```
import numpy as np
```

+ Code + Text

```
# np.array is used to create a NumPy matrix
# 1 dimensional array
a1=np.array([1, 4, 5, 7.8, 9, 5, 3])
print(type(a1))
```

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

```
# 2 dimensional array
a2=np.array([[2, 3], [5, 8], [8, 9]])   # r * c matrix, 3 x 2 matrix
print(a2)
```

```
[[2 3]
 [5 8]
 [8 9]]
```

```
# np.zeros() to make a matrix containing only zeros
a3=(2,2)
print(np.zeros(a3))
```

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

```
# np.ones() to make a matrix containing only ones
a4=(2,2)
print(np.ones(a3))
```

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

```
# to create a array containing all the numbers between a lower and upper bound
a5=np.arange(5, 12) # 5 is included but 12 is not included
print(a5)
```

```
[ 5  6  7  8  9 10 11]
```

```
a6=np.random.randint(low=0, high=101, size=(5))
print(a6)
# Note that the highest generated integer np.random.randint is one less than the high argu
```

```
[88 24 51 64 28]
```

```
# To create random floating-point values between 0.0 and 1.0, call np.random.random.
a7=np.random.random([5])
print(a7)
```

```
[0.44706797 0.17606624 0.12805897 0.46729196 0.11863898]
```

```
# mathematical operations on matrices
# If you want to add or subtract two vectors or matrices, linear algebra requires that the
# Furthermore, if you want to multiply two vectors or matrices, linear algebra imposes str
# Fortunately, NumPy uses a trick called broadcasting to virtually expand the smaller oper
a8=a7+2
print(a8)
a9=a7*3
print(a9)
```

```
[2.44706797 2.17606624 2.12805897 2.46729196 2.11863898]
[1.34120391 0.52819872 0.38417692 1.40187589 0.35591694]
```

```
# Task 1: Create a Linear Dataset
# Your goal is to create a simple dataset consisting of a single feature and a label as fo
# 1. Assign a sequence of integers from 6 to 20 (inclusive) to a NumPy array named feature
# 2. Assign 15 values to a NumPy array named label such that: label = (3)(feature) + 4
feature = np.arange(6, 21)
print(feature)
label = (3*feature)+4
print(label)
```

```
[ 6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
[22 25 28 31 34 37 40 43 46 49 52 55 58 61 64]
```

```
# Task 2: Add Some Noise to the Dataset
# To make your dataset a little more realistic, insert a little random noise into each ele
# To be more precise, modify each value assigned to label by adding a different random flo
noise = (np.random.random([15]) * 4) - 2
print(noise)
label = label + noise
print(label)
```

```
[ 0.72507294 -1.92801494  1.41395433 -1.54584829  1.66641674 -0.54999582
 -1.58599986 -0.17834759  1.11460044 -1.33215644  0.36530526 -0.31494639
 -0.35164051  0.03714895 -1.79619833]
[22.72507294 23.07198506 29.41395433 29.45415171 35.66641674 36.45000418
 38.41400014 42.82165241 47.11460044 47.66784356 52.36530526 54.68505361
 57.64835949 61.03714895 62.20380167]
```