Day 2 Objectives

- List and Dictionary Comprehensions
- Lambda, map(), filter()
- Numpy
- Pandas

In [1]:

```
'14A81A04'+str(0)+str(5)
```

Out[1]:

```
'14A81A0405'
```

In [ ]:

```
roll = '18A51A04'

for num in range(1,101,1):
    if num<10:
        s = roll + str(0) + str(num)
    else:
        s = roll + str(num)
    print(s)
```

In [3]:

```
fh = open('RollNumbers.txt','w')

for num in range(1,101,1):
    if num<10:
        s = roll + str(0) + str(num)
    else:
        s = roll + str(num)
    fh.write(s + ' - ')


fh.close()
```

In [4]:

```
roll = '18A51A04'
roll_numbers =[]
for num in range(1,101,1):
    if num<10:
        s = roll + str(0) + str(num)
    else:
        s = roll + str(num)
    roll_numbers.append(s)

print(roll_numbers)
```

```
['18A51A0401', '18A51A0402', '18A51A0403', '18A51A0404', '18A51A0405', '18
A51A0406', '18A51A0407', '18A51A0408', '18A51A0409', '18A51A0410', '18A51A
0411', '18A51A0412', '18A51A0413', '18A51A0414', '18A51A0415', '18A51A041
6', '18A51A0417', '18A51A0418', '18A51A0419', '18A51A0420', '18A51A0421',
'18A51A0422', '18A51A0423', '18A51A0424', '18A51A0425', '18A51A0426', '18A
51A0427', '18A51A0428', '18A51A0429', '18A51A0430', '18A51A0431', '18A51A0
432', '18A51A0433', '18A51A0434', '18A51A0435', '18A51A0436', '18A51A043
7', '18A51A0438', '18A51A0439', '18A51A0440', '18A51A0441', '18A51A0442',
'18A51A0443', '18A51A0444', '18A51A0445', '18A51A0446', '18A51A0447', '18A
51A0448', '18A51A0449', '18A51A0450', '18A51A0451', '18A51A0452', '18A51A0
453', '18A51A0454', '18A51A0455', '18A51A0456', '18A51A0457', '18A51A045
8', '18A51A0459', '18A51A0460', '18A51A0461', '18A51A0462', '18A51A0463',
'18A51A0464', '18A51A0465', '18A51A0466', '18A51A0467', '18A51A0468', '18A
51A0469', '18A51A0470', '18A51A0471', '18A51A0472', '18A51A0473', '18A51A0
474', '18A51A0475', '18A51A0476', '18A51A0477', '18A51A0478', '18A51A047
9', '18A51A0480', '18A51A0481', '18A51A0482', '18A51A0483', '18A51A0484',
'18A51A0485', '18A51A0486', '18A51A0487', '18A51A0488', '18A51A0489', '18A
51A0490', '18A51A0491', '18A51A0492', '18A51A0493', '18A51A0494', '18A51A0
495', '18A51A0496', '18A51A0497', '18A51A0498', '18A51A0499', '18A51A0410
0']
```

List Comprehensions

In addition to sequence operations and list methods,

Python includes a more advanced operation called a list comprehension.

List comprehensions allow us to build out lists using a different notation.

You can think of it as essentially a one line for loop built inside of brackets.

In [5]:

```
li = [i for i in range(1,101)]
print(li)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 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, 5
8, 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, 9
5, 96, 97, 98, 99, 100]
```

In [6]:

```
even = [i for i in range(1,101) if i % 2 == 0]
print(even)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 4
0, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76,
78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

In [7]:

```
even2 = [i if i % 2 == 0 else 0 for i in range(1,101)]
print(even2)
```

```
[0, 2, 0, 4, 0, 6, 0, 8, 0, 10, 0, 12, 0, 14, 0, 16, 0, 18, 0, 20, 0, 22,
0, 24, 0, 26, 0, 28, 0, 30, 0, 32, 0, 34, 0, 36, 0, 38, 0, 40, 0, 42, 0, 4
4, 0, 46, 0, 48, 0, 50, 0, 52, 0, 54, 0, 56, 0, 58, 0, 60, 0, 62, 0, 64,
0, 66, 0, 68, 0, 70, 0, 72, 0, 74, 0, 76, 0, 78, 0, 80, 0, 82, 0, 84, 0, 8
6, 0, 88, 0, 90, 0, 92, 0, 94, 0, 96, 0, 98, 0, 100]
```

In [8]:

```
even_squares = [i ** 2 for i in range(1,101) if i % 2 == 0]
print(even_squares)
```

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 900, 102
4, 1156, 1296, 1444, 1600, 1764, 1936, 2116, 2304, 2500, 2704, 2916, 3136,
3364, 3600, 3844, 4096, 4356, 4624, 4900, 5184, 5476, 5776, 6084, 6400, 67
24, 7056, 7396, 7744, 8100, 8464, 8836, 9216, 9604, 10000]
```

Dictionary Comprehensions

Syntax

{key:value **for** key,value **in** group_of_items}

{1:1,2:4,3:9,.....,100:10000}

In [9]:

```
numsq = {i:i**2 for i in range(1,101)}
print(numsq)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11:
121, 12: 144, 13: 169, 14: 196, 15: 225, 16: 256, 17: 289, 18: 324, 19: 36
1, 20: 400, 21: 441, 22: 484, 23: 529, 24: 576, 25: 625, 26: 676, 27: 729,
28: 784, 29: 841, 30: 900, 31: 961, 32: 1024, 33: 1089, 34: 1156, 35: 122
5, 36: 1296, 37: 1369, 38: 1444, 39: 1521, 40: 1600, 41: 1681, 42: 1764, 4
3: 1849, 44: 1936, 45: 2025, 46: 2116, 47: 2209, 48: 2304, 49: 2401, 50: 2
500, 51: 2601, 52: 2704, 53: 2809, 54: 2916, 55: 3025, 56: 3136, 57: 3249,
58: 3364, 59: 3481, 60: 3600, 61: 3721, 62: 3844, 63: 3969, 64: 4096, 65:
4225, 66: 4356, 67: 4489, 68: 4624, 69: 4761, 70: 4900, 71: 5041, 72: 518
4, 73: 5329, 74: 5476, 75: 5625, 76: 5776, 77: 5929, 78: 6084, 79: 6241, 8
0: 6400, 81: 6561, 82: 6724, 83: 6889, 84: 7056, 85: 7225, 86: 7396, 87: 7
569, 88: 7744, 89: 7921, 90: 8100, 91: 8281, 92: 8464, 93: 8649, 94: 8836,
95: 9025, 96: 9216, 97: 9409, 98: 9604, 99: 9801, 100: 10000}
```

In [10]:

```
evensq = {i:i**2 for i in range(1,101) if i % 2 ==0}
print(evensq)
```

```
{2: 4, 4: 16, 6: 36, 8: 64, 10: 100, 12: 144, 14: 196, 16: 256, 18: 324, 2
0: 400, 22: 484, 24: 576, 26: 676, 28: 784, 30: 900, 32: 1024, 34: 1156, 3
6: 1296, 38: 1444, 40: 1600, 42: 1764, 44: 1936, 46: 2116, 48: 2304, 50: 2
500, 52: 2704, 54: 2916, 56: 3136, 58: 3364, 60: 3600, 62: 3844, 64: 4096,
66: 4356, 68: 4624, 70: 4900, 72: 5184, 74: 5476, 76: 5776, 78: 6084, 80:
6400, 82: 6724, 84: 7056, 86: 7396, 88: 7744, 90: 8100, 92: 8464, 94: 883
6, 96: 9216, 98: 9604, 100: 10000}
```

{1:'odd',2:'even',.....,100:'even'}

In [11]:

```
num = {i:'even' if i % 2 == 0 else 'odd' for i in range(1,101)}
print(num)
```

```
{1: 'odd', 2: 'even', 3: 'odd', 4: 'even', 5: 'odd', 6: 'even', 7: 'odd',
8: 'even', 9: 'odd', 10: 'even', 11: 'odd', 12: 'even', 13: 'odd', 14: 'ev
en', 15: 'odd', 16: 'even', 17: 'odd', 18: 'even', 19: 'odd', 20: 'even',
21: 'odd', 22: 'even', 23: 'odd', 24: 'even', 25: 'odd', 26: 'even', 27:
'odd', 28: 'even', 29: 'odd', 30: 'even', 31: 'odd', 32: 'even', 33: 'od
d', 34: 'even', 35: 'odd', 36: 'even', 37: 'odd', 38: 'even', 39: 'odd', 4
0: 'even', 41: 'odd', 42: 'even', 43: 'odd', 44: 'even', 45: 'odd', 46: 'e
ven', 47: 'odd', 48: 'even', 49: 'odd', 50: 'even', 51: 'odd', 52: 'even',
53: 'odd', 54: 'even', 55: 'odd', 56: 'even', 57: 'odd', 58: 'even', 59:
'odd', 60: 'even', 61: 'odd', 62: 'even', 63: 'odd', 64: 'even', 65: 'od
d', 66: 'even', 67: 'odd', 68: 'even', 69: 'odd', 70: 'even', 71: 'odd', 7
2: 'even', 73: 'odd', 74: 'even', 75: 'odd', 76: 'even', 77: 'odd', 78: 'e
ven', 79: 'odd', 80: 'even', 81: 'odd', 82: 'even', 83: 'odd', 84: 'even',
85: 'odd', 86: 'even', 87: 'odd', 88: 'even', 89: 'odd', 90: 'even', 91:
'odd', 92: 'even', 93: 'odd', 94: 'even', 95: 'odd', 96: 'even', 97: 'od
d', 98: 'even', 99: 'odd', 100: 'even'}
```

In [12]:

```
li = [i for i in range(1,101)]
print(li)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 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, 5
8, 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, 9
5, 96, 97, 98, 99, 100]
```

In [13]:

```
#str(int(li))
```

In [14]:

```
str(li)
```

Out[14]:

```
'[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 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, 5
8, 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, 9
5, 96, 97, 98, 99, 100]'
```

In [15]:

```
strnum = [str(num) for num in li]
print(strnum)
```

```
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '1
4', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '2
6', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '3
8', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '5
0', '51', '52', '53', '54', '55', '56', '57', '58', '59', '60', '61', '6
2', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '7
4', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '8
6', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '9
8', '99', '100']
```

[[1],[2],[3],[4],[5].....,[100]]

In [16]:

```
strnum = [[num] for num in li]
print(strnum)
```

```
[[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [1
4], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [2
6], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [3
8], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [5
0], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [6
2], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [7
4], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [8
6], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [9
8], [99], [100]]
```

map()

The **map** function allows you to "map" a function to an iterable object.

Syntax:

$map(function, group\_of\_elements)$

In [17]:

```
li2 = list(map(str,li))
print(li2)
```

```
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '1
4', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '2
6', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '3
8', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '5
0', '51', '52', '53', '54', '55', '56', '57', '58', '59', '60', '61', '6
2', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '7
4', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '8
6', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '9
8', '99', '100']
```

In [18]:

```
def squares(a):
    return a ** 2
```

In [19]:

```
li3 = list(map(squares,li))
print(li3)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289,
324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 102
4, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936,
2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 32
49, 3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 476
1, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561,
6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 88
36, 9025, 9216, 9409, 9604, 9801, 10000]
```

In [20]:

```
def cubes(b):
    return b **3

print(list(map(cubes,li)))
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744, 337
5, 4096, 4913, 5832, 6859, 8000, 9261, 10648, 12167, 13824, 15625, 17576,
19683, 21952, 24389, 27000, 29791, 32768, 35937, 39304, 42875, 46656, 5065
3, 54872, 59319, 64000, 68921, 74088, 79507, 85184, 91125, 97336, 103823,
110592, 117649, 125000, 132651, 140608, 148877, 157464, 166375, 175616, 18
5193, 195112, 205379, 216000, 226981, 238328, 250047, 262144, 274625, 2874
96, 300763, 314432, 328509, 343000, 357911, 373248, 389017, 405224, 42187
5, 438976, 456533, 474552, 493039, 512000, 531441, 551368, 571787, 592704,
614125, 636056, 658503, 681472, 704969, 729000, 753571, 778688, 804357, 83
0584, 857375, 884736, 912673, 941192, 970299, 1000000]
```

Lambda

- lambda keyword to declare an anonymous function,
- which is why we refer to them as "lambda functions".
- An anonymous function refers to a function declared with no name

In [21]:

```
sq = lambda a:a**2
```

In [22]:

```
sq(54)
```

Out[22]:

```
2916
```

In [ ]:

```
list(map(lambda x:x**2,li))
```

Numpy (numpy.org) (Numerical Python)

In [24]:

```
import numpy
```

In [25]:

```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\jesus\anaconda3\lib\site-
packages (1.16.2)
Note: you may need to restart the kernel to use updated packages.
```

In [26]:

```
a1 = numpy.array([1,2,3,4])
a1
```

Out[26]:

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

In [27]:

```
type(a1)
```

Out[27]:

```
numpy.ndarray
```

In [28]:

```
a1.ndim
```

Out[28]:

```
1
```

In [29]:

```
a2 = numpy.arange(1,101,1)
a2
```

Out[29]:

```
array([  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, 100])
```

In [30]:

```
a2.size
```

Out[30]:

```
100
```

In [31]:

```
a2.shape
```

Out[31]:

```
(100,)
```

In [ ]:

```
a3 = a2.reshape(50,2)
a3
```

In [33]:

```
a3.shape
```

Out[33]:

```
(50, 2)
```

In [34]:

```
a2
```

Out[34]:

```
array([  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, 100])
```

In [35]:

```
a3.ndim
```

Out[35]:

```
2
```

In [36]:

```
l1 = [i for i in range(1,101)]
l2 = [i for i in range(100,0,-1)]
%timeit [l1[i] + l2[i] for i in range(0,len(l1))]
```

```
26.3 µs ± 7.52 µs per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

In [37]:

```
ar1 = numpy.arange(1,101,1)
ar2 = numpy.arange(100,0,-1)
%timeit ar1 + ar2
```

```
1.08 µs ± 245 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

In [38]:

```
import numpy as np
```

In [39]:

```
ar3 = np.zeros(10)
ar3
```

Out[39]:

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

In [40]:

```
ar3 = np.zeros(10,dtype=int)
ar3
```

Out[40]:

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

In [41]:

```
one = np.ones(10,dtype=int)
one
```

Out[41]:

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

In [42]:

```
5 * one
```

Out[42]:

```
array([5, 5, 5, 5, 5, 5, 5, 5, 5, 5])
```

In [43]:

```
5<one
```

Out[43]:

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

In [44]:

```
1 == one
```

Out[44]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

In [45]:

```
one[0]
```

Out[45]:

```
1
```

In [46]:

```
one[0:]
```

Out[46]:

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

In [47]:

```
a2 = np.arange(1,101)
bol = a2 % 2 == 0
```

In [48]:

```
a2[bol]
```

Out[48]:

```
array([  2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,  24,  26,
        28,  30,  32,  34,  36,  38,  40,  42,  44,  46,  48,  50,  52,
        54,  56,  58,  60,  62,  64,  66,  68,  70,  72,  74,  76,  78,
        80,  82,  84,  86,  88,  90,  92,  94,  96,  98, 100])
```

In [49]:

```
a2[a2 % 3 == 0]
```

Out[49]:

```
array([ 3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51,
       54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99])
```

In [50]:

```
i = np.eye(5,5,dtype = int)
```

In [51]:

```
i.diagonal()
```

Out[51]:

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

In [52]:

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

Out[52]:

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

In [53]:

```
d1 = np.diag([1,2,3,4])
d2 = np.diag([100,200,300,400])
print(d1)
print(d2)
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
[[100   0   0   0]
 [  0 200   0   0]
 [  0   0 300   0]
 [  0   0   0 400]]
```

In [54]:

```
d3 = np.hstack(d1)
d3.shape
```

Out[54]:

```
(16,)
```

In [55]:

```
one.shape
```

Out[55]:

```
(10,)
```

In [56]:

```
d4 = np.vstack(one)
d4.shape
```

Out[56]:

```
(10, 1)
```

In [57]:

```
d4.dtype
```

Out[57]:

```
dtype('int32')
```

In [58]:

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

Out[58]:

```
array(['1', '2', '3', '4', 'apssdc'], dtype='<U11')
```

In [59]:

```
np.array([1,2,3,4,5,6.0])
```

Out[59]:

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

In [60]:

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

Out[60]:

```
array([1.+0.j, 2.+0.j, 3.+0.j, 4.+0.j, 5.+6.j])
```

In [61]:

```
img.astype(str)
```

Out[61]:

```
array(['(1+0j)', '(2+0j)', '(3+0j)', '(4+0j)', '(5+6j)'], dtype='<U64')
```

In [62]:

```
img.astype(int)
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Comple
xWarning: Casting complex values to real discards the imaginary part
  """Entry point for launching an IPython kernel.
```

Out[62]:

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

In [63]:

```
img.dtype
```

Out[63]:

```
dtype('complex128')
```

In [64]:

```
np.logical_not(one)
```

Out[64]:

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

$a^2 + b^2 + 2 * a * b$

$a_0 + b_0 + c_0$

Day Outcomes

Discussed

- about few examples in files
- list comprehensions
- dictionary comprehensions
- map()
- lambda
- numpy