## NumPy Crash Course

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

*Creating Arrays*

In [2]:
```
my_list = [1, 2, 3]
```

In [3]:
```
np.array(my_list)
```

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

In [4]:
```
type(np.array(my_list))
```

Out[4]:
```
numpy.ndarray
```

In [5]:
```
arr = np.array(my_list)
```

In [6]:
```
arr
```

Out[6]:
```
array([1, 2, 3])
```

In [7]:
```
np.arange(0, 10)
```

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

In [8]:
```
np.arange(0, 11)
```

Out[8]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [9]:
```
np.arange(0, 11, 2)
```

Out[9]:
```
array([ 0,  2,  4,  6,  8, 10])
```

In [10]:
```
np.arange(0, 10, 2)
```

Out[10]:
```
array([0, 2, 4, 6, 8])
```

In [11]:
```
np.zeros(5)
```

Out[11]:
```
array([0., 0., 0., 0., 0.])
```

In [12]:
```
np.zeros(2, 2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-d029f1f63cc8> in <module>()
----> 1 np.zeros(2, 2)

TypeError: data type not understood
```

OUTPUT:

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-d029f1f63cc8> in <module>()
----> 1 np.zeros(2, 2)

TypeError: data type not understood
```

In [13]:
```
np.zeros((2, 2))
```

Out[13]:
```
array([[0., 0.],
       [0., 0.]])
```

In [14]:
```
np.zeros((3, 5))
```

Out[14]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [15]:
```
np.zeros((5, 5))
```

Out[15]:
```
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.]])
```

In [16]:
```
np.ones(3) # note they're floating point numbers
```

Out[16]:
```
array([1., 1., 1.])
```

In [17]:
```
np.ones((3, 5))
```

Out[17]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [18]:
```
np.ones((2, 4))
```

Out[18]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [19]:
```
np.linspace(0, 11, 10)
```

Out[19]:
```
array([ 0.        ,  1.22222222,  2.44444444,  3.66666667,  4.88888889,
        6.11111111,  7.33333333,  8.55555556,  9.77777778, 11.        ])
```

In [20]:
```
np.linspace(0, 11, 11)
```

Out[20]:
```
array([ 0. ,  1.1,  2.2,  3.3,  4.4,  5.5,  6.6,  7.7,  8.8,  9.9, 11. ])
```

In [21]:
```
np.linspace(0, 11, 100)
```

Out[21]:
```
array([ 0.        ,  0.11111111,  0.22222222,  0.33333333,  0.44444444,
        0.55555556,  0.66666667,  0.77777778,  0.88888889,  1.        ,
        1.11111111,  1.22222222,  1.33333333,  1.44444444,  1.55555556,
        1.66666667,  1.77777778,  1.88888889,  2.        ,  2.11111111,
        2.22222222,  2.33333333,  2.44444444,  2.55555556,  2.66666667,
        2.77777778,  2.88888889,  3.        ,  3.11111111,  3.22222222,
        3.33333333,  3.44444444,  3.55555556,  3.66666667,  3.77777778,
        3.88888889,  4.        ,  4.11111111,  4.22222222,  4.33333333,
        4.44444444,  4.55555556,  4.66666667,  4.77777778,  4.88888889,
        5.        ,  5.11111111,  5.22222222,  5.33333333,  5.44444444,
        5.55555556,  5.66666667,  5.77777778,  5.88888889,  6.        ,
        6.11111111,  6.22222222,  6.33333333,  6.44444444,  6.55555556,
        6.66666667,  6.77777778,  6.88888889,  7.        ,  7.11111111,
        7.22222222,  7.33333333,  7.44444444,  7.55555556,  7.66666667,
        7.77777778,  7.88888889,  8.        ,  8.11111111,  8.22222222,
        8.33333333,  8.44444444,  8.55555556,  8.66666667,  8.77777778,
        8.88888889,  9.        ,  9.11111111,  9.22222222,  9.33333333,
        9.44444444,  9.55555556,  9.66666667,  9.77777778,  9.88888889,
       10.        , 10.11111111, 10.22222222, 10.33333333, 10.44444444,
       10.55555556, 10.66666667, 10.77777778, 10.88888889, 11.        ])
```

In [22]:
```
np.linspace(0, 10, 6) #  6 gives the number of elements
                      #+ contrast with arange
```

Out[22]:
```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

In [23]:
```
np.linspace(0, 10, 101)
```

Out[23]:
```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
        1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
        2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
        3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
        4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
        5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
        6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
        7.7,  7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
        8.8,  8.9,  9. ,  9.1,  9.2,  9.3,  9.4,  9.5,  9.6,  9.7,  9.8,
        9.9, 10. ])
```

In [24]:
```
np.random.randint(0, 10)
```

Out[24]:
```
6
```

In [25]:
```
np.random.randint(0, 10)
```

Out[25]:
```
4
```

In [26]:
```
np.random.randint(0, 10)
```

Out[26]:
```
0
```

In [27]:
```
np.random.randint(0, 1000, (3, 3))
```

Out[27]:
```
array([[876, 633, 703],
       [451, 888, 379],
       [755, 736, 444]])
```

In [28]:
```
np.random.randint(0, 1000)
```

Out[28]:
```
569
```

In [29]:
```
np.random.randint(0, 10, (3, 3))
```

Out[29]:
```
array([[3, 2, 9],
       [6, 9, 0],
       [8, 4, 9]])
```

In [30]:
```
np.random.randint(0, 10, (3, 3))
```

Out[30]:
```
array([[8, 0, 5],
       [1, 7, 7],
       [2, 6, 2]])
```

In [31]:
```
np.random.normal()
```

Out[31]:
```
0.4172485423906348
```

In [32]:
```
np.random.normal(loc=[3, 7032, -2.64e9],
                 scale=[5, 500, 1e8],
                 size=(8, 11, 3)) # Last one in size
                                  # is 3 b/c of arrays' size
```

Out[32]:
```
array([[[-3.86644518e+00,  6.29988966e+03, -2.59031573e+09],
        [ 7.30700199e+00,  7.39230154e+03, -2.58833299e+09],
        [ 1.89465497e+00,  6.06946932e+03, -2.72954372e+09],
        [-1.47193502e-01,  7.42070239e+03, -2.82477352e+09],
        [ 2.94617621e+00,  7.24880034e+03, -2.70701886e+09],
        [-1.84583807e+00,  6.82857904e+03, -2.67652307e+09],
        [ 1.01376872e+01,  6.84376177e+03, -2.58513308e+09],
        [ 6.10941980e+00,  7.58093960e+03, -2.62136275e+09],
        [-1.14667261e+00,  7.38548352e+03, -2.46573743e+09],
        [ 5.87535012e+00,  7.14341414e+03, -2.65937722e+09],
        [ 4.89081691e-01,  6.83349694e+03, -2.56318234e+09]],

       [[ 1.09484904e+00,  6.94450809e+03, -2.43776277e+09],
        [ 3.34235656e+00,  7.21282076e+03, -2.59506799e+09],
        [-3.59334951e+00,  7.29782671e+03, -2.65527113e+09],
        [-3.62494176e+00,  7.37339955e+03, -2.73970995e+09],
        [-3.91168734e+00,  7.36282461e+03, -2.63943410e+09],
        [ 1.25409853e+01,  5.98382243e+03, -2.81031757e+09],
        [ 2.21294530e-01,  7.67581155e+03, -2.56120341e+09],
        [-1.22788285e+00,  7.35085227e+03, -2.48469829e+09],
        [-5.28214431e-01,  6.98045012e+03, -2.46146723e+09],
        [-5.18218119e+00,  7.32965537e+03, -2.44672368e+09],
        [ 4.18754630e+00,  7.39608230e+03, -2.71626906e+09]],

       [[-6.02941174e+00,  7.45244240e+03, -2.58077942e+09],
        [ 8.93773664e+00,  6.92492321e+03, -2.54387323e+09],
        [ 3.91109048e+00,  7.45098840e+03, -2.51081757e+09],
        [ 4.48986582e+00,  6.98877253e+03, -2.58294180e+09],
        [ 6.50048906e-01,  7.01891534e+03, -2.65356102e+09],
        [-2.36198803e+00,  7.39319373e+03, -2.68767107e+09],
        [ 1.13799785e+01,  6.66486251e+03, -2.82877071e+09],
        [-6.73986987e+00,  6.94112281e+03, -2.55221037e+09],
        [ 7.11743128e+00,  6.97813381e+03, -2.49507200e+09],
        [ 3.37469089e-01,  6.90893497e+03, -2.66029546e+09],
        [-1.60679048e+00,  6.79083973e+03, -2.56526701e+09]],

       [[-5.82344804e+00,  7.49689703e+03, -2.75011968e+09],
        [ 4.84165969e+00,  6.92078297e+03, -2.66030258e+09],
        [ 2.96931058e+00,  7.02131228e+03, -2.59137664e+09],
        [ 4.07537937e+00,  7.18129958e+03, -2.63057262e+09],
        [ 5.07598202e+00,  8.25936339e+03, -2.53116538e+09],
        [ 3.93293854e+00,  6.23806084e+03, -2.83876291e+09],
        [ 5.46554005e+00,  7.75086576e+03, -2.52290035e+09],
```

```
       [ 2.32790070e+00,  7.82833407e+03, -2.61017458e+09],
       [ 7.24462504e+00,  7.44445631e+03, -2.53715200e+09],
       [ 8.37483354e+00,  6.75665711e+03, -2.62967558e+09],
       [ 3.99765246e+00,  7.46494110e+03, -2.68333112e+09]],

      [[ 1.02238258e+01,  6.77287860e+03, -2.73389481e+09],
       [-3.65664472e-01,  6.91686555e+03, -2.64042680e+09],
       [ 3.90692756e+00,  7.00811591e+03, -2.63162280e+09],
       [ 1.21457020e+01,  7.29788010e+03, -2.73131139e+09],
       [ 3.39807977e+00,  7.41073671e+03, -2.72853825e+09],
       [ 1.27703366e+01,  7.68514368e+03, -2.57425745e+09],
       [ 4.41447734e+00,  7.48874209e+03, -2.73844484e+09],
       [ 8.20260039e+00,  6.73457253e+03, -2.63722506e+09],
       [ 2.26701276e-01,  6.25025598e+03, -2.49100858e+09],
       [-3.32440392e+00,  5.93626925e+03, -2.65630115e+09],
       [ 7.06619225e+00,  7.23891653e+03, -2.62900502e+09]],

      [[ 6.80730648e+00,  7.37934573e+03, -2.55089491e+09],
       [ 4.41223622e+00,  6.70582575e+03, -2.63730819e+09],
       [ 4.43427266e+00,  6.48576501e+03, -2.67030695e+09],
       [ 2.98008895e+00,  6.58801815e+03, -2.62854336e+09],
       [ 5.44426130e+00,  7.98286166e+03, -2.55687497e+09],
       [ 8.83899441e+00,  7.35681932e+03, -2.95196027e+09],
       [-1.24906252e-01,  7.78266327e+03, -2.58630427e+09],
       [ 9.55193455e+00,  7.24169512e+03, -2.60395150e+09],
       [ 5.58774456e+00,  6.51184665e+03, -2.68114878e+09],
       [ 3.58953332e+00,  7.43871466e+03, -2.74755916e+09],
       [ 1.40784113e+01,  6.58318591e+03, -2.59589147e+09]],

      [[-3.38679370e+00,  6.97403352e+03, -2.36942050e+09],
       [-4.09716681e-01,  7.63546132e+03, -2.67273498e+09],
       [ 1.87100811e+00,  7.30168734e+03, -2.71211349e+09],
       [ 4.09264455e-02,  6.81634732e+03, -2.62682293e+09],
       [ 2.42804886e+00,  6.88253529e+03, -2.69381014e+09],
       [ 4.79042822e+00,  6.36946039e+03, -2.63238818e+09],
       [ 1.20512880e+01,  7.15735833e+03, -2.57792671e+09],
       [ 9.43989421e+00,  6.83089825e+03, -2.63271688e+09],
       [ 4.24590966e+00,  6.32251979e+03, -2.47380688e+09],
       [ 5.57302164e+00,  7.47320632e+03, -2.55506160e+09],
       [ 3.89731483e+00,  6.70283607e+03, -2.77125729e+09]],

      [[ 5.88778793e+00,  7.41788924e+03, -2.65892519e+09],
       [ 3.12487313e+00,  6.68665030e+03, -2.52537683e+09],
```

```
[ 3.55691380e+00,  7.21923313e+03, -2.60065990e+09],
[-6.38718472e+00,  7.50101594e+03, -2.67451102e+09],
[-3.83258365e+00,  6.93418649e+03, -2.70454118e+09],
[ 6.97643055e+00,  7.64971520e+03, -2.50771078e+09],
[ 4.35281828e+00,  7.07638795e+03, -2.52406185e+09],
[ 5.25018028e+00,  8.55366469e+03, -2.45285678e+09],
[ 6.78030066e+00,  6.91772443e+03, -2.67499838e+09],
[ 3.81727440e+00,  6.69952247e+03, -2.62771086e+09],
[-5.83024685e+00,  6.32315741e+03, -2.58097088e+09]]])
```

