× **Hopper RLHF Assesssment Quiz** NB ▾

Assessment Task 3

This is an assessment task. The Assessment Exam is designed to comprehensively assess your knowledge and understanding of our project's requirements.

You will be assessed on your numerical ratings, explanations, and the ideal response you write (if there are issues in the response provided). Always be specific about any issues you find in the given response!

Please take this seriously and attempt it as you might do on a real task - if you fail, you will not be able to access the real tasks.

Remember, the use of LLMs, chatbots, AI tools, plagiarism, or copy/pasting responses is strictly prohibited and is a violation of the Outlier Terms of Service. This will lead to your removal from the project and the deactivation of your Outlier account.

Please read the prompt carefully:

```
Our website has a 400x500 space we’d like to replace with something fun.
Please build a Tic Tac Toe game with those exact dimensions. Build it in
python using the pygame library. Do not include the implementation for
embedding it on the website, we’ll handle that part, the game should work
by itself. The game should display a legible state (whose turn it is or win
/lose/draw) at all times. Make sure the game resets when someone wins or
when there’s a draw. After a winning move, a red line should cross the
center of the three X’s or O’s that end in the winning move, it should
then pause for 10 seconds and restart the game.
```

Please read the response to the above prompt carefully:

Absolutely! Below is a Pygame implementation of a Tic Tac Toe game with dimensions 400×500. The game will wait and then reset whenever there is a

---

~~~
```python
import pygame as pg
import sys
import time
from pygame.locals import *



# initialize variables
current_player = 'x'
current_winner = None
is_draw = None

WIDTH = 400
HEIGHT = 400 # Add 100 later for the
BACKGROUND = (255, 255, 255)
LINE_COLOR = (0, 0, 0)



grid = [[None]*3, [None]*3, [None]*3]

pg.init()
FPS = 30
clock = pg.time.Clock()

screen = pg.display.set_mode((WIDTH, HEIGHT + 100), 0, 32) # 400 x 500 display
pg.display.set_caption("Tic Tac Toe")

size = 80 # size of the X / O marks

def game_initiating_window():
    """initializes game window"""
    pg.display.update()
~~~

```
    pg.draw.line(screen, LINE_COLOR, (WIDTH / 3, 0), (WIDTH / 3, HEIGHT), 7)
    pg.draw.line(screen, LINE_COLOR, (WIDTH / 3 * 2, 0), (WIDTH / 3 * 2, HEIGHT), 7)
    pg.draw.line(screen, LINE_COLOR, (0, HEIGHT / 3), (WIDTH, HEIGHT / 3), 7)
    pg.draw.line(screen, LINE_COLOR, (0, HEIGHT / 3 * 2), (WIDTH, HEIGHT / 3 * 2), 7)
    draw_status()


def draw_status():
    """draws the status bar"""
    global is_draw
    if current_winner is None:
        message = current_player.upper() + "'s Turn"
    else:
        message = current_winner.upper() + " won !"
    if is_draw:
        message = "Game Draw !"
    font = pg.font.Font(None, 30)
    text = font.render(message, 1, (255, 255, 255))
    screen.fill((0, 0, 0), (0, 400, 475, 100))
    text_rect = text.get_rect(center=(WIDTH / 2, HEIGHT/2))
    screen.blit(text, text_rect)
    pg.display.update()


def check_win():
    """checks game grid for wins or draws"""
    global grid, current_winner, is_draw
    for row in range(0, 3):
        if (grid[row][0] == grid[row][1] == grid[row][2]) and (grid[row][0] is not None):
            current_winner = grid[row][0]
            pg.draw.line(screen, (250, 0, 0),
                         (0, (row + 1)*HEIGHT / 3 - HEIGHT / 6),
    for col in range(0, 3):
        if (grid[0][col] == grid[1][col] == grid[2][col]) and (grid[0][col] is not None):
            current_winner = grid[0][col]
            pg.draw.line(screen, (250, 0, 0),
                        ((col + 1) * WIDTH / 3 - WIDTH / 6, 0),
                        ((col + 1) * WIDTH / 3 - WIDTH / 6, HEIGHT), 4)
            break
    if (grid[0][0] == grid[1][1] == grid[2][2]) and (grid[0][0] is not None):
        current_winner = grid[0][0]
        pg.draw.line(screen, (250, 70, 70), (50, 50), (350, 350), 4)
    if (grid[0][2] == grid[1][1] == grid[2][0]) and (grid[0][2] is not None):
        current_winner = grid[0][2]
        pg.draw.line(screen, (250, 70, 70), (350, 50), (50, 350), 4)
    if (all([all(row) for row in grid]) and current_winner is None):
        is_draw = True
    draw_status()


