﻿"""

Created on Mon Feb 4 13:50:49 2019

@author: roryeggleston

"""

import numpy as np

﻿a = np.array([1, 2, 3, 4])

a + 1

2 \*\* a

Out[2]: array([ 2, 4, 8, 16])

import matplotlib.pyplot as plt

a = np.arange(10)

a

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

plt.plot(a, 3\*a + 5)

plt.xlabel('a')

plt.ylabel('3a+5')

plt.title('my plot')

plt.show()



﻿b = np.zeros((5,6))

b

Out[6]:

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.]])

b[0,0] = 10

b[0,1] = 13

b

Out[7]:

array([[10., 13., 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.]])

﻿plt.pcolormesh(b)

plt.colorbar()

Out[8]: <matplotlib.colorbar.Colorbar at 0x11d046f28>



﻿b[1 , :] = 5

b

Out[9]:

array([[10., 13., 0., 0., 0., 0.],

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

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

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

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

plt.pcolormesh(b, cmap = 'inferno')

plt.colorbar()

Out[10]: <matplotlib.colorbar.Colorbar at 0x11d149d68>



﻿c = b[0:2,:]

c

Out[11]:

array([[10., 13., 0., 0., 0., 0.],

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

plt.pcolormesh(c)

plt.colorbar()

Out[12]: <matplotlib.colorbar.Colorbar at 0x11d1fdb38>



﻿d = np.random.randn(1000)

d

Out[13]:

