Robert Heeter
ELEC 576 Introduction to Deep Learning
13 September 2023

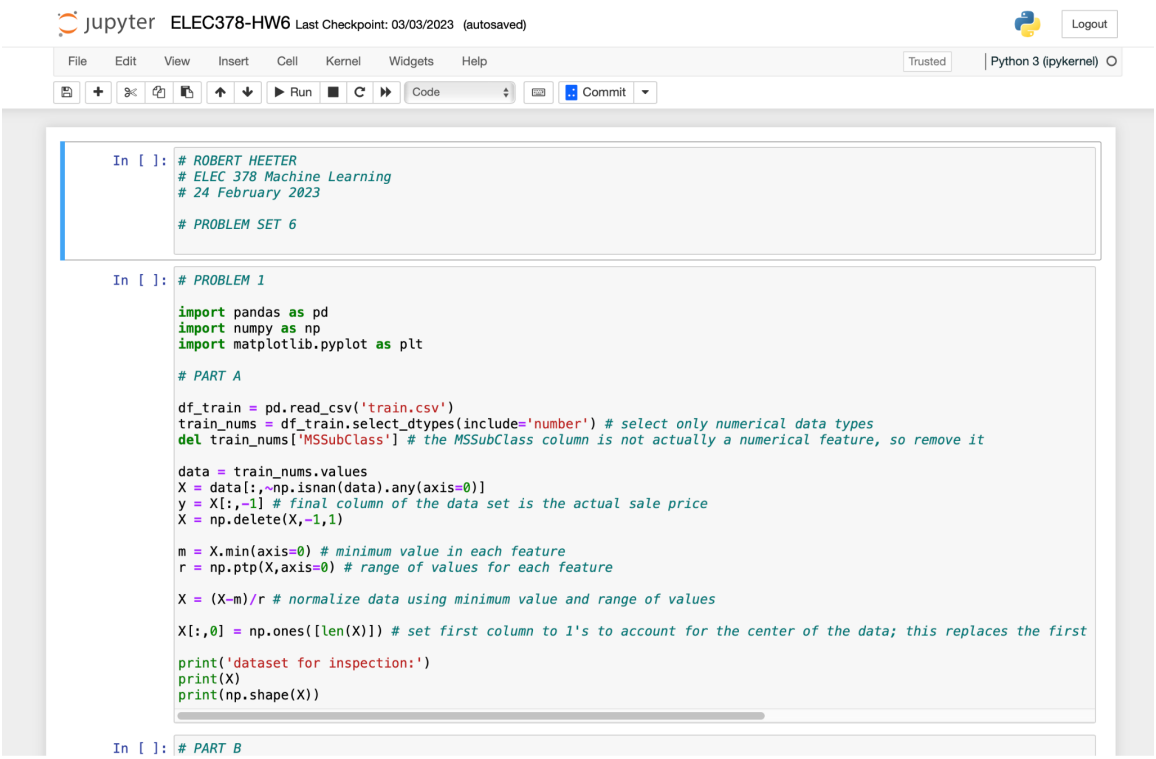
Problem Set #0

1. **Terminal output from >> conda info**

```
Last login: Tue Sep 12 10:46:30 on ttys000
(base) rch@Robert-C-Heeters-MacBook-Pro-3 ~ % conda info

     active environment : base
    active env location : /Users/rch/opt/anaconda3
            shell level : 1
       user config file : /Users/rch/.condarc
 populated config files : /Users/rch/.condarc
          conda version : 23.7.3
    conda-build version : 3.24.0
         python version : 3.9.7.final.0
       virtual packages : __archspec=1=x86_64
                          __osx=10.16=0
                          __unix=0=0
       base environment : /Users/rch/opt/anaconda3  (writable)
      conda av data dir : /Users/rch/opt/anaconda3/etc/conda
  conda av metadata url : None
          channel URLs : https://repo.anaconda.com/pkgs/main/osx-64
                         https://repo.anaconda.com/pkgs/main/noarch
                         https://repo.anaconda.com/pkgs/r/osx-64
                         https://repo.anaconda.com/pkgs/r/noarch
         package cache : /Users/rch/opt/anaconda3/pkgs
                         /Users/rch/.conda/pkgs
       envs directories : /Users/rch/opt/anaconda3/envs
                         /Users/rch/.conda/envs
               platform : osx-64
             user-agent : conda/23.7.3 requests/2.31.0 CPython/3.9.7
Darwin/21.5.0 OSX/10.16
                UID:GID : 501:20
             netrc file : None
           offline mode : False
```

IPython/Jupyter setup. Example notebook:



2. **Results from "Linear Algebra Equivalents" attached at end of document.**
3. **Figure from provided Matplotlib code attached at end of document.**
4. **Matplotlib code and figure from code attached at end of document.**
5. **GitHub (VCS) account/profile: https://github.com/rcheeter**
6. **Test project in GitHub using PyCharm: https://github.com/rcheeter/ELEC576**

