lab2_np

September 5, 2024

0.0.1 NumPy Version

[ ]:
```
%pip install numpy
```

```
Requirement already satisfied: numpy in
c:\users\julian\appdata\local\programs\python\python311\lib\site-packages
(1.25.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.1.2 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[ ]:
```
import numpy as np
```

[ ]:
```
print(np.__version__)
```

```
1.25.0
```

[ ]:
```
# Define the housing price related feature for location
list_row1  = [-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473]
list_row2  = [-121.12, 39.03, 17, 838, 161, 388, 142, 3.6563]
```

[ ]:
```
# create Two numpy arrays from row1 and row2
np_row1 = np.array(list_row1)
np_row2 = np.array(list_row2)
```

[ ]:
```
# check variables' data type
print(type(np_row1), ' vs ', type(list_row1))
```

```
<class 'numpy.ndarray'>  vs  <class 'list'>
```

[ ]:
```
# check 1D Numpy array' dimension
print("np_row1.shape: ", np_row1.shape)
```

```
np_row1.shape:  (8,)
```

[ ]:
```
# Indexing in 1 dimension numpy vector (rank 1 -tensor)
np_row1 = np.array([-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473])
print("np_row1: ", np_row1)
```

```
np_row1:  [-1.2187e+02  3.7230e+01  1.9000e+01  7.3570e+03  9.6300e+02
3.0180e+03
  9.8100e+02  6.9473e+00]
```

[ ]:
```
# Each element in the 1d array can be accessed by passing the positional index␣
 ↪of the element.
print("np_row1[0]: ", np_row1[0])
print("np_row1[1]: ", np_row1[1])
```

```
np_row1[0]:  -121.87
np_row1[1]:  37.23
```

[ ]:
```
# Created 2D Numpy Array based on nested list
np_rows = np.array([
  [-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473],
  [-121.12, 39.03, 17, 838, 161, 388, 142, 3.6563]
])
print("np_rows: ", np_rows)
print("np_rows.shape: ", np_rows.shape)
```

```
np_rows:  [[-1.2187e+02  3.7230e+01  1.9000e+01  7.3570e+03  9.6300e+02
3.0180e+03
   9.8100e+02  6.9473e+00]
 [-1.2112e+02  3.9030e+01  1.7000e+01  8.3800e+02  1.6100e+02  3.8800e+02
   1.4200e+02  3.6563e+00]]
np_rows.shape:  (2, 8)
```

[ ]:
```
# Indexing in 2 dimensions (rank 2 - tensor)
# We can retrieve an element of the 2D Numpy array using two indices i and j -␣
 ↪i selects the row, and j selects the column:
# we can pass i-th row and j-th column in either one bracket or separate␣
 ↪brackets ([])
print("np_rows[1, 2]: ", np_rows[1, 2]) # option 1
print("np_rows[1][2]: ", np_rows[1][2]) # option 2
```

```
np_rows[1, 2]:  17.0
np_rows[1][2]:  17.0
```

[ ]:
```
# Slicing in 1 dimension numpy vector (rank 1 -tensor)
# pick the second, third, and forth element from the array
np_location1 = np.array([-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473])
print("np_location1[1:4]: ", np_location1[1:4])
```

```
np_location1[1:4]:  [  37.23   19.   7357.  ]
```

[ ]:
```
# define 2d array
np_locations = np.array([
  [-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473],
  [-121.12, 39.03, 17, 838, 161, 388, 142, 3.6563],
```

```
  [-119.31, 36.06, 20, 2236, 434, 1405, 412, 1.8827]
])
print("np_locations: ", np_locations)
```

```
np_locations:  [[-1.2187e+02  3.7230e+01  1.9000e+01  7.3570e+03  9.6300e+02
3.0180e+03
   9.8100e+02  6.9473e+00]
 [-1.2112e+02  3.9030e+01  1.7000e+01  8.3800e+02  1.6100e+02  3.8800e+02
   1.4200e+02  3.6563e+00]
 [-1.1931e+02  3.6060e+01  2.0000e+01  2.2360e+03  4.3400e+02  1.4050e+03
   4.1200e+02  1.8827e+00]]
```

[ ]:
```
# select all rows except 1st row
# select 3rd and 4th column
print("np_locations[1:,2:4]: ", np_locations[1:,2:4])
```

```
np_locations[1:,2:4]:  [[  17.  838.]
 [  20. 2236.]]
```

[ ]:
```
## Peform matrix addition

A = np.random.rand(2, 2)
print(f"Matrix A:\n\n{A}\n")
B= np.random.rand(2, 2)
print(f"Matrix B:\n\n{B}\n")

C = A + B
print(f"Matrix addition is:\n\n{C}\n")
```

```
Matrix A:

[[0.31586734 0.34775084]
 [0.82875602 0.04354343]]

Matrix B:

[[0.95988284 0.32425257]
 [0.70455136 0.81895288]]

Matrix addition is:

