## Logistics

1. 80% May Intermediate (Python+DSA) + 20% June Advanced (Python) --> June Common Core
2. If you **want to revise Python+DSA** --> **Shift to Intermediate** Batch (Reach out to support team)
3. 2.5 hrs class + 15-30 mins QnA (optional)
4. Live Lecture to start from **9:03 PM**
5. 4-5 Quizzes per class
6. Mini break at 10:30PM for 10 mins (usually).
7. **Questions in the "Question Tab"** - Instructor may miss it in the chat
8. Use chat window for interaction and answering.
9. **Proper Revision Notes will be provided** - check on the dashboard for this class
10. NumPy - 2-2.5 lectures

https://colab.research.google.com/drive/1U5IzEkNgqHJhS9xZXdZrpeyJtM58N9kx?usp=sharing

## Agenda

1. Introduce the use-case - FitBit
2. Why to use Numpy?
3. Creating a Basic Numpy Array - array(), arange(), linspace(), shape, ndim, type
4. How numpy works under the hood? Why are they called arrays (not lists)
5. 2-D arrays - reshape(), how to transpose, flatten()
6. Creating some special matrices - zeroes, ones, identity, diagonal
7. Indexing and Slicing - Fancy Indexing (Masking)
8. Some Handy Functions (ufunc)
    - Aggregate Function/ Reduction functions - sum(), mean(), min(), max()
    - Logical functions - any() , all()
    - Sorting function - sort(), argsort()
9. Use Case: Fitness Data analysis

# Imagine that you are a Data Scientist at Fitbit

Link: https://drive.google.com/file/d/1kXqcJo4YzmwF1G2BPoA17CI49TZVHANF/view?usp=sharing

```
#date   step_count      mood    calories_burned hours_of_sleep  bool_of_active  weight_kg
06-10-2017      5464    200     181     5       0       66
07-10-2017      6041    100     197     8       0       66
08-10-2017      25      100     0       5       0       66
09-10-2017      5461    100     174     4       0       66
10-10-2017      6915    200     223     5       500     66
11-10-2017      4545    100     149     6       0       66
12-10-2017      4340    100     140     6       0       66
13-10-2017      1230    100     38      7       0       66
14-10-2017      61      100     1       5       0       66
15-10-2017      1258    100     40      6       0       65
16-10-2017      3148    100     101     8       0       65
17-10-2017      4687    100     152     5       0       65
18-10-2017      4732    300     150     6       500     65
19-10-2017      3519    100     113     7       0       65
20-10-2017      1580    100     49      5       0       65
21-10-2017      2822    100     86      6       0       65
22-10-2017      181     100     6       8       0       65
23-10-2017      3158    200     99      5       0       65
```

```
import numpy as np
```

```
a = [1, 2, 3, 4, 5]
```

```
[i**2 for i in a]
```

```
[1, 4, 9, 16, 25]
```

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

```
b
```

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

```
b**2 # elementwise operation
```

```
array([ 1,  4,  9, 16, 25])
```

```
l = range(1000000)
```

```
%timeit [i**2 for i in l]
```

```
1 loop, best of 5: 328 ms per loop
```

```
np_l = np.arange(1000000)
```

```
%timeit np_l**2
```

```
100 loops, best of 5: 1.59 ms per loop
```

```
arr1 = np.array([1, 2, 3])
```

```
arr1
```

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

```
arr1 * 2
```

```
array([2, 4, 6])
```

```
arr1.ndim
```

```
1
```

```
arr1.shape
```

```
(3,)
```

```
a = np.array([1,2,3,4,5,6,7,8])
print(a.ndim, a.shape)
```

```
1 (8,)
```

```
np.arange(1, 5, 2)
```

```
array([1, 3])
```

```
np.arange(1, 5, 1.5)
```

```
array([1. , 2.5, 4. ])
```

```
list(range(1, 5, 2.5))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-8c52bca9d3ed> in <module>()
----> 1 list(range(1, 5, 2.5))

TypeError: 'float' object cannot be interpreted as an integer
```

SEARCH STACK OVERFLOW

```
np.linspace(0, 10, 13)
```

```
array([ 0.        ,  0.83333333,  1.66666667,  2.5       ,  3.33333333,
        4.16666667,  5.        ,  5.83333333,  6.66666667,  7.5       ,
        8.33333333,  9.16666667, 10.        ])
```

```
type(arr1)
```

```
numpy.ndarray
```

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

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

```
np.array([1, 2, 3, 4.5])
```

```
array([1. , 2. , 3. , 4.5])
```

```
np.array([1, 2, 3, 4.5], dtype="float")
```

```
array([1. , 2. , 3. , 4.5])
```

```
np.array([1, 2, 3, 4.5], dtype="int")
```