ELEC576-HW0

September 12, 2023

1 PROBLEM SET 0

- ROBERT HEETER
- ELEC 576 Introduction to Deep Learning
- 13 September 2023

1.1 TASK 2

[1]:
```
import numpy as np
import scipy

a = np.reshape(np.arange(32), (4,8))
b = np.reshape(np.arange(32)+32, (4,8))
c = np.reshape(np.arange(32)+64, (4,8))
d = np.reshape(np.arange(32)+96, (4,8))

v = np.array([1,0.1,1.2,0.4,0.2,10.2,100,4.2])
x = np.reshape(np.arange(10), (2,5))

print('a = \n' + str(a))
print('b = \n' + str(b))
print('c = \n' + str(c))
print('d = \n' + str(d))

print('v = \n' + str(v))
print('x = \n' + str(x))
```

```
a =
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
b =
[[32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
c =
```

```
[[64 65 66 67 68 69 70 71]
 [72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87]
 [88 89 90 91 92 93 94 95]]
d =
[[ 96  97  98  99 100 101 102 103]
 [104 105 106 107 108 109 110 111]
 [112 113 114 115 116 117 118 119]
 [120 121 122 123 124 125 126 127]]
v =
[  1.    0.1   1.2   0.4   0.2  10.2 100.    4.2]
x =
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

[2]:
```
print(np.ndim(a))
print(a.ndim)
```

```
2
2
```

[3]:
```
print(np.size(a))
print(a.size)
```

```
32
32
```

[4]:
```
print(np.shape(a))
print(a.shape)
```

```
(4, 8)
(4, 8)
```

[5]:
```
n = 1
print(a.shape[n-1])
n = 2
print(a.shape[n-1])
```

```
4
8
```

[6]:
```
print(np.array([[1., 2., 3.], [4., 5., 6.]]))
```

```
[[1. 2. 3.]
 [4. 5. 6.]]
```

[7]:
```
print(np.block([[a, b], [c, d]]))
```

```
[[  0   1   2   3   4   5   6   7  32  33  34  35  36  37  38  39]
 [  8   9  10  11  12  13  14  15  40  41  42  43  44  45  46  47]
 [ 16  17  18  19  20  21  22  23  48  49  50  51  52  53  54  55]
 [ 24  25  26  27  28  29  30  31  56  57  58  59  60  61  62  63]
 [ 64  65  66  67  68  69  70  71  96  97  98  99 100 101 102 103]
 [ 72  73  74  75  76  77  78  79 104 105 106 107 108 109 110 111]
 [ 80  81  82  83  84  85  86  87 112 113 114 115 116 117 118 119]
 [ 88  89  90  91  92  93  94  95 120 121 122 123 124 125 126 127]]
```

[8]:
```
print(a[-1])
```

```
[24 25 26 27 28 29 30 31]
```

[9]:
```
print(a[1, 4])
```

```
12
```

[10]:
```
print(a[1])
print(a[1,:])
```

```
[ 8  9 10 11 12 13 14 15]
[ 8  9 10 11 12 13 14 15]
```

[11]:
```
print(a[0:5])
print(a[:5])
print(a[0:5, :])
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
```

[12]:
```
print(a[-5:])
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
```

[13]:
```
print(a[0:3, 4:9])
```

```
[[ 4  5  6  7]
 [12 13 14 15]
 [20 21 22 23]]
```

[14]:
```
a = a.reshape((8,4))
print(a)
print(a[np.ix_([1, 3, 4], [0, 2])])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]
 [24 25 26 27]
 [28 29 30 31]]
[[ 4  6]
 [12 14]
 [16 18]]
```

[15]:
```
print(a[2:21:2,:])
```

```
[[ 8  9 10 11]
 [16 17 18 19]
 [24 25 26 27]]
```

[16]:
```
print(a[::2, :])
```

```
[[ 0  1  2  3]
 [ 8  9 10 11]
 [16 17 18 19]
 [24 25 26 27]]
```

[17]:
```
print(a[::-1,:])
```

```
[[28 29 30 31]
 [24 25 26 27]
 [20 21 22 23]
 [16 17 18 19]
 [12 13 14 15]
 [ 8  9 10 11]
 [ 4  5  6  7]
 [ 0  1  2  3]]
```

[18]:
```
print(a[np.r_[:len(a),0]])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
```

```
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]
 [24 25 26 27]
 [28 29 30 31]
 [ 0  1  2  3]]
```

[19]:
```
print(a.transpose())
print(a.T)
```