array([ 2.85136261e-01, -1.08298797e+00, -1.23892688e+00, -1.36163829e-01,

1.14674718e+00, -8.17053286e-01, 5.27758684e-01, -1.41213699e+00,

5.28295821e-01, -6.85940629e-01, 9.88854748e-01, -3.63424005e-01,

4.85758934e-01, 3.70302146e-01, 1.58550187e-01, 3.44380928e-01,

-5.25617953e-01, -1.70980395e+00, 5.53082548e-01, 3.17883117e-02,

2.84770843e-02, -8.53965265e-01, 1.48495062e+00, -9.69554632e-01,

-4.15021660e-01, -1.46954304e+00, 1.86226886e+00, 1.78839192e+00,

-1.94161918e-01, -2.08860451e-01, -2.51626374e-01, 8.67058812e-01,

-1.00051763e+00, -9.82261281e-01, 1.65074046e+00, -9.91044930e-03,

-1.76870507e+00, 1.30660336e+00, 9.03939431e-02, -5.30651741e-01,

2.28484966e+00, -1.06028294e+00, -3.14158314e-01, -1.42370977e+00,

-5.11110883e-01, 3.73581915e-01, -3.17845243e-02, -6.93388503e-01,

-5.41114902e-01, 2.21017731e-01, -2.39545737e-01, 2.26067790e+00,

-1.09599140e-01, 2.50954612e-01, -1.29832130e+00, 8.78520584e-01,

-3.11079373e-01, -2.67187711e-01, 5.40695866e-01, 1.10268749e+00,

2.93453987e-02, -1.07541492e+00, 2.31091441e-01, 7.57535458e-01,

-1.36943406e+00, -1.53722475e+00, 2.72869204e-01, 2.52512594e+00,

1.08627163e+00, -5.85430444e-01, 5.07451806e-01, 2.17591048e+00,

1.51291111e+00, -1.15648681e+00, 3.29852725e-01, -4.03152387e-01,

-1.21330217e+00, 3.11100212e-01, 2.07867953e-01, 2.52843008e-01,

-1.19156445e+00, -2.26203524e-01, 4.90386328e-01, -8.10885459e-01,

-2.79768911e-01, 8.05521687e-01, 3.21997409e-01, 1.83112014e-01,

3.24091252e-01, 4.06064354e-01, 2.61892940e-01, 9.40095220e-01,

-1.44294211e+00, -2.07531953e+00, 5.85379075e-01, -1.33133393e+00,

7.46946343e-01, 1.54412677e-01, -8.63685421e-01, 2.99452702e-01,

-1.35841525e-01, 2.03895450e-01, -2.59608607e-01, -1.54522952e+00,

-9.06325366e-01, 3.02663157e-01, 1.85942233e-01, 2.13371769e-02,

-2.29442113e+00, -1.94403973e-01, 3.24814280e-01, 4.04174585e-01,

-9.93875093e-01, -2.27799530e+00, 8.68851888e-01, -1.45841607e+00,

1.39692739e+00, 2.94097776e-01, -4.52009156e-01, 4.97582639e-01,

-2.06806711e-01, -9.17708232e-01, 6.60267482e-01, -7.30037745e-01,

2.67252783e-02, -3.76665798e-01, 6.80987615e-01, -3.61576236e-01,

-8.59030274e-01, 5.81326587e-01, 7.14528279e-01, 6.63069656e-01,

9.14866751e-01, 5.39687339e-01, 4.94542918e-02, 1.08091831e-01,

3.48784629e-01, 8.33403569e-02, 1.75930399e+00, -2.23819020e-01,

-1.11285136e+00, 1.35256433e-01, 8.72960799e-01, -2.45025665e+00,

-1.28776128e+00, 1.39331985e+00, -2.37243408e-01, 8.01800638e-02,

-2.20907300e-01, -2.02607634e+00, 8.69785762e-01, 4.74391476e-01,

-9.62186115e-01, -1.83168683e+00, -4.37324961e-01, 1.07360224e+00,

7.67294280e-01, -3.68995842e-01, -5.04604810e-01, 1.07267036e+00,

-8.97754044e-01, -3.53352919e-01, -2.05355017e+00, -4.95123261e-01,

3.01064424e-01, -2.49448542e+00, 1.32405157e+00, 7.14642649e-01,

5.93339891e-01, -4.80596351e-01, -1.99521411e-01, 9.75546745e-01,

-7.16088925e-01, 2.98454208e-02, 8.74825688e-01, 1.38544062e+00,

6.42512984e-01, 7.84360152e-01, 2.85577166e-01, -1.52693884e+00,

8.54779172e-01, 1.01603746e+00, 1.75122389e-01, -1.77712136e+00,

-2.42500796e-01, 2.46482819e+00, -1.12678042e+00, -8.26181947e-01,

-9.19504355e-01, -9.42505432e-01, 4.12668678e-02, -5.26041303e-01,

-9.99106357e-02, 1.30209097e+00, -2.03431219e+00, 2.29351292e+00,

-8.71409290e-01, -9.29728981e-01, -4.12074397e-01, 1.38834171e-01,

4.74348339e-01, 5.85221690e-01, 5.54540602e-01, 4.70789781e-01,

3.67017981e-02, 1.00301158e+00, 1.08743237e+00, -7.20802069e-01,

9.24701155e-01, -2.03175469e-01, -4.77661577e-01, -1.15018412e+00,

-1.99548804e+00, 5.65336783e-01, 1.92582923e-01, 1.22855649e-02,

-1.69775033e-01, 1.10714789e+00, 3.23432734e-01, 1.66373683e+00,

-7.77476835e-01, 1.95965928e+00, -8.35974923e-01, -7.84796272e-01,

-1.32185308e+00, 1.23530109e+00, -1.36302494e+00, 2.44061740e-01,

7.39612548e-01, 2.50180677e-01, -8.08861950e-01, -3.16991247e-01,

-7.01322568e-01, -3.42230036e-01, 2.74132693e-01, 1.78585195e+00,

-2.30788959e-02, 5.79705559e-01, -8.06940592e-01, -1.31562511e-01,

2.56045214e-02, -3.66312359e-01, -4.97331387e-01, -4.05058063e-01,

6.29750264e-01, -1.93193495e+00, -1.27454128e+00, -1.26789006e-01,

