assignment4

October 9, 2024

Assignment 4 - NumPy and Plotting

Python Programming Lab

Name: Soumyadeep Ganguly
Reg No.: 24MDT0082

[2]:
```
import numpy as np
```

1 Exercise 1.

1.1 a) Using the list comprehension technique generate the NumPy array

[3]:
```
A = np.array([[j for j in range(i, 82,10) if j != i+30] for i in range(15,22)␣
  ↪if i != 17 ])
print(A)
```

```
[[15 25 35 55 65 75]
 [16 26 36 56 66 76]
 [18 28 38 58 68 78]
 [19 29 39 59 69 79]
 [20 30 40 60 70 80]
 [21 31 41 61 71 81]]
```

1.2 b) Extract a sub matrices of order 4 × 3 and 4 × 5 from the above matrix A and name them as B and C.

[4]:
```
B = A[:4, 3:]
B
```

[4]:
```
array([[55, 65, 75],
       [56, 66, 76],
       [58, 68, 78],
       [59, 69, 79]])
```

[5]:
```
C = A[:4, 1:]
C
```

[5]:
```
array([[25, 35, 55, 65, 75],
       [26, 36, 56, 66, 76],
       [28, 38, 58, 68, 78],
       [29, 39, 59, 69, 79]])
```

1.3 c) Extract the submatrix

[6]:
```
D = A[-2::-4, ::2]
D
```

[6]:
```
array([[20, 40, 70],
       [15, 35, 65]])
```

1.4 d) Generate random integer matrices $E_{3\times3}$, $F_{5\times3}$ with entries between 10 and 60 and G a random matrix containing 5 elements.

[7]:
```
E = np.random.randint(10, 60, size=(3,3))
print(E)
```

```
[[49 47 50]
 [39 23 44]
 [12 23 49]]
```

[8]:
```
F = np.random.randint(10, 60, size=(5,3))
print(F)
```

```
[[47 51 19]
 [53 56 53]
 [15 35 54]
 [18 46 35]
 [45 56 17]]
```

[9]:
```
G = np.random.rand(5)
print(G)
```

```
[0.00371302 0.30494738 0.74175373 0.2653716  0.8379159 ]
```

1.5 e) Compute the matrix $H = BF^TC + G$. Whileevaluatingtheproductuse appropriate product such as matrix product and dot product wherever applicable.

[10]:
```
H = B@(F.T)*C + G
print(H)
```

```
[[183125.00371302 368550.30494738 393250.74175373 429325.2653716
  554250.8379159 ]
 [193492.00371302 384912.30494738 406224.74175373 442464.2653716
```

```
 570608.8379159 ]
[214928.00371302 418608.30494738 432796.74175373 469336.2653716
 604032.8379159 ]
[225997.00371302 435942.30494738 446394.74175373 483069.2653716
 621098.8379159 ]]
```

1.6 f) Round the entries of H to two decimal points and then find the square root of these elements and store the resultant into HS.

[11]:
```
HS = np.sqrt(np.round(H, 2))
print(HS)
```

```
[[427.93106922 607.08343743 627.09707383 655.22917365 744.48024823]
 [439.87725561 620.4130076  637.35762332 665.17987793 755.38655005]
 [463.60327868 646.99945904 657.87289046 685.08121416 777.19549664]
 [475.39141768 660.25926726 668.12778718 695.03184819 788.09824261]]
```

1.7 g) Find the inverse of the matrix, eigenvalues and eigenvectors of the matrix E.

[12]:
```
E_inv = np.linalg.inv(E)
E_evalues, E_evectors = np.linalg.eig(E)

print(f'Inverse Matrix :{E_inv}')
print(f'\n\nEigen Values :{E_evalues}')
print(f'\n\nEigen Vectors :{E_evectors}')
```

```
Inverse Matrix :[[-0.00406131  0.04071903 -0.03241983]
 [ 0.04884164 -0.06360362  0.00727504]
 [-0.02193106  0.01988275  0.0249329 ]]


Eigen Values :[107.25734632  24.5126334  -10.76997972]


Eigen Vectors :[[-0.75460484 -0.72042895  0.47413878]
 [-0.54224606 -0.29324745 -0.84942343]
 [-0.36951421  0.62848076  0.23167272]]
```

1.8 h) Find the row-wise and column-wise sum and product of the elements of the matrix $(ED^T)^T F^T$.