```
[[ 0  4  8 12 16 20 24 28]
 [ 1  5  9 13 17 21 25 29]
 [ 2  6 10 14 18 22 26 30]
 [ 3  7 11 15 19 23 27 31]]
[[ 0  4  8 12 16 20 24 28]
 [ 1  5  9 13 17 21 25 29]
 [ 2  6 10 14 18 22 26 30]
 [ 3  7 11 15 19 23 27 31]]
```

[20]:
```
print(a.conj().transpose())
print(a.conj().T)
```

```
[[ 0  4  8 12 16 20 24 28]
 [ 1  5  9 13 17 21 25 29]
 [ 2  6 10 14 18 22 26 30]
 [ 3  7 11 15 19 23 27 31]]
[[ 0  4  8 12 16 20 24 28]
 [ 1  5  9 13 17 21 25 29]
 [ 2  6 10 14 18 22 26 30]
 [ 3  7 11 15 19 23 27 31]]
```

[21]:
```
print(a @ b)
```

```
[[ 304  310  316  322  328  334  340  346]
 [1008 1030 1052 1074 1096 1118 1140 1162]
 [1712 1750 1788 1826 1864 1902 1940 1978]
 [2416 2470 2524 2578 2632 2686 2740 2794]
 [3120 3190 3260 3330 3400 3470 3540 3610]
 [3824 3910 3996 4082 4168 4254 4340 4426]
 [4528 4630 4732 4834 4936 5038 5140 5242]
 [5232 5350 5468 5586 5704 5822 5940 6058]]
```

[22]:
```
a = a.reshape((4,8))
print(a)
print(a * b)
```

```
[[ 0  1  2  3  4  5  6  7]
```

```
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
[[   0   33   68  105  144  185  228  273]
 [ 320  369  420  473  528  585  644  705]
 [ 768  833  900  969 1040 1113 1188 1265]
 [1344 1425 1508 1593 1680 1769 1860 1953]]
```

[23]:
```
print(a/b)
```

```
[[0.         0.03030303 0.05882353 0.08571429 0.11111111 0.13513514
  0.15789474 0.17948718]
 [0.2        0.2195122  0.23809524 0.25581395 0.27272727 0.28888889
  0.30434783 0.31914894]
 [0.33333333 0.34693878 0.36       0.37254902 0.38461538 0.39622642
  0.40740741 0.41818182]
 [0.42857143 0.43859649 0.44827586 0.45762712 0.46666667 0.47540984
  0.48387097 0.49206349]]
```

[24]:
```
print(a**3)
```

```
[[    0     1     8    27    64   125   216   343]
 [  512   729  1000  1331  1728  2197  2744  3375]
 [ 4096  4913  5832  6859  8000  9261 10648 12167]
 [13824 15625 17576 19683 21952 24389 27000 29791]]
```

[25]:
```
print((a > 0.5))
```

```
[[False  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]]
```

[26]:
```
print(np.nonzero(a > 0.5))
```

```
(array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 3, 3, 3, 3, 3, 3, 3, 3]), array([1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4,
5, 6, 7, 0, 1, 2, 3, 4, 5, 6,
       7, 0, 1, 2, 3, 4, 5, 6, 7]))
```

[27]:
```
print(a[:,np.nonzero(v > 0.5)[0]])
```

```
[[ 0  2  5  6  7]
 [ 8 10 13 14 15]
 [16 18 21 22 23]
 [24 26 29 30 31]]
```

[28]:
```
print(a[:, v.T > 0.5])
```

```
[[ 0  2  5  6  7]
 [ 8 10 13 14 15]
 [16 18 21 22 23]
 [24 26 29 30 31]]
```

[29]:
```
a = a-10
print(a)
a[a < 0.5]=0
print(a)
```

```
[[-10  -9  -8  -7  -6  -5  -4  -3]
 [ -2  -1   0   1   2   3   4   5]
 [  6   7   8   9  10  11  12  13]
 [ 14  15  16  17  18  19  20  21]]
[[ 0  0  0  0  0  0  0  0]
 [ 0  0  0  1  2  3  4  5]
 [ 6  7  8  9 10 11 12 13]
 [14 15 16 17 18 19 20 21]]
```

[30]:
```
print(a * (a > 0.5))
```

```
[[ 0  0  0  0  0  0  0  0]
 [ 0  0  0  1  2  3  4  5]
 [ 6  7  8  9 10 11 12 13]
 [14 15 16 17 18 19 20 21]]
```

[31]:
```
a[:] = 3
print(a)
```

```
[[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]]
```

[32]:
```
y = x.copy()
print(y)
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

[33]:
```
y = x[1, :].copy()
print(y)
```

```
[5 6 7 8 9]
```

[34]:
```
y = x.flatten()
print(y)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

[35]:
```
print(np.arange(1., 11.))
print(np.r_[1.:11.])
print(np.r_[1:10:10j])
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

[36]:
```
print(np.arange(10.))
print(np.r_[:10.])
print(np.r_[:9:10j])
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

[37]:
```
print(np.arange(1.,11.)[:, np.newaxis])
```