*Operations*

In [33]:
```
np.random.seed(101) #  "Watch video for details"
                    #+ Typically, Jose uses the value, 101

np.random.randint(0, 100, 10) #  Note that seed is in same cell
                              #+ => same array each time.
```

Out[33]:
```
array([95, 11, 81, 70, 63, 87, 75,  9, 77, 40])
```

In [34]:
```
np.random.seed(101)
np.random.randint(0, 100, 10)
```

Out[34]:
```
array([95, 11, 81, 70, 63, 87, 75,  9, 77, 40])
```

In [35]:
```
#  Now, we generate 10 random integers between 0 and 100
#+ without the seed in the same cell
np.random.randint(0, 100)
```

Out[35]:
```
4
```

In [36]:
```
# Let's do it again ... -ish

np.random.seed(101)
arr = np.random.randint(0, 100, 10)
```

In [37]:
```
arr
```

Out[37]:
```
array([95, 11, 81, 70, 63, 87, 75,  9, 77, 40])
```

In [38]:
```
arr2 = np.random.randint(0, 100, 10)
```

In [39]:
```
arr2
```

Out[39]:
```
array([ 4, 63, 40, 60, 92, 64,  5, 12, 93, 40])
```

In [40]:
```
arr.max()
```

Out[40]:
```
95
```

In [41]:
```
arr.min()
```

Out[41]:
```
9
```

In [42]:
```
arr.mean()
```

Out[42]:
```
60.8
```

In [43]:
```
arr.argmax() # index location of the maximum value
```

Out[43]:
```
0
```

In [44]:
```
arr.argmin() # index location of the minimum value
```

Out[44]:
```
7
```

In [45]:
```
arr # note we have 10, elements
```

Out[45]:
```
array([95, 11, 81, 70, 63, 87, 75,  9, 77, 40])
```

In [46]:
```
arr.reshape(2, 5) # 2*5=10, so we're okay
```

Out[46]:
```
array([[95, 11, 81, 70, 63],
       [87, 75,  9, 77, 40]])
```

*Indexing*

In [47]:
```
mat = np.arange(0, 100).reshape(10, 10)
```

In [48]:
```
mat
```

Out[48]:
```
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]])
```

In [49]:
```
# 1st row, 2nd column
mat[0,1]
```

Out[49]:
```
1
```

In [50]:
```
#DWB
mat[0][1]
```

Out[50]:
```
1
```

In [51]:
```
# Let's try to index the 43
mat[4, 3]
```

Out[51]:
```
43
```

In [52]:
```
row = 0
col = 1
```

In [53]:
```
mat[row, col]
```

Out[53]:
```
1
```

In [54]:
```
# With slicing (to grab chunks)

# all the rows in the first column
mat[:, 0]
```

Out[54]:
```
array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [55]:
```
# Grab all the columns in a particular row - let's get the 50s
mat[5, :]
```

Out[55]:
```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59])
```

In [56]:
```
mat[:, col]
```

Out[56]:
```
array([ 1, 11, 21, 31, 41, 51, 61, 71, 81, 91])
```

In [57]:
```
mat[row, :]
```

Out[57]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [58]:
```
#  get the 3 x 3 matrix in the upper-left
#+ first, let's remember our matrix
mat
```

Out[58]:
```
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]])
```

In [59]:
```
mat[0:3, 0:3]
```

Out[59]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22]])
```

*Masking*

In [60]:
```
#remember
mat
```

Out[60]:
```
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]])
```

In [61]:
```
mat > 50
```

Out[61]:
```
array([[False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        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]])
```

In [62]:
```
import pprint

bool_mat = mat > 50
pprint.pprint(bool_mat)

#DWB# meh
```

```
array([[False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        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]])
```

In [63]:
```
# remember again
mat
```

Out[63]:
```
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]])
```

In [64]:
```
my_filter = mat > 50
```

In [65]:
```
mat[my_filter]
```

Out[65]:
```
array([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])
```

In [66]:
```
mat[mat > 50]
```

Out[66]:
```
array([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])
```

*That's all for now*

https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982580 (https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982580)