**Task1:**

active environment : None

user config file : C:\Users\maver\.condarc

populated config files : C:\Users\maver\.condarc

conda version : 23.7.2

conda-build version : 3.26.0

python version : 3.11.4.final.0

virtual packages : \_\_archspec=1=x86\_64

\_\_cuda=12.0=0

\_\_win=0=0

base environment : C:\Users\maver\Anaconda3 (writable)

conda av data dir : C:\Users\maver\Anaconda3\etc\conda

conda av metadata url : None

channel URLs : https://repo.anaconda.com/pkgs/main/win-64

https://repo.anaconda.com/pkgs/main/noarch

https://repo.anaconda.com/pkgs/r/win-64

https://repo.anaconda.com/pkgs/r/noarch

https://repo.anaconda.com/pkgs/msys2/win-64

https://repo.anaconda.com/pkgs/msys2/noarch

package cache : C:\Users\maver\Anaconda3\pkgs

C:\Users\maver\.conda\pkgs

C:\Users\maver\AppData\Local\conda\conda\pkgs

envs directories : C:\Users\maver\Anaconda3\envs

C:\Users\maver\.conda\envs

C:\Users\maver\AppData\Local\conda\conda\envs

platform : win-64

user-agent : conda/23.7.2 requests/2.31.0 CPython/3.11.4 Windows/10 Windows/10.0.22621

administrator : True

netrc file : None

offline mode : False

Task 2:

In [1]: import numpy as np

...: import scipy.linalg

In [2]: a = np.array([[1, 2, 3, 4],

...: [5, 6, 7, 8],

...: [9, 10, 11, 12],

...: [13, 14, 15, 16]])

In [3]: np.ndim(a)

Out[3]: 2

In [4]: np.size(a)

Out[4]: 16

In [5]: np.shape(a)

Out[5]: (4, 4)

In [6]: n = 2

In [7]: a.shape[n-1]

Out[7]: 4

In [8]: np.array([[1., 2., 3.], [4., 5., 6.]])

Out[8]:

array([[1., 2., 3.],

[4., 5., 6.]])

In [9]: b = np.array([[17, 18, 19, 20],

...: [21, 22, 23, 24],

...: [25, 26, 27, 28],

...: [29, 30, 31, 32]])

In [10]: c = np.array([[33, 34, 35, 36],

...: [37, 38, 39, 40],

...: [41, 42, 43, 44],

...: [45, 46, 47, 48]])

In [11]: d = np.array([[49, 50, 51, 52],

...: [53, 54, 55, 56],

...: [57, 58, 59, 60],

...: [61, 62, 63, 64]])

In [12]: np.block([[a, b], [c, d]])

Out[12]:

array([[ 1, 2, 3, 4, 17, 18, 19, 20],

[ 5, 6, 7, 8, 21, 22, 23, 24],

[ 9, 10, 11, 12, 25, 26, 27, 28],

[13, 14, 15, 16, 29, 30, 31, 32],

[33, 34, 35, 36, 49, 50, 51, 52],

[37, 38, 39, 40, 53, 54, 55, 56],

[41, 42, 43, 44, 57, 58, 59, 60],

[45, 46, 47, 48, 61, 62, 63, 64]])

In [13]: a[-1]

Out[13]: array([13, 14, 15, 16])

In [15]: a[1, 3]

Out[15]: 8

In [16]: a[1]

Out[16]: array([5, 6, 7, 8])

In [17]: a[0:3]

Out[17]:

array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12]])

In [18]: a[-2:]

Out[18]:

array([[ 9, 10, 11, 12],

[13, 14, 15, 16]])

In [19]: a[0:2, 1:3]

Out[19]:

array([[2, 3],

[6, 7]])

In [20]: a[np.ix\_([0, 2, 3], [0, 1])]

Out[20]:

array([[ 1, 2],

[ 9, 10],

[13, 14]])

In [22]: a[1:3:1,:]

Out[22]:

array([[ 5, 6, 7, 8],

[ 9, 10, 11, 12]])

In [23]: a[::2, :]

Out[23]:

array([[ 1, 2, 3, 4],

[ 9, 10, 11, 12]])

In [24]: a[::-1,:]

Out[24]:

array([[13, 14, 15, 16],

[ 9, 10, 11, 12],

[ 5, 6, 7, 8],

[ 1, 2, 3, 4]])

In [25]: a[np.r\_[:len(a),0]]

Out[25]:

array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12],

[13, 14, 15, 16],

[ 1, 2, 3, 4]])

In [26]: a.transpose()

Out[26]:

array([[ 1, 5, 9, 13],

[ 2, 6, 10, 14],

[ 3, 7, 11, 15],

[ 4, 8, 12, 16]])

In [27]: a.conj().transpose()

Out[27]:

array([[ 1, 5, 9, 13],

[ 2, 6, 10, 14],

[ 3, 7, 11, 15],

[ 4, 8, 12, 16]])

In [28]: a @ b

Out[28]:

array([[ 250, 260, 270, 280],

[ 618, 644, 670, 696],

[ 986, 1028, 1070, 1112],

[1354, 1412, 1470, 1528]])