```
[[ 1.]
 [ 2.]
 [ 3.]
 [ 4.]
 [ 5.]
 [ 6.]
 [ 7.]
 [ 8.]
 [ 9.]
 [10.]]
```

[38]:
```
print(np.zeros((3, 4)))
```

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

[39]:
```
print(np.zeros((3, 4, 5)))
```

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

[40]:
```
print(np.ones((3, 4)))
```

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

[41]:
```
print(np.eye(3))
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[42]:
```
print(np.diag(a))
```

```
[3 3 3 3]
```

[43]:
```
print(np.diag(v, 0))
```

```
[[  1.    0.    0.    0.    0.    0.    0.    0. ]
 [  0.    0.1   0.    0.    0.    0.    0.    0. ]
 [  0.    0.    1.2   0.    0.    0.    0.    0. ]
 [  0.    0.    0.    0.4   0.    0.    0.    0. ]
 [  0.    0.    0.    0.    0.2   0.    0.    0. ]
 [  0.    0.    0.    0.    0.   10.2   0.    0. ]
 [  0.    0.    0.    0.    0.    0.  100.    0. ]
 [  0.    0.    0.    0.    0.    0.    0.    4.2]]
```

[44]:
```
from numpy.random import default_rng
rng = default_rng(42)
print(rng.random((3,4)))
```

```
[[0.77395605 0.43887844 0.85859792 0.69736803]
 [0.09417735 0.97562235 0.7611397  0.78606431]
 [0.12811363 0.45038594 0.37079802 0.92676499]]
```

[45]:
```
print(np.linspace(1,3,4))
```

```
[1.         1.66666667 2.33333333 3.        ]
```

[46]:
```
print(np.mgrid[0:9.,0:6.])
```

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

[47]:
```
print(np.meshgrid(np.r_[0:9.],np.r_[0:6.]))
```

```
[array([[0., 1., 2., 3., 4., 5., 6., 7., 8.],
       [0., 1., 2., 3., 4., 5., 6., 7., 8.],
       [0., 1., 2., 3., 4., 5., 6., 7., 8.],
       [0., 1., 2., 3., 4., 5., 6., 7., 8.],
       [0., 1., 2., 3., 4., 5., 6., 7., 8.],
       [0., 1., 2., 3., 4., 5., 6., 7., 8.]]), array([[0., 0., 0., 0., 0., 0.,
0., 0., 0.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5., 5., 5., 5.]])]
```

[48]:
```
print(np.ogrid[0:9.,0:6.])
```

```
[array([[0.],
       [1.],
       [2.],
       [3.],
       [4.],
       [5.],
       [6.],
       [7.],
       [8.]]), array([[0., 1., 2., 3., 4., 5.]])]
```

[49]:
```
print(np.ix_(np.r_[0:9.],np.r_[0:6.]))
```

```
(array([[0.],
       [1.],
       [2.],
```

```
       [3.],
       [4.],
       [5.],
       [6.],
       [7.],
       [8.]]), array([[0., 1., 2., 3., 4., 5.]]))
```

[50]:
```
print(np.meshgrid([1,2,4],[2,4,5]))
```

```
[array([[1, 2, 4],
       [1, 2, 4],
       [1, 2, 4]]), array([[2, 2, 2],
       [4, 4, 4],
       [5, 5, 5]])]
```

[51]:
```
print(np.ix_([1,2,4],[2,4,5]))
```

```
(array([[1],
       [2],
       [4]]), array([[2, 4, 5]]))
```

[52]:
```
m = 2
n = 3
a = np.reshape(np.arange(32), (4,8))
print(a)
print(np.tile(a, (m, n)))
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
[[ 0  1  2  3  4  5  6  7  0  1  2  3  4  5  6  7  0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15  8  9 10 11 12 13 14 15  8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23 16 17 18 19 20 21 22 23 16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31 24 25 26 27 28 29 30 31 24 25 26 27 28 29 30 31]
 [ 0  1  2  3  4  5  6  7  0  1  2  3  4  5  6  7  0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15  8  9 10 11 12 13 14 15  8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23 16 17 18 19 20 21 22 23 16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31 24 25 26 27 28 29 30 31 24 25 26 27 28 29 30 31]]
```

[53]:
```
print(np.concatenate((a,b),1))
```

```
[[ 0  1  2  3  4  5  6  7 32 33 34 35 36 37 38 39]
 [ 8  9 10 11 12 13 14 15 40 41 42 43 44 45 46 47]
 [16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55]
 [24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63]]
```

