- **Purpose**: The code creates a dynamic, colorful spiral pattern using the turtle graphics library.

SKILLBHAVAN GET YOUR STUDYROOM!!

- **Random Colors**: Both the pen color and background color change randomly with each iteration of the loop, creating a vibrant and continuously changing visual effect.
- **Spiral Pattern**: The turtle draws circles with increasing radii and turns left by 30 degrees after each circle, resulting in a spiral pattern.
- **Interactive Window**: The turtle graphics window remains open and interactive until manually closed by the user.

Key Concepts

- **turtle.Screen()**: Initializes a drawing window.
- **turtle.Turtle()**: Creates a turtle object for drawing.
- **turtle speed and width**: Controls the drawing speed and pen width.
- **Random Colors**: Uses random.random() to generate random RGB values for colors.
- **Loop and Drawing**: Uses a for-loop to draw the spiral pattern and change colors.
- **Interactive Graphics**: screen.mainloop() keeps the graphics window open and responsive.

SKILLBHAVAN
GET YOUR STUDYROOM!!

Typing Speed Game :

Source code:

```
from time import time
def tperror(prompt):
      global inwords
      words = prompt.split()
      error = 0
      for i in range(len(inwords)):
            if i in (0, len(inwords)-1):
                  if inwords[i] == words[i]:
                        continue
                  else:
                        error += 1
            else:
                  if inwords[i] == words[i]:
                        if    (inwords[i+1]    ==   words[i+1])    and   (inwords[i-1]     ==
words[i-1]):              continue

                        else:
                              error += 1
                  else:
                        error += 1
      return error
def calc_speed(inprompt, stime, etime):
      global inwords
      inwords = inprompt.split()
      twords = len(inwords)
      speed = twords / (etime - stime) * 60
      return speed
def elapsed_time(stime, etime):
elapsed = etime - stime
return elapsed
if __name__ == '__main__':
prompt = ("Python is an interpreted, high-level, general-purpose
programming language. Created by Guido van Rossum and first released in
1991, Python's design philosophy emphasizes code readability with its
notable use of significant whitespace. Its language constructs and object-
oriented approach.")

      print("Type this:- ", prompt,"")
      input("Press Enter when you are ready to check your speed!!!")
      stime = time()
      inprompt = input()
      etime = time()
      elapsed = round(elapsed_time(stime, etime), 2)
      typing_speed = calc_speed(inprompt, stime, etime)
      errors = tperror(prompt)
      print("Total time elapsed:", elapsed, "seconds")
      print("Your Average Typing speed was", typing_speed, "words per minute
(w/m)")
      print("with a total of", errors, "errors")
```

Notes:

This Python program measures typing speed and accuracy. It prompts the user to type a given text, calculates the typing speed in words per minute (wpm), and

SKILLBHAVAN GET YOUR STUDYROOM!!

counts the number of errors compared to the prompt. The elapsed time, typing speed, and number of errors are displayed to the user.

- **Purpose**: This code measures and reports your typing speed and accuracy.
- **Functionality**:
  - **Typing Speed**: Calculates the typing speed in words per minute (wpm).
  - **Typing Errors**: Counts the number of errors in the typed text compared to the given prompt.
  - **Elapsed Time**: Measures the total time taken to type the prompt.
- **Output**: Displays the total elapsed time, average typing speed in words per minute, and the total number of typing errors.

SKILLBHAVAN GET YOUR STUDYROOM!!

Rock-Paper Game :

Source code:

```
from tkinter import *

import random

def random_computer_choice():

    return random.choice(['rock', 'paper', 'scissors'])

def result(human_choice, comp_choice):

    global user_score

    global comp_score

    if human_choice == comp_choice:

        print("Tie")

    elif (human_choice == "rock" and comp_choice == "scissors") or \

         (human_choice == "paper" and comp_choice == "rock") or \

         (human_choice == "scissors" and comp_choice == "paper"):

        print("You Win")

        user_score += 1

    else:

        print("Computer wins")

        comp_score += 1

def rock():

    global user_choice

    global comp_choice

    user_choice = "rock"

    comp_choice = random_computer_choice()

    result(user_choice, comp_choice)

def paper():

    global user_choice

    global comp_choice

    user_choice = "paper"
```

