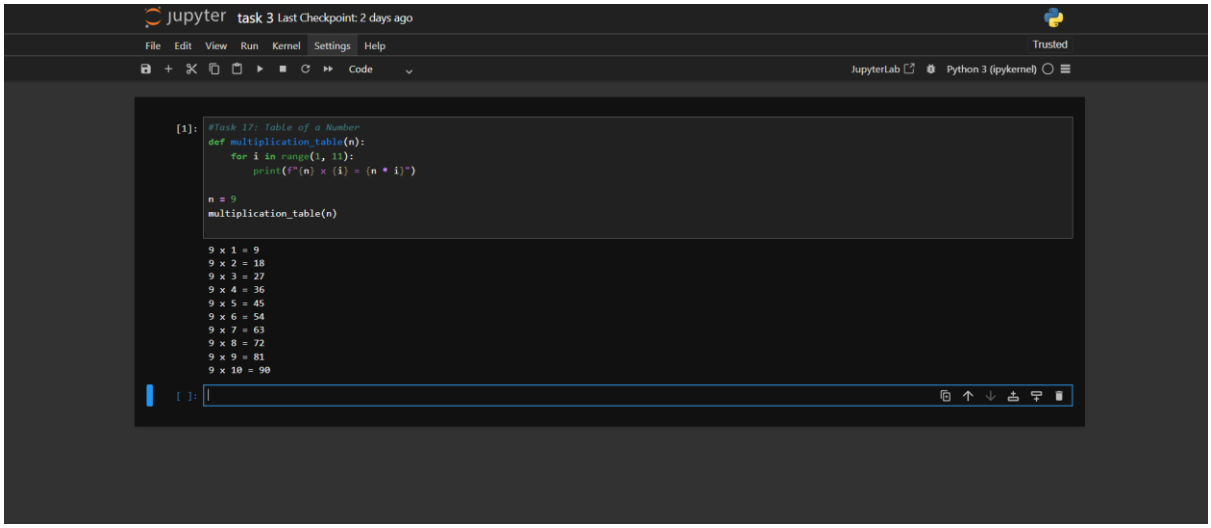
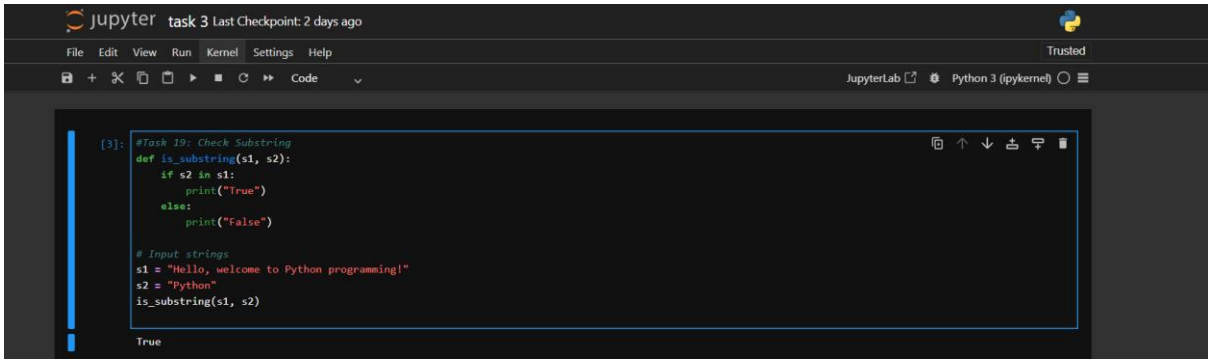
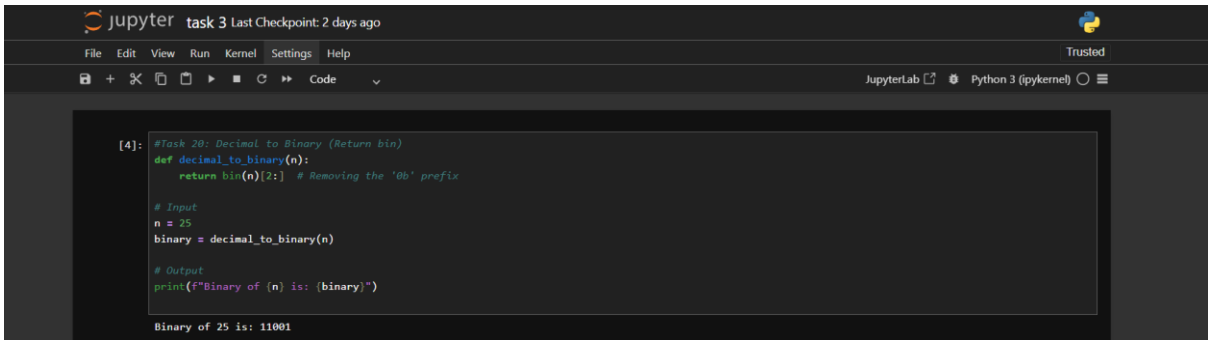

jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]:
#Task 17: Table of a Number
def multiplication_table(n):
for i in range(1, 11):
print(f"{n} x {i} = {n * i}")
n = 9
multiplication_table(n)
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
[ ]:


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[2]:
#Task 18: Swap Two Numbers (without using a third variable)
def swap_numbers(a, b):
a = a + b
b = a - b
a = a - b
print(f"After swapping: a = {a}, b = {b}")
# Input values
a, b = 3, 6
swap_numbers(a, b)
After swapping: a = 6, b = 3


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[3]:
#Task 19: Check Substring
def is_substring(s1, s2):
if s2 in s1:
print("True")
else:
print("False")
# Input strings
s1 = "Hello, welcome to Python programming!"
s2 = "Python"
is_substring(s1, s2)
True


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[4]:
#Task 20: Decimal to Binary (Return bin)
def decimal_to_binary(n):
return bin(n)[2:] # Removing the '0b' prefix
# Input
n = 25
binary = decimal_to_binary(n)
# Output
print(f"Binary of {n} is: {binary}")
Binary of 25 is: 11001

jupyter task 3 Last Checkpoint: 2 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[5]:
```python
#Task 21: Matrix Addition
def add_matrices(matrix1, matrix2):
    result = []  # to store the result matrix
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix1[0])):
            row.append(matrix1[i][j] + matrix2[i][j])
        result.append(row)
    return result

# Input Matrices
matrix1 = [[1, 2, 3],
           [4, 5, 6]]

matrix2 = [[7, 8, 9],
           [1, 2, 3]]

# Function Call
sum_matrix = add_matrices(matrix1, matrix2)

# Output
print("Sum of matrices:")
for row in sum_matrix:
    print(row)
```

```
Sum of matrices:
[8, 10, 12]
[5, 7, 9]
```

jupyter task 3 Last Checkpoint: 2 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[6]:
```python
#Task 22: Matrix Multiplication
def multiply_matrices(A, B):
    result = []
    for i in range(len(A)):
        row = []
        for j in range(len(B[0])):
            sum = 0
            for k in range(len(B)):
                sum += A[i][k] * B[k][j]
            row.append(sum)
        result.append(row)
    return result

# Input Matrices
A = [[1, 2],
     [3, 4]]

B = [[5, 6],
     [7, 8]]

# Function Call
product = multiply_matrices(A, B)

# Output
print("Product of matrices:")
for row in product:
    print(row)
```

```
Product of matrices:
[19, 22]
[43, 50]
```

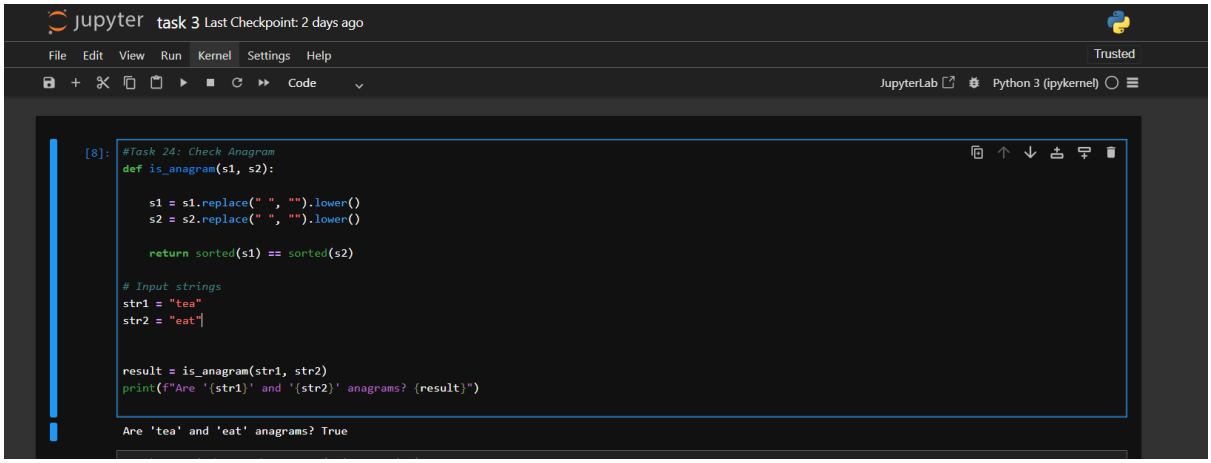
jupyter task 3 Last Checkpoint: 2 days ago

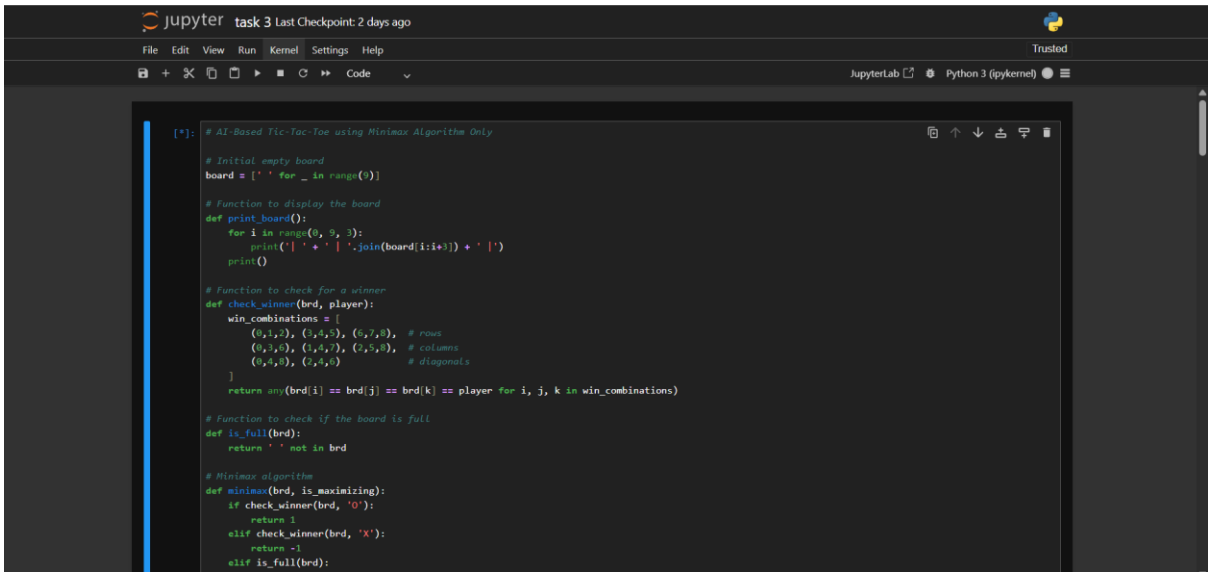
File Edit View Run Kernel Settings Help Trusted

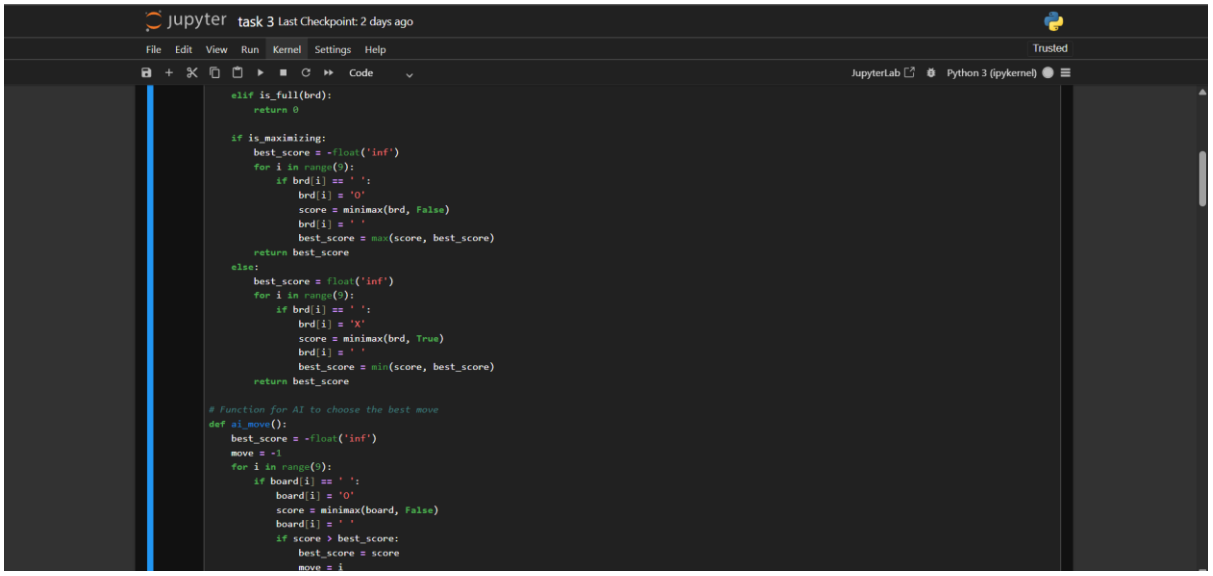
Code JupyterLab Python 3 (ipykernel)

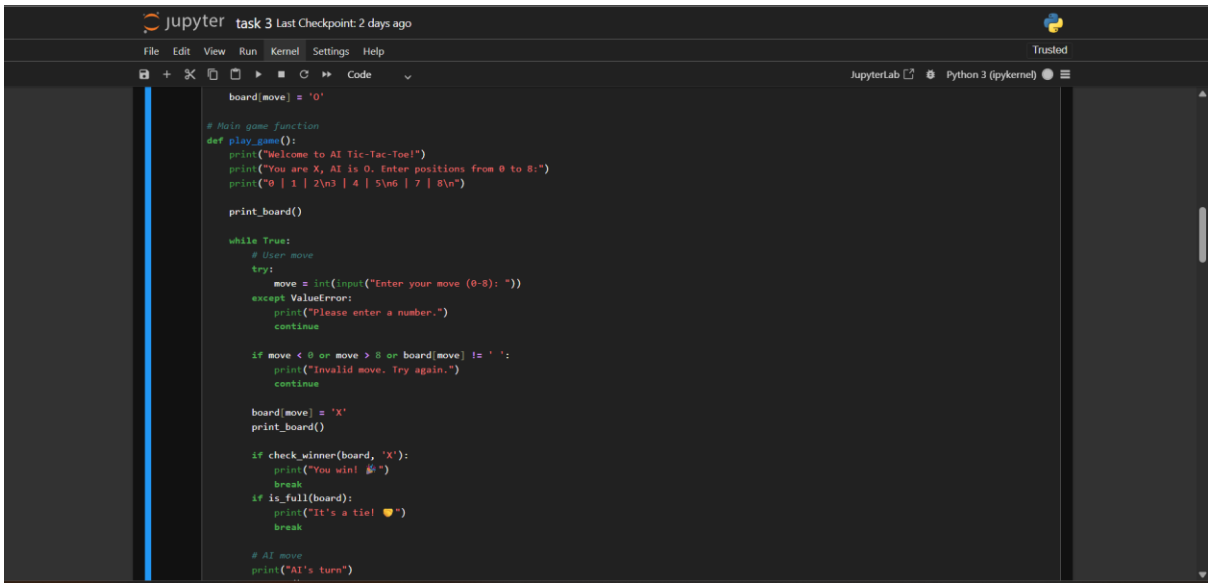
[7]:
```python
#Task 23: Find Second Largest (using sorting)
def second_largest(nums):
    nums = list(set(nums))
    nums.sort(reverse=True)
    if len(nums) >= 2:
        return nums[1]
    else:
        return None

# Input list
numbers = [4, 1, 9, 7, 9, 3]
result = second_largest(numbers)

# Output
print(f"Second largest number is: {result}")
```

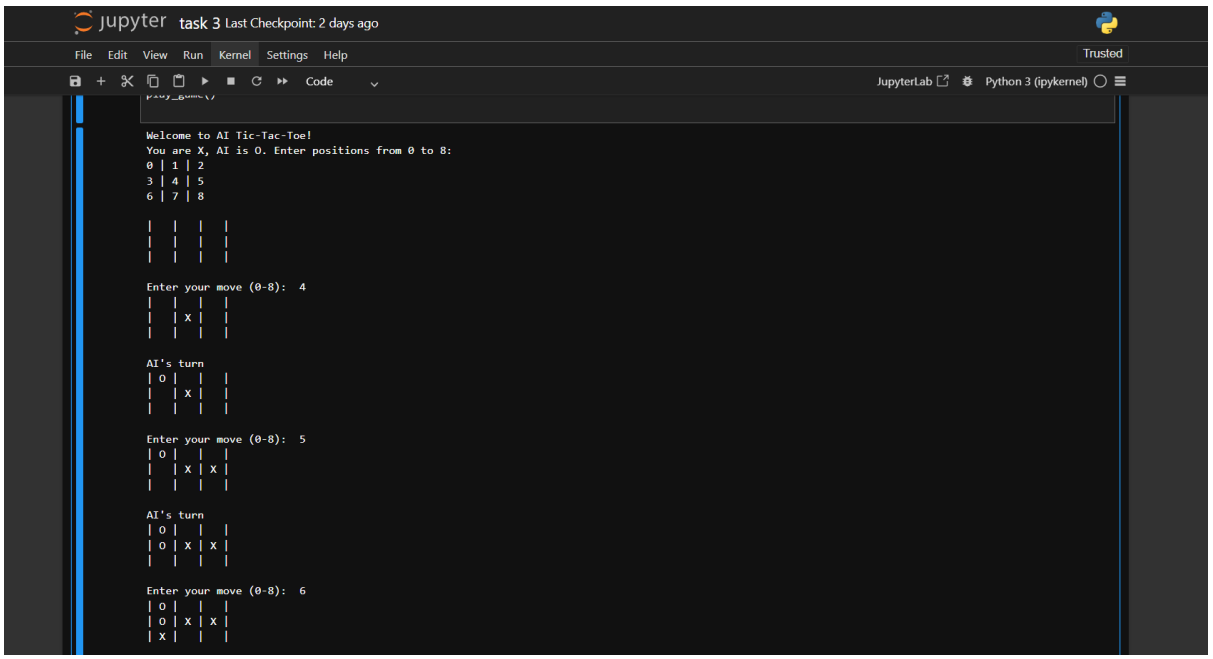
```
Second largest number is: 7
```

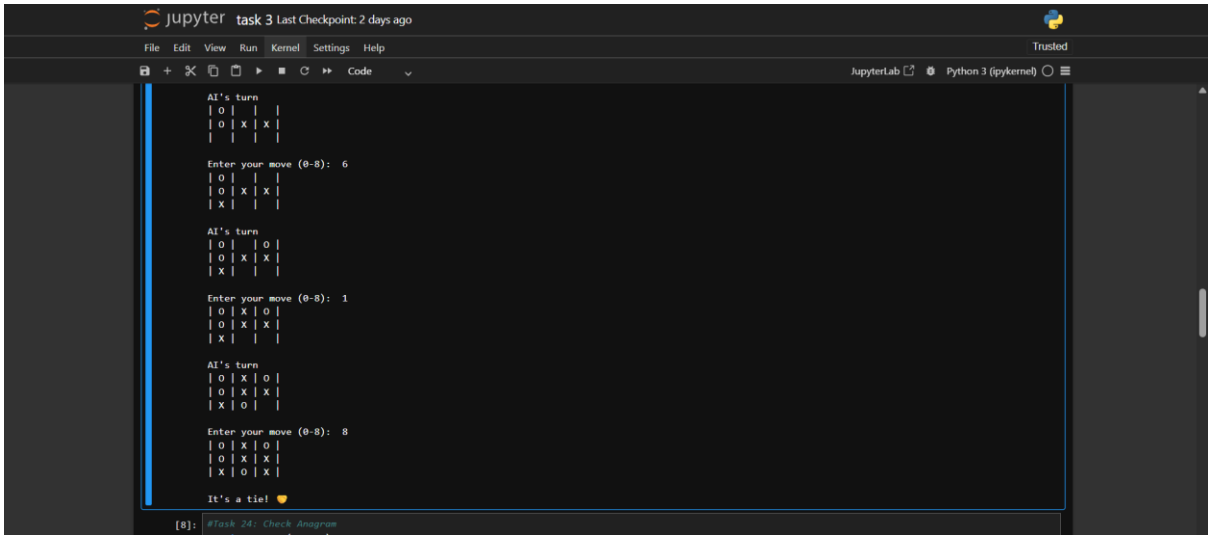

jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[8]:
#Task 24: Check Anagram
def is_anagram(s1, s2):
s1 = s1.replace(" ", "").lower()
s2 = s2.replace(" ", "").lower()
return sorted(s1) == sorted(s2)
# Input strings
str1 = "tea"
str2 = "eat"
result = is_anagram(str1, str2)
print(f"Are '{str1}' and '{str2}' anagrams? {result}")
Are 'tea' and 'eat' anagrams? True


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[*]:
# AI-Based Tic-Tac-Toe using Minimax Algorithm Only
# Initial empty board
board = [' ' for _ in range(9)]
# Function to display the board
def print_board():
for i in range(0, 9, 3):
print('| ' + ' | '.join(board[i:i+3]) + ' |')
print()
# Function to check for a winner
def check_winner(brd, player):
win_combinations = [
(0,1,2), (3,4,5), (6,7,8), # rows
(0,3,6), (1,4,7), (2,5,8), # columns
(0,4,8), (2,4,6) # diagonals
]
return any(brd[i] == brd[j] == brd[k] == player for i, j, k in win_combinations)
# Function to check if the board is full
def is_full(brd):
return ' ' not in brd
# Minimax algorithm
def minimax(brd, is_maximizing):
if check_winner(brd, 'O'):
return 1
elif check_winner(brd, 'X'):
return -1
elif is_full(brd):


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
elif is_full(brd):
return 0
if is_maximizing:
best_score = -float('inf')
for i in range(9):
if brd[i] == ' ':
brd[i] = 'O'
score = minimax(brd, False)
brd[i] = ' '
best_score = max(score, best_score)
return best_score
else:
best_score = float('inf')
for i in range(9):
if brd[i] == ' ':