[54]:

```
print(np.hstack((a,b)))
print(np.column_stack((a,b)))
print(np.c_[a,b])
```

```
[[ 0  1  2  3  4  5  6  7 32 33 34 35 36 37 38 39]
 [ 8  9 10 11 12 13 14 15 40 41 42 43 44 45 46 47]
 [16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55]
 [24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63]]
[[ 0  1  2  3  4  5  6  7 32 33 34 35 36 37 38 39]
 [ 8  9 10 11 12 13 14 15 40 41 42 43 44 45 46 47]
 [16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55]
 [24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63]]
[[ 0  1  2  3  4  5  6  7 32 33 34 35 36 37 38 39]
 [ 8  9 10 11 12 13 14 15 40 41 42 43 44 45 46 47]
 [16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55]
 [24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63]]
```

[55]:

```
print(np.concatenate((a,b)))
print(np.vstack((a,b)))
print(np.r_[a,b])
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]
 [32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]
 [32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]
 [32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
```

[56]:
```
print(a.max())
print(np.nanmax(a))
```

```
31
31
```

[57]:
```
print(a.max(0))
```

```
[24 25 26 27 28 29 30 31]
```

[58]:
```
print(a.max(1))
```

```
[ 7 15 23 31]
```

[59]:
```
print(np.maximum(a, b))
```

```
[[32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
```

[60]:
```
print(np.sqrt(v @ v))
print(np.linalg.norm(v))
```

```
100.61972967564562
100.61972967564562
```

[61]:
```
print(a)
print(b)
print(np.logical_and(a,b))
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
[[32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
[[False  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]]
```

[62]:
```
print(np.logical_or(a,b))
```

```
[[ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]
```

```
 [ True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True]]
```

[63]:
```
print(a & b)
```

```
[[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]
 [16 17 18 19 20 21 22 23]
 [24 25 26 27 28 29 30 31]]
```

[64]:
```
print(a | b)
```

```
[[32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47]
 [48 49 50 51 52 53 54 55]
 [56 57 58 59 60 61 62 63]]
```

[65]:
```
a = np.reshape(np.arange(16), (4,4))
print(a)
print(np.linalg.inv(a))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 9.00719925e+14 -4.50359963e+14 -1.80143985e+15  1.35107989e+15]
 [-2.40191980e+15  2.70215978e+15  1.80143985e+15 -2.10167983e+15]
 [ 2.10167983e+15 -4.05323966e+15  1.80143985e+15  1.50119988e+14]
 [-6.00479950e+14  1.80143985e+15 -1.80143985e+15  6.00479950e+14]]
```

[66]:
```
print(np.linalg.pinv(a))
```

```
[[-2.62500000e-01 -1.37500000e-01 -1.25000000e-02  1.12500000e-01]
 [-1.00000000e-01 -5.00000000e-02  7.80625564e-18  5.00000000e-02]
 [ 6.25000000e-02  3.75000000e-02  1.25000000e-02 -1.25000000e-02]
 [ 2.25000000e-01  1.25000000e-01  2.50000000e-02 -7.50000000e-02]]
```

[67]:
```
print(np.linalg.matrix_rank(a))
```

```
2
```

[68]:
```
a = np.reshape(np.arange(16), (4,4))
print(a)
print(np.linalg.solve(a, b))
# print(np.linalg.lstsq(a, b))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
```

```
 [12 13 14 15]]
[[ -1.2  -1.2  -2.8  -4.4  -1.2  -2.8  -4.4  -1.2]
 [-12.8 -18.8 -15.2 -11.6 -20.8 -17.2 -13.6 -22.8]
 [  3.2  14.2  10.8   7.4  15.2  11.8   8.4  16.2]
 [ 12.8   7.8   9.2  10.6   8.8  10.2  11.6   9.8]]
```

[69]:
```
# a.T x.T = b.T
```

[70]:
```
U, S, Vh = np.linalg.svd(a)
V = Vh.T
print(U)
print(S)
print(Vh)
print(V)
```

```
[[-0.09184212 -0.83160389  0.52939495  0.14050262]
 [-0.31812733 -0.44586433 -0.8105844   0.20725087]
 [-0.54441254 -0.06012478  0.03298396 -0.8360096 ]
 [-0.77069775  0.32561478  0.2482055   0.48825611]]
[3.51399637e+01 2.27661021e+00 8.80118491e-16 4.41188001e-17]
[[-0.42334086 -0.47243254 -0.52152422 -0.57061589]
 [ 0.72165263  0.27714165 -0.16736932 -0.6118803 ]
 [ 0.5427818  -0.66899815 -0.29034911  0.41656546]
 [ 0.0734024  -0.50243554  0.78466387 -0.35563073]]
[[-0.42334086  0.72165263  0.5427818   0.0734024 ]
 [-0.47243254  0.27714165 -0.66899815 -0.50243554]
 [-0.52152422 -0.16736932 -0.29034911  0.78466387]
 [-0.57061589 -0.6118803   0.41656546 -0.35563073]]
```