-2.78587299e-01, 4.20011875e-01, -6.43812747e-02, -5.77766019e-01,

-4.61515085e-01, -9.14729760e-01, -1.66923233e+00, -2.18046510e+00,

-5.42252645e-01, 8.34997524e-01, 9.93854297e-01, -1.48927700e+00,

4.33980220e-01, 2.66282270e-01, 1.25820820e+00, 3.97586791e-01,

-2.24153481e-01, -5.06014292e-01, -1.00917762e+00, 3.04510570e-01,

1.87991927e+00, 3.61234999e-01, 8.57955069e-01, -5.04213092e-01,

-4.52917722e-01, -7.71651503e-01, 4.81775968e-01, -1.36292968e+00,

-4.62973891e-01, 1.39227936e+00, -8.82431279e-02, 6.83853386e-01,

6.51461910e-01, 4.57350750e-01, 3.39349850e-01, 2.86472069e-01,

-1.76228042e-02, -1.64960364e+00, -6.80914314e-01, -7.74988800e-01,

-1.62456385e+00, -8.46993546e-01, 5.45185381e-02, -3.97763589e-01,

1.25548297e+00, -1.09332990e+00, 9.09436875e-01, -1.76843628e+00,

-1.36754339e+00, -2.14603397e-01, -4.91776942e-01, 4.40991370e-01,

-5.31813382e-01, 6.87147355e-01, 8.76298340e-01, -2.26464915e+00,

-1.05634509e-01, -2.75583006e+00, 2.19231558e+00, 6.98369336e-01,

-4.30710384e-01, -1.50674578e+00, 2.67181452e-01, 2.76652141e-01,

-1.94456959e+00, 1.42881933e+00, 1.26668108e+00, -5.53410150e-01,

-2.21832989e+00, -5.12235338e-01, 9.29624352e-01, -2.29587870e-02,

-3.16000320e-01, 4.43044773e-02, 7.42357085e-01, -4.62497898e-01,

2.28153985e+00, 1.43294319e-01, 1.85386174e+00, -3.87161806e-01,

3.93158887e-01, -5.07323414e-01, 1.62002735e+00, -1.14419325e+00,

-3.41625342e-01, -2.35961891e-01, 1.06283263e+00, 1.20319314e-02,

-1.35202767e+00, 5.03662662e-01, -3.10987038e+00, 2.72819509e-01,

2.66018253e-02, 3.29203809e-01, -5.47477035e-01, 3.29931745e+00,

-9.37639287e-01, 8.28845797e-01, -7.11419705e-01, 2.71360467e-01,

-1.52709604e+00, 9.01445612e-01, -1.85949653e+00, -4.98283012e-01,

6.20090728e-01, -6.00700400e-01, -4.42709205e-02, 3.73799770e-01,

-1.65468735e+00, 1.72354974e+00, -3.16980474e-01, 3.28700372e-01,

1.02385150e+00, 1.97543754e-01, 2.32660042e+00, 9.73659218e-01,

9.59551347e-01, 8.11726167e-02, -1.24110392e+00, -9.85873239e-01,

-2.00332222e-01, 5.47205124e-01, 1.25173139e+00, 1.04587613e+00,

4.12083542e-02, 2.37537832e+00, -1.09969180e-01, -1.64507397e-01,

5.76845771e-01, -4.06059195e-01, 8.69117752e-01, 6.43914516e-01,

1.74236120e-01, 1.08015698e+00, 1.97602199e+00, 7.14714826e-01,

1.68781178e-02, -1.31492836e+00, 1.33533453e+00, -7.58352791e-01,

2.96110856e-01, -1.13006235e+00, -1.10239042e+00, 5.57232090e-01,

1.44694811e+00, -6.63297543e-01, 1.19732919e+00, 1.20801578e+00,

1.03565365e+00, -1.91313080e+00, 8.64674093e-01, 9.36631882e-01,

1.45870565e+00, 2.64755087e-01, -2.33666098e-01, 2.33287534e-01,

1.08535216e+00, 1.39273001e+00, 3.98625290e-01, -6.27718932e-01,

1.74612184e-01, -4.39065158e-01, -1.31607678e+00, -1.23801468e+00,

-1.90633527e+00, 1.23129968e+00, 1.67403202e-01, 6.91596552e-01,

2.99157900e-01, 1.37971428e+00, -7.47660888e-01, -8.65844814e-01,

2.16044680e+00, -3.97680255e-01, -5.25652811e-01, -1.22261216e+00,

4.25603677e-02, 2.26046564e-01, 1.16892108e+00, 5.74304556e-01,

-1.82319368e-01, -9.46762100e-01, 5.11934793e-02, -2.43357437e-01,

-1.18523930e-02, -2.43678629e-01, 1.27682135e+00, 2.45729187e+00,

-1.02592739e-01, -7.80556499e-01, 1.56480703e+00, -2.52238044e-01,

2.70976414e+00, 3.51646480e-01, -3.39408014e-01, 1.46818529e+00,

-1.11600293e+00, -3.81147728e-01, -1.98593623e+00, 1.55424472e-01,

7.45272930e-02, -7.27851905e-01, -1.48600948e-01, -1.07757821e-01,

1.73750837e+00, -1.12949477e+00, 1.91129860e+00, 1.07302358e+00,

-1.31540419e+00, -6.28107671e-01, 9.57458440e-01, 4.47925588e-01,

6.19638815e-01, -5.79383069e-01, -6.90825659e-01, -1.05323037e+00,

1.61727768e+00, 3.95199224e-01, -1.17029315e+00, 1.91657930e+00,

6.16586972e-01, -6.46313520e-01, -6.01617138e-01, -1.36377173e-01,

3.91478675e-01, 1.38409093e+00, 2.22770772e+00, 2.33091809e+00,

-8.86534185e-02, 1.65040422e-01, -9.29070075e-01, 1.26228322e+00,

1.02978138e+00, -7.46275700e-03, -1.26346140e-01, -4.84671725e-01,

-6.41588735e-01, 1.38644033e+00, -3.84310281e-01, -8.67759966e-01,

-1.32070248e+00, -1.17029381e-01, 4.18944666e-02, 8.28423658e-01,

-2.50830491e-01, 1.73785977e-01, -1.09871272e-01, 1.08400481e+00,

5.43360163e-01, -6.27298682e-02, -2.21328524e+00, -3.87546503e-01,

