Double-click (or enter) to edit

Double-click (or enter) to edit

```
import numpy as np
```

```
matrix=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)
print(type(matrix))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
<class 'numpy.ndarray'>
```

```
matrix=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)
print(matrix.reshape(1,9))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 2 3 4 5 6 7 8 9]]
```

```
matrix=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)
print(np.shape(matrix))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
```

```
matrix=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(matrix)
print(np.shape(matrix))
print(matrix.reshape(1,9))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
[[1 2 3 4 5 6 7 8 9]]
```

```
matrix_A=[[5,6,7],[8,9,10],[15,16,17]]
matrix_B=[[12,13,14],[3,7,6],[19,20,3]]
matrix_C=np.add(matrix_A,matrix_B)
print(matrix_C)
```

```
[[17 19 21]
 [11 16 16]
 [34 36 20]]
```

```
matrix_A=[[5,6,7],[8,9,10],[15,16,17]]
matrix_B=[[12,13,14],[3,7,6],[19,20,3]]
matrix_C=np.subtract(matrix_A,matrix_B)
print(matrix_C)
```

```
[[-7 -7 -7]
 [ 5  2  4]
 [-4 -4 14]]
```

```
matrix_A=[[5,6,7],[8,9,10],[15,16,17]]
matrix_B=[[12,13,14],[3,7,6],[19,20,3]]
matrix_C=np.dot(matrix_A,matrix_B)
matrix_D=np.dot(matrix_A,5)
print(matrix_C)
print(matrix_D)
```

```
[[211 247 127]
 [313 367 196]
 [551 647 357]]
[[25 30 35]
 [40 45 50]
 [75 80 85]]
```

```
matrix=np.random.randint(50,size=(4,4))
print(matrix)
```

```
[[ 4 15 29 31]
 [ 3 21 14 24]
 [47 34 28 30]
 [23 43  6 29]]
```

```
matrix=np.zeros((3,3))
print(matrix)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
matrix=np.ones((4,4))
print(matrix)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```
matrix_A=[[5,6,7],[8,9,10],[15,16,17]]
matrix_B=[[12,13,14]]
matrix_C=np.add(matrix_A,matrix_B)
print(matrix_C)
```

```
[[17 19 21]
 [20 22 24]
 [27 29 31]]
```

Task 1 Create a program that reads in two 5x6 numpy.arrays, adds them together, and then stores the result in a third 5x6 matrix. The program should then print out the first two matrices and the result matrix.

```
#Akriti Kumari Dev
matrix_A=np.random.randint(50,size=(5,6))
matrix_B=np.random.randint(50,size=(5,6))
matrix_C=matrix_A + matrix_B
print("First array:", matrix_A)
print("Second array:", matrix_B)
print("Third array:", matrix_C)
```

```
First array: [[34 32 32 22 30  7]
 [40 18  4  3 25 44]
 [40  2  8 46 27 45]
 [26 26 46 42 11 17]
 [42 16 28 49  7  3]]
Second array: [[ 7 43 48 43 16  6]
 [14 24 35  6 37 26]
 [ 0  0 11  2 42 18]
 [ 6 20 12 47 15 15]
 [10 28 29 32 35 27]]
Third array: [[41 75 80 65 46 13]
 [54 42 39  9 62 70]
 [40  2 19 48 69 63]
 [32 46 58 89 26 32]
 [52 44 57 81 42 30]]
```

Task 2 Create a program that reads in three 2x3 numpy.arrays, subtracts the second from the first, and adds the third to the result. The program should then print out all three matrices and the result matrix.

```
#Akriti Kumari Dev
matrix_1=np.random.randint(20,size=(2,3))
matrix_2=np.random.randint(20,size=(2,3))
```

```
matrix_3=np.random.randint(20,size=(2,3))
matrix_sub = matrix_1 - matrix_2
matrix_add = matrix_sub + matrix_3
print("The first matrix is ")
print(matrix_1)
print("The second matrix is ")
print(matrix_2)
print("The third matrix is ")
print(matrix_3)
print("The final result is ")
print(matrix_add)
```

```
The first matrix is
[[ 7 10 14]
 [ 7 16 14]]
The second matrix is
[[ 1 14  9]
 [ 4  2  7]]
The third matrix is
[[16  4  7]
 [ 2 14 17]]
The final result is
[[22  0 12]
 [ 5 28 24]]
```

Task 3 Create a 4x4 matrix containing random numbers in the range of 1 to 20. Have the user prompted for a number, check that the imputed value is a number and then check if the number is in the matrix. If the number is in the matrix, return the number of times that number is in the matrix.

```
#Akriti Kumari Dev
matrix=np.random.randint(1,20,size=(4,4))
print("The random generated matrix: ",matrix)
number=input("Enter the number to search for: ")
if number.isdigit():
   if number in matrix:
      count=np.count_nonzero(matrix == number)
      print(number, " is in the array ",count," times.")
   else:
      print(number," is not in the array.")
else:
    print("The imputed value is not a number.Enter new number.")
```

```
The random generated matrix:  [[ 4  1 10  1]
 [ 8  9  9 15]
 [17  4 13 19]
 [10 17  8  8]]
Enter the number to search for: 9
9  is not in the array.
<ipython-input-12-3b03f5099da5>:6: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perfor
  if number in matrix:
```

