AI -Startup Assignment

Use Python to solve the following challenges-

1. To guess a randomly selected word within a limited number of tries.
2. To implement Rock Paper Scissor game.
3. Check if two PDF/Text documents are identical.
4. Create a Screen recorder.
5. Cows and Bulls

   [Hint: Cows and Bulls is a pen and paper code-breaking game usually played between 2 players. In this, a player tries to guess a secret code number chosen by the second player. The rules are as follows:

   - A player will create a secret code, usually a 4-digit number. This number should have no repeated digits.
   - Another player makes a guess (4 digit number) to crack the secret number. Upon making a guess, 2 hints will be provided- Cows and Bulls.
   - Bulls indicate the number of correct digits in the correct position and cows indicates the number of correct digits in the wrong position. For example, if the secret code is 1234 and the guessed number is 1246 then we have 2 BULLS (for the exact matches of digits 1 and 2) and 1 COW (for the match of digit 4 in the wrong position)
   - The player keeps on guessing until the secret code is cracked. The player who guesses in the minimum number of tries wins.]
6. To Create a Countdown Timer Using Python.
7. Create a payment receipts as shown below [use *reportlab* library]-

Date	Name	Subscription	Price (Rs.)
16/11/2020	Full Stack Development with React & Node JS - Live	Lifetime	10,999.00/-
16/11/2020	Geeks Classes: Live Session	6 months	9,999.00/-
Sub Total			20,9998.00/-
Discount			-3,000.00/-
Total			17,998.00/-

Tentative Solution

1

```
import random
name = input("What is your name? ")
print("Good Luck ! ", name)
words = ['rainbow', 'computer', 'science', 'programming',
        'python', 'mathematics', 'player', 'condition',
        'reverse', 'water', 'board', 'geeks']
word = random.choice(words)
print("Guess the characters")
guesses = ''
turns = 12
while turns > 0:
    failed = 0
    for char in word:
        if char in guesses:
            print(char, end=" ")
        else:
            print("_")
            failed += 1
    if failed == 0:
        print("You Win")
        print("The word is: ", word)
        break
    print()
    guess = input("guess a character:")
    guesses += guess
    if guess not in word:
        turns -= 1
        print("Wrong")
        print("You have", + turns, 'more guesses')
        if turns == 0:
            print("You Loose")
```

2

```
import random
# Print multiline instruction
print('Winning rules of the game ROCK PAPER SCISSORS are:\n'
      + "Rock vs Paper -> Paper wins \n"
      + "Rock vs Scissors -> Rock wins \n"
      + "Paper vs Scissors -> Scissors wins \n")
while True:
    print("Enter your choice \n 1 - Rock \n 2 - Paper \n 3 - Scissors \n")
    # Take the input from user
    choice = int(input("Enter your choice: "))
    # Looping until user enters valid input
    while choice > 3 or choice < 1:
        choice = int(input('Enter a valid choice please : '))
```

```
    # Initialize value of choice_name variable corresponding to the choice value
    if choice == 1:
        choice_name = 'Rock'
    elif choice == 2:
        choice_name = 'Paper'
    else:
        choice_name = 'Scissors'
    # Print user choice
    print('User choice is:', choice_name)
    print("Now it's Computer's Turn...")

    # Computer chooses randomly any number among 1, 2, and 3
    comp_choice = random.randint(1, 3)
    # Initialize value of comp_choice_name variable corresponding to the choice
value
    if comp_choice == 1:
        comp_choice_name = 'Rock'
    elif comp_choice == 2:
        comp_choice_name = 'Paper'
    else:
        comp_choice_name = 'Scissors'
    print("Computer choice is:", comp_choice_name)
    print(choice_name, 'vs', comp_choice_name)
    # Determine the winner
    if choice == comp_choice:
        result = "DRAW"
    elif (choice == 1 and comp_choice == 2) or (comp_choice == 1 and choice == 2):
        result = 'Paper'
    elif (choice == 1 and comp_choice == 3) or (comp_choice == 1 and choice == 3):
        result = 'Rock'
    elif (choice == 2 and comp_choice == 3) or (comp_choice == 2 and choice == 3):
        result = 'Scissors'
    # Print the result
    if result == "DRAW":
        print("<== It's a tie! ==>")
    elif result == choice_name:
        print("<== User wins! ==>")
    else:
        print("<== Computer wins! ==>")
    # Ask if the user wants to play again
    print("Do you want to play again? (Y/N)")
    ans = input().lower()
    if ans == 'n':
        break
# After coming out of the while loop, print thanks for playing
print("Thanks for playing!")
```

3

```
import hashlib
```

```
from difflib import SequenceMatcher
def hash_file(fileName1, fileName2):
   # Use hashlib to store the hash of a file
   h1 = hashlib.sha1()
   h2 = hashlib.sha1()
   with open(fileName1, "rb") as file:
      # Use file.read() to read the size of file
      # and read the file in small chunks
      # because we cannot read the large files.
      chunk = 0
      while chunk != b'':
         chunk = file.read(1024)
         h1.update(chunk)

   with open(fileName2, "rb") as file:
      # Use file.read() to read the size of file a
      # and read the file in small chunks
      # because we cannot read the large files.
      chunk = 0
      while chunk != b'':
         chunk = file.read(1024)
         h2.update(chunk)
      # hexdigest() is of 160 bits
      return h1.hexdigest(), h2.hexdigest()
msg1, msg2 = hash_file("pd1.pdf ", "pd1.pdf")
if(msg1 != msg2):
   print("These files are not identical")
else:
   print("These files are identical")
```

