

Assessment Task 1

Assessment Task 1

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, justifications, and rewrite (if applicable). For the latter, specific details on why the response failed or succeeded on the dimension are recommended.

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:

```
I'm working on some login infra for my application and I want to use Python and Pandas to build it. Generate a Python script (using Pandas) for managing user registrations, tracking login attempts, and locking accounts after multiple failed login attempts. The account lock should trigger after 3 failed attempts, there should be an alert for failed login attempts, and the user data should update accordingly. Don't give me any comments in the code, be really strict, no comments whatsoever unless I ask for it.
```

Rate Instruction Following

The response has IF issues



The response has does not have IF issues



If you said the response has Instruction Following issues, write your justification here. Be specific and detailed. If there weren't any instruction following issues, write "N/A".

3 min char

The response has instruction following issues, there is a comment other than what the user requested in the prompt which is "Add new user to DataFrame" while the prompt requested to not include any comments other than what it requested which is "Never alter this login function" above the login function.
Also, the prompt required in the login, the username should be case sensitive. However, the response converted both usernames to lowercase which is incorrect and makes it case-insensitive.

Char count: 493

Answered

The response has instruction following issues, there is a comment other than what the user requested in the prompt which is "Add new user to DataFrame" while the prompt requested to not include any comments other than what it requested which is "Never alter this login function" above the login function.

Also, the prompt required in the login, the username should be case sensitive. However, the response converted both usernames to lowercase which is incorrect and makes it case-insensitive.

Rate Truthfulness

The response has truthfulness issues



The response does not have truthfulness issues



If you said the response has truthfulness issues, write your justification here. Be specific and detailed. If there weren't any truthfulness issues, write "N/A".

3 min char

The response has truthfulness issues, in the login method, the response compares the same variable of password "password == password" which leads to always increment the failed attempts. In addition, when the "password == password" in the if statement it increments the failed attempts which is wrong, it should remove it from the if statement and move it to the else statement above the "update_user" method, and remove the reset method in the else statement to make the logic correct. Also, modify the if statement to "if password == user.password:" for accurate comparison.

Also, the response didn't implement the failed attempts correctly in the AuthenticationSystem. In the User class, the failed_attempts were initialized to 0. The counter should be incremented and when it reaches 3 attempts it will lock the account by setting the "is_locked" to true. However, in the AuthenticationSystem class, it was declared as "fialed_attempts_left" at initialized to 3. And this is the issue, because in the last test case when it tried to log in with the correct username and password for "helios" the "fialed_attempts_left" was already 3, and even though the username and password were correct, it triggered an Account locked and failed to login.

Char count: 1246

Answered

The response has truthfulness issues, in the login method, the response compares the same variable of password "password == password" which leads to always increment the failed attempts. In addition, when the "password == password" in the if statement it increments the failed attempts which is wrong, it should remove it from the if statement and move it to the else statement above the "update_user" method, and remove the reset method in the else statement to make the logic correct. Also, modify the if statement to "if password == user.password:" for accurate comparison.

Also, the response didn't implement the failed attempts correctly in the AuthenticationSystem. In the User class, the failed_attempts were initialized to 0. The counter should be incremented and when it reaches 3 attempts it will lock the account by setting the "is_locked" to true. However, in the AuthenticationSystem class, it was declared as "fialed_attempts_left" at initialized to 3. And this is the issue, because in the last test case when it tried to log in with the correct username and password for "helios" the "fialed_attempts_left" was already 3, and even though the username and password were correct, it triggered an Account locked and failed to login.

