Google Account | (no subject) - punamava | SE - COMD - Lab - Goog | Dear all, Lab program to | Intro | Upload files - Punamava | DFS 8-puzzle solution | Convert JPG to PDF, Imag

jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Guest

JupyterLite Intro Last Checkpoint: 2 months ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python (Pyodide)

```
Move: left

(1, 2, 3)
(4, 6, 8)
```

[10]:

```
def vacuum_world():
    # Initialize goal_state: '0' means clean, '1' means dirty
    goal_state = {'A': '0', 'B': '0'}
    cost = 0

    # Input: Get the current location of the vacuum cleaner
    location_input = input("Enter Location of Vacuum (A or B): ")

    # Input: Get the status of the current location (clean or dirty)
    status_input = input(f"Enter status of {location_input} (0 for clean, 1 for dirty): ")

    # Input: Get the status of the other location
    location_input_complement = 'B' if location_input == 'A' else 'A'
    status_input_complement = input(f"Enter status of {location_input_complement} (0 for clean, 1 for dirty): ")

    print("Initial Location Condition: ", goal_state)

    # If the vacuum cleaner is at location A
    if location_input == 'A':
        # Location A is dirty
        if status_input == '1':
            print("Vacuum is placed in Location A")
            print("Location A is Dirty.")
            goal_state['A'] = '0'  # Clean location A
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING A:", cost)
            print("Location A has been Cleaned.")

        # If location B is dirty
        if status_input_complement == '1':
            print("Location B is Dirty.")
            print("Moving right to Location B.")
            cost += 1  # Increment cost for movement
            print("Cost for moving RIGHT:", cost)
            goal_state['B'] = '0'  # Clean location B
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING B:", cost)
            print("Location B has been Cleaned.")
        else:
            print("Location B is already clean. No action.")
```

26°C Mostly cloudy

Search

ENG IN 10:20:00 01-09-2025

Google Account | (no subject) - punamava | SE - COMD - Lab - Goog | Dear all, Lab program to | Intro | Upload files - Punamava | DFS 8-puzzle solution | Convert JPG to PDF, Imag

jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Guest

JupyterLite Intro Last Checkpoint: 2 months ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python (Pyodide)

