Viral Shaileshkumar Jariwala

Question 1 :

The three gestures used in base Rock Paper Scissors are rock, paper, and scissors. The way these are scored is as such: Rock beats Scissors, Scissors beats Paper, Paper beats Rock. It gets a lot more complicated when you introduce new gestures, but let's keep it simple for now.

We're definitely going to need a way to decide who has won and who has lost, or whether the round has ended in a draw. Using the rules provided, give us an engine for deciding this based on the player's moves.

Rock Beats Scissors

As a player, I want rock to beat scissors. So that I can play with rules I'm familiar with.

• Given I have chosen rock When the opponent chooses scissors Then I should win.

• Given I have chosen scissors When the opponent chooses rock Then they should win

Scissors Beats Paper

As a player, I want scissors to beat paper. So that I can play with rules I'm familiar with.

• Given I have chosen scissors When the opponent chooses paper Then I should win

• Given I have chosen paper When the opponent chooses scissors Then they should win.

Paper Beats Rock

As a player, I want paper to beat rock. So that I can play with rules I'm familiar with.

• Given I have chosen paper When the opponent chooses rock Then I should win

• Given I have chosen rock When the opponent chooses paper Then they should win.

Same move results in Draw

As a player, I want the same moves to draw. So that I can play with rules I'm familiar with.

• Given I have chosen rock When the opponent chooses rock

• Given I have chosen paper When the opponent chooses paper Then it should be a draw

• Given I have chosen scissors When the opponent chooses scissors Then it should be a draw

Some rules to keep in mind

• In a Single file the developer needs to store the player's name and the highest score

• Developer need to display the highest score when any person starts a new game

• If any player beats the highest score then his/her score should be updated as the highest score for the game

• The score cannot be in negative value and the file in which developer is storing user's data should provide the information when it is required to be fetched and if not able to find then proper handling of this scenario should be there

• Until and unless user wants to quit the game the playing option should be available

Answer :

import os

def who_wins(player1, player2):

   if player1 == player2:

      return "draw"

   elif player1 == "rock" and player2 == "scissors":

      return "player1"

   elif player1 == "scissors" and player2 == "paper":

      return "player1"

   elif player1 == "paper" and player2 == "rock":

      return "player1"

   else:

      return "player2"

def get_high_score(filename):

   if os.path.exists(filename):

      with open(filename, "r") as file:

         return int(file.read())

   else:

      return 0

```python
def update_high_score(filename, score):
    with open(filename, "w") as file:
        file.write(str(score))


def play_game():
    player1 = input("Player 1, make your move (rock/paper/scissors): ")
    player2 = input("Player 2, make your move (rock/paper/scissors): ")
    result = who_wins(player1, player2)
    if result == "draw":
        print("It's a draw!")
    elif result == "player1":
        print("Player 1 wins!")
    else:
        print("Player 2 wins!")


def main():
    high_score = get_high_score("high_score.txt")
    print("The current high score is:", high_score)
    while True:
        play_game()
        player1_score = int(input("Player 1, enter your score: "))
        player2_score = int(input("Player 2, enter your score: "))
        if player1_score < 0 or player2_score < 0:
            print("Score cannot be negative!")
            continue
        if player1_score > high_score:
            update_high_score("high_score.txt", player1_score)
            high_score = player1_score
```

```python
        elif player2_score > high_score:

            update_high_score("high_score.txt", player2_score)

            high_score = player2_score

        answer = input("Do you want to continue playing? (y/n): ")

        if answer == "n":

            break


if name == "main":

    main()
```

Question 2 :

You need to write the software to calculate the minimum number of coins required to return an amount of change to a user of Acme Vending machines. For example, the vending machine has coins 1,2,5 and 10 what is the minimum number of coins required to make up the change of 43 cents?

The coin denominations will be supplied as a parameter. This is so the algorithm is not specific to one country. You may not hardcode these into the algorithm, they must be passed as a parameter.

The country's denominations to use for the Question are:

● British Pound ○ 1,2,5,10,20,50

● US Dollar ○ 1,5,10,25

● Norwegian Krone ○ 1,5,10,20

The Question assumes an infinite number of coins of each denomination. You are to return an array with each coin to be given as change.

Increment- Remove the assumption that there are infinite coins of each denomination. Modify the code to accept a fixed number of each denomination. It will affect the change calculation in that you now need to consider the availability of coins when calculating change.

Answer :

Question 3 :

There are varying opinions amongst writers as to when one should spell out a number vs. using figures. However there are a few generally agreed upon rules for doing so. 1 There are 2 of them:

● Numbers beginning a sentence should be spelt out.

● Numbers beginning a sentence should be spelt out.

Write an implementation of these rules for numbers starting a sentence. It should take an input of figures and return the number spelt out. Assume all the numbers given start sentences and your code will be used like a spell checker for grammar issues related to numbers. Only worry about up numbers up to 4 digits long. Assume all numbers are positive.

Examples

 1 digit

 0 = zero

 5 = five

 8 = eight

 2 digits

 10 = ten

 21 = twenty-one

 77 = seventy-seven

 3 digits

 100 = one hundred

 303 = three hundred three

 555 = five hundred fifty-five

 4 digits

 2000 = two thousand

 3466 = three thousand four hundred sixty-six

 2400 = two thousand four hundred

additional rule: The simplest way to express large numbers is usually best. Instead of 5300 being equal to five thousand three hundred it should return fifty-three hundred

Answer :