1.61899976e+00, -5.19864309e-02, -4.91203641e-01, 1.36725399e-01,

-1.64919763e-01, -1.48046042e+00, -1.18593376e+00, -1.50206453e+00,

-1.18118753e+00, 2.39254173e-01, -1.00248417e+00, -8.23811665e-01,

1.77773259e-01, -6.91017891e-01, -1.78855168e+00, 8.26914920e-02,

-3.86626134e-01, -1.62689501e+00, -7.50375639e-02, -2.07092729e-01,

1.84525865e+00, 1.43235896e+00, -2.34069964e-01, -1.05409157e+00,

-3.09124200e-01, 7.37912463e-01, 7.29477962e-01, 1.23208611e+00,

-1.01378828e+00, -1.29778348e+00, -6.99078505e-01, -9.04959065e-01,

6.47950838e-01, 4.48875945e-02, -3.56174337e-01, 1.12731540e+00,

7.89711727e-01, -2.25555301e-01, 1.57649911e+00, 3.00861622e+00,

-1.50203200e+00, -1.32037192e-01, 4.14620071e-01, -1.33315880e+00,

-6.40637012e-01, -5.87491690e-01, -1.67228723e-01, 8.69790734e-01,

1.36303421e+00, 1.72300410e-01, -7.26663518e-01, -7.65223944e-01,

1.82635730e-01, -1.01870832e+00, -1.86114932e-01, 1.11941811e+00,

2.12823037e-01, 8.57154059e-01, -1.41612311e+00, 1.58569983e-01,

4.42060101e-01, 6.63486719e-01, -2.41469855e+00, -1.00981218e+00,

8.59265473e-01, 4.36415109e-01, -4.40228998e-01, -3.83201479e-01,

2.58400475e-01, -6.78533689e-02, -7.06102556e-01, -8.53436936e-02,

-1.89815273e+00, -6.39489576e-01, 3.31430898e-02, -7.07548011e-01,

2.82940035e-01, -7.74481345e-01, 3.17880762e-02, 2.42839916e-01,

1.37966508e+00, -1.60214510e+00, -3.28928405e-01, 1.22570453e+00,

-7.89166254e-01, -3.68021328e-01, -7.82922498e-01, -9.96614130e-01,

-2.51188215e-01, 1.63825794e+00, 1.81161125e+00, 3.19634225e-01,

5.55183598e-01, -9.66426570e-01, -5.17242869e-01, -1.07657166e-01,

1.15463856e+00, 2.98275220e-01, -1.53164099e+00, 3.69570745e-01,

-2.63810248e-01, 4.19712361e-01, 1.29215597e+00, -1.11616443e+00,

-5.92329195e-01, 4.55955630e-01, -1.52700478e+00, -1.41916517e+00,

5.77213526e-01, -1.31734563e-01, -1.78390826e+00, 3.81967852e-01,

-1.48645710e+00, -1.80699922e-01, -8.55593412e-01, 3.52129572e-01,

2.06674699e+00, -4.52551331e-01, 7.07290412e-01, 1.43702510e-01,

1.30306676e+00, 4.76972828e-02, 2.37633222e+00, 1.45550874e+00,

5.81187306e-02, 3.67002181e-01, -7.71081941e-01, -1.47964069e+00,

-9.74738326e-01, -1.00137820e+00, -1.30970657e+00, 6.67355430e-01,

1.23498674e+00, -1.27082591e+00, 1.17593780e+00, -1.35204838e+00,

1.13665705e-01, -8.31498308e-01, 1.15614129e+00, 6.90387191e-01,

1.81352891e-01, 7.06128640e-01, 5.49660858e-01, -7.69297057e-01,

2.37748492e-01, 8.17305198e-01, 9.11639895e-01, -1.59194178e+00,

1.47183206e+00, -1.37011449e+00, -3.62552929e-01, 1.38596162e+00,

7.55725401e-01, -1.61011533e+00, -1.75459250e-02, -8.15601835e-01,

7.78134568e-01, -1.47017089e+00, 8.08658480e-01, 2.10236793e-01,

-9.50015838e-01, 8.52680540e-01, -2.39263689e-01, 3.58953765e-02,

5.71199018e-01, -1.63096450e+00, -1.57050902e+00, -5.42253371e-01,

-1.13076884e-01, 2.62864033e-01, 7.09058709e-01, -4.55627744e-01,

-9.79129158e-01, -9.11460323e-01, -5.49158283e-01, 2.48478996e-01,

6.27552788e-02, 1.73905775e+00, 8.51474177e-01, 8.39128081e-02,

1.60393384e+00, -1.89639890e+00, 1.45300623e+00, 1.66434136e-01,

-5.68024094e-01, 1.80432324e+00, 3.60132064e-01, 6.34010876e-03,

-1.07036161e+00, 1.78310452e-01, 7.88015815e-02, 6.04792191e-01,

1.22248119e+00, -1.85536975e+00, 6.90964425e-01, -8.09748116e-01,

-8.85819150e-01, -4.47445147e-02, -1.52737708e+00, -7.99988553e-01,

-9.18665839e-02, 1.60114052e+00, 1.29693008e+00, 1.94201574e+00,

1.21451320e-01, 1.86264325e-01, -2.06389713e+00, 1.44435664e+00,

4.17149513e-01, 4.50942373e-01, -1.01813969e-02, -6.61462758e-01,

-1.32562055e-01, -1.35641063e+00, -3.19549092e-02, 1.94531081e+00,

-2.70160984e-01, -2.07718026e-01, -9.34892903e-01, -1.41922769e+00,

5.75291083e-01, -2.05030565e-01, 1.17564675e+00, -4.67450775e-02,

7.53189858e-01, 7.68696698e-01, -5.45927895e-01, 7.45767151e-01,

-3.34391459e-01, 3.83430960e-01, 1.10594888e+00, -1.52234651e+00,

8.55326949e-01, -9.72677768e-01, -3.23601570e-01, -1.88573958e-01,

1.25062380e+00, -5.73028820e-01, 9.77982916e-01, -1.38114216e-01,

1.14796263e+00, 9.79442268e-01, -4.39876768e-01, 1.45203208e+00,

-7.37985523e-01, 2.77095710e+00, 5.81535791e-01, -8.27406112e-01,

