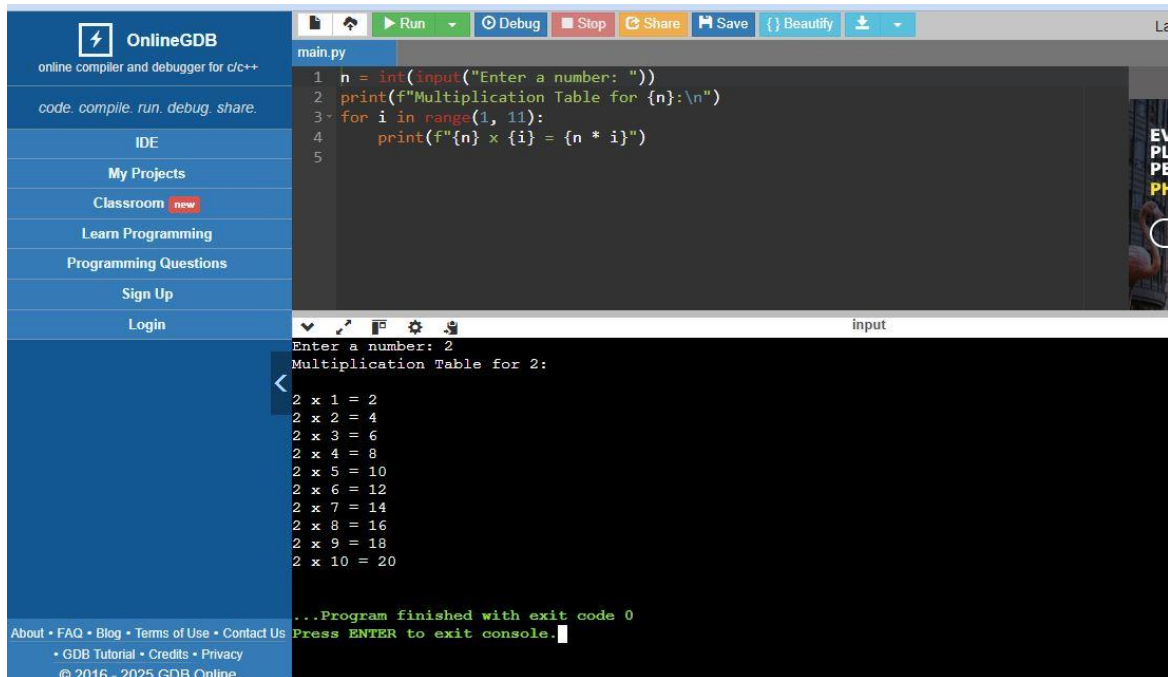


PYTHON DEVELOPER

TASK - 3

17. Table of a Number:



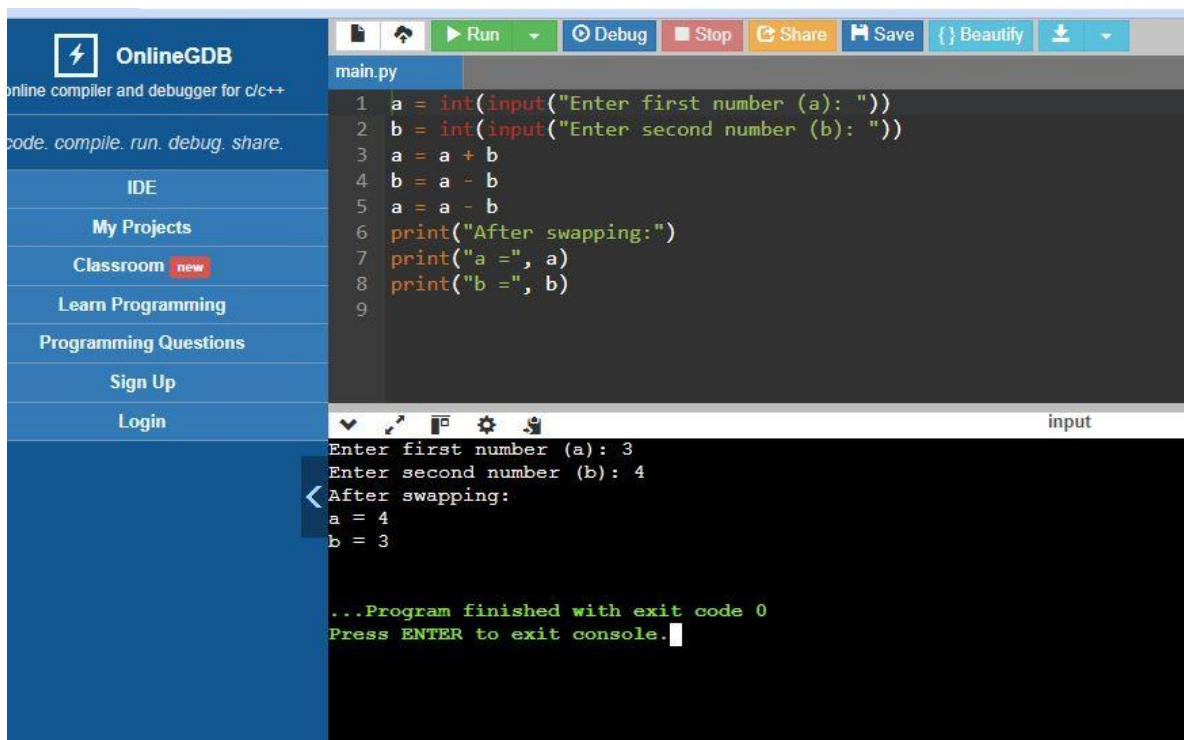
The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: OnlineGDB, code.compile.run.debug.share., IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a Python file named 'main.py' with the following code:

```
1 n = int(input("Enter a number: "))
2 print(f"Multiplication Table for {n}:\n")
3 for i in range(1, 11):
4     print(f"{n} x {i} = {n * i}")
5
```

The console output shows the program execution for input '2':

```
Enter a number: 2
Multiplication Table for 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
...Program finished with exit code 0
Press ENTER to exit console.
```

18. Swap Two Numbers:



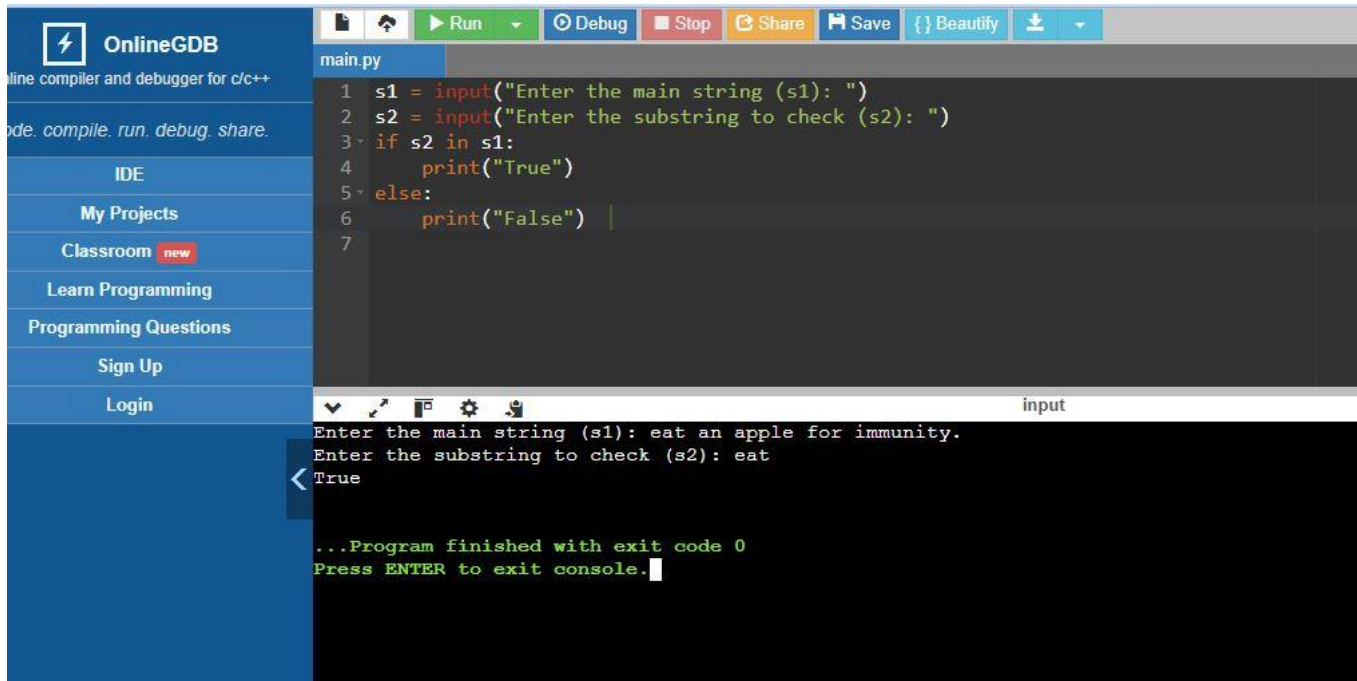
The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: OnlineGDB, code.compile.run.debug.share., IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a Python file named 'main.py' with the following code:

```
1 a = int(input("Enter first number (a): "))
2 b = int(input("Enter second number (b): "))
3 a = a + b
4 b = a - b
5 a = a - b
6 print("After swapping:")
7 print("a =", a)
8 print("b =", b)
9
```

The console output shows the program execution for inputs '3' and '4':

```
Enter first number (a): 3
Enter second number (b): 4
After swapping:
a = 4
b = 3
...Program finished with exit code 0
Press ENTER to exit console.
```

19. Check Substring:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The top toolbar contains icons for Run, Debug, Stop, Share, Save, and Beautify. The main editor displays a Python file named 'main.py' with the following code:

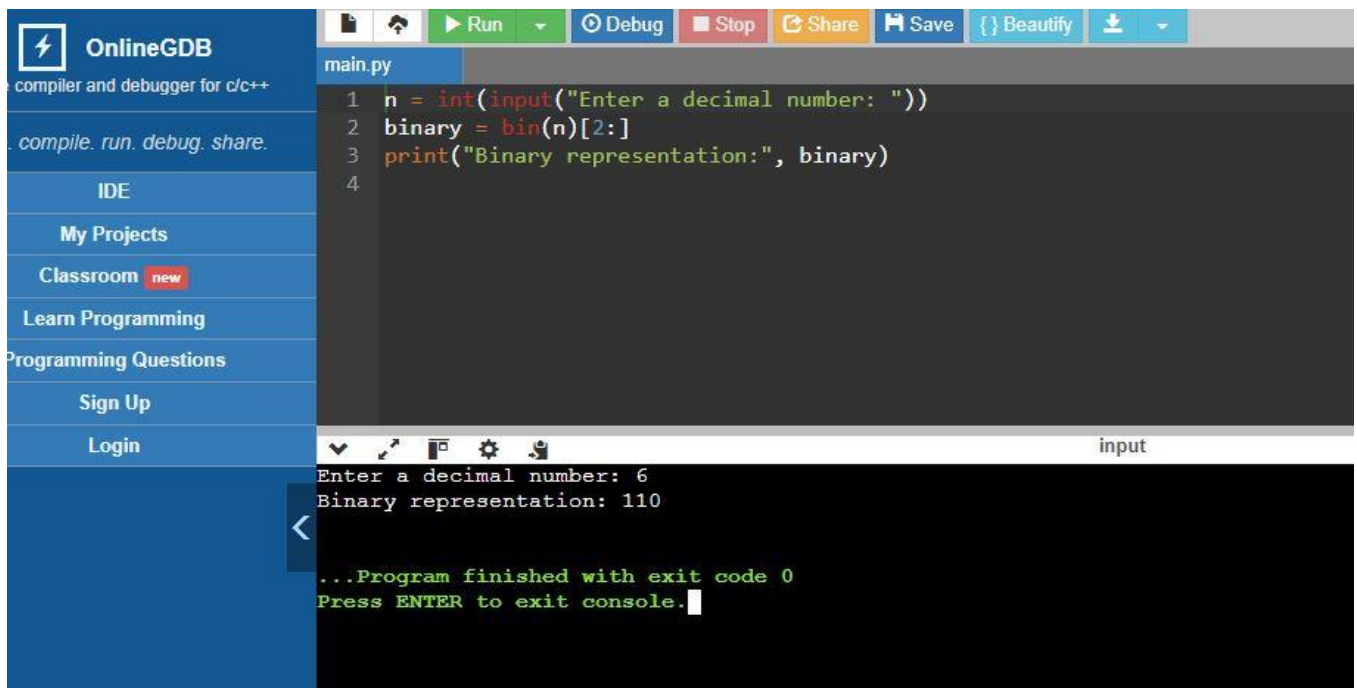
```
1 s1 = input("Enter the main string (s1): ")
2 s2 = input("Enter the substring to check (s2): ")
3 if s2 in s1:
4     print("True")
5 else:
6     print("False")
7
```

The console output shows the program execution with the following input and output:

```
Enter the main string (s1): eat an apple for immunity.
Enter the substring to check (s2): eat
True

...Program finished with exit code 0
Press ENTER to exit console.
```

20. Decimal to Binary:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The top toolbar contains icons for Run, Debug, Stop, Share, Save, and Beautify. The main editor displays a Python file named 'main.py' with the following code:

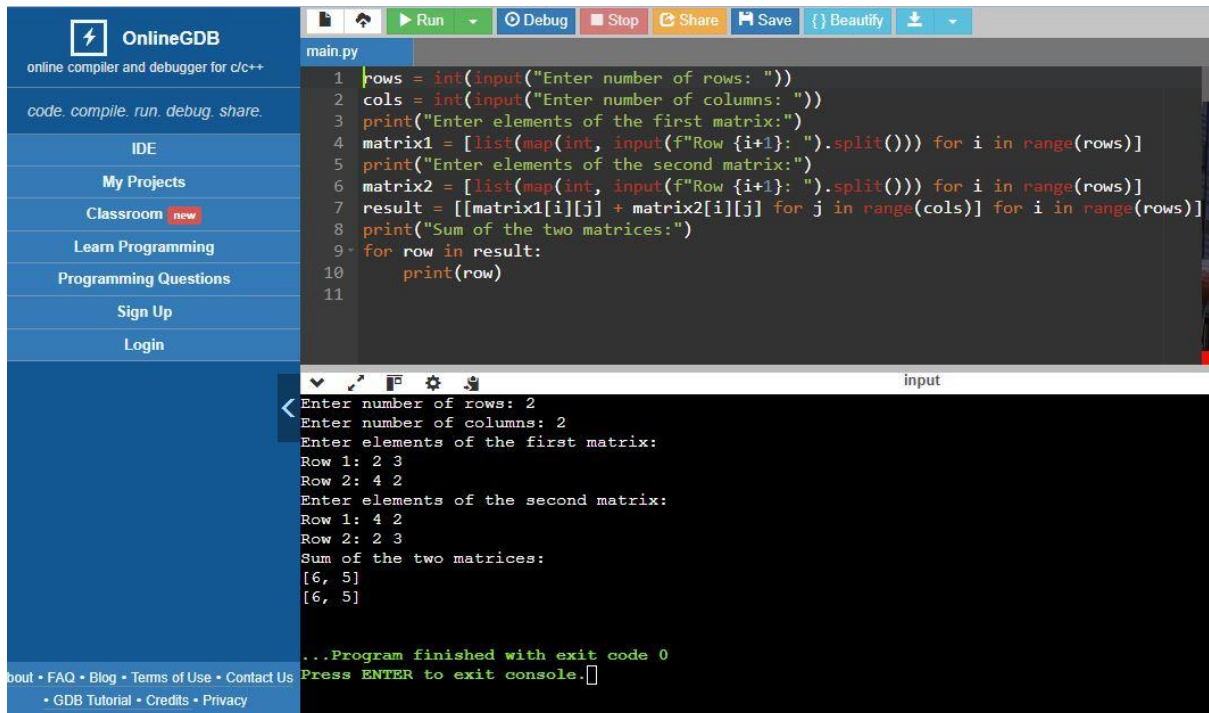
```
1 n = int(input("Enter a decimal number: "))
2 binary = bin(n)[2:]
3 print("Binary representation:", binary)
4
```

The console output shows the program execution with the following input and output:

```
Enter a decimal number: 6
Binary representation: 110

...Program finished with exit code 0
Press ENTER to exit console.
```

21. Matrix Addition:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code, compile, run, debug, share, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor displays a Python script for matrix addition. The code prompts the user for the number of rows and columns, then for the elements of two matrices, and finally prints the sum of the two matrices. The output window shows the execution results, including the input values and the resulting sum matrix.

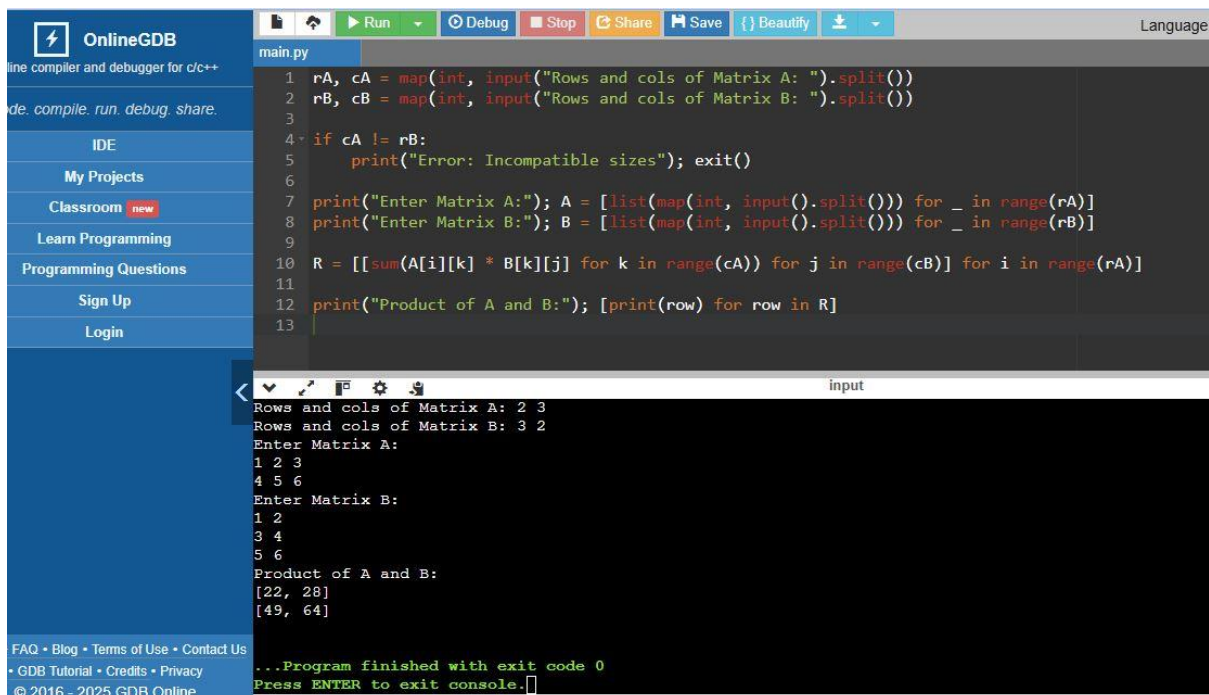
```
main.py
1 rows = int(input("Enter number of rows: "))
2 cols = int(input("Enter number of columns: "))
3 print("Enter elements of the first matrix:")
4 matrix1 = [list(map(int, input(f"Row {i+1}: ").split())) for i in range(rows)]
5 print("Enter elements of the second matrix:")
6 matrix2 = [list(map(int, input(f"Row {i+1}: ").split())) for i in range(rows)]
7 result = [[matrix1[i][j] + matrix2[i][j] for j in range(cols)] for i in range(rows)]
8 print("Sum of the two matrices:")
9 for row in result:
10     print(row)
11
```

input

```
< Enter number of rows: 2
Enter number of columns: 2
Enter elements of the first matrix:
Row 1: 2 3
Row 2: 4 2
Enter elements of the second matrix:
Row 1: 4 2
Row 2: 2 3
Sum of the two matrices:
[6, 5]
[6, 5]

...Program finished with exit code 0
Press ENTER to exit console.
```

22. Matrix Multiplication:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code, compile, run, debug, share, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor displays a Python script for matrix multiplication. The code prompts the user for the rows and columns of two matrices, then for the elements of each matrix, and finally prints the product of the two matrices. The output window shows the execution results, including the input values and the resulting product matrix.

```
main.py
1 rA, cA = map(int, input("Rows and cols of Matrix A: ").split())
2 rB, cB = map(int, input("Rows and cols of Matrix B: ").split())
3
4 if cA != rB:
5     print("Error: Incompatible sizes"); exit()
6
7 print("Enter Matrix A:"); A = [list(map(int, input().split())) for _ in range(rA)]
8 print("Enter Matrix B:"); B = [list(map(int, input().split())) for _ in range(rB)]
9
10 R = [[sum(A[i][k] * B[k][j] for k in range(cA)) for j in range(cB)] for i in range(rA)]
11
12 print("Product of A and B:"); [print(row) for row in R]
13
```

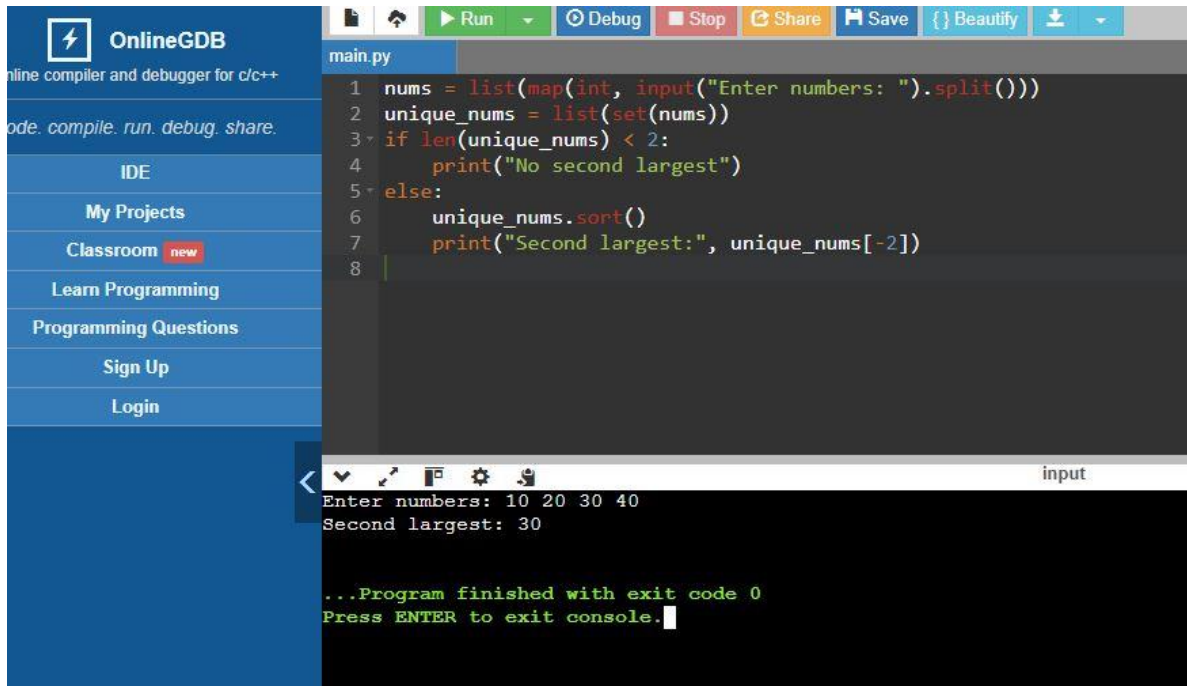
Language

input

```
< Rows and cols of Matrix A: 2 3
Rows and cols of Matrix B: 3 2
Enter Matrix A:
1 2 3
4 5 6
Enter Matrix B:
1 2
3 4
5 6
Product of A and B:
[22, 28]
[49, 64]

...Program finished with exit code 0
Press ENTER to exit console.
```

23. Find Second Larger:

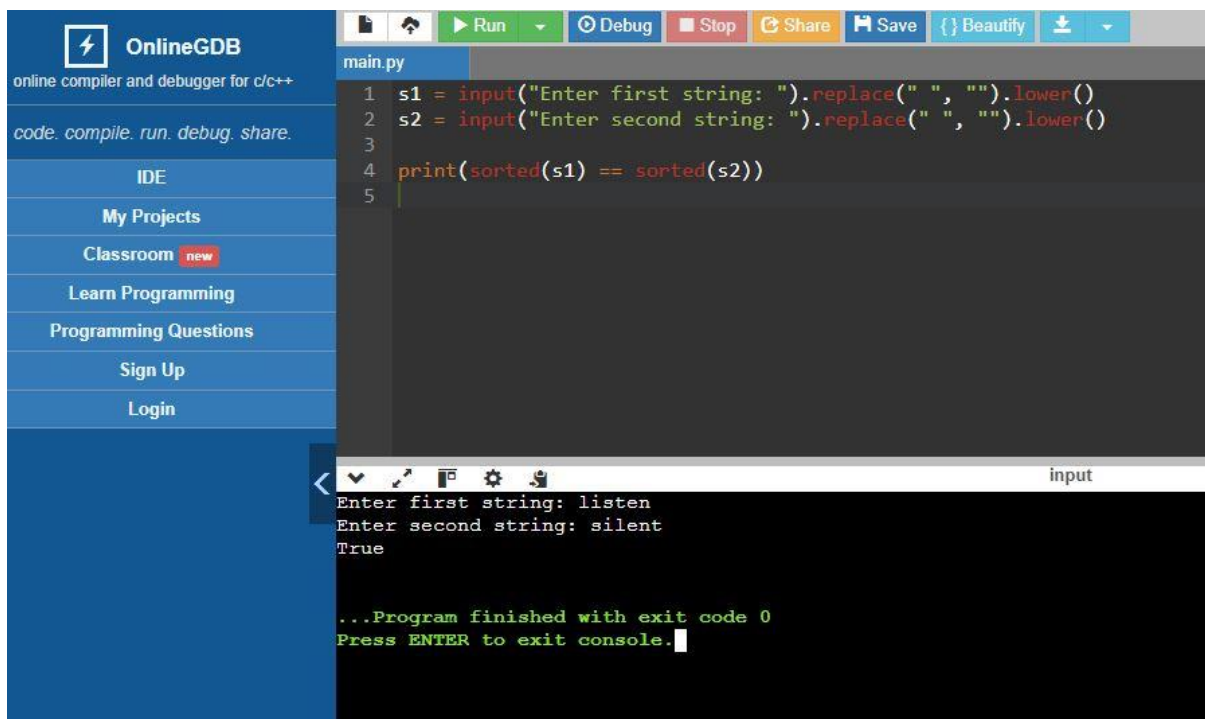


The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, online compiler and debugger for c/c++, code. compile. run. debug. share., IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a Python file named 'main.py' with the following code:

```
1 nums = list(map(int, input("Enter numbers: ").split()))
2 unique_nums = list(set(nums))
3 if len(unique_nums) < 2:
4     print("No second largest")
5 else:
6     unique_nums.sort()
7     print("Second largest:", unique_nums[-2])
8
```

Below the code editor is a console window showing the program's execution. The input '10 20 30 40' is entered, and the output 'Second largest: 30' is displayed. The console also shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

24. Check Anagram:



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, online compiler and debugger for c/c++, code. compile. run. debug. share., IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a Python file named 'main.py' with the following code:

```
1 s1 = input("Enter first string: ").replace(" ", "").lower()
2 s2 = input("Enter second string: ").replace(" ", "").lower()
3
4 print(sorted(s1) == sorted(s2))
5
```

Below the code editor is a console window showing the program's execution. The input 'listen' is entered for the first string and 'silent' for the second string. The output 'True' is displayed, indicating that the two strings are anagrams. The console also shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

3. AI-Based Tic-Tac-Toe

● **Description:** Create a Tic-Tac-Toe game where the computer plays against the user and uses a minimax algorithm to make decisions.

```
import math
```

```
def print_board(board):
```

```
    for row in board:
```

```
        print(" | ".join(row))
```

```
    print("-" * 9)
```

```
def is_winner(board, player):
```

```
    # Check rows, columns, diagonals
```

```
    for i in range(3):
```

```
        if all(board[i][j] == player for j in range(3)): return True
```

```
        if all(board[j][i] == player for j in range(3)): return True
```

```
    if all(board[i][i] == player for i in range(3)): return True
```

```
    if all(board[i][2 - i] == player for i in range(3)): return True
```

```
    return False
```

```
def is_board_full(board):
```

```
    return all(cell != ' ' for row in board for cell in row)
```

```
def minimax(board, depth, is_maximizing, ai_player, human_player):
```

```
    if is_winner(board, ai_player):
```

```
        return 10 - depth
```

```
    if is_winner(board, human_player):
```

```
        return depth - 10
```

```
    if is_board_full(board):
```

```
        return 0
```

```

if is_maximizing:

    best_score = -math.inf

    for i in range(3):

        for j in range(3):

            if board[i][j] == ' ':

                board[i][j] = ai_player

                score = minimax(board, depth + 1, False, ai_player, human_player)

                board[i][j] = ' '

                best_score = max(score, best_score)

    return best_score

else:

    best_score = math.inf

    for i in range(3):

        for j in range(3):

            if board[i][j] == ' ':

                board[i][j] = human_player

                score = minimax(board, depth + 1, True, ai_player, human_player)

                board[i][j] = ' '

                best_score = min(score, best_score)

    return best_score

```

```

def best_move(board, ai_player, human_player):

    best_score = -math.inf

    move = (-1, -1)

    for i in range(3):

        for j in range(3):

            if board[i][j] == ' ':

                board[i][j] = ai_player

                score = minimax(board, 0, False, ai_player, human_player)

```

return move

```
def play_game():
```

```
board = [[' ' for _ in range(3)] for _ in range(3)]
```

human_player = "

```
ai_player = ''
```

```
while human_player not in ['X', 'O']:
```

```
human_player = input("Choose your symbol (X/O): ").upper()
```

```
ai_player = 'O' if human_player == 'X' else 'X'
```

```
current_turn = 'X' # X always starts
```

```
print("\nBoard positions are numbered 1-9 as below:")
```

```
print("1 | 2 | 3\n4 | 5 | 6\n7 | 8 | 9\n")
```

while True:

```
print_board(board)
```

```
if current_turn == human_player:
```

```
valid_move = False
```

```
while not valid_move:
```

try:

```
move = int(input("Your move (1-9): ")) - 1
```

```
row, col = divmod(move, 3)
```

```
if board[row][col] == '':
```

```
board[row][col] = human_player
```

```

        valid_move = True

    else:

        print("Cell occupied! Try again.")

    except (ValueError, IndexError):

        print("Invalid input! Enter a number from 1 to 9.")

else:

    print("AI is making a move...")

    row, col = best_move(board, ai_player, human_player)

    board[row][col] = ai_player


# Check for win/tie

if is_winner(board, current_turn):

    print_board(board)

    if current_turn == human_player:

        print("Congratulations! You won!")

    else:

        print("AI wins! Better luck next time.")

    break


if is_board_full(board):

    print_board(board)

    print("It's a tie!")

    break


# Switch turns

current_turn = ai_player if current_turn == human_player else human_player


if __name__ == "__main__":

    play_game()

```


program:

The image displays three sequential screenshots of the OnlineGDB IDE, showing the development of a Python program for a 3x3 Tic Tac Toe game. The program uses a minimax algorithm to determine the best move for the AI player.

Screenshot 1 (Top): Shows the initial code structure. It includes imports, a board printing function, a winner checking function, a board full check, and the minimax function.

```
1 import math
2
3 def print_board(board):
4     for row in board:
5         print(" | ".join(row))
6         print("-" * 9)
7
8 def is_winner(board, player):
9     # Check rows, columns, diagonals
10    for i in range(3):
11        if all(board[i][j] == player for j in range(3)): return True
12        if all(board[j][i] == player for j in range(3)): return True
13    if all(board[i][i] == player for i in range(3)): return True
14    if all(board[i][2-i] == player for i in range(3)): return True
15    return False
16
17 def is_board_full(board):
18     return all(cell != ' ' for row in board for cell in row)
19
20 def minimax(board, depth, is_maximizing, ai_player, human_player):
21     if is_winner(board, ai_player):
22         return 10 - depth
23     if is_winner(board, human_player):
24         return depth - 10
25     if is_board_full(board):
26         return 0
27
28     if is_maximizing:
29         best_score = -math.inf
30         for i in range(3):
31             for j in range(3):
```

Screenshot 2 (Middle): Continues the minimax function, showing the recursive calls for both AI and human players.

```
32             if board[i][j] == ' ':
33                 board[i][j] = ai_player
34                 score = minimax(board, depth + 1, False, ai_player, human_player)
35                 board[i][j] = ' '
36                 best_score = max(score, best_score)
37             return best_score
38     else:
39         best_score = math.inf
40         for i in range(3):
41             for j in range(3):
42                 if board[i][j] == ' ':
43                     board[i][j] = human_player
44                     score = minimax(board, depth + 1, True, ai_player, human_player)
45                     board[i][j] = ' '
46                     best_score = min(score, best_score)
47             return best_score
48
49 def best_move(board, ai_player, human_player):
50     best_score = -math.inf
51     move = (-1, -1)
52     for i in range(3):
53         for j in range(3):
54             if board[i][j] == ' ':
55                 board[i][j] = ai_player
56                 score = minimax(board, 0, False, ai_player, human_player)
57                 board[i][j] = ' '
```

Screenshot 3 (Bottom): Shows the game loop and the play_game function.

```
58         board[i][j] = ' '
59         if score > best_score:
60             best_score = score
61             move = (i, j)
62     return move
63
64 def play_game():
65     board = [[' ' for _ in range(3)] for _ in range(3)]
66     human_player = ''
67     ai_player = ''
68
69     while human_player not in ['X', 'O']:
70         human_player = input("Choose your symbol (X/O): ").upper()
71         ai_player = 'O' if human_player == 'X' else 'X'
72
73     current_turn = 'X' # X always starts
74     print("\nBoard positions are numbered 1-9 as below:")
75     print("1 | 2 | 3\n4 | 5 | 6\n7 | 8 | 9\n")
76
77     while True:
78         print_board(board)
79
80         if current_turn == human_player:
81             valid_move = False
82             while not valid_move:
83                 try:
84                     move = int(input("Your move (1-9): ")) - 1
85                     row, col = divmod(move, 3)
86                     if board[row][col] == ' ':
87                         board[row][col] = human_player
88                         valid_move = True
89                 except ValueError:
```

OnlineGDB

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

FAQ • Blog • Terms of Use • Contact Us

GDB Tutorial • Credits • Privacy

Run

Debug

Stop

Share

Save

Beautify

Language Python 3

main.py

```
85-
86-
87-         if board[row][col] == ' ':
88-             board[row][col] = human_player
89-             valid_move = True
90-         else:
91-             print("Cell occupied! Try again.")
92-     except (ValueError, IndexError):
93-         print("Invalid input! Enter a number from 1 to 9.")
94-
95- else:
96-     print("AI is making a move...")
97-     row, col = best_move(board, ai_player, human_player)
98-     board[row][col] = ai_player
99-
100- # Check for win/tie
101- if is_winner(board, current_turn):
102-     print_board(board)
103-     if current_turn == human_player:
104-         print("Congratulations! You won!")
105-     else:
106-         print("AI wins! Better luck next time.")
107-     break
108-
109- if is_board_full(board):
110-     print_board(board)
111-     print("It's a tie!")
112-     break
113-
114- # Switch turns
115- current_turn = ai_player if current_turn == human_player else human_player
```

OnlineGDB

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

FAQ • Blog • Terms of Use • Contact Us

GDB Tutorial • Credits • Privacy

© 2016 - 2025 GDB Online

Run

Debug

Stop

Share


Save

Beautify

Language Python 3

main.py

```
87-         valid_move = True
88-     else:
89-         print("Cell occupied! Try again.")
90-     except (ValueError, IndexError):
91-         print("Invalid input! Enter a number from 1 to 9.")
92-
93- else:
94-     print("AI is making a move...")
95-     row, col = best_move(board, ai_player, human_player)
96-     board[row][col] = ai_player
97-
98- # Check for win/tie
99- if is_winner(board, current_turn):
100-     print_board(board)
101-     if current_turn == human_player:
102-         print("Congratulations! You won!")
103-     else:
104-         print("AI wins! Better luck next time.")
105-     break
106-
107- if is_board_full(board):
108-     print_board(board)
109-     print("It's a tie!")
110-     break
111-
112- # Switch turns
113- current_turn = ai_player if current_turn == human_player else human_player
114-
115- if __name__ == "__main__":
116-     play_game()
117-
```

**OnlineGDB**
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

[FAQ](#) • [Blog](#) • [Terms of Use](#) • [Contact Us](#)
[GDB Tutorial](#) • [Credits](#) • [Privacy](#)
© 2016 - 2025 GDB Online

Choose your symbol (X/O): 0
Choose your symbol (X/O): x

Board positions are numbered 1-9 as below:
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9


Your move (1-9): 1
X | | |
| | |
| | |

AI is making a move...
X | | |
O	

Your move (1-9): 2
X | X | |
O	

AI is making a move...

00

**OnlineGDB**
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

[FAQ](#) • [Blog](#) • [Terms of Use](#) • [Contact Us](#)
[GDB Tutorial](#) • [Credits](#) • [Privacy](#)
© 2016 - 2025 GDB Online

AI is making a move...
X | X | O
| O |
| | |

Your move (1-9): 7
X | X | O
| O |
| | |

X | | |
| | |
| | |


AI is making a move...
X | X | O
| O |
| O |
| | |

X | | |
| | |
| | |

Your move (1-9): 6
X | X | O
| O | O | X
| | |
X | | |
| | |

AI is making a move...
X | X | O
| O | O | X
| | |
X | O | |
| | |

Your move (1-9): 9

 **OnlineGDB**

Online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

[FAQ](#) • [Blog](#) • [Terms of Use](#) • [Contact Us](#)

[GDB Tutorial](#) • [Credits](#) • [Privacy](#)

© 2016 - 2025 GDB Online

```
-----
X |  | 
-----
AI is making a move...
X | X | O
-----
O | O | 
-----
X |  | 
-----
Your move (1-9): 6
X | X | O
-----
O | O | X
-----
X |  | 
-----
AI is making a move...
X | X | O
-----
O | O | X
-----
X | O | 
-----
Your move (1-9): 9
X | X | O
-----
O | O | X
-----
X | O | X
-----
It's a tie!

...Program finished with exit code 0
Press ENTER to exit console.
```