Array: Collection of element with same data type

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

- **Signed Integer (#i):** Represents positive and negative whole numbers in an array, like temperatures or scores etc. [-10, 20, 30].
- **Unsigned Integer (#l):** Stores non-negative whole numbers in an array, such as counts or indices [0, 1, 2, 3].
- **Float (#f):** Holds decimal numbers in an array, commonly used for precise measurements [1.5, 3.14, 2.718].
- **Unicode Character (#u):** Represents textual data in an array, typically used for storing strings of text ["Hello", "World"].
- **'d':** The type code 'd' specifies that the array will contain double-precision floating-point numbers.

Type Code	C Type	Python Type	Minimum Size (Bytes)
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	Py_UNICODE	Unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	int	2
'l'	signed long	int	4
'L'	unsigned long	int	4
'q'	signed long long	int	8
'Q'	unsigned long long	int	8
'f'	float	float	4
'd'	double	float	8

In [2]:
```
num=[1,2,3,4,5,6]
arr1=np.array(num)
arr=ar.array("i",num)
print(arr1,'\n',arr)
```

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

In [3]:
```
print(type(arr1),type(arr))
```

```
<class 'numpy.ndarray'> <class 'array.array'>
```

In [4]:
```
one_d=np.array([1,2,3,4])
two_d=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(one_d,"\nDim:",one_d.ndim)
print(two_d,"\nDim:",two_d.ndim)
```

```
[1 2 3 4]
Dim: 1
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Dim: 2
```

In [5]:
```
print(two_d[0,2])
print(two_d[2,-1])
print(two_d[-1,-1])
```

```
3
9
9
```

In [6]:
```
o_d=np.array([1,2,3,4,5,6,7,8,9,10])
print("Every 3rd element:",o_d[::3],"\nFrom the 5th element:",o_d[4:],
      "\nLast 3 element:",o_d[-3:],"\nTill 5th element:",o_d[:5])
```

```
Every 3rd element: [ 1  4  7 10]
From the 5th element: [ 5  6  7  8  9 10]
Last 3 element: [ 8  9 10]
Till 5th element: [1 2 3 4 5]
```

In [7]:
```
t_d=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(t_d,'\n','\n Whole Matrix ":" and Element upto 0th column: ','\n',t_d[:,0:1],'\n M
atrix starting from 1st row "1:" and Element upto 1st column: ','\n',t_d[1:,:2],'\n Whol
e Matrix ":" and Element from 2nd column to 3rd column excluding 3rd column: ','\n',t_d
[:,2:3])
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

 Whole Matrix ":" and Element upto 0th column:
 [[1]
 [4]
 [7]]
 Matrix starting from 1st row "1:" and Element upto 1st column:
 [[4 5]
 [7 8]]
 Whole Matrix ":" and Element from 2nd column to 3rd column excluding 3rd column:
 [[3]
 [6]
 [9]]
```

In [8]:
```
# Printing each element of the 1-D array
for i in o_d:
    print(i,end=" ")
```

```
1 2 3 4 5 6 7 8 9 10
```

In [9]:
```
# Printing 2-D matrix
for i in t_d:
    print(i)
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

In [10]:
```
# Printing each element of the 2-D array row-wise
for row in t_d:
    for i in row:
        print(i,end=" ")
```

```
1 2 3 4 5 6 7 8 9
```

In [11]:
```
# Concatenating two arrays
print(o_d)
new_arr=np.array([11,12,13,14,15])
np.concatenate((o_d,new_arr))
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

Out[11]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [12]:
```
# hstack : example of horizontal stacking of array
a=np.hstack((o_d,new_arr))
print(a)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```

In [13]:
```
# vstack :example of vertical stacking of array resulting in 1-D to 2-D array
oe_d=np.array([1,2,3,4,5])
print(oe_d)
print('Shape array: ',oe_d.shape)
b=np.vstack((oe_d,new_arr))  #when columns are same we can do vstack
print(b)
print('Shape array after vertical stacking: ',b.shape)
```

```
[1 2 3 4 5]
Shape array:  (5,)
[[ 1  2  3  4  5]
 [11 12 13 14 15]]
Shape array after vertical stacking:  (2, 5)
```

In [14]:
```
# use of np.where function
print(o_d)
print('Showing numbers from array which are divisble by 3: ',o_d[np.where(o_d%3==0)])
```

```
[ 1  2  3  4  5  6  7  8  9 10]
Showing numbers from array which are divisble by 3:  [3 6 9]
```

In [15]:
```
# Reversing the given array
print(o_d)
print('Reversing the given array: ',o_d[::-1])
```

```
[ 1  2  3  4  5  6  7  8  9 10]
Reversing the given array:  [10  9  8  7  6  5  4  3  2  1]
```

In [16]:
```
# Will delete the element at index 4 i.e. 5 here as indexing starts from 0 in array
np.delete(o_d,4)
```

Out[16]:
```
array([ 1,  2,  3,  4,  6,  7,  8,  9, 10])
```

In [17]:
```
print(t_d.ndim)
print(t_d.shape)
```

