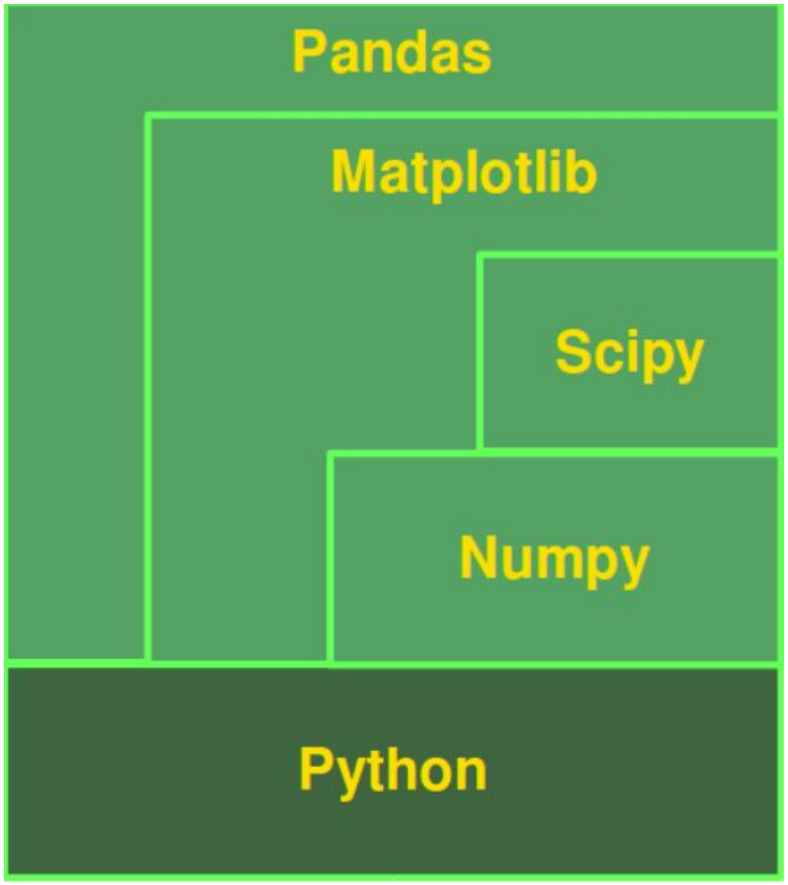
Relation between Python packages for scientific computation

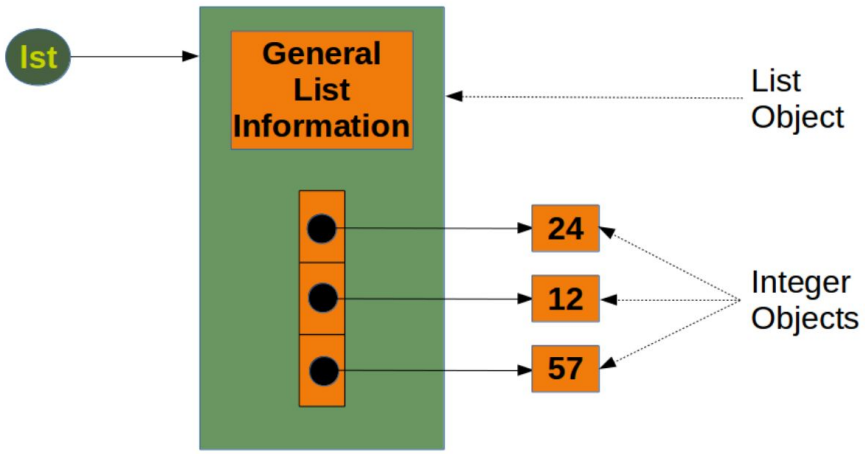


Numpy

https://www.python-course.eu/numpy.php
https://numpy.org/doc/stable/reference/arrays.ndarray.html

- Central data structure is a numpy array.
  - N-dimensional array
  - Homogeneous elements - have the same type
  - Individual elements operation
  - Whole array operations
  - Very efficient

- Python lists:

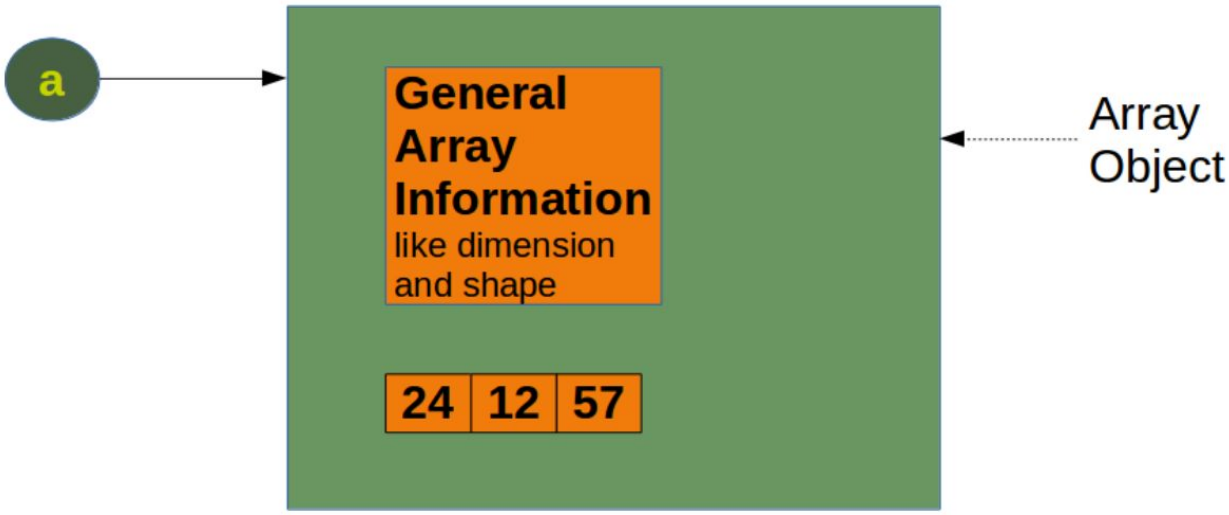



```
from sys import getsizeof as size

lst = [24, 12, 57]

size_of_list_object = size(lst)    # only green box
size_of_elements = len(lst) * size(lst[0]) # the elements
```

  - The size of the list (the green box) is the size of the general list information (64 bytes) plus the size of the references to the individual elements of the list (8 bytes/element)

    - The elements themselves. Small integers usually take 28 bytes whereas large integers can take 48 bytes.

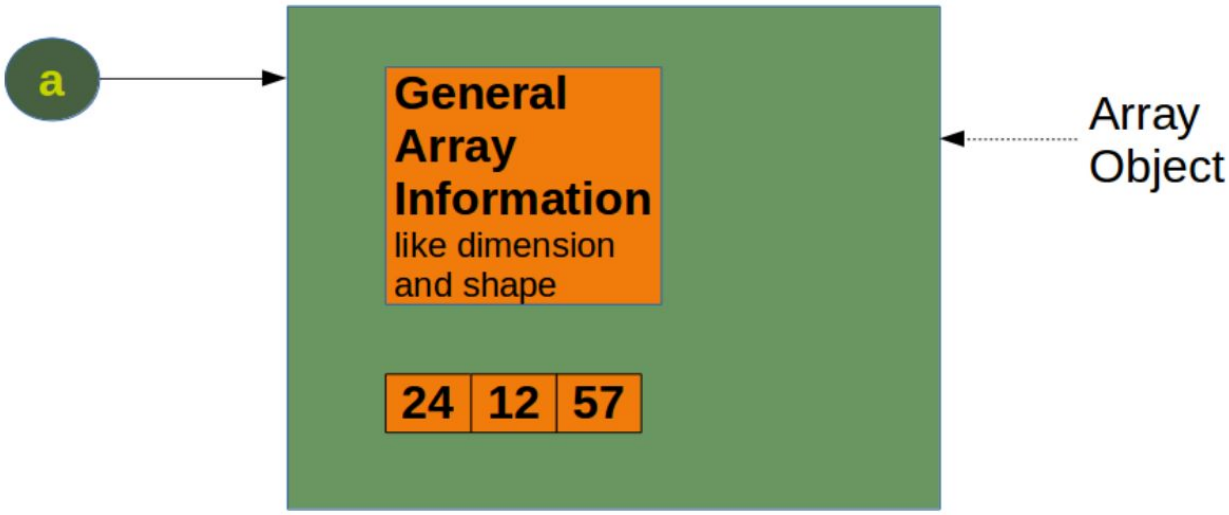
- Python `array`:




```
from array import array
arr = array('i', [1, 2, 3, 4, 5])
```

    - General array information is 64 bytes
    - Each integer takes 4 bytes
    - Fixed size integers. Size depend on typecode
    - Integers packed within the array itself

- Numpy arrays: Homogeneous arrays. Representation similar to array.array



```
import numpy as np

narr = np.array([])
narr.shape
(0,)
size(narr) => 96 # General info
narr = np.array([1])
size(narr) => 96 + 8*1
narr.dtype => dtype('int64')
narr = np.array([23,  2,  3,  4])
narr.shape => (4,0)
size(narr) => 96 + 8*4
```

```
narr1 = np.array([[]])
narr1.shape => (1, 0)
size(narr1) =>  112
narr1 = np.array([[1,2,3],[4,5,6]])
narr1.shape => (2, 3)
size(narr1) => Out[118]: 112 + 6*8
narr1=np.array([[[1],[2],[3]],[[4],[5]
               ,[6]]])
narr1.shape => (2, 3, 1)
size(narr1) => 128 + 6*8
```

Question: Consider the python list [[1,2,3,4], [5,6,7,8]] and the same list as a numpy array. What are the sizes required in the two cases?

- **Conclusion**
  - **One can think of numpy arrays as consisting of a shape and the data elements that make up the array.**
  - **An array with shape (x,y,z) and dtype of size b bytes would be of the following size:**
    - **The general array information would require the same size as a (1,1,0) array.**
    - **The data elements would require x*y*z*b bytes.**

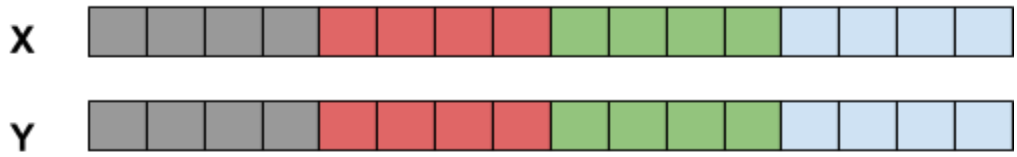
- Time comparison

```
import numpy as np
from timeit import Timer

size_of_vec = 1000

X_list = range(size_of_vec)
Y_list = range(size_of_vec)

X = np.arange(size_of_vec)
Y = np.arange(size_of_vec)

def pure_python_version():
    Z = [X_list[i] + Y_list[i] for i in
range(len(X_list)) ]

def numpy_version():
    Z = X + Y   # numpy whole array operation


#timer_obj = Timer("x = x + 1", "x = 0")
#"x = x + 1" Statement to be timed
#"x = 0"     Setup statement

timer_obj1 = Timer("pure_python_version()",
                   "from __main__ import
pure_python_version")
timer_obj2 = Timer("numpy_version()",
                   "from __main__ import
numpy_version")
```

```
for i in range(3):
    t1 = timer_obj1.timeit(10)
    t2 = timer_obj2.timeit(10)
    print("time for pure Python version: ", t1)
    print("time for Numpy version: ", t2)
    print(f"Numpy was {t1 / t2:7.2f} times
faster!")
```

```
time for pure Python version:
0.00242765503935515 9
time for Numpy version:  3.624614328145981e-05
Numpy was   66.98 times faster!
time for pure Python version:
0.00237795407883822 9
time for Numpy version:  2.5078188627958298e-05
Numpy was   94.82 times faster!
time for pure Python version:
0.00283168489113450 05
time for Numpy version:  2.4903099983930588e-05
Numpy was  113.71 times faster!
```

- Why are computations on numpy arrays faster than python lists?
  - Less type checking at execution time. We have talked about this earlier.
  - Very efficient implementation of whole array operations using vector support.



X + Y can take 4 or even 2 SIMD operations

Structured Arrays

(https://www.python-course.eu/numpy_dtype.php)

- The elements of an array are described by a class called `dtype`

```
narr1 = np.array([[[1],[2],[3]],[[4],[5],[6]]])
narr1.dtype => dtype('int64')
```

Suppose we wanted to represent the following table---the column names are important:

Country	Population Density	Area	Population
Netherlands	393	41526	16,928,800
Belgium	337	30510	11,007,020
United Kingdom	256	243610	62,262,000
Germany	233	357021	81,799,600
Liechtenstein	205	160	32,842
Italy	192	301230	59,715,625
Switzerland	177	41290	7,301,994
Luxembourg	173	2586	512,000

```
dt = np.dtype([('country', 'S20'), ('density',
'i4'), ('area', 'i4'), ('population', 'i4')])
```

- This is just like an array of records or structs. The column names are like fieldnames.

```
struct entry {
    char  country[20];
    int   density;
    int   area;
    int   population;
 };
```

```
population_table = np.array([
    ('Netherlands', 393, 41526, 16928800),
    ('Belgium', 337, 30510, 11007020),
    ('United Kingdom', 256, 243610, 62262000),
    ('Germany', 233, 357021, 81799600),
    ('Liechtenstein', 205, 160, 32842),
    ('Italy', 192, 301230, 59715625),
    ('Switzerland', 177, 41290, 7301994),
    ('Luxembourg', 173, 2586, 512000)],
    dtype=dt)
```

```
population_table[:4] =>
array([(b'Netherlands', 393,  41526, 16928800),
       (b'Belgium', 337,  30510, 11007020),
       (b'United Kingdom', 256, 243610, 62262000),
       (b'Germany', 233, 357021, 81799600)],
      dtype=[('country', 'S20'), ('density',
'<i4'), ('area', '<i4'), ('population', '<i4')])
```

```
population_table['density'] =>
array([393, 337, 256, 233, 205, 192, 177,
173], dtype=int32)
population_table['country'] =>
array([b'Netherlands', b'Belgium', b'United
Kingdom', b'Germany',
       b'Liechtenstein', b'Italy',
b'Switzerland', b'Luxembourg'], dtype='|S20')
population_table['area'][2:5] =>
array([243610, 357021,    160], dtype=int32)
```

Numerical operations on numpy arrays
https://www.python-course.eu/numpy_numerical_operations_on_numpy_arrays.php

```
narr = np.array([2,3, 7.9, 3.3, 6.9, 0.11, 10.3, 12.9])
narr = narr + 2
narr => array([ 4., 5., 9.9 , 5.3, 8.9, 2.11, 12.3 ,
14.9])

A = np.array([ [11, 12, 13], [21, 22, 23], [31, 32, 33] ])
B = np.ones((3,3))
B => array([[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]])
A * (B + 1) =>     #This is not matrix multiplication
     array([[22., 24., 26.],
            [42., 44., 46.],
            [62., 64., 66.]])

A = np.array([[1,2,3]])
A.shape => (1, 3)

B =np.array([[1],[2],[3]])
B.shape => (3, 1)

np.dot(B, A) =>    #This is matrix multiplication
array([[1, 2, 3],
       [2, 4, 6],
       [3, 6, 9]])
np.dot(A, B) => array([[14]])
```

Broadcasting

https://www.python-course.eu/numpy_numerical_operations_on_numpy_arrays.php#Broadcasting

https://jakevdp.github.io/PythonDataScienceHandbook/02.05-computation-on-arrays-broadcasting.html

```
A = np.array([[11, 12, 13],[21, 22, 23],[31, 32, 33]])
B = np.array([1, 2, 3])
print(A * B) =>
    [[11 24 39]
     [21 44 69]
     [31 64 99]]
```

Here is the idea:

`A.shape` is `(3,3)`

`B.shape` is `(3)`

First convert `B` to have shape `(1,3)`:

We get `[[1, 2, 3]]`

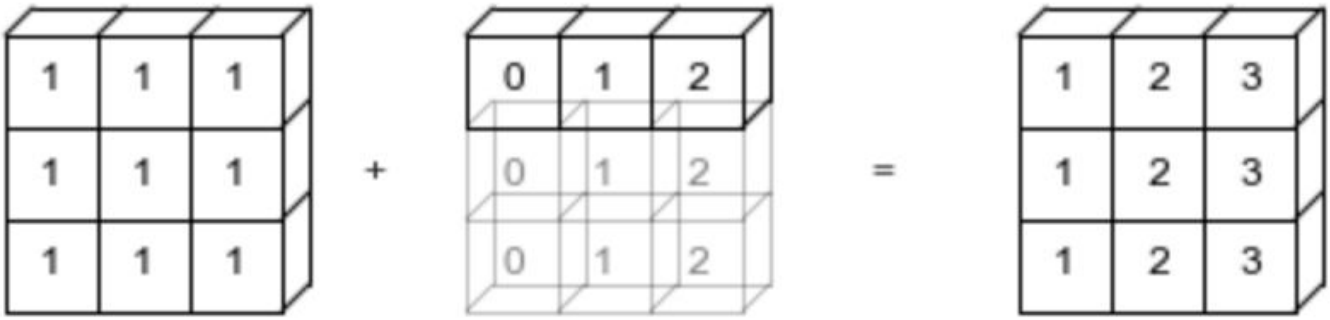
Then convert `(1,3)` to `(3,3)` by replication

We get `[[1, 2, 3],[1, 2, 3],[1, 2, 3]]`

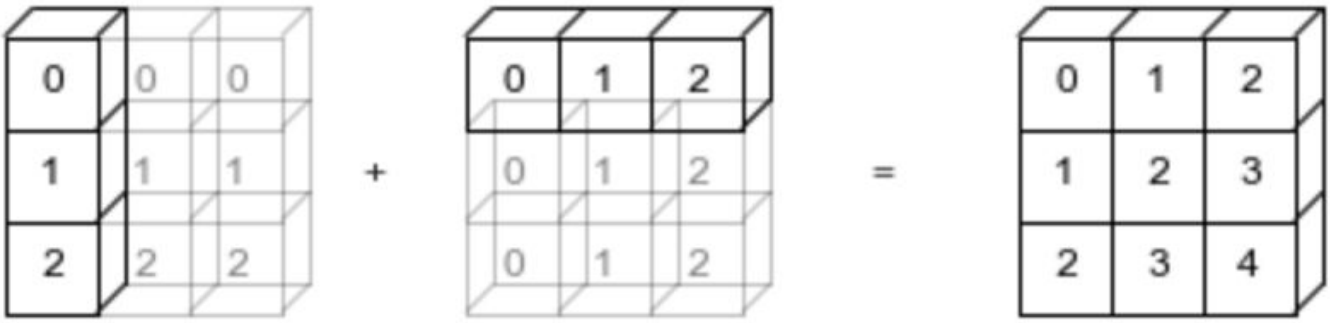
Now do element wise multiplication.

`np.arange(3)+5`

`np.ones((3,3))+np.arange(3)`



`np.arange(3).reshape((3,1))+np.arange(3)`



Array indexing

https://docs.scipy.org/doc/numpy-1.10.1/user/basics.indexing.html
https://www.pythonlikeyoumeanit.com/Module3_IntroducingNumpy/AdvancedIndexing.html#:~:text=Boolean%2DArray%20Indexing,-NumPy%20also%20permits&text=In%20its%20simplest%20form%2C%20boolean,ind%20using%20row%2Dmajor%20ordering.

1. **Simple indexing: Indexing using numbers, which can also be negative.**

```
x = np.arange(10)
x.shape = (2,5)
x => array([[0, 1, 2, 3, 4],
            [5, 6, 7, 8, 9]])

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

2. **Slicing and striding:**

```
x = np.arange(10)
x[2:5] => array([2, 3, 4])
x[:-7] => array([0, 1, 2])
x[1:7:2] => array([1, 3, 5])   #2 is the stride

y = np.arange(35).reshape(5,7)
y =>
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]])
```

```
y[1:5:2,::3] =>                    #2 and 3 are strides
array([[ 7, 10, 13],
       [21, 24, 27]])
```

Slices of arrays do not copy the internal array data but also produce new views of the original data.

3. **Index arrays:**

```
x = np.arange(10,1,-1)
x => array([10, 9, 8, 7, 6, 5, 4, 3, 2])
x[np.array([3, 3, 1, 8])] => array([7, 7, 9, 2])
x[np.array([[1,1],[2,3]])] =>
   array([[9, 9],
          [8, 7]])
```

Returns an array with the same shape as the index array, but with the values of the array being indexed. Returns a copy of the array unlike slices.

4. **Indexing Multi-dimensional arrays**

```
y => 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]])

y[np.array([0,2,4]), np.array([0,1,2])] =>
  array([ 0, 15, 30])
```

`[0,2,4]` selects the rows 0th, 2nd and the 4th rows, and `[0,1,2]` selects the 0th, the 1st and the second columns from these rows. In effect selects (0,0), (2,1), (4,2)

```
y[np.array([0,2,4]), 1] =>  array([ 1, 15, 29])
```

because of broadcasting.

```
y[np.array([0,2,4])] =>
array([[ 0,  1,  2,  3,  4,  5,  6],
       [14, 15, 16, 17, 18, 19, 20],
       [28, 29, 30, 31, 32, 33, 34]])
```

```
x = np.array([[ 0.58,  0.05,  0.84,  0.21],
              [ 0.88,  0.98,  0.45,  0.13],
              [ 0.1 ,  0.52,  0.58,  0.38],
              [ 0.84,  0.76,  0.25,  0.07]])
```

to make the diagonals of `x 0,` we do:

```
x[np.arange(4), np.arange(4)] = np.zeroes(4)
```

5. Boolean Masks

```
y =>  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]])
```

```
b = y % 3 == 0
b =>
```

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

```
y[b] =>
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33])
```

How is this obtained? By indexing `y` with `np.where(b)`.

```
np.where(b) =>
    (array([0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 4, 4]),
     array([0, 3, 6, 2, 5, 1, 4, 0, 3, 6, 2, 5]))
```

Altering the array:

```
y[y % 3 == 0] = 0
y => array([[ 0,  1,  2,  0,  4,  5,  0],
            [ 7,  8,  0, 10, 11,  0, 13],
            [14,  0, 16, 17,  0, 19, 20],
            [ 0, 22, 23,  0, 25, 26,  0],
            [28, 29,  0, 31, 32,  0, 34]])
```

This is the same as `y[np.where(b)] = 0`

6. newaxis() and ellipsis …

```
x = np.arange(10).reshape(2,5)

x[:,np.newaxis,:] =>
array([[[0, 1, 2, 3, 4]],
       [[5, 6, 7, 8, 9]]])
```

```
x = np.arange(5)
x[:,np.newaxis] + x[np.newaxis,:] =>
array([[0, 1, 2, 3, 4],
       [1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [3, 4, 5, 6, 7],
       [4, 5, 6, 7, 8]])

z = np.arange(81).reshape(3,3,3,3)
z[1,...,2]  # equivalent to z[1,:, :, 2]
array([[29, 32, 35],
       [38, 41, 44],
       [47, 50, 53]])
```

Sudoku validation

```
x = np.array([
    [1,2,6,4,3,7,9,5,8],
    [8,9,5,6,2,1,4,7,3],
    [3,7,4,9,8,5,1,2,6],
    [4,5,7,1,9,3,8,6,2],
    [9,8,3,2,4,6,5,1,7],
    [6,1,2,5,7,8,3,9,4],
    [2,6,9,3,1,4,7,8,5],
    [5,4,8,7,6,9,2,3,1],
    [7,3,1,8,5,2,6,4,9],
], dtype=np.int8)
```

```
# Boolean arrays
# Each array functions as a mask, denoting the presence of
an element
# eg for y1, if ith element of a row is 0, this element is
missing in the ith row of x
# for y2, if ith element of a row is 0, this element is
missing in ith column of x
y1 = np.zeros((9,9), dtype=np.int8)
y2 = np.zeros((9,9), dtype=np.int8)
y3 = np.zeros((9,9), dtype=np.int8)


for i in range(9):
    y1[i,x[i]-1]   = 1    # Index into ith row of y1
using ith row of x
    y2[i,x[:,i]-1] = 1    # Index into ith row of y2
using ith column of x

for i in range(3):
    for j in range(3):
     t1 =  x[ 3*i : 3*i+3 , 3*j : 3*j+3].reshape(-1)
     y3[i*3+j, t1-1] = 1

# find the number of locations where the elements are not
1 (actually 0)
# if this number exceeds 0, duplicates exist
if (y1!=1).sum() or (y2!=1).sum() or (y3!=1).sum():
    print("Duplicates exist")
else:
    print("No duplicates exist")
```

Scipy Links:
https://realpython.com/python-scipy-cluster-optimize/#minimizing-a-function-with-one-variable
https://realpython.com/python-scipy-cluster-optimize/#minimizing-a-function-with-many-variables
https://realpython.com/linear-programming-python/#using-scipy
https://docs.scipy.org/doc/scipy/reference/stats.html#module-scipy.stats
https://blog.eduonix.com/artificial-intelligence/scientific-python-scipy-machine-learning/
https://www.nature.com/articles/s41592-019-0686-2#Sec12
https://docs.scipy.org/doc/scipy/reference/spatial.html#module-scipy.spatial

i. https://meet.google.com/linkredirect?authuser=0&dest=https%3A%2F%2Fwww.geeksforgeeks.org%2Fpython-lists-vs-numpy-arrays%2F
ii. https://meet.google.com/linkredirect?authuser=0&dest=https%3A%2F%2Fwebcourses.ucf.edu%2Fcourses%2F1249560%2Fpages%2Fpython-lists-vs-numpy-arrays-what-is-the-difference%23%3A~%3Atext%3DA%2520numpy%25

20array%2520is%2520a%2CPython%2520core%2520library%2520provided%2520Lists

iii. Vectorization and locality

2. API for numpy and scipy are similar.
    a. Special functions (`scipy.special`)
    b. Integration (`scipy.integrate`)
    c. Optimization (`scipy.optimize`)
    d. Interpolation (`scipy.interpolate`)
    e. Fourier Transforms (`scipy.fftpack`)
    f. Signal Processing (`scipy.signal`)
    g. Linear Algebra (`scipy.linalg`)
    h. Sparse Eigenvalue Problems with ARPACK
    i. Compressed Sparse Graph Routines (`scipy.sparse.csgraph`)
    j. Spatial data structures and algorithms (`scipy.spatial`)
    k. Statistics (`scipy.stats`)
    l. Multidimensional image processing (`scipy.ndimage`)
    m. File IO (`scipy.io`)

LInks :You3:03 PM
https://docs.google.com/document/d/1onic05xQ_m67JXSzLy0Lf513q3Nx0I2fAws4_YcKLv4/edit?usp=sharing

Vipin Mahawar3:19 PM
https://www.geeksforgeeks.org/python-lists-vs-numpy-arrays/

Shashank shet3:22 PM
https://stackoverflow.com/questions/111983/python-array-versus-numpy-array

Vipin Mahawar3:34 PM
https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference#:~:text=A%20numpy%20array%20is%20a,Python%20core%20library%20provided%20Lists.

Shashank shet3:36 PM

https://stackoverflow.com/questions/8385602/why-are-numpy-arrays-so-fast

Vipin Mahawar3:37 PM
https://numpy.org/
Shashank shet3:40 PM

https://github.com/numpy/numpy/blob/master/numpy/core/src/umath/simd.inc.src
Shashank shet3:52 PM
https://www.quora.com/What-is-the-difference-between-NumPy-and-SciPy?share=1
Vipin Mahawar3:53 PM
https://www.edureka.co/blog/scipy-

https://matplotlib.org/3.2.0/tutorials/index.html
Vipin Mahawar3:58 PM
https://www.educative.io/edpresso/what-is-matplotlib

Vipin Mahawar3:53 PM
https://www.edureka.co/blog/scipy-tutorial/#:~:text=SciPy%20is%20an%20open%2Dsource,range%20of%20high%2Dlevel%20commands.

Shashank shet3:54 PM
https://scipy.org/scipylib/faq.html

https://scipy.org/scipylib/faq.html#id15

Shashank shet3:57 PM
https://matplotlib.org/3.2.0/tutorials/index.html

Vipin Mahawar3:58 PM
https://www.educative.io/edpresso/what-is-matplotlib

Shashank shet4:00 PM
https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/index.html

Vipin Mahawar4:01 PM
https://pandas.pydata.org/
https://www.scipy.org/

Vipin Mahawar4:02 PM
https://www.learnpython.org/en/Pandas_Basics

Shashank shet4:04 PM

https://pandas.pydata.org/docs/user_guide/basics.html#basics

https://docs.google.com/document/d/1onic05xQ_m67JXSzLy0Lf513q3Nx0l2fAws4_YcKLv4/edit?usp=sharing

Shashank shet3:21 PM

https://docs.python.org/3.8/library/sys.html#sys.getsizeof

Vipin Mahawar3:30 PM

https://www.edureka.co/blog/arrays-in-python/

Shashank shet3:32 PM

https://docs.python.org/2/library/array.html

Shashank shet3:36 PM

https://pythonspeed.com/articles/python-integers-memory/

Shashank shet3:40 PM

https://stackoverflow.com/questions/14329794/get-size-of-integer-in-python

int vs int_object

Second answer

Vipin Mahawar8:59 PM
https://docs.google.com/document/d/1GFDCnLFCJPNrsXt-C20np4TBqDbsUssinlmedLaJpC8/edit?usp=sharing
Vipin Mahawar9:01 PM
https://docs.google.com/document/d/1GFDCnLFCJPNrsXt-C20np4TBqDbsUssinlmedLaJpC8/edit?usp=sharing
Vipin Mahawar9:08 PM
https://matplotlib.org/gallery/statistics/hist.html#sphx-glr-gallery-statistics-hist-py