The rewritten Response:

Here's a Python script for managing user registrations, login attempts, and account locking after multiple failed attempts. The script includes the requested functionality, along with the test cases:

```
```python
```

```
import pandas as pd
```

```
class User:
```

```
 def __init__(self, user_id, username, password, failed_attempts=0, is_locked=False):
```

```
 self.user_id = user_id
```

```
 self.username = username
```

```
 self.password = password
```

```
 self.failed_attempts = failed_attempts
```

```
 self.is_locked = is_locked
```

```
 def reset_failed_attempts(self):
```

```
 self.failed_attempts = 0
```

```
 print(f"Failed attempts reset for user {self.username}.")
```

```
 def increment_failed_attempts(self):
```

```
 self.failed_attempts += 1
```

```
 print(f"Failed attempts for {self.username}: {self.failed_attempts}")
```

```
 if self.failed_attempts >= 3:
```

```
 self.lock_account()
```

```
 def lock_account(self):
```

```
 self.is_locked = True
```

```
 print(
```

```
 f"Account for {self.username} has been locked due to too many failed login attempts.")
```

```
class AuthenticationSystem:
```

```

def __init__(self):
 self.users = pd.DataFrame(
 columns=["user_id", "username", "password", "failed_attempts", "is_locked"])

def register_user(self, user_id, username, password):
 self.users = pd.concat([self.users, pd.DataFrame({
 "user_id": [user_id],
 "username": [username],
 "password": [password],
 "failed_attempts": [0],
 "is_locked": [False]
 })], ignore_index=True)
 print(f"User {username} registered successfully.")

Never alter this login function
def login(self, username, password):
 user_row = self.users[self.users['username'] == username]
 if user_row.empty:
 print(f"User {username} not found.")
 return

 user = User(user_row['user_id'].values[0], user_row['username'].values[0],
user_row['password'].values[0],
 user_row['failed_attempts'].values[0], user_row['is_locked'].values[0])

 if user.is_locked:
 print(f"Account for {username} is locked. Please contact support.")
 return

```

```

if password == user.password:

 print(f"User {username} logged in successfully.")

 user.reset_failed_attempts()

 self.update_user(user)

else:

 user.increment_failed_attempts()

 self.update_user(user)

```

```

def update_user(self, user):

 self.users.loc[self.users['username'] == user.username,

 'failed_attempts'] = user.failed_attempts

 self.users.loc[self.users['username'] ==

 user.username, 'is_locked'] = user.is_locked

 print(f"User {user.username}'s data updated.")

```

```

auth_system = AuthenticationSystem()

auth_system.register_user(1, "neena", "password123")

auth_system.register_user(2, "helios", "mysecurepassword")

```

```

auth_system.login("neena", "password321")

auth_system.login("Neena", "password123")

auth_system.login("neena", "password321")

auth_system.login("neena", "password123")

```

```

auth_system.login("helios", "mysecurepassword")

...

```

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

3 min char

- Removed the "new\_user" initialization in the "register\_user" method because it is not necessary and it is not used in the code.
- Modified the "login" method in comparing the "username" from "user\_row = self.users[self.users['username'].str.lower() == username.lower()]" to "user\_row = self.users[self.users['username'] == username]" by removing the "str.lower()" in the dataframe and "lower" in the username variable.
- In the if statement that compares the password, modified "passsword == password" to "password == user.password".
- In the if statement, removed the "user.increment\_failed\_attempts()" and wrote it in the else statement above the "self.update\_user(user)". In addition, removed the "user.reset\_failed\_attempts()" from the else statement. Also, moved the "print(f"User {username} logged in successfully.")" in the if statement to the top to make it readable.
- Removed the comment "# Add new user to DataFrame." because it is not required in the prompt.

Char count: 973

Answered

- Removed the "new\_user" initialization in the "register\_user" method because it is not necessary and it is not used in the code.

- Modified the "login" method in comparing the "username" from "user\_row = self.users[self.users['username'].str.lower() == username.lower()]" to "user\_row = self.users[self.users['username'] == username]" by removing the "str.lower()" in the dataframe and "lower" in the username variable.

- In the if statement that compares the password, modified "passsword == password" to "password == user.password".

- In the if statement, removed the "user.increment\_failed\_attempts()" and wrote it in the else statement above the "self.update\_user(user)". In addition, removed the "user.reset\_failed\_attempts()" from the else statement. Also, moved the "print(f"User {username} logged in successfully.")" in the if statement to the top to make it readable.

- Removed the comment "# Add new user to DataFrame." because it is not required in the prompt.

## Assessment Task 2

### Assessment Task 2

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, justifications, and rewrite (if applicable). For the latter, specific details on why the response failed or succeeded on the dimension are recommended.**

*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:**