[[1.27575017 0.6720034 ]
 [1.53330738 0.86249631]]
```

[ ]:
```
# adding the two arrays
np_location1 = np.array([-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473])
np_location2 = np.array([-121.12, 39.03, 17, 838, 161, 388, 142, 3.6563])
```

```
np_location_sum = np_location1 + np_location2

print("np_location_sum: ",np_location_sum)
```

```
np_location_sum:  [-242.99     76.26     36.     8195.     1124.     3406.
1123.
   10.6036]
```

[ ]:
```
# get sum of values in array
print("np_location1.sum(): ", np_location1.sum())
```

```
np_location1.sum():  12260.3073
```

[ ]:
```
# get min of values in array
print("np_location1.min(): ", np_location1.min())
```

```
np_location1.min():  -121.87
```

[ ]:
```
# concatenate two 1D arrays
list_location1 = [-121.87, 37.23, 19, 7357, 963, 3018, 981, 6.9473]
list_location2 = [-121.12, 39.03, 17, 838, 161, 388, 142, 3.6563]

np_location_concat = np.concatenate([list_location1, list_location2])

print("np_location_concat: ", np_location_concat)
```

```
np_location_concat:  [-1.2187e+02  3.7230e+01  1.9000e+01  7.3570e+03
9.6300e+02  3.0180e+03
  9.8100e+02  6.9473e+00 -1.2112e+02  3.9030e+01  1.7000e+01  8.3800e+02
  1.6100e+02  3.8800e+02  1.4200e+02  3.6563e+00]
```

0.1 NumPy Problem 1

Task 1

[ ]:
```
longitude = np.array([-121.87, -121.12, -119.31, -118.03, -120.97,-118.18, -117.
↪7])
latitude = np.array([37.23, 39.03, 36.06, 33.78, 37.61, 34.02, 33.6])
total_rooms = np.array([7357, 838, 2236, 3554, 1326, 2631, 2092])
population = np.array([3018, 388, 1405, 1600, 884, 3228, 877])
households = np.array([981, 142, 412, 537, 375, 701, 392])

#print the shapes

print(f"longitude shape: {longitude.shape}")
print(f"latitude shape: {latitude.shape}")
print(f"total_rooms shape: {total_rooms.shape}")
print(f"population shape: {population.shape}")
print(f"households shape: {households.shape}")
```

```
longitude shape: (7,)
latitude shape: (7,)
total_rooms shape: (7,)
population shape: (7,)
households shape: (7,)

latitude shape: (7,)
total_rooms shape: (7,)
population shape: (7,)
households shape: (7,)
```

Task 2

```
[ ]: #calculate sum,mean,min,max of all the arrays
print(f"longitude sum: {longitude.sum()}, mean: {longitude.mean()}, min:␣
  ↪{longitude.min()}, max: {longitude.max()}")
print(f"latitude sum: {latitude.sum()}, mean: {latitude.mean()}, min: {latitude.
  ↪min()}, max: {latitude.max()}")
print(f"total_rooms sum: {total_rooms.sum()}, mean: {total_rooms.mean()}, min:␣
  ↪{total_rooms.min()}, max: {total_rooms.max()}")
print(f"population sum: {population.sum()}, mean: {population.mean()}, min:␣
  ↪{population.min()}, max: {population.max()}")
print(f"households sum: {households.sum()}, mean: {households.mean()}, min:␣
  ↪{households.min()}, max: {households.max()}")
```

```
longitude sum: -837.1800000000001, mean: -119.59714285714287, min: -121.87, max:
-117.7
latitude sum: 251.32999999999998, mean: 35.90428571428571, min: 33.6, max: 39.03
total_rooms sum: 20034, mean: 2862.0, min: 838, max: 7357
population sum: 11400, mean: 1628.5714285714287, min: 388, max: 3228
households sum: 3540, mean: 505.7142857142857, min: 142, max: 981
```

Task 3

```
[ ]: #create a 2d array from the above arrays
np_2d_array = np.array([[-117.7,33.6,16,2092,489,877,392,3.0461],[-121.87, 37.
  ↪23, 19, 7357, 963, 3018, 981, 6.9473],[-121.12, 39.03, 17, 838, 161, 388,␣
  ↪142, 3.6563]])
print(f"2d array shape: {np_2d_array.shape}")
```

```
2d array shape: (3, 8)
```

0.2 NumPy Problem 2

```
[ ]: import tensorflow as tf
(mnist_images_training,_),(mnist_images_test,_) = tf.keras.datasets.mnist.
  ↪load_data(path="mnist.npz")
```

Task 1

[ ]:
```
print(f"mnist_images_training shape: {mnist_images_training.shape}")#tensor 3
print(f"mnist_images_test shape: {mnist_images_test.shape}") #tensor 3
```

```
mnist_images_training shape: (60000, 28, 28)
mnist_images_test shape: (10000, 28, 28)
```

Task 2

[ ]:
```
image = mnist_images_training[0]
print(f"image shape: {image.shape}")
```

```
image shape: (28, 28)
```

Task 3

[ ]:
```
images = mnist_images_training[0:10]
print(f"images shape: {images.shape}")
```

```
images shape: (10, 28, 28)
```