```
2
(3, 3)
```

Here's a comparison of `.ndim` and `.shape` in tabular form:

Attribute	Purpose	Example Usage
`.ndim`	Returns the number of dimensions of the array	`arr.ndim` (1 for a 1D array, 2 for 2D, etc.)
`.shape`	Returns a tuple representing the size of each dimension	`arr.shape` (e.g., `(3,)` for 1D, `(2, 3)` for 2D)

These attributes are both useful for understanding the structure and dimensionality of NumPy arrays.

In [18]:
```
#Iterates over each element in 1-D array and returns last stored number i.e. 10
lent=0
print(o_d)
for i in range(len(o_d)):
#     print(o_d[i],end='  ')
    lent+=1
#     print(lent)
lent
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

Out[18]:
```
10
```

In [19]:
```
st=[ar.array("u",s) for s in ["yash","jai","yogesh","rudra"]]
st
```

Out[19]:
```
[array('u', 'yash'),
 array('u', 'jai'),
 array('u', 'yogesh'),
 array('u', 'rudra')]
```

In [20]:
```
st=np.array(["yash","jai","yogesh","rudra"])
s=np.sort(st)
#Sorted the numpy array in alphabetical order
print(type(s))
print(st,"\n",s)
```

```
<class 'numpy.ndarray'>
['yash' 'jai' 'yogesh' 'rudra']
 ['jai' 'rudra' 'yash' 'yogesh']
```

In [21]:
```
d=np.array([1,2,3,5,1,2,5,6,2])
q=np.sort(d)
#Sorted the numpy array in numerical order
print(type(q))
print(d,"\n",q)
```

```
<class 'numpy.ndarray'>
[1 2 3 5 1 2 5 6 2]
 [1 1 2 2 2 3 5 5 6]
```

In [22]:
```
to_d=np.array([[11,2,3],[17,8,9],[4,15,6]])
print(to_d,"\nIt does sorting row-wise: ",'\n',np.sort(to_d))
```

```
[[11  2  3]
 [17  8  9]
 [ 4 15  6]]
It does sorting row-wise:
 [[ 2  3 11]
 [ 8  9 17]
 [ 4  6 15]]
```

In [23]:
```
print(o_d)
print('Even number in numpy array: ',o_d[np.where(o_d%2==0)])
print('Odd number in numpy array: ',o_d[np.where(o_d%2!=0)])
```

```
[ 1  2  3  4  5  6  7  8  9 10]
Even number in numpy array:  [ 2  4  6  8 10]
Odd number in numpy array:  [1 3 5 7 9]
```

In [24]:
```
print(two_d)
print()
print(to_d)
t=two_d+to_d
print()
print(t)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[11  2  3]
 [17  8  9]
 [ 4 15  6]]

[[12  4  6]
 [21 13 15]
 [11 23 15]]
```

In [25]:
```
print(o_d)
o_d.max()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

Out[25]: 10

In [26]:
```
q=np.array([1,2,3,4,5])
y=(lambda x:x*x)
s=y(q)
s
```

Out[26]:
```
array([ 1,  4,  9, 16, 25])
```

In [27]:
```
np.array(list(map(lambda x:x*x,q)))
```

Out[27]:
```
array([ 1,  4,  9, 16, 25])
```

In [28]:
```
np.vectorize(lambda x:x*x)(q)
```

Out[28]:
```
array([ 1,  4,  9, 16, 25])
```

Insert & Append

In [38]:
```
a = ar.array('i', [1, 2, 3])
print("Array before insertion : ", end=" ")
for i in range(0, 3):
    print(a[i], end=" ")
print()
a.insert(1, 4)
print("Array after insertion : ", end=" ")
for i in (a):
    print(i, end=" ")
print()

b = ar.array('d', [2.5, 3.2, 3.3])
print("Array before insertion : ", end=" ")
for i in range(0, 3):
    print(b[i], end=" ")
print()
b.append(4.4)
print("Array after insertion : ", end=" ")
for i in (b):
    print(i, end=" ")
print()
```

```
Array before insertion :  1 2 3
Array after insertion :  1 4 2 3
Array before insertion :  2.5 3.2 3.3
Array after insertion :  2.5 3.2 3.3 4.4
```

- append() is also used to add the value mentioned in its arguments at the end of the Python array.
- insert() is used to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array.

Accessing the Elements from the Array

In [42]:
```
a = ar.array('i', [1, 2, 3, 4, 5, 6])
print("Access element is: ", a[0])
print("Access element is: ", a[3])
b = ar.array('d', [2.5, 3.2, 3.3])
print("Access element is: ", b[1])
print("Access element is: ", b[2])
```

```
Access element is:  1
Access element is:  4
Access element is:  3.2
Access element is:  3.3
```

In [78]:
```
arr = ar.array('i', [1, 2, 3, 1, 5])
print("The new created array is : ", end="")
for i in range(len(arr)):
    print(arr[i], end=" ")

print("\r")
print("The popped element is : ", end="")
print(arr.pop(2))  # removes element of the 2nd index
print("The array after popping is : ", end="")
for i in range(len(arr)):
    print(arr[i], end=" ")

print('\r')
print("The popped element is : ", end="")
print(arr.pop())  # removes  last element
print("The array after popping last element is : ",*arr, end="")



print("\r")
arr.remove(1) # removes 1 from the array which is at 0th index
print("The array after removing is : ", end="")
for i in range(len(arr)):
    print(arr[i], end=" ")
```