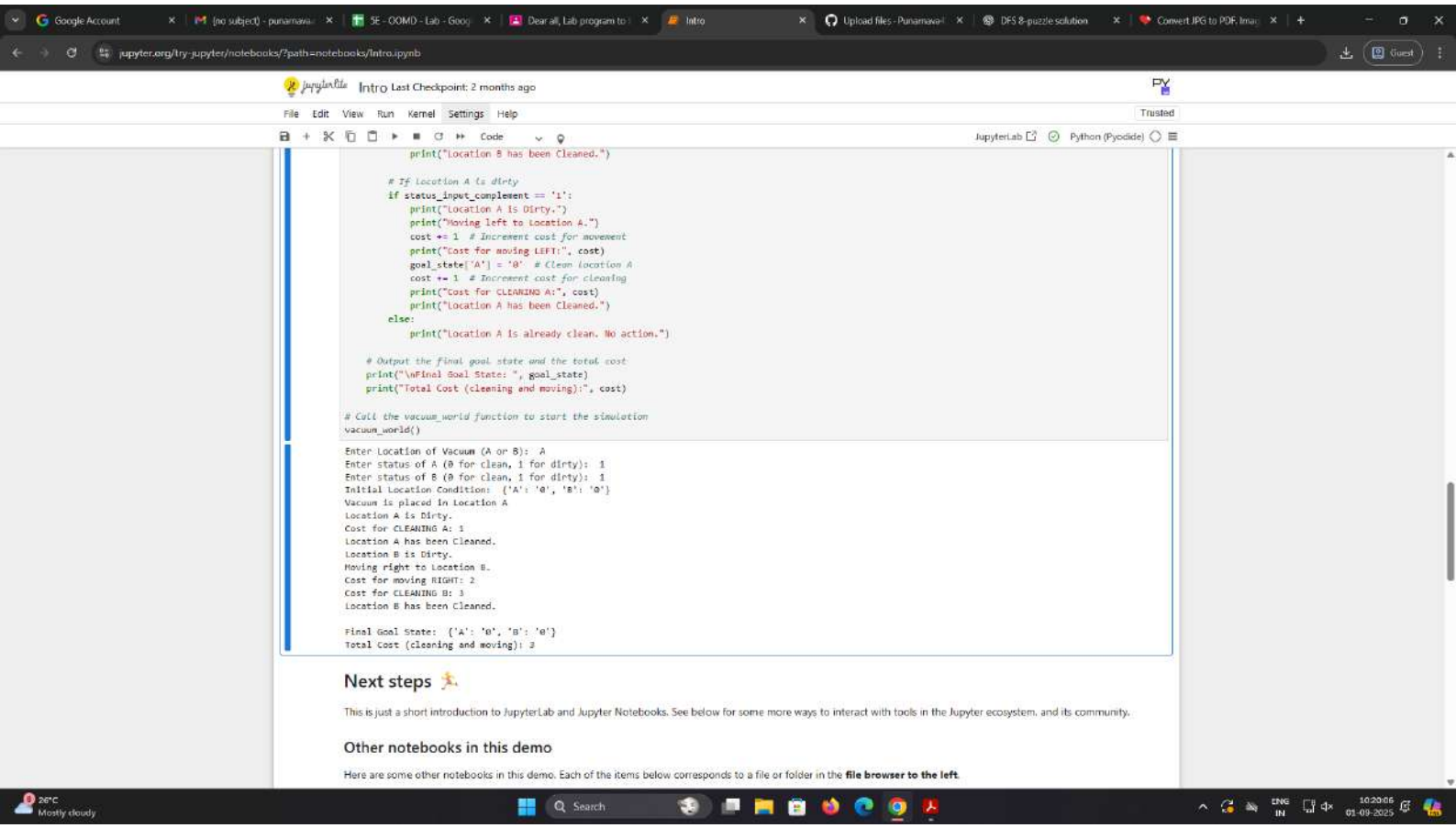
```
        # If location B is dirty
        if status_input_complement == '1':
            print("Location B is Dirty.")
            print("Moving right to Location B.")
            cost += 1  # Increment cost for movement
            print("Cost for moving RIGHT:", cost)
            goal_state['B'] = '0'  # Clean location B
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING B:", cost)
            print("Location B has been Cleaned.")
        else:
            print("Location B is already clean. No action.")

    # If the vacuum cleaner is at location B
    elif location_input == 'B':
        if status_input == '1':
            print("Vacuum is placed in Location B")
            print("Location B is Dirty.")
            goal_state['B'] = '0'  # Clean location B
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING B:", cost)
            print("Location B has been Cleaned.")

        # If location A is dirty
        if status_input_complement == '1':
            print("Location A is Dirty.")
            print("Moving left to Location A.")
            cost += 1  # Increment cost for movement
            print("Cost for moving LEFT:", cost)
            goal_state['A'] = '0'  # Clean location A
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING A:", cost)
            print("Location A has been Cleaned.")
        else:
            print("Location A is already clean. No action.")

    # Output the final goal state and the total cost
    print("\nFinal Goal State: ", goal_state)
    print("Total Cost (cleaning and moving):", cost)

# Call the vacuum_world function to start the simulation
vacuum_world()
```

```
Enter Location of Vacuum (A or B):  A
Enter status of A (0 for clean, 1 for dirty):  1
Enter status of B (0 for clean, 1 for dirty):  1
```

26°C Mostly cloudy

Search

ENG IN 10:20:04 01-09-2025



```
            print("Location B has been Cleaned.")

        # If Location A is dirty
        if status_input_complement == '1':
            print("Location A is Dirty.")
            print("Moving left to Location A.")
            cost += 1  # Increment cost for movement
            print("Cost for moving LEFT:", cost)
            goal_state['A'] = '0'  # Clean location A
            cost += 1  # Increment cost for cleaning
            print("Cost for CLEANING A:", cost)
            print("Location A has been Cleaned.")
        else:
            print("Location A is already clean. No action.")

    # Output the final goal state and the total cost
    print("\nFinal Goal State: ", goal_state)
    print("Total Cost (cleaning and moving):", cost)

# Call the vacuum_world function to start the simulation
vacuum_world()
```