SKILLBHAVAN
GET YOUR STUDYROOM!!

```
    comp_choice = random_computer_choice()

    result(user_choice, comp_choice)

def scissors():

    global user_choice

    global comp_choice

    user_choice = "scissors"

    comp_choice = random_computer_choice()

    result(user_choice, comp_choice)

# Create the Tkinter window

rps = Tk()

rps.geometry("300x300")

rps.title("Rock Paper Scissors")

# Initialize scores

user_score = 0

comp_score = 0

# Create buttons for each choice

button_rock = Button(text="Rock", bg="#888487", font=("arial", 15,
"italic bold"), relief=RIDGE, activebackground="#059458",
activeforeground="white", width=24, command=rock)

button_rock.grid(column=0, row=0)

button_paper = Button(text="Paper", bg="#808487", font=("arial", 15,
"italic bold"), relief=RIDGE,activebackground="#85945B",
activeforeground="white", width=24, command=paper)

button_paper.grid(column=0, row=1)

button_scissors = Button(text="Scissors", bg="#808487", font=("arial",
15, "italic bold"), relief=RIDGE,activebackground="#059458",
activeforeground="white", width=24, command=scissors)

button_scissors.grid(column=0, row=2)

# Start the main event loop

rps.mainloop()
```

Notes:

Importing Libraries

SKILLBHAVAN GET YOUR STUDYROOM!!

- **Tkinter**: Used for creating graphical user interfaces (GUI) in Python.
- **random**: Allows generation of random choices, used here for the computer's choices in the game.

Functions

Generate Random Computer Choice

- **random_computer_choice()**: Returns a random choice among 'rock', 'paper', and 'scissors' for the computer.

Determine the Result

- **result(human_choice, comp_choice)**: Compares the user's choice and the computer's choice to determine the result of each round.
    - o Handles ties, user wins, and computer wins based on the game rules.
    - o Updates global variables user_score and comp_score accordingly.

User Chooses Rock, Paper, or Scissors

- Functions rock(), paper(), and scissors():
    - Set user_choice based on the user's selection.
    - o Generate comp_choice using random_computer_choice().
    - o Call result(user_choice, comp_choice) to determine the outcome of the round.

GUI Setup

Create the Tkinter Window

- Initializes the main window (rps) for the Rock-Paper-Scissors game.
- Sets window dimensions (300x300) and title ("Rock Paper Scissors").

Initialize Scores

- Initializes user_score and comp_score to 0 to keep track of wins for both the user and the computer.

Create Buttons for User Choices

- Buttons (button_rock, button_paper, button_scissors) allow the user to select 'rock', 'paper', or 'scissors'.
- Each button is styled with specific background colors, font settings, relief (border style), and active backgrounds.
- Buttons are positioned in a grid layout within the Tkinter window (rps).

Start the Main Event Loop

SKILLBHAVAN GET YOUR STUDYROOM!!

- rps.mainloop(): Initiates the Tkinter event loop, which listens for user interactions (button clicks) and updates the GUI accordingly.

SKILLBHAVAN
GET YOUR STUDYROOM!!

Indian Flag :

Source code:

```
import turtle

import time

from turtle import *

speed(0)

setup(800,500)

penup()

goto(-400,250)

pendown()

color("orange")

begin_fill()

forward(800)

right(90)

forward(167)

right(90)

forward(800)

end_fill()

left(90)

forward(167)

color('green')

begin_fill()

forward(167)

left(90)

forward(800)

left(90)

forward(167)

end_fill()

penup()
```

SKILLBHAVAN GET YOUR STUDYROOM!!

```
goto(70,0)
pendown()
color("navy")
begin_fill()
circle(70)
end_fill()
penup()
goto(60,0)
color("white")
begin_fill()
circle(60)
end_fill()
penup()
goto(-57 , -8)
pendown()
color("navy")
for i in range (24):
    begin_fill()
    circle(3)
    end_fill()
    penup()
    forward(15)
    right(15)
    pendown()
penup()
goto(20,0)
pendown()
begin_fill()
circle(20)
```

SKILLBHAVAN GET YOUR STUDYROOM!!

```
end_fill()

penup()

goto(0,0)

pendown()

pensize(2)

for i in range(24):

    forward(60)

    backward(60)

    left(15)

hideturtle()
```

Notes:

Turtle Graphics Setup

- **Imports**:
  - o The code imports the turtle module, which provides functionality for drawing graphics using a turtle metaphor.
- **Speed and Setup**:
  - o speed(0): Sets the drawing speed to the maximum (fastest).
  - o setup(800, 500): Sets up the drawing window with dimensions 800 pixels wide and 500 pixels high.