3.62329298e-01, 3.90448425e-01, 5.79699101e-01, -1.21267608e+00,

-1.02344786e+00, -1.47450847e+00, 6.24870076e-01, -1.16775504e+00,

-1.67113780e+00, 5.94386526e-01, -1.51705214e+00, 1.60177684e+00,

-8.98210845e-02, -3.75641759e-03, -1.15201659e+00, -2.38909268e-02,

1.06693359e+00, 1.76162505e-01, -2.08311259e-01, -5.36530754e-01,

3.93314333e-01, 1.49005142e+00, 7.70110426e-01, 7.59980906e-01,

-1.47304240e+00, 1.26010110e+00, -2.42444732e-01, 2.44739668e-01,

2.93161018e-02, 8.93309950e-01, 1.24973327e-01, -7.59028826e-01,

1.24114515e-01, -9.13565589e-01, -5.55436422e-01, -5.25247859e-01,

-3.93109406e-01, 8.71628577e-01, -7.11698287e-01, 3.38802419e-01,

1.21664173e+00, -8.31857668e-01, -1.74067821e+00, 7.75417358e-01,

-9.96945123e-01, 2.55879812e+00, -3.15994876e-01, 2.10341628e+00,

-9.09186963e-01, 1.40143019e+00, 6.43995083e-01, 1.88614315e+00,

-4.89191376e-01, -1.60095911e-01, 5.43478189e-02, 3.65251951e-01,

1.83974247e+00, 1.55605485e+00, -3.67758317e-01, 3.14873221e-01,

5.16651897e-01, 1.12243027e+00, 4.36780354e-01, -1.28856622e-03,

5.98540207e-01, -1.93891315e-01, 8.11033390e-01, -7.03801994e-01,

-1.13737885e+00, 7.87010658e-02, 2.70611887e-01, -1.24906905e+00,

1.14740243e+00, -6.74501824e-01, 2.32209651e-01, -4.03347543e-01,

4.04853736e-01, -6.66865706e-01, 9.43801036e-01, -5.06548932e-01,

1.73647004e+00, -4.60475732e-01, 9.76860753e-01, 3.50968066e-01,

1.16518452e+00, -3.13570190e-01, -1.30728242e-01, -2.26656251e-01,

5.67914252e-02, -3.48256719e-01, -9.46172787e-01, -1.80148694e-01,

-1.56551370e+00, -1.08264173e+00, -1.59176548e-01, 4.81845896e-01,

2.13301950e+00, 9.74616349e-01, -6.49531444e-01, 5.77212137e-01,

-5.64332803e-01, -1.87000007e-01, 2.44796901e+00, 2.51289695e-01,

5.50059967e-01, -1.53590749e-01, 1.44045974e+00, 3.95425005e-01,

3.18785751e-01, -2.65388176e-01, 6.92214735e-01, 3.22243414e-01,

-1.27440479e+00, 2.22881851e+00, -6.44783148e-01, 9.86691963e-01,

-6.17677809e-01, -5.50680320e-01, 1.27537078e+00, -6.40841599e-01,

-3.81774688e-01, 1.53653992e+00, -2.08330894e-01, 1.17473652e-01,

7.05245885e-01, 1.11188939e+00, -4.36126546e-01, 5.08221048e-02,

-1.23868471e+00, -1.28252389e+00, 7.52382499e-01, -4.63914777e-01,

-6.71287078e-01, -6.79565362e-01, 2.21489919e-02, 1.44439180e+00,

1.04735372e+00, -1.33724185e-01, -7.94260148e-01, 4.11961994e-01,

8.31946485e-01, 6.97512494e-01, 1.52390968e+00, -7.83498689e-01,

4.61195227e-01, 7.71076703e-01, -1.12110972e+00, 6.63687298e-01,

4.79158659e-01, -9.85988728e-02, 5.34721232e-01, 7.40288299e-01,

9.76273155e-01, 1.13450411e+00, -2.79383834e-01, 5.48754395e-01,

1.06269078e+00, -1.01408791e+00, -5.36586667e-02, 3.08796252e-01,

2.27350934e-01, -3.03494565e-01, -6.33097545e-01, 4.47490606e-01,

-7.56216014e-02, -9.22224035e-01, -6.01165827e-01, 1.20236384e+00,

2.87014937e-01, -3.58718051e-01, 6.64075431e-02, -6.55840883e-01,

-1.08467820e+00, -8.10493901e-01, -7.72549161e-01, 2.85812988e-01,

8.29315892e-01, -1.78055557e+00, -1.72635649e+00, -3.25785256e-01,

9.84127414e-01, 2.28023752e-01, -4.09328090e-01, 1.26551177e+00,

-5.52771994e-01, 2.41628702e-03, 7.29816369e-02, 1.02867056e+00,

9.66614797e-01, -1.00578045e+00, -3.68816294e-01, 1.85049424e+00,

7.04813639e-01, 1.51378221e-01, -6.45778414e-01, -2.88484343e-02,

1.66583737e+00, -4.64081095e-02, 4.74248767e-01, 8.90922486e-01,

-1.58846997e+00, -1.58226350e+00, -2.41038779e+00, -8.52455021e-01,

1.63388627e+00, 1.25794555e+00, 1.64789087e+00, -4.51110406e-01,

-3.66788059e-01, -1.54841157e+00, 1.09202643e+00, -5.25997139e-02,

4.61631400e-01, 9.60414513e-02, 2.98464460e-01, 2.75021624e-01,

2.88688572e-01, -7.83762203e-01, -1.34919963e+00, 5.08006754e-01,

5.68397698e-01, 7.19526451e-01, -4.91398999e-01, 1.42793774e+00,

2.27916534e+00, -3.63033711e-01, 2.96809936e-01, 2.70533873e-01,

1.28947448e+00, -6.47525427e-01, -7.19896580e-01, 3.19143586e-01,

4.50431429e-01, -6.64146123e-01, 9.93486566e-01, 8.50349204e-01,

-1.34020794e-02, 8.94758082e-01, 1.20500556e+00, -1.05553286e-01,

-4.10186094e-01, -5.52004121e-01, 4.05953039e-01, 1.21637121e+00,

2.63861900e-02, 5.26968602e-01, 1.30610854e+00, 4.65553795e-01,

-1.15990038e+00, 1.43151817e+00, 1.21845836e+00, -2.18516333e-02])