brd[i] = 'X'
score = minimax(brd, True)
brd[i] = ' '
best_score = min(score, best_score)
return best_score
# Function for AI to choose the best move
def ai_move():
best_score = -float('inf')
move = -1
for i in range(9):
if board[i] == ' ':
board[i] = 'O'
score = minimax(board, False)
board[i] = ' '
if score > best_score:
best_score = score
move = i


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
board[move] = 'O'
# Main game function
def play_game():
print("Welcome to AI Tic-Tac-Toe!")
print("You are X, AI is O. Enter positions from 0 to 8:")
print("0 | 1 | 2\n3 | 4 | 5\n6 | 7 | 8\n")
print_board()
while True:
# User move
try:
move = int(input("Enter your move (0-8): "))
except ValueError:
print("Please enter a number.")
continue
if move < 0 or move > 8 or board[move] != ' ':
print("Invalid move. Try again.")
continue
board[move] = 'X'
print_board()
if check_winner(board, 'X'):
print("You win! 🎉")
break
if is_full(board):
print("It's a tie! 🤝")
break
# AI move
print("AI's turn")


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
play_game()
Welcome to AI Tic-Tac-Toe!
You are X, AI is O. Enter positions from 0 to 8:
0 | 1 | 2
3 | 4 | 5
6 | 7 | 8
Enter your move (0-8): 4
X	
AI's turn
O	
X	
Enter your move (0-8): 5
O	
X	X
AI's turn
O	
O	X	X
Enter your move (0-8): 6
O	
O	X	X
X	


jupyter task 3 Last Checkpoint: 2 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
AI's turn
O		
O	X	X
Enter your move (0-8): 6
O		
O	X	X
X		
AI's turn
O		O
O	X	X
X		
Enter your move (0-8): 1
O	X	O
O	X	X
X		
AI's turn
O	X	O
O	X	X
X	O	
Enter your move (0-8): 8
O	X	O
O	X	X
X	O	X
It's a tie!
[8]: #Task 24: Check Anagram