Background Drawing

- **Orange Rectangle**:
  - o Draws a large orange rectangle across the top portion of the screen, establishing a background.
- **Green Rectangle**:
  - o Draws a smaller green rectangle below the orange one, extending down to fill the lower part of the window. This likely serves as a ground or base for the scene.

Sun Drawing

- **Sun**:
  - o Draws a sun shape using circles.
  - o A navy-colored circle is drawn first to outline the sun, followed by a smaller white circle inside to depict the sun's brightness.

Small Circles Around the Sun

SKILLBHAVAN IGNITE YOUR STUDYROOM!!

- **Decorative Circles**:
  - Draws multiple small navy-colored circles arranged in a circular pattern around the sun.
  - These circles add detail and decoration to the sun, enhancing its visual appeal.

Additional Decorative Patterns

- **Pattern Around the Sun**:
  - Creates a symmetrical pattern around the sun using alternating straight lines.
  - These lines likely form a geometric or floral design around the central sun shape, adding further artistic detail to the scene.

Hiding the Turtle

- **Hiding the Turtle**:
  - After completing the drawing, hides the turtle cursor from view.
  - This action ensures that only the final graphical output is visible without the distraction of the turtle icon.

Summary

- **Purpose**:
  - The code uses turtle graphics to generate a visual scene with colored rectangles, a sun, and decorative elements.
- **Design Elements**:
  - Utilizes circles and lines to create shapes and patterns, employing different colors (orange, green, navy, white) for visual distinction.
- **Technical Details**:
  - Relies on turtle-specific functions to position the drawing cursor and render various graphical elements.
  - Configures the drawing speed and window dimensions to control the appearance and performance of the graphical output.

SKILLBHAVAN
GET YOUR STUDY ROOM!!

Voting System :

Source code:

```
nominee1 = input("Enter the name of the 1st nominee: ")

nominee2 = input("Enter the name of the 2nd nominee: ")


# Initially, vote count for both must be 0

nm1_votes = 0

nm2_votes = 0


voter_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

no_of_voter = len(voter_id)


while True:

    voter = int(input("Enter your voter id: "))

    if voter in voter_id:

        print("You are a voter.")

        voter_id.remove(voter)

        vote = int(input(f"Enter your vote (1 for {nominee1}, 2 for
{nominee2}): "))

        if vote == 1:

            nm1_votes += 1

        elif vote == 2:

            nm2_votes += 1

        else:

            print("Invalid vote! Please enter 1 or 2.")

    else:

        print("You are not a registered voter or you have already
voted.")


    if len(voter_id) == 0:
```

SKILLBHAVAN GET YOUR STUDYROOM!!

```
        print("Voting session is over!!!")
        if nm1_votes > nm2_votes:
            percent = (nm1_votes / no_of_voter) * 100
            print(f"{nominee1} has won the election with {percent}% of votes.")
            break
        elif nm2_votes > nm1_votes:
            percent = (nm2_votes / no_of_voter) * 100
            print(f"{nominee2} has won the election with {percent}% of votes.")
            break
        else:
            print("Both have an equal number of votes! Government will decide the winner.")
            break
```

Notes:

Voting System Explanation

Initialization

- **Nominee Names**:
  - nominee1 and nominee2 are initialized to store the names of the two nominees. These names are input by the user at the beginning of the program.
- **Vote Count Initialization**:
  - nm1_votes and nm2_votes are initialized to zero. They will be used to keep track of the number of votes received by nominee1 and nominee2, respectively.
- **Voter IDs**:
  - voter_id is a list containing voter IDs [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], representing the registered voters.
  - no_of_voter stores the initial number of voters, which is the length of voter_id list (10 in this case).

Voting Process

- **While Loop**:
  - The program uses a while True loop to continuously prompt voters for their IDs and votes until all registered voters have cast their votes (len(voter_id) == 0).

SKILLBHAVAN
GET YOUR STUDYROOM!!