In [29]: a \* b

Out[29]:

array([[ 17, 36, 57, 80],

[105, 132, 161, 192],

[225, 260, 297, 336],

[377, 420, 465, 512]])

In [30]: a/b

Out[30]:

array([[0.05882353, 0.11111111, 0.15789474, 0.2 ],

[0.23809524, 0.27272727, 0.30434783, 0.33333333],

[0.36 , 0.38461538, 0.40740741, 0.42857143],

[0.44827586, 0.46666667, 0.48387097, 0.5 ]])

In [31]: a\*\*3

Out[31]:

array([[ 1, 8, 27, 64],

[ 125, 216, 343, 512],

[ 729, 1000, 1331, 1728],

[2197, 2744, 3375, 4096]], dtype=int32)

In [32]: (a > 0.5)

Out[32]:

array([[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True]])

In [33]: np.nonzero(a > 0.5)

Out[33]:

(array([0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3], dtype=int64),

array([0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3], dtype=int64))

In [35]: v = np.array([1, 2, 3, 4])

In [36]: a[:,np.nonzero(v > 0.5)[0]]

Out[36]:

array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12],

[13, 14, 15, 16]])

In [37]: a[:, v.T > 0.5]

Out[37]:

array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12],

[13, 14, 15, 16]])

In [38]: a[a < 0.5]=0

In [39]: a \* (a > 0.5)

Out[39]:

array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12],

[13, 14, 15, 16]])

In [40]: a[:] = 3

In [41]: x = [1, 2, 3]

In [42]: y = x.copy()

In [43]: y = x[1, :].copy()

In [45]: x = a

In [46]: y = x[1, :].copy()

In [47]: y = x.flatten()

In [48]: np.arange(1., 11.)

Out[48]: array([ 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.])

In [49]: np.arange(10.)

Out[49]: array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

In [50]: np.arange(1.,11.)[:, np.newaxis]

Out[50]:

array([[ 1.],

[ 2.],

[ 3.],

[ 4.],

[ 5.],

[ 6.],

[ 7.],

[ 8.],

[ 9.],

[10.]])

In [51]: np.zeros((3, 4))

Out[51]:

array([[0., 0., 0., 0.],

[0., 0., 0., 0.],

[0., 0., 0., 0.]])

In [52]: np.zeros((3, 4, 5))

Out[52]:

array([[[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.]],

[[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.]],

[[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0.]]])

In [54]: np.ones((3, 4))

Out[54]:

array([[1., 1., 1., 1.],

[1., 1., 1., 1.],

[1., 1., 1., 1.]])

In [55]: np.eye(3)

Out[55]:

array([[1., 0., 0.],

[0., 1., 0.],

[0., 0., 1.]])

In [56]: np.diag(a)

Out[56]: array([3, 3, 3, 3])

In [57]: np.diag(v, 0)

Out[57]:

array([[1, 0, 0, 0],

[0, 2, 0, 0],

[0, 0, 3, 0],

[0, 0, 0, 4]])

In [64]: from numpy.random import default\_rng

...: rng = default\_rng(42)

...: rng.random((3, 4))

Out[64]:

array([[0.77395605, 0.43887844, 0.85859792, 0.69736803],

[0.09417735, 0.97562235, 0.7611397 , 0.78606431],

[0.12811363, 0.45038594, 0.37079802, 0.92676499]])

In [65]: np.linspace(1,3,4)

Out[65]: array([1. , 1.66666667, 2.33333333, 3. ])

In [66]: np.mgrid[0:9.,0:6.]

Out[66]:

array([[[0., 0., 0., 0., 0., 0.],

[1., 1., 1., 1., 1., 1.],

[2., 2., 2., 2., 2., 2.],

[3., 3., 3., 3., 3., 3.],

[4., 4., 4., 4., 4., 4.],

[5., 5., 5., 5., 5., 5.],

[6., 6., 6., 6., 6., 6.],

[7., 7., 7., 7., 7., 7.],

[8., 8., 8., 8., 8., 8.]],

[[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.],

[0., 1., 2., 3., 4., 5.]]])

In [70]: np.ogrid[0:9.,0:6.]

Out[70]:

[array([[0.],

[1.],

[2.],

[3.],

[4.],

[5.],

[6.],

[7.],

[8.]]),

array([[0., 1., 2., 3., 4., 5.]])]

In [71]: np.meshgrid([1,2,4],[2,4,5])

Out[71]:

[array([[1, 2, 4],

[1, 2, 4],

[1, 2, 4]]),

array([[2, 2, 2],

[4, 4, 4],

[5, 5, 5]])]

In [72]: np.ix\_([1,2,4],[2,4,5])

Out[72]:

(array([[1],

[2],

[4]]),

array([[2, 4, 5]]))

In [74]: np.tile(a, (3, n))

Out[74]:

array([[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3],

[3, 3, 3, 3, 3, 3, 3, 3]])

In [75]: np.concatenate((a,b),1)

Out[75]:

array([[ 3, 3, 3, 3, 17, 18, 19, 20],

[ 3, 3, 3, 3, 21, 22, 23, 24],

[ 3, 3, 3, 3, 25, 26, 27, 28],

[ 3, 3, 3, 3, 29, 30, 31, 32]])

In [76]: a.max()

Out[76]: 3

In [77]: a.max(0)

Out[77]: array([3, 3, 3, 3])

In [78]: a.max(1)

Out[78]: array([3, 3, 3, 3])

In [79]: np.maximum(a, b)

Out[79]:

array([[17, 18, 19, 20],

[21, 22, 23, 24],

[25, 26, 27, 28],

[29, 30, 31, 32]])

In [80]: np.sqrt(v @ v)

Out[80]: 5.477225575051661

In [82]: np.logical\_and(a,b)

Out[82]:

array([[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True]])

In [83]: np.logical\_or(a,b)

Out[83]:

array([[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True],

[ True, True, True, True]])

In [84]: a & b

Out[84]:

array([[1, 2, 3, 0],

[1, 2, 3, 0],

[1, 2, 3, 0],

[1, 2, 3, 0]])

In [85]: a | b

Out[85]:

array([[19, 19, 19, 23],

[23, 23, 23, 27],

[27, 27, 27, 31],

[31, 31, 31, 35]])

In [91]: a = np.array([[1, 2],

...: [3, 4]])

In [92]: np.linalg.inv(a)

Out[92]:

array([[-2. , 1. ],

[ 1.5, -0.5]])

In [94]: np.linalg.pinv(a)

Out[94]:

array([[-2. , 1. ],

[ 1.5, -0.5]])

In [95]: np.linalg.matrix\_rank(a)

Out[95]: 2

In [97]: b = np.array([[5, 6],

...: [7, 8]])

...:

In [98]: np.linalg.solve(a, b)

Out[98]:

array([[-3., -4.],

[ 4., 5.]])

Missing command here

In [103]: U, S, Vh = np.linalg.svd(a); V = Vh.T

In [107]: a = np.array([[4, 2],

...: [2, 5]])

...:

In [108]: np.linalg.cholesky(a)

Out[108]:

array([[2., 0.],

[1., 2.]])

In [109]: D,V = np.linalg.eig(a)

D,V = np.linalg.eig(a)

In [116]: np.eigs(a, k=3)

---------------------------------------------------------------------------

AttributeError Traceback (most recent call last)

Cell In[116], line 1

----> 1 np.eigs(a, k=3)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\numpy\\_\_init\_\_.py:322, in \_\_getattr\_\_(attr)

319 "Removed in NumPy 1.25.0"

320 raise RuntimeError("Tester was removed in NumPy 1.25.")

--> 322 raise AttributeError("module {!r} has no attribute "

323 "{!r}".format(\_\_name\_\_, attr))

AttributeError: module 'numpy' has no attribute 'eigs'

In [117]: Q,R = np.linalg.qr(a)

In [121]: from scipy.linalg import lu

In [122]: P, L, U = lu(a)

In [126]: from scipy.sparse.linalg import cg

...: A = np.array([[4.0, 2.0],

...: [2.0, 5.0]])

...: b = np.array([1.0, 2.0])

...: x, info = cg(A, b)

In [127]: np.fft.fft(a)

Out[127]:

array([[ 6.+0.j, 2.+0.j],

[ 7.+0.j, -3.+0.j]])

In [128]: np.fft.ifft(a)

Out[128]:

array([[ 3. +0.j, 1. +0.j],

[ 3.5+0.j, -1.5+0.j]])

In [129]: np.sort(a)

Out[129]:

array([[2, 4],

[2, 5]])

In [130]: np.sort(a, axis=1)

Out[130]:

array([[2, 4],

[2, 5]])

In [131]: I = np.argsort(a[:, 0]); b = a[I,:]

In [133]: Z = np.array([[1, 2],

...: [3, 4],

...: [5, 6]])

...:

...: y = np.array([7, 8, 9])

In [134]: x = np.linalg.lstsq(Z, y)

<ipython-input-134-2b18ce8b977f>:1: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.

To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.

x = np.linalg.lstsq(Z, y)

In [137]: from scipy.signal import resample

In [138]: x = np.array([1, 2, 3, 4, 5])

...: q = 2

In [139]: resample(x, int(np.ceil(len(x) / q)))

Out[139]: array([2. , 2.30801829, 4.69198171])

In [140]: np.unique(a)

Out[140]: array([2, 4, 5])

In [141]: a.squeeze()

Out[141]:

array([[4, 2],

[2, 5]])

**Task 3:**

A graph with a line

Description automatically generated

**Task 4:**

In [1]: import numpy as np

...: import matplotlib.pyplot as plt

...: x = np.linspace(0, 10, 100)

...: y = np.sin(x)

...: plt.figure(figsize=(8, 6))

...: plt.plot(x, y, label='sin(x)', color='black', linestyle='-', linewidth=2)

...: plt.xlabel('x')

...: plt.ylabel('y')

...: plt.grid(True)

...: plt.show()

A graph of a function

Description automatically generated

**Task 5:**

maverickxl123