﻿plt.plot(d)

plt.xlabel('index')

plt.ylabel('d')

Out[14]: Text(0, 0.5, 'd')



﻿d.max()

d.min()

d.mean()

d.std()

Out[15]: 0.9905773765828989

plt.hist(d, bins = 20)

Out[16]:

(array([ 1., 2., 9., 16., 33., 51., 69., 97., 118., 135., 133.,

98., 82., 63., 45., 22., 15., 7., 2., 2.]),

array([-3.10987038, -2.78941099, -2.46895159, -2.1484922 , -1.82803281,

-1.50757342, -1.18711403, -0.86665464, -0.54619524, -0.22573585,

0.09472354, 0.41518293, 0.73564232, 1.05610171, 1.37656111,

1.6970205 , 2.01747989, 2.33793928, 2.65839867, 2.97885806,

3.29931745]),

<a list of 20 Patch objects>)



﻿a = np.arange(10)

a

Out[17]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

b = a[::2]

b

Out[18]: array([0, 2, 4, 6, 8])

np.may\_share\_memory(a, b)

Out[19]: True

b[0] = 12

b

Out[20]: array([12, 2, 4, 6, 8])

a

#a now includes the changes made to b

Out[21]: array([12, 1, 2, 3, 4, 5, 6, 7, 8, 9])

﻿a = np.arange(10)

c = a[::2].copy()

c[0] = 12

a

Out[22]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

np.may\_share\_memory(a, c)

Out[23]: False

﻿np.may\_share\_memory(a, c)

Out[23]: False

is\_prime = np.ones((100,), dtype = bool)

﻿is\_prime[:2] = 0

is\_prime

Out[25]:

array([False, 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, 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, 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, True, True,

True])

N\_max = int(np.sqrt(len(is\_prime) - 1))

for j in range(2, N\_max + 1):

is\_prime[2\*j::j] = False

N\_max

Out[26]: 9

﻿n [27]: print(is\_prime)

[False False True True False True False True False False False True

False True False False False True False True False False False True

False False False False False True False True False False False False

False True False False False True False True False False False True

False False False False False True False False False False False True

False True False False False False False True False False False True

False True False False False False False True False False False True

False False False False False True False False False False False False

False True False False]

﻿np.random.seed(3)

a = np.random.randint(0, 21, 15)

a

Out[28]: array([10, 3, 8, 0, 19, 10, 11, 9, 10, 6, 0, 20, 12, 7, 14])

(a % 3 == 0)

Out[29]:

array([False, True, False, True, False, False, False, True, False,

True, True, False, True, False, False])

mask = (a % 3 == 0)

extract\_from\_a = a[mask]

extract\_from\_a

Out[30]: array([ 3, 0, 9, 6, 0, 12])

﻿a[a % 3 == 0] = -1

a

Out[31]: array([10, -1, 8, -1, 19, 10, 11, -1, 10, -1, -1, 20, -1, 7, 14])

a = np.arange(0, 100, 10)

a

Out[32]: array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])

a[[2, 3, 2, 4, 2]]

Out[33]: array([20, 30, 20, 40, 20])

﻿n [34]: a[[9, 7]] = -100

a

Out[34]: array([ 0, 10, 20, 30, 40, 50, 60, -100, 80, -100])

a = np.arange(10)

idx = np.array([[3, 4], [9, 7]])

idx.shape

Out[35]: (2, 2)

a[idx]

Out[36]:

array([[3, 4],

[9, 7]])

﻿a = np.array([1, 2, 3, 4])

a + 1

Out[37]: array([2, 3, 4, 5])

2 \*\* a

Out[38]: array([ 2, 4, 8, 16])

﻿b = np.ones(4) + 1

a - b

Out[39]: array([-1., 0., 1., 2.])

a \* b

Out[40]: array([2., 4., 6., 8.])

j = np.arange(5)

2\*\*(j + 1) - j

Out[41]: array([ 2, 3, 6, 13, 28])

﻿a = np.arange(10000)

%timeit a + 1

5.98 µs ± 181 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

c = np.ones((3, 3))

c \* c

879 µs ± 29.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

Out[44]:

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

[1., 1., 1.],

[1., 1., 1.]])Out[45]:

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

[3., 3., 3.],

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

c.dot(c)

﻿a = np.arange(100)

a

Out[46]:

array([ 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, 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, 96, 97, 98, 99])

a[0:99] + a[1:100]

Out[47]:

array([ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,

27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51,

53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77,

79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103,

105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129,

131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155,

157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181,

183, 185, 187, 189, 191, 193, 195, 197])

b = np.arange(5)

b

Out[48]: array([0, 1, 2, 3, 4])

﻿2\*\*b

Out[49]: array([ 1, 2, 4, 8, 16])

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

b = np.array([4, 2, 2, 4])

a == b

Out[50]: array([False, True, False, True])

a > b

Out[51]: array([False, False, True, False])

﻿n [52]: a = np.array([1, 2, 3, 4])

b = np.array([4, 2, 2, 4])

c = np.array([1, 2, 3, 4])

np.array\_equal(a, b)

#the arrays are the not the same and therefore are not equal

Out[52]: False

np.array\_equal(a, c)

#the arrays are the same and are therefore equal

Out[53]: True

﻿a = np.array([1, 1, 0, 0], dtype=bool)

b = np.array([1, 0, 1, 0], dtype=bool)

np.logical\_or(a, b)

Out[54]: array([ True, True, True, False])

np.logical\_and(a, b)

Out[55]: array([ True, False, False, False])

a = np.arange(5)

np.sin(a)

Out[56]: array([ 0. , 0.84147098, 0.90929743, 0.14112001, -0.7568025 ])

np.log(a)

\_\_main\_\_:1: RuntimeWarning: divide by zero encountered in log

Out[57]: array([ -inf, 0. , 0.69314718, 1.09861229, 1.38629436])

﻿np.exp(a)

Out[58]: array([ 1. , 2.71828183, 7.3890561 , 20.08553692, 54.59815003])

a = np.arange(4)

a + np.array([1, 2])

Traceback (most recent call last):

File "<ipython-input-59-7ded08b2ed74>", line 2, in <module>

a + np.array([1, 2])

ValueError: operands could not be broadcast together with shapes (4,) (2,)

a = np.triu(np.ones((3, 3)), 1)

a

Out[60]:

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

[0., 0., 1.],

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

﻿n [61]: x = np.array([1, 2, 3, 4])

np.sum(x)

Out[61]: 10

x.sum()

Out[62]: 10

x = np.array([[1, 1], [2, 2]])

x

Out[63]:

array([[1, 1],

[2, 2]])

x.sum(axis=0)

Out[64]: array([3, 3])

x[:, 0].sum(), x[:, 1].sum()

Out[65]: (3, 3)

﻿x.sum(axis=1)

Out[66]: array([2, 4])

x[0, :].sum(), x[1, :].sum()

Out[67]: (2, 4)

x = np.random.rand(2, 2, 2)

x.sum(axis=2)[0, 1]

Out[68]: 1.2671177193964822

x[0, 1, :].sum()

Out[69]: 1.2671177193964822

x = np.array([1, 3, 2])

x.min()

Out[70]: 1

x.max()

Out[71]: 3

﻿x.argmin()

#index(location) of minimum value of the array

Out[72]: 0

x.argmax()

#index(location) of maximum value of the array

Out[73]: 1

np.all([True, True, False])

Out[74]: False

np.any([True, True, False])

Out[75]: True

a = np.zeros((100, 100))

np.any(a !=0)

Out[76]: False

np.all(a == a)

Out[77]: True

a = np.array([1, 2, 3, 2])

b = np.array([2, 2, 3, 2])

c = np.array([6, 4, 4, 5])

((a <= b) & (b <= c)).all()

Out[78]: True

﻿a = np.array([1, 2, 3, 2])

b = np.array([2, 2, 3, 2])

c = np.array([6, 4, 4, 5])

((a <= b) & (b <= c)).all()

Out[78]: True

x = np.array([1, 2, 3, 1])

y = np.array([[1, 2, 3], [5, 6, 1]])

x.mean()

Out[79]: 1.75

np.median(x)

Out[80]: 1.5

np.median(y, axis=-1)

Out[81]: array([2., 5.])

x.std()

Out[82]: 0.82915619758885

x = np.array([1, 12, 34, 54, 68, 86])

x.cumsum()

#cumulatively sums each element in the array

Out[83]: array([ 1, 13, 47, 101, 169, 255])

infile = open("/Users/roryeggleston/Documents/populations.txt")

data = np.loadtxt("/Users/roryeggleston/Documents/populations.txt")

year, hares, lynxes, carrots = data.T

# assigning columns to variables

﻿from matplotlib import pyplot as plt

plt.axes([0.2, 0.1, 0.5, 0.8])

plt.plot(year, hares, year, lynxes, year, carrots)

plt.legend(('Hare', 'Lynx', 'Carrot'), loc=(1.05, 0.5))

Out[86]: <matplotlib.legend.Legend at 0x11d53fac8>



﻿populations = data[:, 1:]

populations.mean(axis=0)

Out[87]: array([34080.95238095, 20166.66666667, 42400. ])

populations.std(axis=0)

Out[88]: array([20897.90645809, 16254.59153691, 3322.50622558])

np.argmax(populations, axis=1)

Out[89]: array([2, 2, 0, 0, 1, 1, 2, 2, 2, 2, 2, 2, 0, 0, 0, 1, 2, 2, 2, 2, 2])

a = np.tile(np.arange(0, 40, 10), (3, 1)).T

a

Out[90]:

array([[ 0, 0, 0],

[10, 10, 10],

[20, 20, 20],

[30, 30, 30]])