[15]:
```
matrix = ((E@D.T).T)@F.T

col_sum = np.sum(matrix, axis=0)
row_sum = np.sum(matrix, axis=1)

col_prod = np.prod(matrix, axis=0)
```

```
row_prod = np.prod(matrix, axis=1)

print(f'Row wise sum: {row_sum}')
print(f'Column wise sum: {col_sum}')
print(f'Row wise product: {row_prod}')
print(f'Column wise product: {col_prod}')
```

```
Row wise sum: [3115420 2781400]
Column wise sum: [1190500 1605430  968940  937800 1194150]
Row wise product: [-834538870497166848 3590099011828555776]
Column wise product: [353121246900 642297922000 234030492800 219185477900
355285078400]
```

1.9 i) Is the matrices E, F are broadcastable? If Yes, what is the shape of the resultant matrix? If No, give the reason.

[16]:
```
E*F
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 1
----> 1 E*F

ValueError: operands could not be broadcast together with shapes (3,3) (5,3)
```

1.9.1 E and F are not broadcastable

Because the shapes of the matrices are different

1.10 j) Provide the Python command to generate a matrix of order 4 × 3 × 5 × 3 by appropriately reshaping the matrices C,E,F and adding them.

[17]:
```
C_reshaped = np.reshape(C,(4,1,5,1))
E_reshaped = np.reshape(E,(1,3,1,3))
F_reshaped = np.reshape(F,(1,1,5,3))
result = C_reshaped + E_reshaped + F_reshaped
print(result.shape)
print(result)
```

```
(4, 3, 5, 3)
[[[[121 123  94]
   [137 138 138]
   [119 137 159]
   [132 158 150]
   [169 178 142]]

  [[111  99  88]
```

```
  [127 114 132]
  [109 113 153]
  [122 134 144]
  [159 154 136]]

 [[ 84  99  93]
  [100 114 137]
  [ 82 113 158]
  [ 95 134 149]
  [132 154 141]]]


[[[122 124  95]
  [138 139 139]
  [120 138 160]
  [133 159 151]
  [170 179 143]]

 [[112 100  89]
  [128 115 133]
  [110 114 154]
  [123 135 145]
  [160 155 137]]

 [[ 85 100  94]
  [101 115 138]
  [ 83 114 159]
  [ 96 135 150]
  [133 155 142]]]


[[[124 126  97]
  [140 141 141]
  [122 140 162]
  [135 161 153]
  [172 181 145]]

 [[114 102  91]
  [130 117 135]
  [112 116 156]
  [125 137 147]
  [162 157 139]]

 [[ 87 102  96]
  [103 117 140]
  [ 85 116 161]
  [ 98 137 152]
  [135 157 144]]]
```

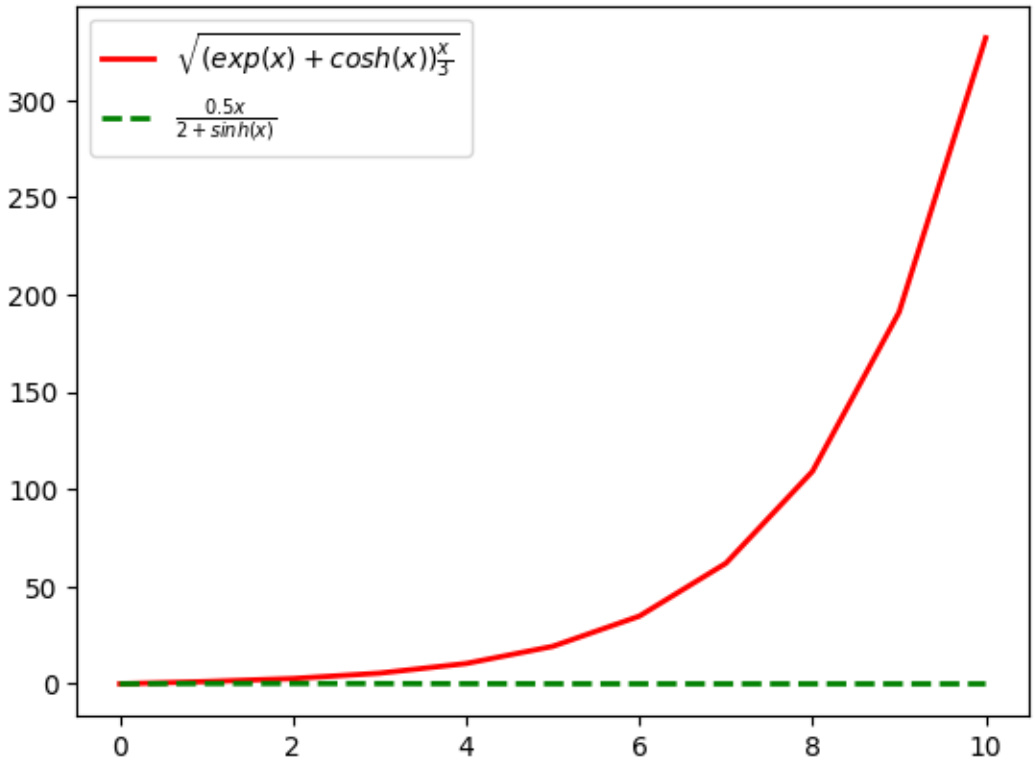
```
[[[125 127  98]
  [141 142 142]
  [123 141 163]
  [136 162 154]
  [173 182 146]]

 [[115 103  92]
  [131 118 136]
  [113 117 157]
  [126 138 148]
  [163 158 140]]

 [[ 88 103  97]
  [104 118 141]
  [ 86 117 162]
  [ 99 138 153]
  [136 158 145]]]
```