4

```
# importing the required packages
import pyautogui
import cv2
import numpy as np
# Specify resolution
resolution = (1920, 1080)
# Specify video codec
codec = cv2.VideoWriter_fourcc(*"XVID")
# Specify name of Output file
filename = "Recording.avi"
# Specify frames rate. We can choose any
# value and experiment with it
fps = 60.0
# Creating a VideoWriter object
out = cv2.VideoWriter(filename, codec, fps, resolution)
# Create an Empty window
cv2.namedWindow("Live", cv2.WINDOW_NORMAL)
```

```
# Resize this window
cv2.resizeWindow("Live", 480, 270)
while True:
    # Take screenshot using PyAutoGUI
    img = pyautogui.screenshot()
    # Convert the screenshot to a numpy array
    frame = np.array(img)
    # Convert it from BGR(Blue, Green, Red) to
    # RGB(Red, Green, Blue)
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Write it to the output file
    out.write(frame)
    # Optional: Display the recording screen
    cv2.imshow('Live', frame)
     # Stop recording when we press 'q'
    if cv2.waitKey(1) == ord('q'):
        break
# Release the Video writer
out.release()
# Destroy all windows
cv2.destroyAllWindows()
```

5

```
# Import required module
import random
# Returns list of digits
# of a number
def getDigits(num):
    return [int(i) for i in str(num)]

# Returns True if number has
# no duplicate digits
# otherwise False
def noDuplicates(num):
    num_li = getDigits(num)
    if len(num_li) == len(set(num_li)):
        return True
    else:
        return False
# Generates a 4 digit number
# with no repeated digits
def generateNum():
    while True:
        num = random.randint(1000,9999)
        if noDuplicates(num):
            return num
# Returns common digits with exact
# matches (bulls) and the common
# digits in wrong position (cows)
```

	def numOfBullsCows(num,guess):
	  bull_cow = [0,0]
	  num_li = getDigits(num)
	  guess_li = getDigits(guess)
	  for i,j in zip(num_li,guess_li):
	  # common digit present
	  if j in num_li:
	  # common digit exact match
	  if j == i:
	  bull_cow[0] += 1
	  # common digit match but in wrong position
	  else:
	  bull_cow[1] += 1
	  return bull_cow
	# Secret Code
	num = generateNum()
	tries =int(input('Enter number of tries: '))
	# Play game until correct guess
	# or till no tries left
	while tries > 0:
	  guess = int(input("Enter your guess: "))
	  if not noDuplicates(guess):
	  print("Number should not have repeated digits. Try again.")
	  continue
	  if guess < 1000 or guess > 9999:
	  print("Enter 4 digit number only. Try again.")
	  continue
	  bull_cow = numOfBullsCows(num,guess)
	  print(f"{bull_cow[0]} bulls, {bull_cow[1]} cows")
	  tries -=1
	  if bull_cow[0] == 4:
	  print("You guessed right!")
	  break
	else:
	  print(f"You ran out of tries. Number was {num}")

6	import time
	def countdown(t):
	  while t:
	  mins, secs = divmod(t, 60)
	  timer = '{:02d}:{:02d}'.format(mins, secs)
	  print(timer, end='\r') # Overwrite the line each second
	  time.sleep(1)
	  t -= 1
	  print("Fire in the hole!!")
	t = input("Enter the time in seconds: ")
	countdown(int(t))

7	# imports module

```
from reportlab.platypus import SimpleDocTemplate, Table, Paragraph, TableStyle
from reportlab.lib import colors
from reportlab.lib.pagesizes import A4
from reportlab.lib.styles import getSampleStyleSheet
# data which we are going to display as tables
DATA = [
   [ "Date" , "Name", "Subscription", "Price (Rs.)" ],
   [
      "16/11/2020",
      "Full Stack Development with React & Node JS - Live",
      "Lifetime",
      "10,999.00/-",
   ],
   [ "16/11/2020", "Geeks Classes: Live Session", "6 months", "9,999.00/-"],
   [ "Sub Total", "", "", "20,9998.00/-"],
   [ "Discount", "", "", "-3,000.00/-"],
   [ "Total", "", "", "17,998.00/-"],
]
# creating a Base Document Template of page size A4
pdf = SimpleDocTemplate( "receipt.pdf" , pagesize = A4 )
# standard stylesheet defined within reportlab itself
styles = getSampleStyleSheet()
# fetching the style of Top level heading (Heading1)
title_style = styles[ "Heading1" ]
# 0: left, 1: center, 2: right
title_style.alignment = 1
# creating the paragraph with
# the heading text and passing the styles of it
title = Paragraph( "GeeksforGeeks" , title_style )
# creates a Table Style object and in it,
# defines the styles row wise
# the tuples which look like coordinates
# are nothing but rows and columns
style = TableStyle(
 [
      ( "BOX" , ( 0, 0 ), ( -1, -1 ), 1 , colors.black ),
      ( "GRID" , ( 0, 0 ), ( 4 , 4 ), 1 , colors.black ),
      ( "BACKGROUND" , ( 0, 0 ), ( 3, 0 ), colors.gray ),
      ( "TEXTCOLOR" , ( 0, 0 ), ( -1, 0 ), colors.whitesmoke ),
      ( "ALIGN" , ( 0, 0 ), ( -1, -1 ), "CENTER" ),
      ( "BACKGROUND" , ( 0 , 1 ) , ( -1 , -1 ), colors.beige ),
   ]
)
# creates a table object and passes the style to it
table = Table( DATA , style = style )
# final step which builds the
# actual pdf putting together all the elements
```

	pdf.build([title , table])