TASK 1

Hangman game

```
import random
def choose_word():
   words = ['python', 'hangman', 'programming', 'computer', 'science']     return
random.choice(words)
def display_word(word, guessed_letters):
   return ' '.join([letter if letter in guessed_letters else '_' for letter in word])
def hangman():
   word = choose_word()     guessed_letters = set()     attempts = 6
  print("Welcome to Hangman!")
  print(display_word(word, guessed_letters))
    while attempts > 0:
      guess = input("Guess a letter: ").lower()
     if guess in guessed_letters:
         print("You already guessed that letter. Try again.")        elif guess in word:
      guessed_letters.add(guess)
   print("Good guess!")
  else:
         attempts -= 1
 print(f"Incorrect! You have {attempts} attempts left.")
       current_display = display_word(word, guessed_letters)
    print(current_display)
      if '_' not in current_display:
         print("Congratulations! You've guessed the word.")
     break
   else:
      print(f"Game over! The word was '{word}'.")
    if __name__ == "__main__":
   hangman()
```

OUTPUT

```
      Welcome to Hangman!
_ _ _ Guess a letter: d Good guess! d _ _
Guess a letter: o Good guess! d o _
Guess a letter: g Good guess! d o g

Congratulations! You've guessed the word.


===  Code Execution Successful  ===
```

TASK2

Stock Portfolio Tracker

```
pip install request
 import requests
import json
 API_KEY = 'YOUR_ALPHA_VANTAGE_API_KEY'  # Replace with your Alpha Vantage API key
 def get_stock_price(symbol):
    url = f'https://www.alphavantage.co/query'
params = {
        'function': 'TIME_SERIES_INTRADAY',
        'symbol': symbol,
        'interval': '1min',
        'apikey': API_KEY
    }
    response = requests.get(url, params=params)
data = response.json()      try:
        latest_time = list(data['Time Series (1min)'].keys())[0]
latest_close = data['Time Series (1min)'][latest_time]['4. close']
return float(latest_close)     except KeyError:         print(f"Error
fetching data for {symbol}.")         return None


class Portfolio:
def __init__(self):
self.stocks = {}


    def add_stock(self, symbol, quantity):
price = get_stock_price(symbol)          if
price is not None:
            self.stocks[symbol] = {
'quantity': quantity,
'purchase_price': price
            }
            print(f"Added {quantity} of {symbol} at {price} each.")
else:
```

```
            print(f"Failed to add {symbol} to portfolio.")


    def remove_stock(self, symbol):         if symbol
in self.stocks:             del self.stocks[symbol]
print(f"Removed {symbol} from portfolio.")
else:

   print(f"{symbol} not found in portfolio.")
 def view_portfolio(self):
     print("Portfolio:")

for symbol, details in self.stocks.items():
current_price = get_stock_price(symbol)
if current_price is not None:

                total_value = details['quantity'] * current_price
print(f"{symbol}: Quantity = {details['quantity']}, Purchase Price =

{ details['purchase_price']}, Current Price = {current_price}, Total Value = {total_value}" )
else:

                print(f"{symbol}: Unable to fetch current price.")


def main():     portfolio
= Portfolio()     while
True:

        print("\nOptions:")

     print("1. Add Stock")

   print("2. Remove Stock")

     print("3. View Portfolio")

    print("4. Exit")

     choice = input("Enter choice (1/2/3/4): ")


        if choice == '1':
            symbol = input("Enter stock symbol: ").

     upper()

           quantity = int(input("Enter quantity: "))
portfolio.add_stock(symbol, quantity)

        elif choice == '2':

            symbol = input("Enter stock symbol to remove: ").
upper()
```

```
    portfolio.remove_stock(symbol)
    elif choice == '3':
         portfolio.view_portfolio()
elif choice == '4':

         break
else:

         print("Invalid choice. Please try again.")


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

TASK 3

Basic chatbot

```
import nltk from nltk.chat.util import Chat,
reflections


# Define pairs of patterns and responses
pairs = [
    (r'hi|hello|hey there',
     ['Hello!How can I assist you today', 'Hey!', 'Hi!']),
    (r'how are you?',
     ['I am good, thank you.', 'Doing well, thanks!']),
    (r'(.*) (age|old) are you',
     ["As an AI language model, I don't have a physical form or age like humans do. I exist to provide information and assist with your questions to the best of my abilities. If there's anything specific you'd like to know or discuss, feel free to ask!."]),
    (r'quit',
     ['Bye!', 'Goodbye.', 'Have a nice day!']),
]


# Define reflections for pronoun replacement
reflections = {     "i am": "you are",
    "i was": "you were",
    "i": "you",
    "i'm": "you are",
    "i'd": "you would",
    "i've": "you have",
    "i'll": "you will",
    'my': 'your',
    'you are': 'I am',
    'you were': 'I was',
    "you've": "I have",
    "you'll": "I will",
    'your': 'my',
    'yours': 'mine',
    'you': 'me',
    'me': 'you'
}

def chatbot():
   print("Hello! I'm your chatbot. How can I help you today?")
chat = Chat(pairs, reflections)     while True:
      user_input = input("You: ")
if user_input.lower() == 'quit':
         break        response =
chat.respond(user_input)
print("Bot:", response)

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

output

```
Hello! I'm your chatbot. How can I help you today?
You: Hi there
Bot: Hello!How can I assist you today
You: How are you?
Bot: Doing well, thanks!
You: How old are you?
Bot: As an AI language model, I don't have a physical form or age like humans do. I exist to
provide information and assist with your questions to the best of my abilities. If there's anything
specific you'd like to know or discuss, feel free to ask!.
You: Bye
Bot: Bye!
```