Experiment No. 13

Aim: To implement programs based on NumPy library in Python.

Questions:

1. Implement a python program to create and print 1D, 2D and 3D array.

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a)
```

```
[1 2 3 4]
```

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

```
[[1 2 3]
 [4 5 6]]
```

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

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

2. Implement a python program to apply indexing and slicing operations on an array.

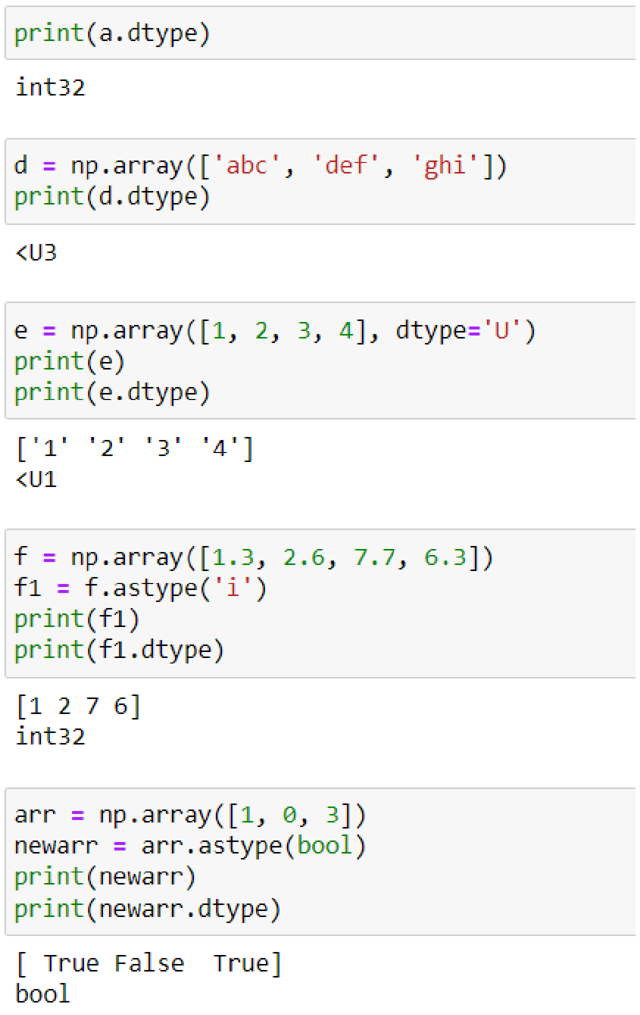
```
print(a[3])
print(b[1, 1])
print(c[0, 0, 1])
```

```
4
5
2
```

```
print(a[:3])
print(b[1, :])
print(c[1, 0, :])
```

```
[1 2 3]
[4 5 6]
[5 6]
```

3. Implement a python program to demonstrate NumPy data types.



```
print(a.dtype)
```

```
int32
```

```
d = np.array(['abc', 'def', 'ghi'])
print(d.dtype)
```

```
<U3
```

```
e = np.array([1, 2, 3, 4], dtype='U')
print(e)
print(e.dtype)
```

```
['1' '2' '3' '4']
<U1
```

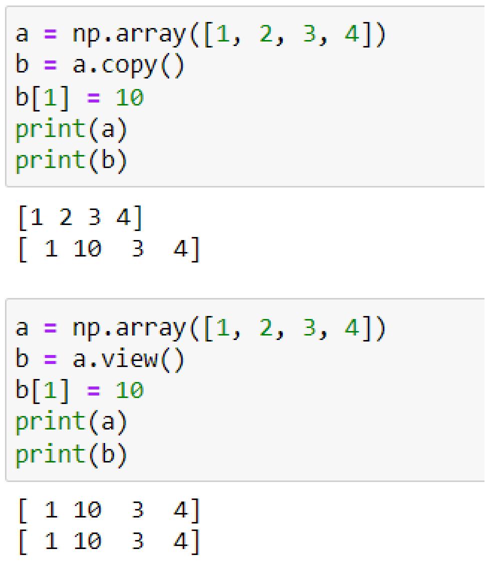
```
f = np.array([1.3, 2.6, 7.7, 6.3])
f1 = f.astype('i')
print(f1)
print(f1.dtype)
```

```
[1 2 7 6]
int32
```

```
arr = np.array([1, 0, 3])
newarr = arr.astype(bool)
print(newarr)
print(newarr.dtype)
```

```
[ True False  True]
bool
```

4. Implement a python program to demonstrate NumPy copy and View.



```
a = np.array([1, 2, 3, 4])
b = a.copy()
b[1] = 10
print(a)
print(b)
```

```
[1 2 3 4]
[ 1 10  3  4]
```

```
a = np.array([1, 2, 3, 4])
b = a.view()
b[1] = 10
print(a)
print(b)
```

```
[ 1 10  3  4]
[ 1 10  3  4]
```

5. Implement a python program to demonstrat   join, split, serach, sort and filter operations.

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

```
[1 2 3 4 5 6]
```

```
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)
```

```
(array([3, 5, 6], dtype=int64),)
```

```
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
```

```
[0 1 2 3]
```

```
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))
```

```
['apple' 'banana' 'cherry']
```

```
arr = np.array([41, 42, 43, 44])
x = [True, False, True, False]
newarr = arr[x]
print(newarr)
```

```
[41 43]
```

6. Implement a python program to demonstrate random number generation in NumPy.

```
from numpy import random
x = random.randint(100)
print(x)
```

```
16
```

```
x = random.rand()
print(x)
```

```
0.1299098318920241
```

```
x=random.randint(100, size=(5))
print(x)
```

```
[49 67 88 43  6]
```

```
x = random.rand(5)
print(x)
```

```
[0.05675683 0.90738527 0.29408829 0.04146051 0.73548929]
```

```
x = random.rand(3, 5)
print(x)
```

```
[[0.23278747 0.849614   0.27976407 0.16974571 0.86299871]
 [0.44444896 0.24605474 0.68902167 0.62613724 0.87142138]
 [0.07221896 0.64091061 0.41995085 0.06591627 0.80487996]]
```

7. Implement a python program to demonstrate use of ufuncs (Universal functions) in NumPy.

```
arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])
newarr = np.add(arr1, arr2)
print(newarr)
```

```
[30 32 34 36 38 40]
```

```
arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 5, 6, 8, 2, 33])
newarr = np.power(arr1, arr2)
print(newarr)
```

```
[      1000     3200000   729000000 -520093696        2500           0]
```

```
arr = np.array([-1, -2, 1, 2, 3, -4])
newarr = np.absolute(arr)
print(newarr)
```

```
[1 2 1 2 3 4]
```

8. Implement a python program to create user defined ufunc and use the same.

```
def myadd(x, y):
    return x+y
myadd = np.frompyfunc(myadd, 2, 1)
print(myadd([1, 2, 3, 4], [5, 6, 7, 8]))
```

```
[6 8 10 12]
```