﻿n [91]: b = np.array([0, 1, 2])

a + b

Out[91]:

array([[ 0, 1, 2],

[10, 11, 12],

[20, 21, 22],

[30, 31, 32]])

a = np.ones((4, 5))

a[0] = 2

a

Out[92]:

array([[2., 2., 2., 2., 2.],

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

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

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

a = np.arange(0, 40, 10)

a.shape

Out[93]: (4,)

a = a[:, np.newaxis]

a.shape

Out[94]: (4, 1)

a

Out[95]:

array([[ 0],

[10],

[20],

[30]])

a + b

Out[96]:

array([[ 0, 1, 2],

[10, 11, 12],

[20, 21, 22],

[30, 31, 32]])

﻿n [97]: mileposts = np.array([0, 198, 303, 736, 871, 1175, 1475, 1544, 1913, 2448])

distance\_array = np.abs(mileposts - mileposts[:, np.newaxis])

distance\_array

Out[97]:

array([[ 0, 198, 303, 736, 871, 1175, 1475, 1544, 1913, 2448],

[ 198, 0, 105, 538, 673, 977, 1277, 1346, 1715, 2250],

[ 303, 105, 0, 433, 568, 872, 1172, 1241, 1610, 2145],

[ 736, 538, 433, 0, 135, 439, 739, 808, 1177, 1712],

[ 871, 673, 568, 135, 0, 304, 604, 673, 1042, 1577],

[1175, 977, 872, 439, 304, 0, 300, 369, 738, 1273],

[1475, 1277, 1172, 739, 604, 300, 0, 69, 438, 973],

[1544, 1346, 1241, 808, 673, 369, 69, 0, 369, 904],

[1913, 1715, 1610, 1177, 1042, 738, 438, 369, 0, 535],

[2448, 2250, 2145, 1712, 1577, 1273, 973, 904, 535, 0]])

x, y = np.arange(5), np.arange(5)[:, np.newaxis]

distance = np.sqrt(x \*\* 2 + y \*\* 2)

distance

Out[98]:

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

[1. , 1.41421356, 2.23606798, 3.16227766, 4.12310563],

[2. , 2.23606798, 2.82842712, 3.60555128, 4.47213595],

[3. , 3.16227766, 3.60555128, 4.24264069, 5. ],

[4. , 4.12310563, 4.47213595, 5. , 5.65685425]])

plt.pcolor(distance)

plt.colorbar()

Out[99]: <matplotlib.colorbar.Colorbar at 0x11d645748>



﻿x, y = np.ogrid[0:5, 0:5]

x, y

Out[100]:

(array([[0],

[1],

[2],

[3],

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

x.shape, y.shape

Out[101]: ((5, 1), (1, 5))

distance = np.sqrt(x \*\* 2 + y \*\* 2)

x, y = np.mgrid[0:4, 0:4]

x

y

Out[103]:

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

[0, 1, 2, 3],

[0, 1, 2, 3],

[0, 1, 2, 3]])

a = np.array([[1, 2, 3], [4, 5, 6]])

a.ravel()

Out[104]: array([1, 2, 3, 4, 5, 6])

a.T

Out[105]:

array([[1, 4],

[2, 5],

[3, 6]])

a.T.ravel()

Out[106]: array([1, 4, 2, 5, 3, 6])

a.shape

Out[107]: (2, 3)

b = a.ravel()

b = b.reshape((2, 3))

b

Out[108]:

array([[1, 2, 3],

[4, 5, 6]])

a.reshape((2, -1))

#LEFT OFF AT SECOND HALF OF RESHAPING IN 1.3.2.4

Out[109]:

array([[1, 2, 3],

[4, 5, 6]])

b[0, 0] = 99

a

Out[110]:

array([[99, 2, 3],

[ 4, 5, 6]])

a = np.zeros((3, 2))

b = a.T.reshape(3\*2)

b[0] = 9

a

Out[111]:

array([[0., 0.],

[0., 0.],

[0., 0.]])

z = np.array([1, 2, 3])

z

Out[112]: array([1, 2, 3])

﻿z[:, np.newaxis]

Out[113]:

array([[1],

[2],

[3]])

z[np.newaxis, :]

Out[114]: array([[1, 2, 3]])

a = np.arange(4\*3\*2).reshape(4, 3, 2)

a.shape

Out[115]: (4, 3, 2)

a[0, 2, 1]

Out[116]: 5

b = a.transpose(1, 2, 0)

b.shape

Out[117]: (3, 2, 4)

b[2, 1, 0]

Out[118]: 5

b[2, 1, 0] = -1

a[0, 2, 1]

Out[119]: -1

a = np.arange(4)

a.resize((8,))

a

Out[120]: array([0, 1, 2, 3, 0, 0, 0, 0])

b = a

a.resize((4,))

Traceback (most recent call last):

File "<ipython-input-121-665bd5cc7e2d>", line 2, in <module>

a.resize((4,))

ValueError: cannot resize an array that references or is referenced

by another array in this way. Use the resize function

a = np.array([[4, 3, 5], [1, 2, 1]])

b = np.sort(a, axis = 1)

b

Out[122]:

array([[3, 4, 5],

[1, 1, 2]])

a.sort(axis = 1)

a

Out[123]:

array([[3, 4, 5],

[1, 1, 2]])

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

j = np.argsort(a)

j

Out[124]: array([2, 3, 1, 0])

a[j]

Out[125]: array([1, 2, 3, 4])

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

j\_max = np.argmax(a)

j\_min = np.argmin(a)

j\_max, j\_min

Out[126]: (0, 2)