def drawXO(row, col):
    """draws the X's and O's on the board"""
    global grid, current_player
    if row == 1:
        pos_x = 50
    if row == 2:
        pos_x = WIDTH / 3 + 30
    if row == 3:
        pos_x = WIDTH / 3 * 2 + 30
    if col == 1:
        pos_y = 25
    if col == 2:
        pos_y = HEIGHT / 3 + 30
    if col == 3:
        pos_y = HEIGHT / 3 * 2 + 30
    grid[row-1][col-1] = current_player
    if current_player == 'x':
        pg.draw.line(screen, (0, 0, 0), (pos_y, pos_x + size), (pos_y + size,
pos_x), 5)
        current_player = 'o'
    else:
        center = (pos_y + size // 2, pos_x + size // 2)
        radius = size // 2 - 5
        pg.draw.circle(screen, (0, 0, 0), center, radius, 5)
        current_player = 'x'
    pg.display.update()

def user_click():
    """finds the position of the user's click"""
    x, y = pg.mouse.get_pos()
    if (x < WIDTH / 3):
        col = 1
    elif (x < WIDTH / 3 * 2):
        col = 2
    elif (x < WIDTH):
        col = 3
    else:
        col = None
    if (y < HEIGHT / 3):
        row = 1
    elif (y < HEIGHT / 3 * 2):
        row = 2
    elif (y < HEIGHT):
        row = 3
    else:
        row = None
    if (row and col and grid[row-1][col-1] is None):
        drawXO(row, col)
        check_win()

def reset_game():
    """restarts game on win or draw"""
    current_player = x
    is_draw = False
    game_initiating_window()
    current_winner = None
    grid = [[None]*3, [None]*3, [None]*3]

game_initiating_window()

while True: # continue loop until user closes the window
    for event in pg.event.get(): # check for new events
        if event.type == QUIT:
            pg.quit()
            sys.exit()
        elif event.type == MOUSEBUTTONDOWN:
            user_click()
            if (current_winner or is_draw):
                reset_game()
    pg.display.update()
    clock.tick(FPS)

```

___

Instructions for testing and local use

* Ensure you have `pygame` installed in your current python environment.
* Run the script from your console by calling `python script_name.py`
* X goes first, a red line will apear once a player has won.
* Play the game until one player wins or there is a draw.
* Close the window once you've finished playing.

Your job is to determine if this is a high-quality, correct response to the prompt!

- Truthfulness: Are all factual claims in the response correct? Does the code work, and is it well-written?
  - Examples of claims: "Python is an object-oriented language.", "O(n) is the runtime of this code"
- Instruction following: Are all the prompt's instructions followed? Does the provided response meet all specifications?

These determine the overall quality of the response.

Is the given response correct and helpful? Consider the prompt's instructions, the response's code quality, etc.

Yes - the response is helpful and correct. ○

No - there are some issues. ○

Submit Answer

How would you change the given response to make it helpful? Be detailed and call out all issues in the response, including the code! If there are no issues, write N/A.

Type answer here...

Char count: 0

Submit Answer

Instruction Following Dimension

[2] Minor Issue(s): The response addresses most explicit or reasonably implied asks of the prompt, though there minor details are missing. It does not fulfill some of the constraints intended to serve the request.

[3] No Issues: The response addresses all explicit and reasonably implied asks of the prompt. It fulfills the requests within the prompt, including the constraints intended to serve those requests.

Rate Instruction Following

The response has IF issues ○

The response has does not have IF issues ○

Submit Answer

Truthfulness Dimension

[1] Major Issue(s): There are major errors in claims in the response. Code is not executable or does not provide correct outputs.

[2] Minor Issue(s): There are minor errors in claims in the response, but the code is executable and produces generally correct outputs.

[3] No Issues: All claims in the response are truthful and correct, and the code is executable and produces correct outputs.

Rate Truthfulness

The response has truthfulness issues ○

The response does not have truthfulness issues ○

Submit Answer

instruction following, or other errors that need fixing? If so, please write and provide an improved response in the field below. Remember, you must complete this step if you flagged issues above.

Please ensure that you only move forward when the:

- The response addresses all explicit and reasonable implied asks of the prompt.
- The response completed the requests within the prompt, including the constraints placed in service of those requests.
- All code and intermediate steps are completely correct and accurate without any errors.

Write an ideal response to the prompt question here! Ensure you correct all issues that you found. If you found no issues, write N/A.

3 min char

Type answer here...

Char count: 0

Submit Answer

If you provided a rewrite, summarize the changes/fixed you made. If you did not provide a rewrite, write "N/A".

3 min char

Type answer here...

Char count: 0