Task 4 Create a text based game that allows a player to play a simple one person game of battleships. With a 6x6 matrix of all 0's add 4 random x,y locations with the values 1, 2, 3 and 4. 1 = 'Battleship' 2 = 'Aircraft Carrier' 3 = 'Submarine' 4 = 'Destroyer' The player then is asked to 'shoot' at 8 matrix locations from the 6x6 matrix. Use a function to assess if the location has a value in it over 0, and if so feedback to the player that they have 'Hit!' and tell them which ship they have sunk. If the location has only a 0 in it, let the player know that their shot was a 'Miss!'. Once all the shots have been taken, show the matrix to the player, with a -1 in the locations they shot at. If any of the ships remain in the matrix, tell the player that they have lost, otherwise, they have won.

```
#Akriti Kumari Dev
import numpy as np

import numpy as np

# Create a 6*6 grid of zeros
game_board = np.zeros((6,6), dtype=int)

# Assign random values to 4 locations in the grid
ship_types = {1: 'Battleship', 2: 'Aircraft Carrier', 3: 'Submarine', 4: 'Destroyer'}

for ship_type, count in zip(ship_types.keys(), range(1, 5)):
    while count > 0:
        x_coord, y_coord = np.random.randint(6, size=2)
        if game_board[x_coord, y_coord] == 0:
```

```
            if game_board[x_coord, y_coord] == 0:
                game_board[x_coord, y_coord] = ship_type
                count -= 1

    # Define a function to check if a given position has been hit
    def check_hit(x_coord, y_coord):
        if game_board[x_coord, y_coord] > 0:
            print('Hit! You sunk the', ship_types[game_board[x_coord, y_coord]])
            game_board[x_coord, y_coord] = -1
            return True
        else:
            print('Miss!')
            game_board[x_coord, y_coord] = -1
            return False

    # Initialize the number of shots taken to 0
    num_shots_taken = 0

    # Allow the player to take 8 shots
    while num_shots_taken < 8:
        x_input, y_input = input('Enter coordinates to shoot (e.g. "2 3"): ').split()
        x_coord, y_coord = int(x_input) - 1, int(y_input) - 1 # convert to 0-based indexing
        if not (0 <= x_coord < 6 and 0 <= y_coord < 6):
            print('Invalid coordinates. Try again.')
            continue
        if check_hit(x_coord, y_coord):
            num_shots_taken += 1

    # Print the final state of the grid
    print(game_board)

    # Check if any ships remain in the grid
    if np.count_nonzero(game_board > 0) == 0:
        print('You won!')
    else:
        print('You lost!')
```

```
Enter coordinates to shoot (e.g. "2 3"): 1 1
Hit! You sunk the Battleship
Enter coordinates to shoot (e.g. "2 3"): 1 2
Miss!
Enter coordinates to shoot (e.g. "2 3"): 1 3
Miss!
Enter coordinates to shoot (e.g. "2 3"): 1 4
Hit! You sunk the Destroyer
Enter coordinates to shoot (e.g. "2 3"): 1 5
Hit! You sunk the Destroyer
Enter coordinates to shoot (e.g. "2 3"): 1 6
Miss!
Enter coordinates to shoot (e.g. "2 3"): 2 1
Hit! You sunk the Submarine
Enter coordinates to shoot (e.g. "2 3"): 2 2
Miss!
Enter coordinates to shoot (e.g. "2 3"): 2 3
Miss!
Enter coordinates to shoot (e.g. "2 3"): 2 4
Miss!
Enter coordinates to shoot (e.g. "2 3"): 2 5
Miss!
Enter coordinates to shoot (e.g. "2 3"): 2 6
Miss!
Enter coordinates to shoot (e.g. "2 3"): 3 1
Miss!
Enter coordinates to shoot (e.g. "2 3"): 3 2
Hit! You sunk the Destroyer
Enter coordinates to shoot (e.g. "2 3"): 3 3
Hit! You sunk the Destroyer
Enter coordinates to shoot (e.g. "2 3"): 3 4
Miss!
Enter coordinates to shoot (e.g. "2 3"): 3 5
Miss!
Enter coordinates to shoot (e.g. "2 3"): 3 6
Miss!
Enter coordinates to shoot (e.g. "2 3"): 4 1
Hit! You sunk the Submarine
Enter coordinates to shoot (e.g. "2 3"): 4 2
Miss!
Enter coordinates to shoot (e.g. "2 3"): 4 3
Miss!
```

```
Enter coordinates to shoot (e.g. "2 3"): 4 4
Miss!
Enter coordinates to shoot (e.g. "2 3"): 4 5
Miss!
Enter coordinates to shoot (e.g. "2 3"): 4 6
Miss!
Enter coordinates to shoot (e.g. "2 3"): 5 1
Miss!
Enter coordinates to shoot (e.g. "2 3"): 5 2
Hit! You sunk the Aircraft Carrier
[[-1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1]
 [-1 -1  0  0  0  2]
 [ 0  0  0  3  0  0]]
```

✓ 1m 59s completed at 3:12 PM