[18]:
```
import matplotlib.pyplot as plt
```

2 Exercise 2.

[19]:
```
x = [j for j in range(11)]
f = [np.sqrt(((np.exp(i)+np.cosh(i))*i)/3) for i in range(11)]
g = [ ((0.5*k)/(2 + np.sinh(k)))  for k in range(11)]


fig, ax = plt.subplots()
ax.plot(x, f, "red", linewidth = '2',␣
 ↪label=r'$\sqrt{(exp(x)+cosh(x))\frac{x}{3}}$')
ax.plot(x, g, "green", linewidth = '2',linestyle = 'dashed', label=r'$\frac{0.
 ↪5x}{2+sinh(x)}$')
plt.legend()
```

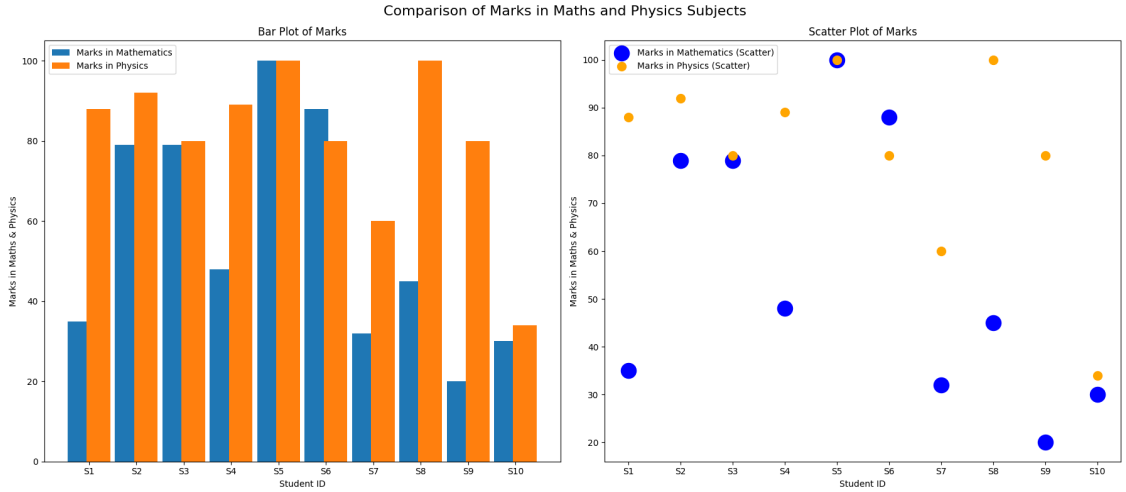
[19]:
```
<matplotlib.legend.Legend at 0x173e97a6a20>
```



3 Exercise 3.

[20]:
```
students = ["S1", "S2", "S3", "S4", "S5", "S6", "S7", "S8", "S9", "S10"]
marks_in_maths = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_in_physics = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]

bar_width = 0.2
index = np.arange(len(students))

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 8))

ax1.bar(index - bar_width, marks_in_maths, width=0.5, label='Marks in␣
 ↪Mathematics')
ax1.bar(index + bar_width, marks_in_physics, width=0.5, label='Marks in␣
 ↪Physics')
ax1.set_xticks(index)
ax1.set_xticklabels(students)
ax1.set_xlabel("Student ID")
ax1.set_ylabel("Marks in Maths & Physics")
ax1.set_title("Bar Plot of Marks")
ax1.legend()
```

```
ax2.scatter(index, marks_in_maths, marker="o", s=300, label='Marks in␣
 ↪Mathematics (Scatter)', color='blue')
ax2.scatter(index, marks_in_physics, marker="o", s=100, label='Marks in Physics␣
 ↪(Scatter)', color='orange')
ax2.set_xticks(index)
ax2.set_xticklabels(students)
ax2.set_xlabel("Student ID")
ax2.set_ylabel("Marks in Maths & Physics")
ax2.set_title("Scatter Plot of Marks")
ax2.legend()

fig.suptitle("Comparison of Marks in Maths and Physics Subjects", fontsize=16)
plt.tight_layout()
plt.show()
```



[ ]: