```
import numpy as np
import numpy as norm

# 1. Create a null vector of size 10 but the fifth value which is 1.
x = np.zeros(10)
x[4] = 1
print(x)
```

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

```
# 2. Create a vector with values ranging from 10 to 49.
x = np.arange(10,49)
print(x)
```

```
[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 35 36 37 38 39 40 41 42 43 44 45 46 47 48]
```

```
# 3. Reverse a vector (first element becomes last).
x = x[::-1]
print(x)
```

```
[48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25
 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]
```

```
# 4. Create a 3x3 matrix with values ranging from 0 to 8.
x = np.arange(0,9).reshape(3,3)
print(x)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

```
# 5. Create a random vector of size 30 and find the mean value.
x = np.random.rand(30)
print(x)
print()
```

```
y = sum(x)/len(x)
print('mean: ', y)
```

```
[0.34004946 0.45239851 0.41414835 0.77796377 0.97859286 0.73383716
 0.1546413  0.78505175 0.91080031 0.89482882 0.60680746 0.39998293
 0.00828513 0.87772013 0.07545884 0.65225267 0.83683822 0.74863961
 0.22535566 0.34679234 0.37042879 0.19007257 0.60918042 0.52468057
 0.90600822 0.98834562 0.53665881 0.81307164 0.6738128  0.5528185 ]

mean:  0.5795174403750846
```

```
# 6. Create a 2d array with 1 on the border and 0 inside.
x = np.ones((3,3))
y = 1 * x
y[1][1] = 0
print(y)
```

```
[[1. 1. 1.]
 [1. 0. 1.]
 [1. 1. 1.]]
```

```
# 7. Create random vector of size 10 and replace the maximum value by 0.
x = np.random.rand(10)
print('original: ', x)
x[x.argmax()] = 0
y = list(x)
print('replaced: ', y)
```

```
original:  [0.06452845 0.7316195  0.00392886 0.489799   0.87316401 0.51101525
 0.32698302 0.36892565 0.34509059 0.82545913]
replaced:  [0.06452845311180688, 0.7316194955388082, 0.003928862097860919, 0.4897989992702326, 0.0, 0.5110152
```

```
# 8. Create an array with a checkerboard pattern of 0 and 1 of size 8x8.
x = np.zeros((8,8))
y = 1 * x
y[1::2,::2] = 1
y[::2,1::2] = 1
print(y)
```

```
[[0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]]
```

```
# 9. Consider a random vector with shape (100,2) representing coordinates,
#    find point by point distances (using L2-norm).
vec = np.random.rand(10,2)
x,y = np.atleast_2d(vec[:,0], vec[:,1])
distances = np.sqrt((x-x.T)**2 + (y-y.T)**2)
print(distances)
```

```
[[0.         0.33740795 0.04931661 0.28725164 0.44548072 0.0118984
  0.41952476 0.32043087 0.39067765 0.36452055]
 [0.33740795 0.         0.3000274  0.17414019 0.78284615 0.34615778
  0.35992063 0.19752113 0.58499005 0.7017341 ]
 [0.04931661 0.3000274  0.         0.27345772 0.48591821 0.05246309
  0.37281795 0.27119829 0.43182193 0.40581507]
 [0.28725164 0.17414019 0.27345772 0.         0.70523178 0.2988889
  0.49411009 0.33166995 0.4235837  0.6240793 ]
 [0.44548072 0.78284615 0.48591821 0.70523178 0.         0.43708072
  0.7697654  0.73407721 0.49405201 0.08163831]
 [0.0118984  0.34615778 0.05246309 0.2988889  0.43708072 0.
  0.41785904 0.32308098 0.39615641 0.35636414]
 [0.41952476 0.35992063 0.37281795 0.49411009 0.7697654  0.41785904
  0.         0.16478706 0.80046838 0.7026653 ]
 [0.32043087 0.19752113 0.27119829 0.33166995 0.73407721 0.32308098
  0.16478706 0.         0.67179066 0.65832317]
 [0.39067765 0.58499005 0.43182193 0.4235837  0.49405201 0.39615641
  0.80046838 0.67179066 0.         0.43694644]
 [0.36452055 0.7017341  0.40581507 0.6240793  0.08163831 0.35636414
  0.7026653  0.65832317 0.43694644 0.        ]]
```