HW 6 – def jam()

Objective

Begin to master the art of writing functions to create reusable code.

General Rules

- Any functions named in the assignment must exist and implement the given functionality
  - You can write additional functions as needed to make your life easier
- Module names should be exactly as specified
- Quit – anytime a user responds with 'Quit' in any case (quit, Quit, qUit) – leave the current game and say "Thank you for playing:" <game_name>
  - (for example – "Thank you for playing: coin_toss")
- Validation – in this code, if we are validating, then we
  - Ask for the input
  - Handle exceptions and invalid input by repeating the request for input
  - Repeat until the input is good or the user types "Quit" (see above)
- Win_message
  - Any time a user wins a game, tell them "Congratulations, you win: " <game_name>
- Lose_message
  - Any time a user wins a game, tell them "I'm so sorry you lost at: " <game_name> ". I bet you'll win next time"
- Optional: If you choose to create an additional module to handle the general rules, please name it casino_tools.py

Imports

- **Module:** coin_toss.py

  Description: Coin_toss contains functionality to support flipping a virtual coin.

  Implementation:

  - **Function** flip() – returns random choice of 'Heads' or 'Tails' – see the Code Samples page in Canvas – you can download the "starter" file
  - **Function** play_game()
    - Asks the user "Heads or Tails"
      - Validates – response must be heads or tails
    - The computer calls flip
- **Module:** roulette.py

  Description: Plays a simplified Roulette game.

  Implementation:

  - **Function** get_random_color()
    - return random choice of 'Black' or 'Red'

- **Function** get_num()
    - return a random number between 0 and 36
- **Function** play_game()
    - Asks the user to choose Red or Black
        - Validates response – must be Red or Black
    - Asks the user to choose a random integer between 0 and 36
        - Validates response
            - Must be integer
            - Must be between 0 and 36
    - Tell the user "I'm spinning the wheel now"
    - Calls get_num() and get_random_color()
    - Compare the random results against the user choices
        - Prints win message if both color and number match user choices
        - Otherwise, prints lose message
- NOTE: Roulette will be really hard to test, since you basically will have odds of 1:72 of winning. You may have to figure out a way to "cheat" at first to make sure the if statements work

- **Module:** casino_games.py

  Description: This is a "driver program" for the other gambling code we have written – it just loops, allowing the user to play one of the games.

  Implementation:

    - Imports coin_toss and roulette
    - Choice – do you want to?
        - 1. Play Roulette
        - 2. Flip a coin
    - Validate the choice – has to be 1, 2 or Quit
    - Once the user has chosen 1 or 2, call the appropriate game (using the play_game() function from the appropriate imported module)
    - Then repeat until the user chooses "Quit"
        - Use the quit message for casino_games