```
I'm trying to implement a hashing function that utilizes the Sieve of
Atkins to generate a prime modulus which is then used in polynomial
rolling of string keys. Please send a script example of how to implement
this hash function in python
```

Rate Instruction Following

The response has IF issues

☐

The response has does not have IF issues

☒

If you said the response has Instruction Following issues, write your justification here. Be specific and detailed. If there weren't any instruction following issues, write "N/A".

3 min char

N/A

Char count: 3

Answered

Rate Truthfulness

The response has truthfulness issues



The response does not have truthfulness issues



If you said the response has truthfulness issues, write your justification here. Be specific and detailed. If there weren't any truthfulness issues, write "N/A".

3 min char

There is a truthfulness issue, the sieve implementation fails to initialize the "res" list correctly, causing an "IndexError: list assignment index out of range" when assigning primality values.

Char count: 194

Answered

There is a truthfulness issue, the sieve implementation fails to initialize the "res" list correctly, causing an "IndexError: list assignment index out of range" when assigning primality values.

Rewritten Response:

Here is an example of a hashing function that utilizes the Sieve of Atkins to generate a list of primes, then uses those primes in polynomial rolling of string keys.

```
```python
```

```
def sieve(limit):
```

```
    if limit < 2:
```

```
        return []
```

```
    if limit == 2:
```

```
        return [False, True]
```

```
    if limit == 3:
```

```
        return [False, True, True]
```

```
    res = [False] * (limit + 1)
```



```
if limit >= 2:
```

```
    res[2] = True
```

```
if limit >= 3:
```

```
    res[3] = True
```

```
for i in range(4, limit + 1):
```

```
    res[i] = False
```

```
i = 1
```

```
while i <= limit:
```

```
    j = 1
```

```
    while j <= limit:
```

```
        n = (4 * i * i) + (j * j)
```

```
        if (n <= limit and (n % 12 == 1 or  
            n % 12 == 5)):
```

```
            res[n] ^= True
```

```
        n = (3 * i * i) + (j * j)
```

```
        if n <= limit and n % 12 == 7:
```

```
            res[n] ^= True
```

```
        n = (3 * i * i) - (j * j)
```

```
        if (i > j and n <= limit and
```

```
            n % 12 == 11):
```

```
            res[n] ^= True
```

```
        j += 1
```

```
    i += 1
```

```

r = 5
while r * r <= limit:
    if res[r]:
        for i in range(r * r, limit + 1, r * r):
            res[i] = False

    r += 1

return [index for index,value in enumerate(res) if value]

```

```

def pick_prime(primes, min_size=1000):
    """returns a suitable prime to use as modulus"""
    for prime in primes:
        if prime >= min_size:
            return prime
    # if no prime large enough exists, use last one on list
    return primes[-1]

```

```

def hash(string, modulus):
    """implements polynomial rolling of string keys"""
    hash_value = 5381
    for char in string:
        # hash = 33 XOR ord(c)
        hash_value = ((hash_value << 5) + hash_value) ^ ord(char)
    return hash_value % modulus

```

```

if __name__ == '__main__':
    # generate primes list to use as modulus
    primes = sieve(10000) # modify limit based on your needs

```

```
modulus = pick_prime(primes, 1000)
```

```
test_array = ["alpha","beta","gamma","delta","epsilon"]
```

```
for string in test_array:
```

```
    hash_value = hash(string, modulus)
```

```
    print(f"Hash of {string} is {hash_value}")
```

```
...
```

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

3 min char

- In the sieve function, there is the array `res`, which serves as a boolean array that tracks prime numbers. The response didn't allocate the correct number of indexes; he only allocated one index.

- The sieve function should return a list of prime numbers that passed the Atkin sieve algorithm, however here the response returns an array of boolean, and each index of that array represents the number, whether it passed the sive algorithm or not, However, the response should have processed the list of boolean and return from it an array of numbers.

