Lab_1

November 8, 2024

1 Lab 1: Numpy Basics

[1]:

```
import numpy as np
```

Question 1 - Generate Natural Numbers Write a function that generates a numpy array of natural numbers from 1 to `n` (both inclusive).

[4]:

```
# Complete the 'generateNNaturalNumbers' function below.
#
# The function is expected to return a numpy array.
# The function accepts an INTEGER n as parameter.
#

def generateNNaturalNumbers(n):
    # Write your code here
    return np.arange(1, n+1)
```

Question 2 - Pacman Score

[10]:

```
# Complete the 'pacmanScore' function below.
#
# The function is expected to return a np scalar int.
# The function accepts following parameters:
#  1. np scalar float time
#  2. np scalar int dots
#  3. np scalar int berries
#  4. np scalar int ghosts
#

def pacmanScore(time, dots, berries, ghosts):
    # Write your code here. REMEMBER TO RETURN AN INTEGER
    S = np.add(np.add(np.multiply(100, ghosts), np.multiply(50, dots)), np.
  ↪multiply(10, np.max(time - 30, 0)))
    S = 10 * max(time - 30, 0) + 20 * dots + 50 * berries + 100 * ghosts
    return np.array(S, int)
```

Question 3 - Calculate BMI

[12]:

```
# Complete the 'calculateBMI' function below.
#
# The function is expected to return an np scalar float.
# The function accepts following parameters:
#  1. np scalar float height
#  2. np scalar float weight
#

def calculateBMI(height, weight):
    # Write your code here
    bmi = np.divide(weight, np.power(np.divide(height, 100), 2))
    return np.round(bmi, 2)
```

Question 4 - Calculate BMI using numpy arrays

[14]:

```
# Complete the 'calculateBMI' function below.
#
# The function is expected to return a numpy vector.
# The function accepts following parameters:
#  1. numpy vector of floats heights
#  2. numpy vector of floats weights
#

def calculateBMI(heights, weights):
    # Write your code here
    bmi = np.divide(weights, np.square(np.divide(heights, 100)))
    return np.round(bmi, 2)
```

Question 5 - Influence, Index and Arrays

[16]:

```
def billionaire_influence(current_landscape, demands):
    # Hint: Column Broadcast
    # Write your code here
    return current_landscape + demands

def politician_influence(current_landscape, index, update_amount):
    # Hint : Column Broadcast
    # write your code here
    current_landscape[:, index] += update_amount
    return current_landscape
```

Question 6 - Generate Multiplication table from two arrays

[19]:

```
# Complete the 'getMultiplicationTable' function below.

def getMultiplicationTable(multiply_with_left, multiply_with_right):
    # We generate a multiplication table
    # we can multiply [1, 2, 3] with [1, 2, 3, 4] to get:
    # [1, 2, 3, 4], [2, 4, 6, 8], [3, 6, 9, 12]
    # The first row contains 1*1, 1*2, 1*3, 1*4
    # Write your code here
    table = np.multiply(multiply_with_left.reshape(-1, 1), multiply_with_right)
    return table
```

Question 7 - Softmax Function Preprocessing

[24]:

```
# Complete the 'softmax_preprocess' function below.
#
# The function is expected to return a 2D numpy array.
# The function accepts 2D numpy array y as parameter.
#
# The function should subtract the maximum value in each row from all the␣
 ↪elements of that row.

def softmax_preprocess(y):
    # Write your code here
    y = np.subtract(y, np.max(y, axis=1).reshape(-1, 1))
    return y
```