[71]:
```
a = np.reshape(np.array([0.2,10,24,21,0.3,14,9,2,2,1,42,49,1,24,1,243]),(4,4))
print(a)
print(np.linalg.cholesky(a))
```

```
[[2.00e-01 1.00e+01 2.40e+01 2.10e+01]
 [3.00e-01 1.40e+01 9.00e+00 2.00e+00]
 [2.00e+00 1.00e+00 4.20e+01 4.90e+01]
 [1.00e+00 2.40e+01 1.00e+00 2.43e+02]]
[[ 0.4472136   0.          0.          0.        ]
 [ 0.67082039  3.68103246  0.          0.        ]
 [ 4.47213595 -0.54332582  4.65884074  0.        ]
 [ 2.23606798  6.11241553 -1.21896564 14.11214013]]
```

[72]:
```
D,V = np.linalg.eig(a)
print(D)
print(V)
```

```
[243.79673241  41.27264443  -0.83800518  14.96862834]
[[-0.10720169 -0.51921882 -0.99910619 -0.56828719]
```

```
 [-0.01777443 -0.27013897 -0.0056662  -0.81328578]
 [-0.23565103 -0.80989932  0.0416487  -0.0882487 ]
 [-0.96574347  0.03872778  0.00448431  0.0884764 ]]
```

[73]:
```
# D,V = np.linalg.eig(a, b)
```

[74]:
```
D,V = scipy.sparse.linalg.eigs(a, k=3)
print(D)
print(V)
```

```
[243.79673241+0.j  41.27264443+0.j  -0.83800518+0.j  14.96862834+0.j]
[[-0.10720169 -0.51921882 -0.99910619 -0.56828719]
 [-0.01777443 -0.27013897 -0.0056662  -0.81328578]
 [-0.23565103 -0.80989932  0.0416487  -0.0882487 ]
 [-0.96574347  0.03872778  0.00448431  0.0884764 ]]
```

```
/Users/rch/opt/anaconda3/lib/python3.9/site-
packages/scipy/sparse/linalg/_eigen/arpack/arpack.py:1272: RuntimeWarning: k >=
N - 1 for N * N square matrix. Attempting to use scipy.linalg.eig instead.
  warnings.warn("k >= N - 1 for N * N square matrix. "
```

[75]:
```
Q,R = np.linalg.qr(a)
print(Q)
print(R)
```

```
[[-0.08830216 -0.3376115   0.83152562  0.43218786]
 [-0.13245324 -0.46780943  0.23364618 -0.84203322]
 [-0.88302157  0.4460596   0.12580271 -0.07400955]
 [-0.44151079 -0.68425408 -0.48800439  0.3141915 ]]
[[  -2.26495033  -14.2166473   -40.83974768 -152.67442969]
 [   0.          -25.90148528    5.7372883  -152.4422804 ]
 [   0.            0.           26.85513978  -94.49140439]
 [   0.            0.            0.           80.11394486]]
```

[76]:
```
P,L,U = scipy.linalg.lu(a)
print(P)
print(L)
print(U)
```

```
[[0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]]
[[1.        0.        0.        0.       ]
 [0.5       1.        0.        0.       ]
 [0.1       0.4212766 1.        0.       ]
 [0.15      0.5893617 0.513267  1.       ]]
[[  2.          1.         42.          49.        ]
 [  0.         23.5       -20.         218.5       ]
```

```
 [  0.           0.          28.22553191 -75.94893617]
 [  0.           0.           0.         -95.14344942]]
```

[77]:
```
# cg
```

[78]:
```
a = np.reshape(np.arange(32), (4,8))
print(np.fft.fft(a))
```

```
[[ 28.+0.j          -4.+9.65685425j  -4.+4.j          -4.+1.65685425j
   -4.+0.j          -4.-1.65685425j  -4.-4.j          -4.-9.65685425j]
 [ 92.+0.j          -4.+9.65685425j  -4.+4.j          -4.+1.65685425j
   -4.+0.j          -4.-1.65685425j  -4.-4.j          -4.-9.65685425j]
 [156.+0.j          -4.+9.65685425j  -4.+4.j          -4.+1.65685425j
   -4.+0.j          -4.-1.65685425j  -4.-4.j          -4.-9.65685425j]
 [220.+0.j          -4.+9.65685425j  -4.+4.j          -4.+1.65685425j
   -4.+0.j          -4.-1.65685425j  -4.-4.j          -4.-9.65685425j]]
```

[79]:
```
print(np.fft.ifft(a))
```

```
[[ 3.5+0.j         -0.5-1.20710678j -0.5-0.5j        -0.5-0.20710678j
  -0.5+0.j         -0.5+0.20710678j -0.5+0.5j        -0.5+1.20710678j]
 [11.5+0.j         -0.5-1.20710678j -0.5-0.5j        -0.5-0.20710678j
  -0.5+0.j         -0.5+0.20710678j -0.5+0.5j        -0.5+1.20710678j]
 [19.5+0.j         -0.5-1.20710678j -0.5-0.5j        -0.5-0.20710678j
  -0.5+0.j         -0.5+0.20710678j -0.5+0.5j        -0.5+1.20710678j]
 [27.5+0.j         -0.5-1.20710678j -0.5-0.5j        -0.5-0.20710678j
  -0.5+0.j         -0.5+0.20710678j -0.5+0.5j        -0.5+1.20710678j]]
```

[80]:
```
a = np.reshape(np.array([0.2,10,24,21,0.3,14,9,2,2,1,42,49,1,24,1,243]),(4,4))
print(np.sort(a))
a.sort(axis=0)
print(a)
```

```
[[2.00e-01 1.00e+01 2.10e+01 2.40e+01]
 [3.00e-01 2.00e+00 9.00e+00 1.40e+01]
 [1.00e+00 2.00e+00 4.20e+01 4.90e+01]
 [1.00e+00 1.00e+00 2.40e+01 2.43e+02]]
[[2.00e-01 1.00e+00 1.00e+00 2.00e+00]
 [3.00e-01 1.00e+01 9.00e+00 2.10e+01]
 [1.00e+00 1.40e+01 2.40e+01 4.90e+01]
 [2.00e+00 2.40e+01 4.20e+01 2.43e+02]]
```

[81]:
```
a = np.reshape(np.array([0.2,10,24,21,0.3,14,9,2,2,1,42,49,1,24,1,243]),(4,4))
print(np.sort(a, axis=1))
a.sort(axis=1)
print(a)
```

```
[[2.00e-01 1.00e+01 2.10e+01 2.40e+01]
```

```
 [3.00e-01 2.00e+00 9.00e+00 1.40e+01]
 [1.00e+00 2.00e+00 4.20e+01 4.90e+01]
 [1.00e+00 1.00e+00 2.40e+01 2.43e+02]]
[[2.00e-01 1.00e+01 2.10e+01 2.40e+01]
 [3.00e-01 2.00e+00 9.00e+00 1.40e+01]
 [1.00e+00 2.00e+00 4.20e+01 4.90e+01]
 [1.00e+00 1.00e+00 2.40e+01 2.43e+02]]
```

[82]:
```
I = np.argsort(a[:, 0]); b = a[I,:]
print(I)
```

```
[0 1 2 3]
```

[83]:
```
y = np.arange(5)
Z = np.reshape(np.arange(25), (5,5))

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

```
(array([ 1.20000000e-01,  8.00000000e-02,  4.00000000e-02,  1.18886171e-16,
       -4.00000000e-02]), array([], dtype=float64), 2, array([6.99085940e+01,
3.57609824e+00, 5.72246903e-15, 2.09124342e-16,
       6.08024818e-17]))

/var/folders/xz/pb7f4dg10fj3tpm9jkly5h600000gn/T/ipykernel_11490/122479090.py:4:
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)
```

[84]:
```
# scipy.signal.resample(x, np.ceil(len(x)/q))
```

[85]:
```
a = np.reshape(np.arange(25), (5,5))
print(np.unique(a))
```

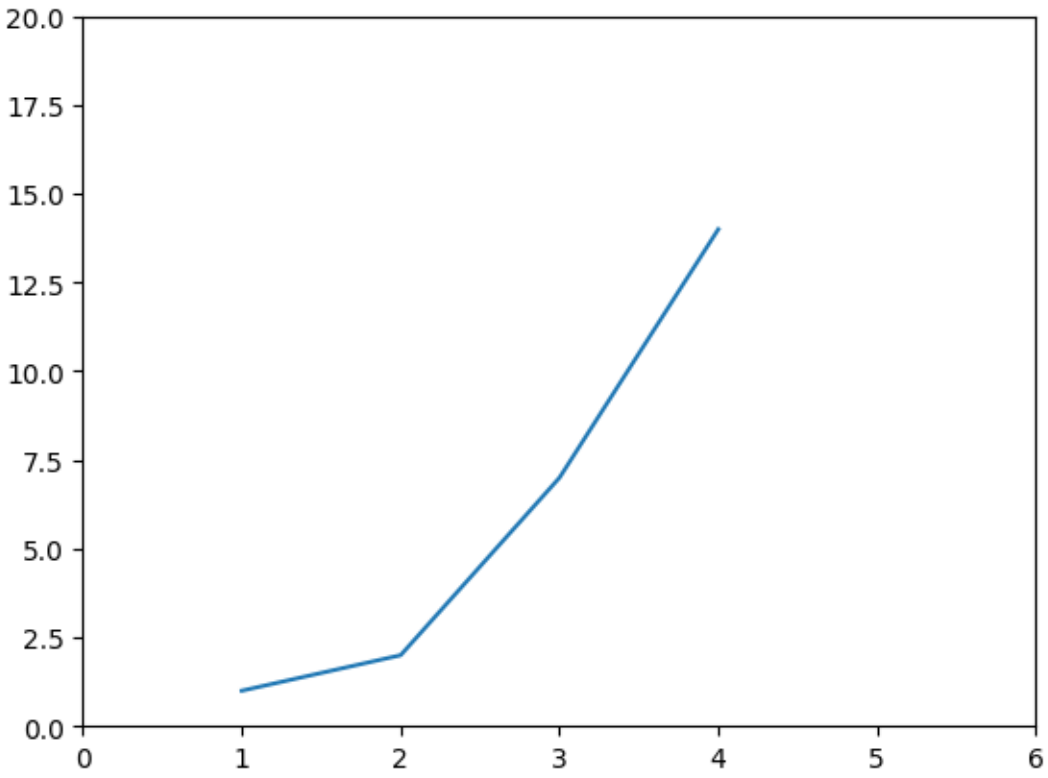
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
```

[86]:
```
print(a.squeeze())
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

1.2 TASK 3

[87]:
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,2,7,14])
plt.axis([0, 6, 0, 20])
plt.show()
```



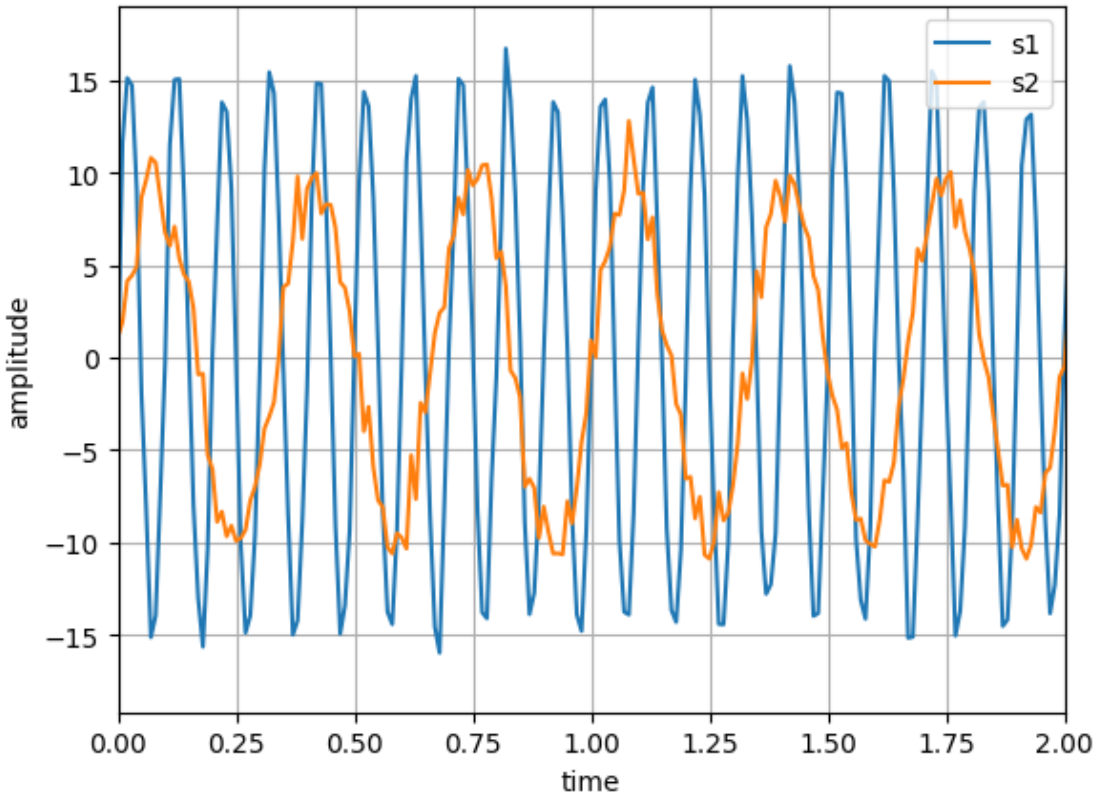
1.3 TASK 4

[88]:
```
import numpy as np
import matplotlib.pyplot as plt

# make sinusoid signals with noise
dt = 0.01
t = np.arange(0, 100, dt)
nse1 = np.random.randn(len(t)) # white noise
nse2 = np.random.randn(len(t))
s1 = 15*np.sin(1 * np.pi * 20 * t) + nse1
s2 = 10*np.sin(0.3 * np.pi * 20 * t) + nse2

# plot signals with noise
plt.plot(t, s1, t, s2)
```

```
plt.xlim(0, 2)
plt.xlabel('time')
plt.ylabel('amplitude')
plt.legend(['s1', 's2'])
plt.grid(True)
plt.show()
```



[ ]: