Numpy

import numpy as np

1) Numpy array

```
#Creating array
np.array(list)  or np.array(tuple)

For multidimensional array creation pass nested list/tuples.
```

2) Attributes

```
# Dimensions
arr.ndim

# Shape
arr.shape

# Size (Number of elements)
arr.size

# Datatype
arr.dtype

# Size of an item (in bytes)
arr.itemsize

# Size of array  (in bytes)
arr.nbytes
```

3) Different types of arrays

```
# Array of zeros
np.zeros(shape)  #shape should be passed in tuple if more than 1D is needed

# Array of ones
np.ones(shape)

# Identity matrix
np.identity(val) or np.eye(val)   # The identity matrix is only possible when it is square matrix.
                                    so, val means, it has a shape of (val, val)

#  Creates an array of random values between 0 and 1.
np.random.random(shape)

# Random Integer values between given range i.e between start and end
# end index exclusive
np.random.randint(start, end, size=shape)
```

```
# Creating a sequential array
np.arange(start, end, step)  #end is exclusive

# Creating event spaced array
np.linspace(start, stop, num)    # it divides the interval between start and stop into num
equally
                                        spaced values and returns an array containing those values.
```

4) Creating deep copy

```
# Assigning reference
arr1 = arr                  # Here, changes in arr1, reflect in arr

# Creating copy
arr1 = arr.copy()           # Here, changes in arr1, do not reflect in arr
```

5) Mathematical operations

```
# Addition
arr + value  # value is added to each element of arr |||ly for others

# Subtraction
arr - value

# Multiplication
arr * value

# Division
arr / value

# Modulo
arr % value

# Power
arr ** value
```

6) Arithmetic operations

```
# Addition
np.add(arr1, arr2)      or      arr1 + arr2

# Subtraction
np.subtract(arr1, arr2)      or      arr1 - arr2

# Multiplication
np.multiply(arr1, arr2)      or      arr1 * arr2
```

Division
np.divide(arr1, arr2) or arr1 / arr2

Modulus
np.mod(arr1, arr2) or arr1 % arr2

Note:
The above operations can be performed on any number of arrays.
Ex: arr1 + arr2 + arr3 + arr4

Power
np.power(arr, val) or arr ** val

# Square root
np.sqrt(arr)

# Cube root
np.cbrt(arr)

# Absolute value
np.abs(arr)

# Sign inversion (+ to -, - to +)
np.negative(arr)

7) Trignometric operations

np.sin(arr) : Sine

np.cos(arr) : Cosine

np.tan(arr) : Tangent

np.arcsin(arr) : Inverse sine

np.arccos(arr) : Inverse cosine

np.arctan(arr) : Inverse tangent

8) Exponential and logarithmic functions

np.exp(arr) : Exponential *i.e* e^x

np.log(arr) : Natural log *i.e* log base e

np.log10(arr) : Log base 10

np.log2(arr) : Log base 2

9) Rounding and floor/ceiling functions

np.round(arr) : Round to nearest integer
np.round(arr, val) : Round to specified number of decimals
np.ceil(arr) : Ceil
np.floor(arr) : Floor
np.trunc(arr) : Truncate decimals

10) Statistical functions

np.mean(arr) : Mean
np.median(arr) : Median
np.std(arr) : Standard deviation
np.var(arr) : Variance
np.max(arr) : Max value
np.min(arr) : Min value

11) Bitwise operations

np.bitwise_and(arr1, arr2) : Bitwise AND
np.bitwise_or(arr1, arr2) : Bitwise OR
np.bitwise_xor(arr1, arr2) : Bitwise XOR
np.bitwise_not(arr1) : Bitwise NOT

12) Comparison functions

#Elementwise comparison is performed in arr1 and arr2 and a boolean array is returned.

np.equal(arr1, arr2)
np.not_equal(arr1, arr2)
np.less(arr1, arr2)
np.less_equal(arr1, arr2)
np.greater(arr1, arr2)
np.greater_equal(arr1, arr2)
np.logical_and(arr1, arr2)
np.logical_or(arr1, arr2)
np.logical_xor(arr1, arr2)
np.logical_not(arr1)

13) Linear Algebra functions

```
arr1 @ arr2   or np.matmul(arr1, arr2)  #MAtrix multiplication
np.dot(arr1, arr2)     # Dot product
np.cross(arr1, arr2)  # Cross product , 1D-shape: (3,) , 2D-shape: (N, 3)
np.linalg.det(arr)       # Determinant, where arr is a square matrix
np.linalg.eigvals(arr)  # Eigenvalue
np.linalg.inv(arr)       # Inverse
np.linalg.svd(arr)       # Singular Value Decomposition
```

14) Reorganizing arrays

```
arr.reshape(shape)  #Reshapes and returns reshaped array.
                    # shape is not a tuple. Just mention shape without enclosing it in tuple
                    # shape should be multiple of size (i.e number of elements)

arr.resize(shape)     #Same as reshape, except that it reshapes original array itself.

arr.flatten()          # Converts multidimensional array to 1D array. Doesn't modify
                         array

np.transpose(arr)   #Transpose

arr.T                  # Transpose
```

15) Sorting

```
# sort() function is used to sort elements in ascending order

np.sort(arr)

np.sort(arr)[::-1]   # sort in descending order
```

16) Concatenating and stacking

```
np.concatenate((arr1, arr2, arr3)) : join arrays along an existing axis.

np.stack((arr1, arr2, arr3))          : join arrays along a new axis.

np.hstack((arr1, arr2, arr3))         :  stack arrays horizontally, column-wise.

np.vstack((arr1, arr2, arr3))         : stack arrays vertically, row-wise.
```

17) Load data from a file

```
data = np.genfromtxt(path, delimiter=',')
```

18) Type conversion

```
# Changing datatype while creation
arr = np.array([34, 56, 78], dtype='int8')   #int8, int16, int32 (default), int64

# Changing datatype after creation
arr.astype(datatype)

# Converting ndarray to list
li =arr.tolist()
```

19) Condition Based Retrieval

```
arr[arr > 10]        # return an arry containin elements less than 10

arr < 30             # returns a boolean array
```

20) Insertion/Deletion

```
# insert single data
np.insert(arr, index, value)

# insert multiple data
np.insert(arr, index,list)     # list = [value1, value2, value3, …]

# delete single data
np.insert(arr, index)

# delete multiple data
np.insert(arr, list)     # list = [index1, index2, index3, …]
```

21) Retrieving information

```
np.info(operation)      # Gives help/info about that operation

Ex: np.info(np.delete)
```

22) Iterating arrays

```
np.nditer(arr)      # Gives help/info about that operation

for x in np.nditer(arr):
        print(x)
```

23) Splitting arrays

```
np.array_split(arr, num)      # Divides arr into num subarrays
```