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

```
100**10
```

```
10000000000000000000
```

```
np.array([0, 10, 100])**10
```

```
array([                  0,         10000000000, 7766279631452241920])
```

```
m1 = np.array([[1, 2, 3], [4, 5, 6]])
m1
```

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

```
m1.shape
```

```
(2, 3)
```

```
m1.ndim
```

```
2
```

```
a = np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])

b = len(a)
print(b)
```

```
3
```

```
a.shape
```

```
(3, 3)
```

```
a.ndim
```

```
2
```

```
m2 = np.arange(1, 13)
m2
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
m3 = m2.reshape(3, 4)
m3
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
m2.reshape(4, 3)
```

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

```
m2.reshape(3, 3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-65-a573ad189aea> in <module>()
----> 1 m2.reshape(3, 3)

ValueError: cannot reshape array of size 12 into shape (3,3)
```

SEARCH STACK OVERFLOW

```
m2.reshape(1, 12)
```

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]])
```

```
m2
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
m2.reshape(12, 1)
```

```
array([[ 1],
```

```
            [ 2],
            [ 3],
            [ 4],
            [ 5],
            [ 6],
            [ 7],
            [ 8],
            [ 9],
            [10],
            [11],
            [12]])
```

```
a = np.arange(3)
a
```

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

```
a.T
```

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

```
a = np.arange(3).reshape(1, 3)
a
```

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

```
a.T
```

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

```
A = np.arange(12).reshape(3, 4)
A
```

```
    array([[ 0,  1,  2,  3],
           [ 4,  5,  6,  7],
           [ 8,  9, 10, 11]])
```

```
A.reshape(12)
```

```
    array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
A.flatten()
```

```
    array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
np.arange(12).reshape(4, 3)
```

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

```
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

```
A.reshape(6, -1)
```

```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

```
A.reshape(6, -4)
```

```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11]])
```

```
A.reshape(-1, 4)
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
np.zeros(4)
```

```
array([0., 0., 0., 0.])
```

```
np.zeros((2, 3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

```
np.ones((2, 3))
```

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

```
a = np.ones((2, 3)) * 5
```

```
a.dtype
```

```
dtype('float64')
```

```
np.identity(3) # eye()
```

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

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

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

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

```
np.diag(a)
```

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

```
### Indexing and slicing
```

```
m1 = np.arange(12)
m1
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
m1[2]
```

```
2
```

```
m1[12]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-95-0abd94d7097d> in <module>()
----> 1 m1[12]

IndexError: index 12 is out of bounds for axis 0 with size 12
```

SEARCH STACK OVERFLOW

```
m2 = np.arange(1, 10).reshape(3, 3)
m2
```

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

```
m2[1][2] # not suggested
```

```
6
```

```
m2[1, 2]
```

```
6
```

```
m2[1, 1]
```

```
5
```

```
m2[1, -1]
```

```
6
```

```
m1 = np.array([100,200,300,400,500,600])
```

```
m1[[2,3,4,1,2,2]]
```

```
array([300, 400, 500, 200, 300, 300])
```

```
m1 = np.arange(9).reshape((3,3))
m1
```

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

```
m1[[0,1,2],[0,1,2]]
```

```
array([0, 4, 8])
```

```
# slicing
```

```
m1 = np.arange(12)
```

```
m1
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
m1[:5]
```

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

```
m1 = np.array([[0,1,2,3],
     [4,5,6,7],
     [8,9,10,11]])
m1
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
m1[0,1:3]
```

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

```
m1[1:,1:3]
```

```
array([[ 5,  6],
       [ 9, 10]])
```

```
m1[:, 2:]
```

```
array([[ 2,  3],
       [ 6,  7],
       [10, 11]])
```

```
m1[:, [1, 3]]
```

```
array([[ 1,  3],
       [ 5,  7],
       [ 9, 11]])
```

```
a = np.array([0,1,2,3,4,5])
a[4:] = 10
a
```

```
array([ 0,  1,  2,  3, 10, 10])
```

```
a = np.array([1,2,3,4,5])
b = np.array([8,7,6])
```

```
a[2:] = b[::-1]
a
```

```
array([1, 2, 6, 7, 8])
```

```
# Fancy Indexing
```

```
m1 = np.arange(12).reshape(3, 4)
print(m1)
m1 < 6
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
array([[ True,  True,  True,  True],
       [ True,  True, False, False],
       [False, False, False, False]])
```

```
m1[m1 < 6]
```

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

```
m1[m1%2 == 0]
```

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

```
m1[(m1%2 == 0) & (m1%5 == 0)]
```

```
array([ 0, 10])
```

```
a = np.array([0,1,2,3,4,5])
a[a%2 == 0] = -1
a
```

```
array([-1,  1, -1,  3, -1,  5])
```

```
np.array([1.2, 1.4, 5.6]).astype("int32")
```

```
array([1, 1, 5], dtype=int32)
```

✓ 0s completed at 23:49