```
The new created array is : 1 2 3 1 5
The popped element is : 3
The array after popping is : 1 2 1 5
The popped element is : 5
The array after popping last element is :  1 2 1
The array after removing is : 2 1
```

In [49]:
```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

a = ar.array('i', l)
print("Initial Array: ")
for i in (a):
    print(i, end=" ")
Sliced_array = a[3:8]
print("\nSlicing elements in a range 3-8: ")
print(Sliced_array)
Sliced_array = a[5:]
print("\nElements sliced from 5th "
      "element till the end: ")
print(Sliced_array)
Sliced_array = a[:]
print("\nPrinting all elements using slice operation: ")
print(Sliced_array)
```

```
Initial Array:
1 2 3 4 5 6 7 8 9 10
Slicing elements in a range 3-8:
array('i', [4, 5, 6, 7, 8])

Elements sliced from 5th element till the end:
array('i', [6, 7, 8, 9, 10])

Printing all elements using slice operation:
array('i', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

In [50]:
```
arr = ar.array('i', [1, 2, 3, 1, 2, 5])
print("The new created array is : ", end="")
for i in range(0, 6):
    print(arr[i], end=" ")

print("\r")
print("The index of 1st occurrence of 2 is : ", end="")
print(arr.index(2))
print("The index of 1st occurrence of 1 is : ", end="")
print(arr.index(1))
```

```
The new created array is : 1 2 3 1 2 5
The index of 1st occurrence of 2 is : 1
The index of 1st occurrence of 1 is : 0
```

Updating Elements in a Array

In [51]:
```
arr = ar.array('i', [1, 2, 3, 1, 2, 5])
print("Array before updation : ", end="")
for i in range(0, 6):
    print(arr[i], end=" ")

print("\r")
arr[2] = 6
print("Array after updation : ", end="")
for i in range(0, 6):
    print(arr[i], end=" ")
print()
arr[4] = 8
print("Array after updation : ", end="")
for i in range(0, 6):
    print(arr[i], end=" ")
```

```
Array before updation : 1 2 3 1 2 5
Array after updation : 1 2 6 1 2 5
Array after updation : 1 2 6 1 8 5
```

Counting Elements in a Array

In [52]:
```
my_array = ar.array('i', [1, 2, 3, 4, 2, 5, 2])
count = my_array.count(2)
print("Number of occurrences of 2:", count)
```

```
Number of occurrences of 2: 3
```

In [57]:
```
my_array = ar.array('i', [1, 2, 3, 4, 5])
print("Original array:", *my_array)
my_array.reverse()
print("Reversed array:", *my_array)
```

```
Original array: 1 2 3 4 5
Reversed array: 5 4 3 2 1
```

The capability of extending an array to include additional elements.

In [59]:
```
a = ar.array('i', [1, 2, 3,4,5])
print("The before array extend  : ", end =" ")
for i in range (0, 5):

    print (a[i], end =" ")

print()
a.extend([6,7,8,9,10])
print("\nThe array after extend :",end=" ")

for i in range(0,10):

    print(a[i],end=" ")

print()
```

```
The before array extend  :  1 2 3 4 5

The array after extend : 1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

In [61]:
```
a=ar.array('i',[1,2,3,4,5,6])
print("The Before extend array is :",end=" ")
for i in range(0,6):

    print(a[i],end=" ")

print()
a.extend([7,8,9,10,11,12])
print("\nThe After extend array is :",end=" ")

for i in range(0,12):

    print(a[i],end=" ")

print()
b = ar.array('d', [2.1,2.2,2.3,2.4,2.5,2.6])

print("\nThe before extend array is :",end=" ")

for i in range(0,6):

  print(b[i],end=" ")
print()
b.extend([2.6,2.7,2.8,2.9])

print("\nThe after extend array is :",end=" ")

for i in range(0,9+1):
    print(b[i],end=" ")
print()
```

```
The Before extend array is : 1 2 3 4 5 6

The After extend array is : 1 2 3 4 5 6 7 8 9 10 11 12

The before extend array is : 2.1 2.2 2.3 2.4 2.5 2.6

The after extend array is : 2.1 2.2 2.3 2.4 2.5 2.6 2.6 2.7 2.8 2.9
```

In [80]:
```
# initializing array with array values
arr = ar.array('i',[1, 2, 3, 1, 2, 5])
li = [1, 2, 3]

# using fromlist() to append list at end of array
arr.fromlist(li)

# printing the modified array
print ("The modified array is : ",*arr,end="")
```

```
The modified array is :  1 2 3 1 2 5 1 2 3
```

In [83]:
```
# initializing array with array values
arr = ar.array('i',[1, 2, 3, 1, 2, 5])

# using tolist() to convert array into list
li2 = arr.tolist()

# printing the new list
print ("The new list created is : ",*li2,end="")
```

```
The new list created is :  1 2 3 1 2 5
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: