```
import numpy as np
a=np.array([]) #Creating empty numpy array
print(a)
b=np.array(['a','b','c',"d"]) #creating numpy array with predefined data
print(b)
```

```
[]
['a' 'b' 'c' 'd']
```

```
#Checkerboard pattern

import numpy as np
x = np.ones((3, 3))
print("Checkerboard pattern:")
x = np.zeros((5, 5), dtype = int)
x[1::2, ::2] = 2
x[::2, 1::2] = 1
print(x)
```

```
Checkerboard pattern:
[[0 1 0 1 0]
 [2 0 2 0 2]
 [0 1 0 1 0]
 [2 0 2 0 2]
 [0 1 0 1 0]]
```

```
# Slicing numpy array
import numpy as np
arr = np.array([11, 12, 13, 14, 15, 16, 17])
print(arr[-3:-1]) #Negative Slicing

a = np.array([[11, 12, 13, 14, 15], [16, 17, 18, 19, 20]]) #a 2-D numpy array
print(a[0:2, 1:4]) #slice index 1 to index 4 (not included)
```

```
[15 16]
[[12 13 14]
 [17 18 19]]
```

```
#update values using slicing given an array value
import numpy as np
a = np.arange(0,36).reshape((6,6)).T; a[2,0] = 4; a[4,0] = 2;
print(a)
a[np.arange(a.shape[1])[None,:] >= a[:,0,None]] = 0
print()
print(a)
```

```
[[ 0  6 12 18 24 30]
 [ 1  7 13 19 25 31]
 [ 4  8 14 20 26 32]
```

```
 [ 3  9 15 21 27 33]
 [ 2 10 16 22 28 34]
 [ 5 11 17 23 29 35]]

[[ 0  0  0  0  0  0]
 [ 1  0  0  0  0  0]
 [ 4  8 14 20  0  0]
 [ 3  9 15  0  0  0]
 [ 2 10  0  0  0  0]
 [ 5 11 17 23 29  0]]
```

```
#Updating 2-D numpy array element
import numpy as np
a = np.array([[1, 1], [2 , 1]], dtype=np.float)
a[0, 0] = 1.5
print(a)
```

```
[[1.5 1. ]
 [2.  1. ]]
```

```
#Transpose numpy array
import numpy as np
a = np.arange(12).reshape(3,4)

print('The original array is:')
print(a)
print ('\n' )

print ('The transposed array is:' )
print (np.transpose(a))
```

```
The original array is:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]


The transposed array is:
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

```
#Shape manipulations in numpy array
import numpy as np
a = np.arange(8).reshape(2,4)

print ('The original array is:' )
print(a)
print ('\n')
```

```
print ('After applying ravel function:')
print (a.ravel())
```

```
The original array is:
[[0 1 2 3]
 [4 5 6 7]]


After applying ravel function:
[0 1 2 3 4 5 6 7]
```

```
#For loop in 2-D array
import numpy as np

arr = np.array([[11, 12, 13], [14, 15, 16]])

for x in arr:
  for y in x:
    print(y)
print(arr)

print()
# while loop in 2-D array
a = np.array([[10.8, 2.3, 3], [14, 15, 16]])

n=len(a)
i=0

while(i<n):
  j=0

  while(j<len(a[i])):
    print("index",i,j,':',a[i][j])
    j+=1
  i+=1
  print()
```

```
11
12
13
14
15
16
[[11 12 13]
 [14 15 16]]

index 0 0 : 10.8
index 0 1 : 2.3
index 0 2 : 3.0

index 1 0 : 14.0
index 1 1 : 15.0
```

```
index 1 2 : 16.0
```

```
#Reading files in numpy
array_from_file = np.genfromtxt("numpy.txt", dtype=str) #Used to load data from a text file,
print(array_from_file)

# skipping last line in the file
ab = np.genfromtxt("numpy.txt", dtype=str,
                   encoding=None, skip_footer=1)
print(ab)
```

```
['w,e,e,k,2' 'n,u,m,p,j']
w,e,e,k,2
```

```
import numpy as np

import time

# generating 1000 x 1000 matrices
np.random.seed(42)

x = np.random.randint(0, 256, size = (1000, 1000)).astype("float64")

y = np.random.randint(0, 256, size = (1000, 1000)).astype("float64")

#computing multiplication time on CPU
tic = time.time()

z = np.matmul(x, y)

toc = time.time()

time_taken = toc - tic #time in s

print("Time taken on CPU (in ms) = {}".format(time_taken * 1000))
```

```
Time taken on CPU (in ms) = 62.15476989746094
```

✓ 0s completed at 3:03 PM