**Activity 1**

Mathematical Foundation for Computer Application

— — — — — — — — — — — — — — — — — — — — —

**Name-** Naman Jain

**Class-** MCA AI & ML

**USN-** JUPGI202210715

Python code to play a game of chomp agains the computer.

Choose a square and all squares to the right and downwards will be eaten. The computer will do the same. The one to eat the poison square loses.

**Code:**

"""

Chomp - a strategy game

"""

import random

import time

NUM\_ROWS = 5

NUM\_COLS = 6

def print\_title():

print(r"""

\_\_\_\_\_\_ \_\_ \_\_ \_\_\_\_\_\_ \_\_ \_\_ \_\_\_\_\_\_

/\ \_\_\_\ /\ \\_\ \ /\ \_\_ \ /\ "-./ \ /\ == \

\ \ \\_\_\_\_ \ \ \_\_ \ \ \ \/\ \ \ \ \-./\ \ \ \ \_-/

\ \\_\_\_\_\_\ \ \\_\ \\_\ \ \\_\_\_\_\_\ \ \\_\ \ \\_\ \ \\_\

\/\_\_\_\_\_/ \/\_/\/\_/ \/\_\_\_\_\_/ \/\_/ \/\_/ \/\_/

""")

def print\_instructions():

print("Welcome to Chomp. Choose a square and all squares to the right")

print("and downwards will be eaten. The computer will do the same.")

print("The one to eat the poison square loses. Good luck!")

print()

def who\_goes\_first():

if random.randint(0, 1) == 0:

return "computer"

else:

return "human"

def play\_again():

print("Would you like to play again? (yes or no)")

return input().lower().startswith("y")

def print\_matrix(matrix):

for row in matrix:

for elem in row:

print(elem, end=" ")

print()

def validate\_user\_input(player\_choice, board):

try:

row, col = player\_choice.split()

except ValueError:

print("Bad input: The input should be exactly two numbers separated by a space.")

return False

try:

row = int(row)

col = int(col)

except ValueError:

print("Input must be two numbers, however non-digit characters were received.")

return False

if row < 0 or row > NUM\_ROWS - 1:

print(f"The first number must be between 0 and {NUM\_ROWS - 1} but {row} was passed.")

return False

if col < 0 or col > NUM\_COLS - 1:

print(f"The second number must be between 0 and {NUM\_COLS - 1} but {col} was passed.")

return False

if board[row][col] == " ":

print("That square has already been eaten!")

return False

return True

def update\_board(board, row, col):

for i in range(row, len(board)):

for j in range(col, len(board[i])):

board[i][j] = " "

def get\_human\_move(board):

valid\_input = False

while not valid\_input:

player\_choice = input("Enter the row and column of your choice separated by a space: ")

valid\_input = validate\_user\_input(player\_choice, board)

row, col = player\_choice.split()

return int(row), int(col)

def get\_computer\_move(board):

valid\_move = False

while not valid\_move:

row = random.randint(0, NUM\_ROWS - 1)

col = random.randint(0, NUM\_COLS - 1)

if board[row][col] == " ":

continue

else:

valid\_move = True

return row, col

def main():

board = []

for i in range(NUM\_ROWS):

row = []

for j in range(NUM\_COLS):

row.append("#")

board.append(row)

board[0][0] = "P"

game\_is\_playing = True

turn = "human"

print\_title()

print\_instructions()

while game\_is\_playing:

if turn == "human":

# Human turn

print("Human turn.")

print()

print\_matrix(board)

print()

row, col = get\_human\_move(board)

if board[row][col] == "P":

print()

print("Too bad, the computer wins!")

game\_is\_playing = False

else:

update\_board(board, row, col)

print()

print\_matrix(board)

print()

turn = "computer"

time.sleep(1)

else:

# Computer turn

row, col = get\_computer\_move(board)

print(f"Computer turn. the computer chooses ({row}, {col})")

print()

if board[row][col] == "P":

print()

print("Yay, you win!")

game\_is\_playing = False

else:

update\_board(board, row, col)

print\_matrix(board)

print()

turn = "human"

if play\_again():

main()

else:

print("Goodbye!")

raise SystemExit

main()