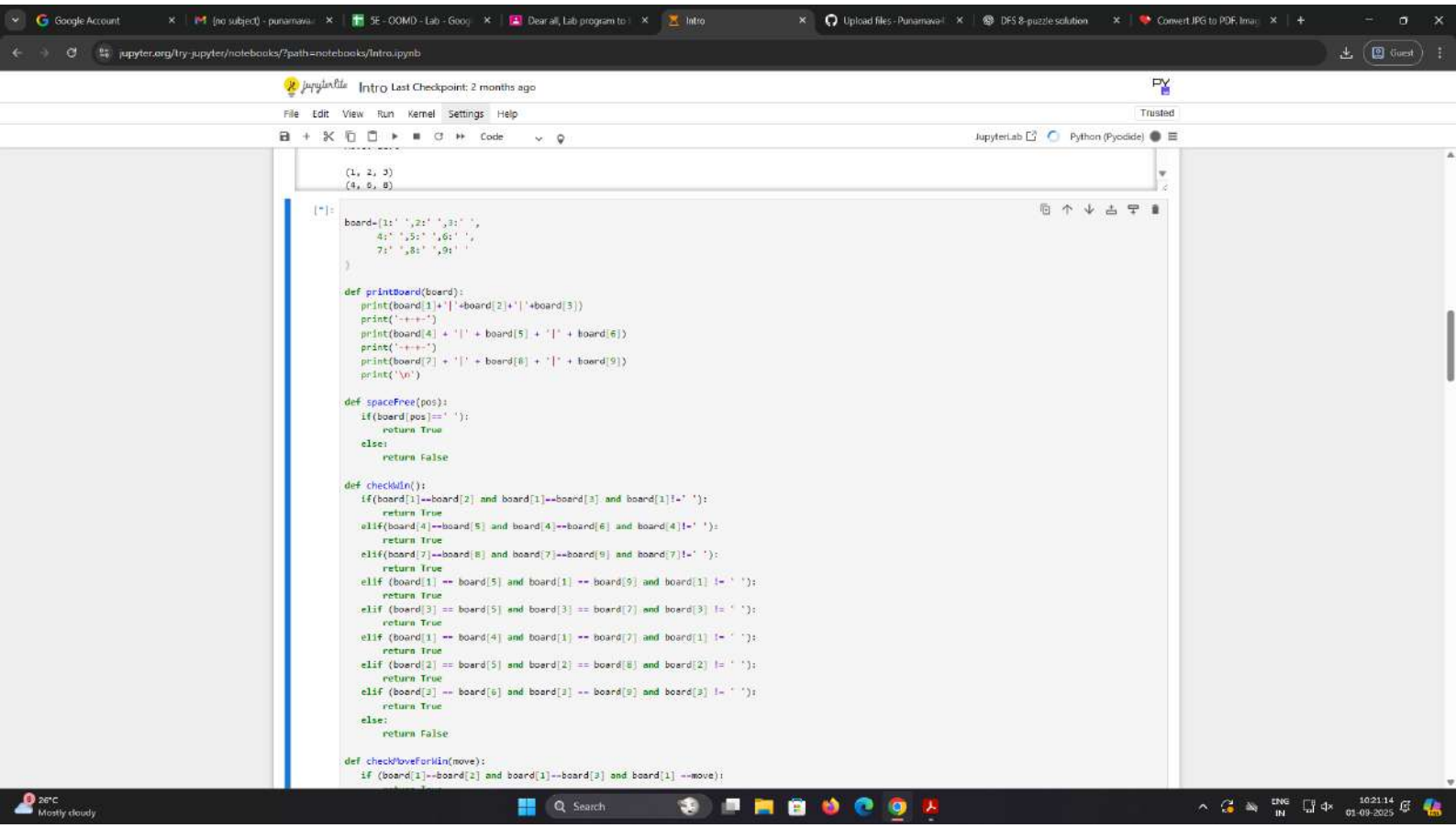
```
Enter Location of Vacuum (A or B):  A
Enter status of A (0 for clean, 1 for dirty):  1
Enter status of B (0 for clean, 1 for dirty):  1
Initial Location Condition:  {'A': '0', 'B': '0'}
Vacuum is placed in Location A
Location A is Dirty.
Cost for CLEANING A: 1
Location A has been Cleaned.
Location B is Dirty.
Moving right to Location B.
Cost for moving RIGHT: 2
Cost for CLEANING B: 3
Location B has been Cleaned.

Final Goal State:  {'A': '0', 'B': '0'}
Total Cost (cleaning and moving): 3
```