- **Input Voter ID**:
   - o Inside the loop, the user is prompted to enter their voter ID (voter).
   - o The program checks if the entered voter ID exists in the voter_id list.
- **Validation**:
   - If the voter ID is valid (i.e., exists in voter_id list):
     - The program removes the voter ID from the voter_id list to mark that the voter has cast their vote and cannot vote again.
     - The voter is then prompted to input their vote (1 for nominee1 or 2 for nominee2).
     - Based on the vote input (1 or 2), the respective nominee's vote count (nm1_votes or nm2_votes) is incremented.
- **Invalid or Duplicate Voting**:
   - If the entered voter ID is invalid (not in voter_id list) or has already been removed (indicating the voter has already voted), the program informs the user that they cannot vote again.

End of Voting Session

- **End Condition**:
   - o Once all voters have cast their votes (len(voter_id) == 0), the voting session ends.
   - o The program prints "Voting session is over!!!" to notify that no more votes can be cast.
   - o It then compares nm1_votes and nm2_votes to determine the winner:
     - If nm1_votes is greater than nm2_votes, nominee1 is declared the winner.
     - If nm2_votes is greater than nm1_votes, nominee2 is declared the winner.
     - The program calculates and prints the percentage of votes each nominee received.
- **Tie Condition**:
   - o If nm1_votes equals nm2_votes, the program declares a tie between nominee1 and nominee2.
   - o It suggests that the government or an external authority will decide the winner in the case of a tie.

Summary

- **Purpose**:
   - o The code simulates a basic voting system where registered voters can cast their votes for one of two nominees.
- **Functionality**:
   - o Manages voter registration, voting, and vote counting.

SKILLBHAVAN GET YOUR STUDYROOM!!

- Determines and declares the winner based on the vote counts or handles ties appropriately.

- **Features**:
    - Utilizes loops, conditional statements, lists, and input/output operations to simulate an election process.
    - Provides feedback to users on their voting status and election results through print statements.

SKILLBHAVAN
GET YOUR STUDYROOM!!

WordCloud :

Source code:

```
import sys
import numpy as np
from PIL import Image
import wikipedia #to extract information
from wordcloud import WordCloud, STOPWORDS

# We will import STOPWORDS to remove common words
a = str(input("Enter the name of which you want to make word cloud: "))
# Search for the title from Wikipedia
title = wikipedia.search(a)[0]
# Search the page related to the given topic on Wikipedia
page = wikipedia.page(title)
# Extract the content of that topic
text = page.content
print(text)

bg = np.array(Image.open("abcd.jpg"))

# Set of unwanted words
unwanted_words = set(STOPWORDS)
# Generate word cloud
wordcloud = WordCloud(background_color="black", max_words=400, mask=bg,
stopwords=unwanted_words)
wordcloud.generate(text)
# Save the word cloud
wordcloud.to_file("sample.png")
```

SKILLBHAVAN
GET YOUR STUDYROOM!!

Notes:

Libraries Used

- **sys**: Standard library used for system-specific parameters and functions.
- **numpy (np)**: Used for numerical operations and handling arrays.
- **PIL (Python Imaging Library)**: Imported as Image from PIL, used for handling images.
- **wikipedia**: Python library for interacting with Wikipedia API to fetch articles and content.
- **wordcloud**: A library used to create word clouds from text data.
- **STOPWORDS**: A predefined set of common words (like "the", "is", "and", etc.) that are typically filtered out from word clouds.

Steps in the Code

1. **User Input**
   - o Prompts the user to enter the name/topic for which they want to generate a word cloud (a).

2. **Search on Wikipedia**
    - Uses wikipedia.search(a) to find the most relevant Wikipedia article title related to the user's input.
    - Selects the first result from the search (wikipedia.search(a)[0]) as the title.
3. **Fetch Wikipedia Page**
    - Retrieves the Wikipedia page content using wikipedia.page(title), where title is the title of the Wikipedia article.
    - Extracts the full content of the Wikipedia page (text = page.content).
4. **Text Preprocessing**
    - Converts the extracted Wikipedia content into a string (text).
5. **Image for Masking**
    - Loads an image (abcd.jpg) to be used as a mask for the word cloud. This image will define the shape of the word cloud.
6. **Stopwords Removal**
    - Initializes unwanted_words with the set of STOPWORDS imported from the wordcloud library. These words will be excluded from the word cloud visualization.
7. **Word Cloud Generation**
    - Creates a WordCloud object:
        - background_color="black": Sets the background color of the word cloud to black.
        - max_words=400: Limits the maximum number of words displayed in the word cloud to 400.
        - mask=bg: Uses bg (the loaded image) as the mask for shaping the word cloud.
        - stopwords=unwanted_words: Excludes the stopwords from appearing in the word cloud.
8. **Generate and Save Word Cloud**
    - Generates the word cloud using wordcloud.generate(text) with the extracted Wikipedia content.
    - Saves the generated word cloud as an image file (sample.png) using wordcloud.to_file("sample.png").

Related Concepts

- **Word Cloud**:
    - A visual representation of text data where the size of each word indicates its frequency or importance within the text.
    - Commonly used for visualizing textual data to quickly understand the most prominent terms.
- **Text Mining and Natural Language Processing (NLP)**:

SKILLBHAWAN GET YOUR STUDYROOM!!

- o Techniques used to extract and analyze information from large amounts of textual data.
  - o In this case, the script uses Wikipedia's content and NLP techniques to generate meaningful insights (word cloud) from the text.

 Image Masking:
  - o Technique where an image is used as a mask to shape the appearance of another image or visual representation.
  - o In the context of word clouds, the mask defines the outline or shape of the cloud, allowing for customized visualizations.

 Stopwords:
  - o Commonly used words (like articles, prepositions, etc.) that are filtered out before or after processing of natural language data.
  - o Helps to focus on the more meaningful and relevant words in text analysis and visualization tasks.

Summary

 The code combines data retrieval from Wikipedia, text processing techniques, and visualization using the wordcloud library to create a customized word cloud based on user input.
 It demonstrates how Python libraries can be integrated to fetch, process, and visualize textual data efficiently, offering insights into the most significant terms related to a given topic sourced from Wikipedia.

SKILLBHAVAN GET YOUR STUDY ROOM!!

Image to Pencil :

Source code:

```
import numpy as np

import imageio

import scipy.ndimage

import cv2


img = "viru.jpg"     # Removed extra space before the filename


def rgb2gray(rgb):

      return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])


def dodge(front, back):

      final_sketch = front * 255 / (255 - back)        # Added '*' operator

      final_sketch[final_sketch > 255] = 255

      final_sketch[back == 255] = 255

      return final_sketch.astype('uint8')


ss = imageio.imread(img)   # Corrected variable name to 'ss'

gray = rgb2gray(ss)

i = 255 - gray

blur = scipy.ndimage.filters.gaussian_filter(i, sigma=15)

r = dodge(blur, gray)


cv2.imwrite('virat-sketch.png', r)
```

Notes:

Libraries Used

- **numpy (np)**: Used for numerical operations on arrays.
- **imageio**: Library for reading and writing image data.

SKILLBHAVAN GET YOUR STUDYROOM!!

 scipy.ndimage: Part of SciPy for image processing operations like filtering.
 cv2: OpenCV library for computer vision tasks.

Functions Defined

1. **rgb2gray(rgb) Function**
   - o Converts a color image (rgb) to grayscale using the formula gray = np.dot(rgb[..., :3], [0.2989, 0.5870, 0.1140]).
   - o This formula computes a weighted sum of the RGB channels to approximate the perceived intensity of the grayscale image.
2. **dodge(front, back) Function**
   - o Implements the dodge blending technique to create the pencil sketch effect.
   - o front is the blurred version of the inverted grayscale image.
   - o back is the original grayscale image.
   - o Formula used: final_sketch = front * 255 / (255 - back).
   - o Ensures pixel values are within the valid range (0 to 255) using array operations.
   - o Returns the final sketch image as an unsigned 8-bit integer array (uint8).

Image Processing Steps

1. **Read Image**
   - o Uses imageio.imread(img) to read the input image (viru.jpg) into a NumPy array (ss).
2. **Convert to Grayscale**
   - o Applies the rgb2gray function to convert the color image (ss) into a grayscale image (gray).
3. **Invert Grayscale Image**
   - o Inverts the grayscale image (gray) by subtracting its values from 255 (i = 255 - gray).
   - o Inversion enhances the contrast and prepares the image for the dodge blending.
4. **Gaussian Blur**
   - o Applies Gaussian blur to the inverted grayscale image (i) using scipy.ndimage.filters.gaussian_filter.
   - o sigma=15 controls the standard deviation of the Gaussian kernel, influencing the blur strength (blur).
5. **Apply Dodge Blending**
   - o Uses the dodge function to blend the blurred image (blur) with the original grayscale image (gray).
   - o Produces the final pencil sketch effect (r).
6. **Save Output**

SKILLBHAVAN GET YOUR STUDYROOM

- o Saves the resulting pencil sketch image (r) as virat-sketch.png using cv2.imwrite.

Related Concepts

- **Image Processing Techniques**:
  - o **Grayscale Conversion**: Transforming color images into grayscale simplifies processing and reduces data complexity.
  - ○ **Inversion**: Inverting grayscale intensities enhances edges and details for certain effects.
  - o **Gaussian Blur**: Smoothing technique to reduce noise and detail, useful for preparing images for further processing.
  - o **Dodge Blending**: Technique to blend two images for special effects, such as creating pencil sketches.
- **Library Usage**:
  - o Demonstrates integration of different libraries (numpy, imageio, scipy, cv2) for efficient image processing tasks.
  - ○ Each library contributes specific functionalities crucial for different stages of the image transformation pipeline.

Summary

 The code leverages Python libraries to transform a color photograph into a pencil sketch effect through a series of well-defined image processing steps.
 Illustrates fundamental concepts like grayscale conversion, image inversion, Gaussian blurring, and dodge blending in the context of creative image manipulation.
 Provides practical insight into using Python for digital image processing and artistic rendering, enhancing understanding of computational photography techniques.

SKILLBHAWAN GET YOUR STUDYROOM!!

Web Map :

Source code:

```
import folium

# Create a feature group for the GeoJson data

fg = folium.FeatureGroup("map")

# Add GeoJson data to the feature group

fg.add_child(folium.GeoJson(data=open("india_states.json", "r",
encoding="utf-8-sig").read()))


# Create a map object

map = folium.Map(location=[20.0000, 75.0000], zoom_start=4)


# Add the feature group to the map

map.add_child(fg)


# Save the map to HTML file

map.save("final.html")
```

SKILLBHAVAN GET YOUR STUDYROOM!!

Notes:

Folium Library

 Folium is a Python library used for creating interactive maps that are rendered as HTML files. It is built upon the Leaflet.js library and allows for easy integration of geographical data and visualization.

Code Breakdown

1. **Importing Folium**
   - o Imports the Folium library using import folium.
2. **Creating a Feature Group for GeoJson Data**
   - o **Feature Group**: A collection of features that can be added to a map as a single layer.
   - o fg = folium.FeatureGroup("map"): Creates a new Feature Group named "map".
3. **Adding GeoJson Data**

o **GeoJson Data**: A format for encoding a variety of geographic data structures.

o open("india_states.json", "r", encoding="utf-8-sig").read(): Opens and reads a GeoJson file (india_states.json) containing state boundaries of India.

o fg.add_child(folium.GeoJson(data=geojson_data)): Adds the GeoJson data to the Feature Group (fg) using folium.GeoJson.

4. **Creating a Map Object**
   - o **Map Object**: Represents the entire map visualization.
   - o map = folium.Map(location=[20.0000, 75.0000], zoom_start=4): Creates a map centered at latitude 20.0000 and longitude 75.0000, with an initial zoom level of 4.
5. **Adding Feature Group to the Map**
   - o map.add_child(fg): Adds the previously created Feature Group (fg) containing GeoJson data to the map (map).
6. **Saving the Map**
   - o **HTML Output**: Folium generates an HTML file that renders the interactive map.
   - o map.save("final.html"): Saves the map (map) as an HTML file named final.html.

SKILLBHAVAN GET YOUR STUDYROOM!!

Related Concepts

 GeoJson Format:
- o **GeoJson** (Geographic JavaScript Object Notation) is an open standard format designed for representing simple geographical features, along with their properties.
- o Used widely for encoding a variety of geographic data structures including points, lines, polygons, and their combinations.

 Interactive Mapping:
- o Folium allows for the creation of interactive maps with features like zooming, panning, and adding layers.
- o Suitable for visualizing spatial data such as population distributions, weather patterns, land use, and more.

 Leaflet.js Integration:
- o Folium leverages Leaflet.js, a leading open-source JavaScript library for interactive maps, providing extensive customization options and seamless integration with Python.

Summary

 The code snippet demonstrates how to use Folium to create an interactive map displaying state boundaries of India from GeoJson data.

 It illustrates key steps such as creating a map object, adding GeoJson data as a feature group, and saving the interactive map as an HTML file.

- Emphasizes the utility of Folium in visualizing geographic data and its integration with Python for spatial analysis and presentation.

SKILLBHAVAN
GET YOUR STUDYROOM!!