Char count: 551

Answered

- In the sieve function, there is the array `res`, which serves as a boolean array that tracks prime numbers. The response didn't allocate the correct number of indexes; he only allocated one index.
- The sieve function should return a list of prime numbers that passed the Atkin sieve algorithm, however here the response returns an array of boolean, and each index of that array represents the number, whether it passed the sive algorithm or not, However, the response should have processed the list of boolean and return from it an array of numbers.

Assessment Task 3

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, justifications, and rewrite (if applicable). For the latter, specific details on why the response failed or succeeded on the dimension are recommended.

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
```

Rate Instruction Following

The response has IF issues



The response has does not have IF issues



If you said the response has Instruction Following issues, write your justification here. Be specific and detailed. If there weren't any instruction following issues, write "N/A".

3 min char

The prompt requested to pause 10 seconds before restarting the game, but the response just paused for 3 seconds. In addition, the prompt requested to draw a red line in the center of the 3 Xs' or 3 Os' that end a winning move.

Char count: 226

Answered

The prompt requested to pause 10 seconds before restarting the game, but the response just paused for 3 seconds. In addition, the prompt requested to draw a red line in the center of the 3 Xs' or 3 Os' that end a winning move.

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



If you said the response has truthfulness issues, write your justification here. Be specific and detailed. If there weren't any truthfulness issues, write "N/A".

3 min char

The response pauses the game for 3 seconds instead of 10 seconds. Also, when the game ends, it doesn't reset the state of the game. It keeps displaying for example "X won!" even after restarting the game, instead of displaying "X's turn". Also, there is an issue with the first

The response pauses the game for 3 seconds instead of 10 seconds. Also, when the game ends, it doesn't reset the state of the game. It keeps displaying for example "X won !" even after restarting the game, instead of displaying "X's turn". Also, there is an issue with the first line when it draws the X or the O, it doesn't draw it in the center of each box in the first line in the "drawXO" function which leads to not drawing a red line passing the center of each letter as the prompt requested.

The rewritten Response:

Absolutely! Below is a Pygame implementation of a Tic Tac Toe game with dimensions 400x500. The game will wait and then reset whenever there is a draw or one of the players wins. When a player wins, it generates a red line crossing all three squares that were a part of the winning move.

```
```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()
```

```
 screen.fill(BACKGROUND)
```

```

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 + 50))
 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),
```

```
 (WIDTH, (row + 1)*HEIGHT / 3 - HEIGHT / 6), 4)
```

```
 break
```

```
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 = 25
```

```
 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),
```

```
 (pos_y + size, pos_x + size), 5)
```

```
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"""
```

```
 global grid, current_winner, current_player, is_draw
```

```
 time.sleep(10)
```

```
 current_player = 'x'
```

```
 is_draw = False
```

```
 current_winner = None
```

```
 game_initiating_window()
```

```
 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 appear 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.

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

3 min char

- Removed the "sleep" function in the "game\_initiating\_window" which is not necessary. And increased the time "sleep" to 10 seconds in the "reset\_game".
- In the "draw\_status" function, modified the height to be "text\_rect = text.get\_rect(center=(WIDTH / 2, HEIGHT + 50))" to allocate the status text in the black area under the playground.
- In the "drawXO", modified the location of the row and the column of each box.
- In the "reset\_game" function, moved the code line "current\_winner = None" above the code line "game\_initiating\_window()" within the same function.

Char count: 571

Answered

- Removed the "sleep" function in the "game\_initiating\_window" which is not necessary. And increased the time "sleep" to 10 seconds in the "reset\_game".
- In the "draw\_status" function, modified the height to be "text\_rect = text.get\_rect(center=(WIDTH / 2, HEIGHT + 50))" to allocate the status text in the black area under the playground.
- In the "drawXO", modified the location of the row and the column of each box.
- In the "reset\_game" function, moved the code line "current\_winner = None" above the code line "game\_initiating\_window()" within the same function.