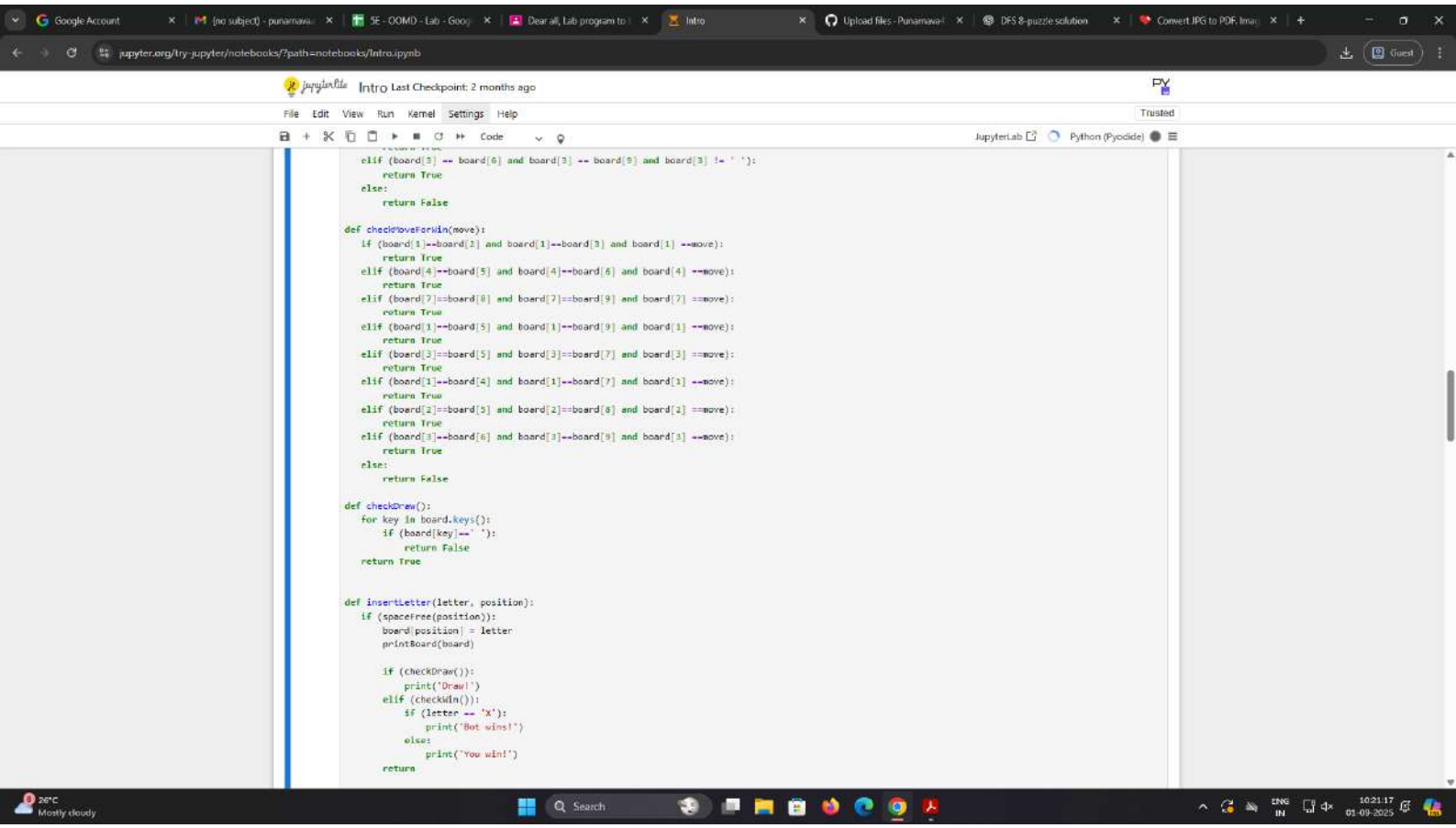
Next steps

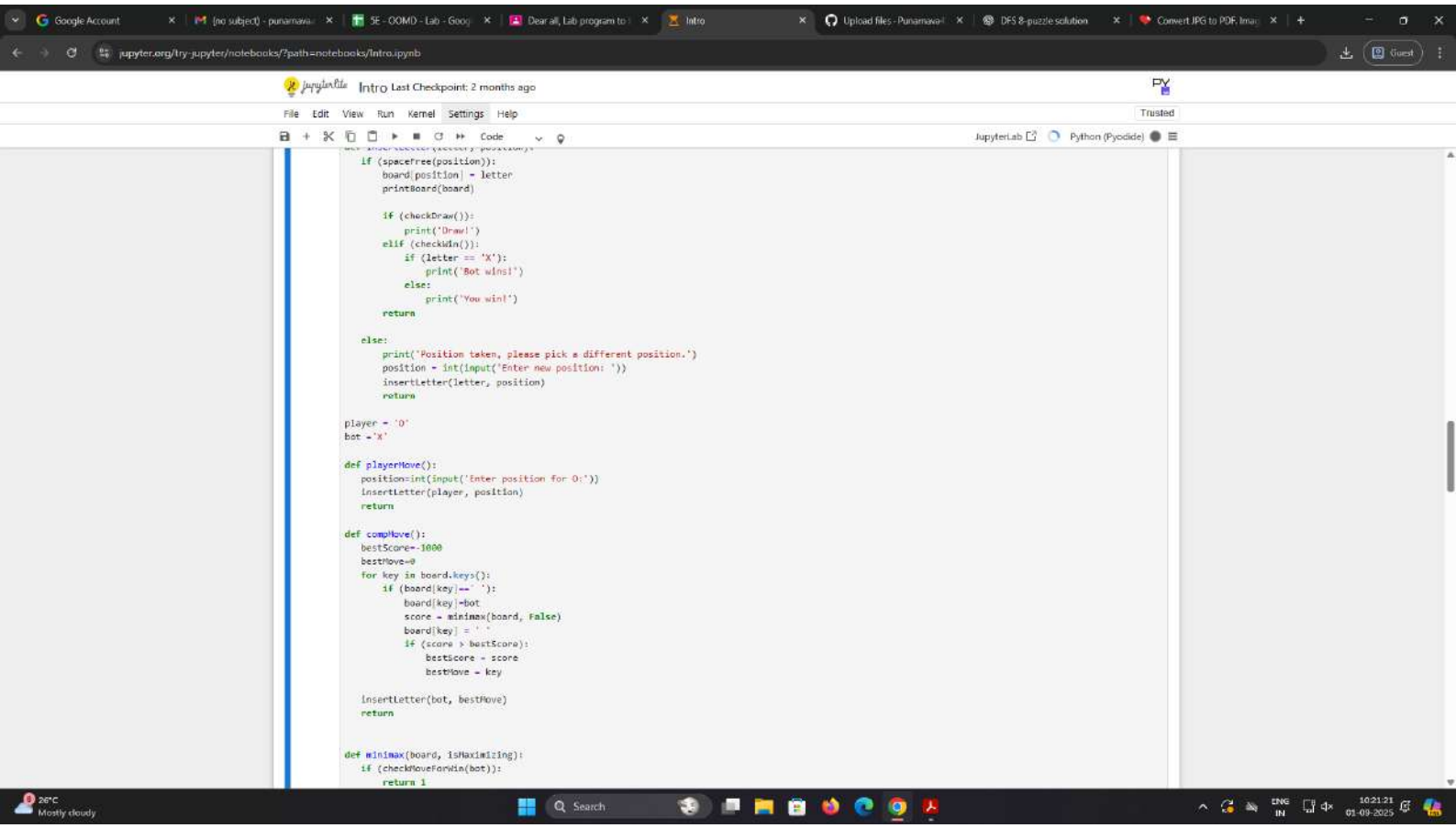
This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with tools in the Jupyter ecosystem, and its community.

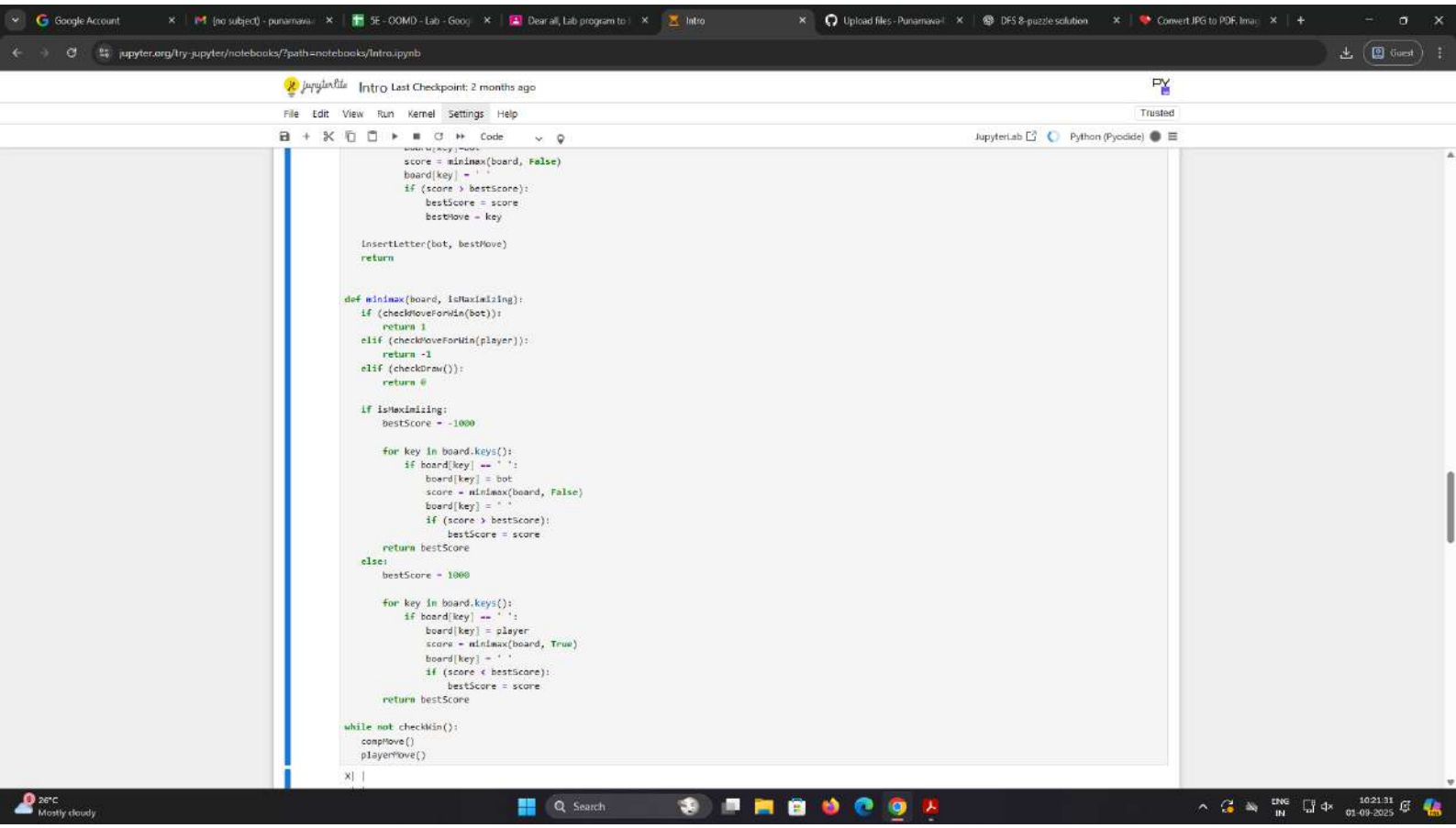
Other notebooks in this demo

Here are some other notebooks in this demo. Each of the items below corresponds to a file or folder in the **file browser to the left**.


Google Account
(no subject) - punamava
SE - OOMD - Lab - Goog
Dear all, Lab program to
Intro
Upload files - Punamava
DFS 8-puzzle solution
Convert JPG to PDF, Imag
jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb
Guest
JupyterLite
Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
(1, 2, 3)
(4, 5, 6)
[*]:
board={1:' ',2:' ',3:' ',
4:' ',5:' ',6:' ',
7:' ',8:' ',9:' '
}
def printBoard(board):
print(board[1]+'|'+board[2]+'|'+board[3])
print('-+-+-')
print(board[4] + '|' + board[5] + '|' + board[6])
print('-+-+-')
print(board[7] + '|' + board[8] + '|' + board[9])
print('\n')
def spaceFree(pos):
if(board[pos]==' '):
return True
else:
return False
def checkWin():
if(board[1]==board[2] and board[1]==board[3] and board[1]!=' '):
return True
elif(board[4]==board[5] and board[4]==board[6] and board[4]!=' '):
return True
elif(board[7]==board[8] and board[7]==board[9] and board[7]!=' '):
return True
elif (board[1] == board[5] and board[1] == board[9] and board[1] != ' '):
return True
elif (board[3] == board[5] and board[3] == board[7] and board[3] != ' '):
return True
elif (board[1] == board[4] and board[1] == board[7] and board[1] != ' '):
return True
elif (board[2] == board[5] and board[2] == board[8] and board[2] != ' '):
return True
elif (board[3] == board[6] and board[3] == board[9] and board[3] != ' '):
return True
else:
return False
def checkMoveForWin(move):
if (board[1]==board[2] and board[1]==board[3] and board[1] ==move):
26°C
Mostly cloudy
Search
ENG
IN
10:21:14
01-09-2025


Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
elif (board[3] == board[6] and board[3] == board[9] and board[3] != ' '):
return True
else:
return False
def checkMoveForWin(move):
if (board[1]==board[2] and board[1]==board[3] and board[1] ==move):
return True
elif (board[4]==board[5] and board[4]==board[6] and board[4] ==move):
return True
elif (board[7]==board[8] and board[7]==board[9] and board[7] ==move):
return True
elif (board[1]==board[5] and board[1]==board[9] and board[1] ==move):
return True
elif (board[3]==board[5] and board[3]==board[7] and board[3] ==move):
return True
elif (board[1]==board[4] and board[1]==board[7] and board[1] ==move):
return True
elif (board[2]==board[5] and board[2]==board[8] and board[2] ==move):
return True
elif (board[3]==board[6] and board[3]==board[9] and board[3] ==move):
return True
else:
return False
def checkDraw():
for key in board.keys():
if (board[key]==' '):
return False
return True
def insertLetter(letter, position):
if (spaceFree(position)):
board[position] = letter
printBoard(board)
if (checkDraw()):
print('Draw!')
elif (checkWin()):
if (letter == 'X'):
print('Bot wins!')
else:
print('You win!')
return
26°C
Mostly cloudy
Search
ENG
IN
10:21:17
01-09-2025


Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
if (spacefree(position)):
board[position] = letter
printBoard(board)
if (checkDraw()):
print('Draw!')
elif (checkWin()):
if (letter == 'X'):
print('Bot wins!')
else:
print('You win!')
return
else:
print('Position taken, please pick a different position.')
position = int(input('Enter new position: '))
insertLetter(letter, position)
return
player = 'O'
bot ='X'
def playerMove():
position=int(input('Enter position for O:'))
insertLetter(player, position)
return
def compMove():
bestScore=-1000
bestMove=0
for key in board.keys():
if (board[key]==' '):
board[key]=bot
score = minimax(board, False)
board[key] = ' '
if (score > bestScore):
bestScore = score
bestMove = key
insertLetter(bot, bestMove)
return
def minimax(board, isMaximizing):
if (checkMoveForWin(bot)):
return 1
26°C
Mostly cloudy
Search
ENG
IN
10:21:21
01-09-2025


Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
score = minimax(board, False)
board[key] = ' '
if (score > bestScore):
bestScore = score
bestMove = key
insertLetter(bot, bestMove)
return
def minimax(board, isMaximizing):
if (checkMoveForWin(bot)):
return 1
elif (checkMoveForWin(player)):
return -1
elif (checkDraw()):
return 0
if isMaximizing:
bestScore = -1000
for key in board.keys():
if board[key] == ' ':
board[key] = bot
score = minimax(board, False)
board[key] = ' '
if (score > bestScore):
bestScore = score
return bestScore
else:
bestScore = 1000
for key in board.keys():
if board[key] == ' ':
board[key] = player
score = minimax(board, True)
board[key] = ' '
if (score < bestScore):
bestScore = score
return bestScore
while not checkWin():
compMove()
playerMove()
26°C
Mostly cloudy
Search
ENG
IN
10:21:31
01-09-2025

Google Account | (no subject) - punamava | SE - OOMD - Lab - Goog | Dear all, Lab program to | Intro | Upload files - Punamava | DFS 8-puzzle solution | Convert JPG to PDF, Imag

jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Guest

JupyterLite Intro Last Checkpoint: 2 months ago

File Edit View Run Kernel Settings Help

Trusted

Code

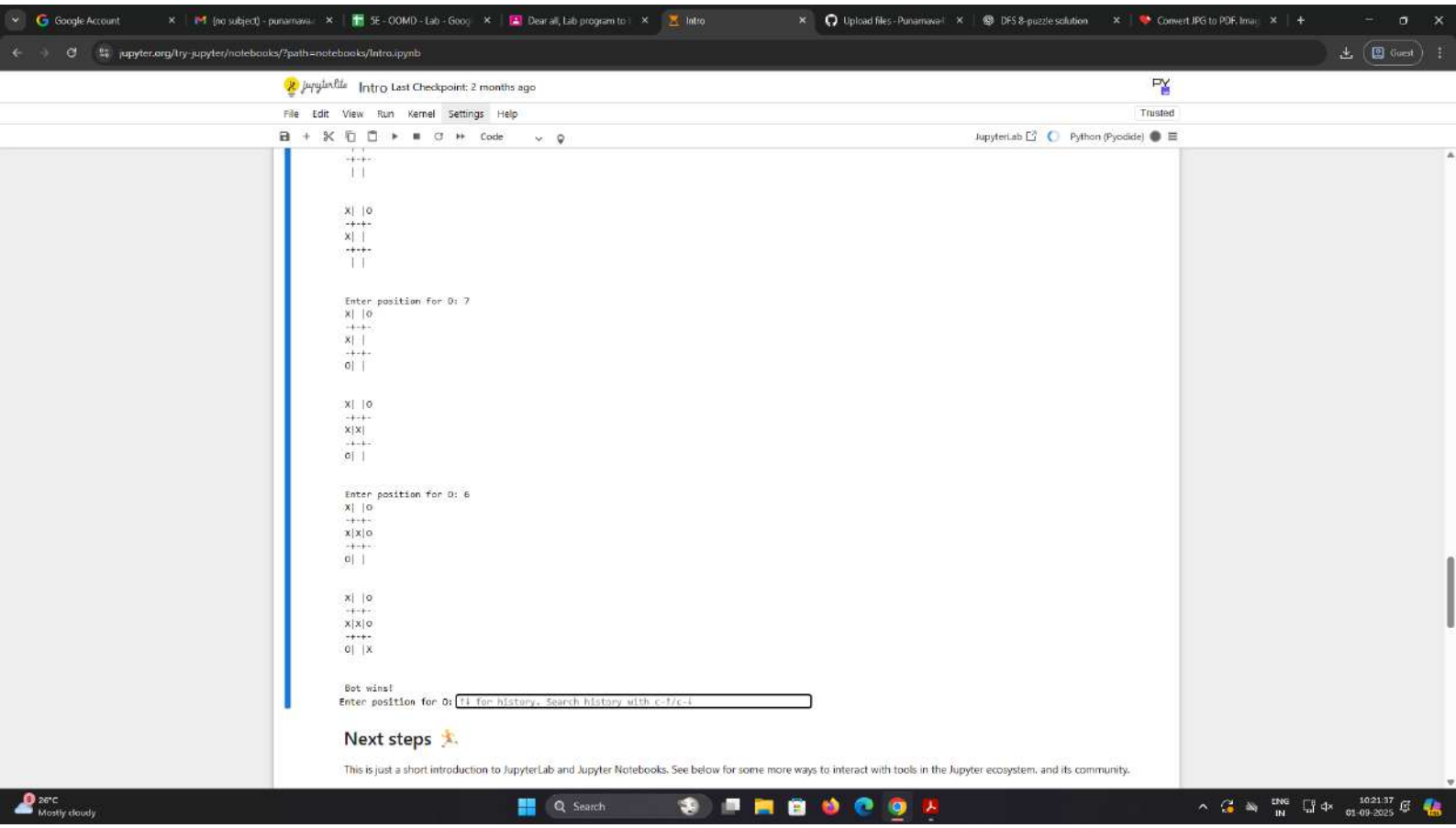
JupyterLab Python (Pyodide)

```
X| |
-+-+-
 | |
-+-+-
 | |


Enter position for O: 3
X| |O
-+-+-
 | |
-+-+-
 | |


X| |O
-+-+-
X| |
-+-+-
 | |


Enter position for O: 7
X| |O
-+-+-
X| |
-+-+-
O| |


X| |O
-+-+-
X|X|
-+-+-
O| |


Enter position for O: 6
X| |O
-+-+-
X|X|O
-+-+-
O| |


X| |O
-+-+-
X|X|O
```

26°C Mostly cloudy

Search

ENG IN 10:21:34 01-09-2025


Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
X| |O
-+-+-
X| |
-+-+-
| |
Enter position for O: 7
X| |O
-+-+-
X| |
-+-+-
O| |
X| |O
-+-+-
X|X|
-+-+-
O| |
Enter position for O: 6
X| |O
-+-+-
X|X|O
-+-+-
O| |
X| |O
-+-+-
X|X|O
-+-+-
O| |X
Bot wins!
Enter position for O: ?i for history, Search history with c-f/c-i
Next steps
This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with tools in the Jupyter ecosystem, and its community.