Vectors

In [ ]:

```
import numpy as np

#Defining a list
lst = [10, 20, 30, 40, 50]

#Defining a vector
vctr = np.array(lst)

#Printing the vector
print("Vector created from a list:")
print(vctr)
```

```
Vector created from a list:
[10 20 30 40 50]
```

In [ ]:

```
#Defining a vector
vctr = np.array('10, 20, 30, 40, 50')

#Printing the vector
print(vctr)
```

```
10, 20, 30, 40, 50
```

In [ ]:

```
#Defining a vector
vctr = np.array([10, 20, 30, 40, 50])

#Printing the vector
print(vctr)
```

```
[10 20 30 40 50]
```

In [ ]:

```
import numpy as np

#Defining a list
lst = [[2],
       [4],
       [6],
       [10]]

#Defining a vector
vctr = np.array(lst)

#Printing the vector
print("Vector created from a list:")
print(vctr)
```

```
Vector created from a list:
[[ 2]
 [ 4]
 [ 6]
 [10]]
```

Basic Operations

Addition

In [ ]:

```
import numpy as np

#Defining lists
lst1 = [10, 20, 30, 40, 50]
lst2 = [1, 2, 3, 4, 5]

#Defining vectors
vctr1 = np.array(lst1)
vctr2 = np.array(lst2)

#Printing the vectors
print(vctr1)
print(vctr2)

#Calculating the addition of the vectors
vctr_add = vctr1 + vctr2

#Printing the result of the addition
print("Addition of the two vectors: ", vctr_add)
```

```
[10 20 30 40 50]
[1 2 3 4 5]
Addition of the two vectors:  [11 22 33 44 55]
```

Subtraction

In [ ]:

```
import numpy as np

#Defining lists
lst1 = [10, 20, 30, 40, 50]
lst2 = [1, 2, 3, 4, 5]

#Defining vectors
vctr1 = np.array(lst1)
vctr2 = np.array(lst2)

#Printing the vectors
print(vctr1)
print(vctr2)

#Calculating the subtraction of the vectors
vctr_sub = vctr1 - vctr2

#Printing the result of the subtration
print("Subtraction of the two vectors: ", vctr_sub)
```

```
[10 20 30 40 50]
[1 2 3 4 5]
Subtraction of the two vectors:  [ 9 18 27 36 45]
```

Division

In [ ]:

```
import numpy as np

#Defining lists
lst1 = [10, 20, 30, 40, 50]
```

```
lst2 = [1, 2, 3, 4, 5]

#Defining vectors
vctr1 = np.array(lst1)
vctr2 = np.array(lst2)

#Printing the vectors
print(vctr1)
print(vctr2)

#Calculating the division of the vectors
vctr_div = vctr1 / vctr2

#Printing the result of the subtration
print("Subtraction of the two vectors: ", vctr_div)
```

```
[10 20 30 40 50]
[1 2 3 4 5]
Subtraction of the two vectors:  [10. 10. 10. 10. 10.]
```

Dot Product

The dot product of two lists is the sum of the products of the corresponding elements of the lists.

For example: A . B = A[0] * B[0] + A[1] * A[1] + ...

In [ ]:

```
import numpy as np

#Defining lists
lst1 = [10, 20, 30, 40, 50]
lst2 = [1, 1, 1, 1, 1]

#Defining vectors
vctr1 = np.array(lst1)
vctr2 = np.array(lst2)

#Printing the vectors
print(vctr1)
print(vctr2)

#Calculating the dot product of the vectors
vctr_dot = np.dot(vctr1, vctr2)

##Alternative Methods for calculating the dot product of the vectors:
#vctr_dot = vctr1.dot(vctr2)
#vctr_dot = vctr1 @ vctr2

#Printing the result of the dot product
print("Dot product of the two vectors: ", vctr_dot)
```

```
[10 20 30 40 50]
[1 1 1 1 1]
Dot product of the two vectors:  150
```

Product

In [ ]:

```
#Defining lists
```

```
p = [4, 2]
q = [5, 6]

#Calculating product
product = np.cross(p, q)

#Printing the result of the product
print(product)
```

```
14
```

p x q = 4 x 6 - 2 x 5 = 24 - 10 = 14

In [ ]:

```
#Defining numpy arrays
p = np.array([1, 2])
q = np.array([1, 3])

#Calculating product
product = np.cross(p, q)

#Printing the result of the product
print(product)
```

```
1
```

Magnitude of a Vector

Method 1: using linalg.norm() from numpy module

In [ ]:

```
import numpy as np

#Define vector
x = np.array([3, 6, 6, 4, 8, 12, 13])

#Calculate the magnitude of the vector
np.linalg.norm(x)
```

Out[ ]:

```
21.77154105707724
```

Method 2: using custom NumPy functions

In [ ]:

```
import numpy as np

#Define vector
x = np.array([3, 6, 6, 4, 8, 12, 13])

#Calculate the magnitude of the vector
np.sqrt(x.dot(x))
```

Out[ ]:

```
21.77154105707724
```

Unit Vectors

Method 1: using unit_vector() from transformations library

```
In [ ]: pip install transformations
```

```
Requirement already satisfied: transformations in /data/data/ru.iiec.pydro
id3/files/aarch64-linux-android/lib/python3.9/site-packages (2022.9.26)
Requirement already satisfied: numpy>=1.19.2 in /data/data/ru.iiec.pydroid
3/files/aarch64-linux-android/lib/python3.9/site-packages (from transforma
tions) (1.21.2)
WARNING: You are using pip version 21.2.4; however, version 23.3.1 is avai
lable.
You should consider upgrading via the '/data/user/0/ru.iiec.pydroid3/file
s/aarch64-linux-android/bin/python3.9 -m pip install --upgrade pip' comman
d.
Note: you may need to restart the kernel to use updated packages.
```

```
In [ ]: import transformations as tr

        #Defining a numpy array (a vector)
        arr = np.array([1, 2, 3])

        #Normalizing the array to unit vector and Printing
        print(tr.unit_vector(arr))
```

```
[0.26726124 0.53452248 0.80178373]
```

Method 2: using normalize() from vg module

```
In [ ]: pip install vg
```

```
Requirement already satisfied: vg in /data/data/ru.iiec.pydroid3/files/aar
ch64-linux-android/lib/python3.9/site-packages (2.0.0)
Requirement already satisfied: numpy in /data/data/ru.iiec.pydroid3/files/
aarch64-linux-android/lib/python3.9/site-packages (from vg) (1.21.2)
WARNING: You are using pip version 21.2.4; however, version 23.3.1 is avai
lable.
You should consider upgrading via the '/data/user/0/ru.iiec.pydroid3/file
s/aarch64-linux-android/bin/python3.9 -m pip install --upgrade pip' comman
d.
Note: you may need to restart the kernel to use updated packages.
```

```
In [ ]: import vg

        #Creating a numpy array
        arr = np.array([1, 2, 3])

        #Normalizing the array to unit vector
        unitVector = vg.normalize(arr)

        #Printing the unit vector
        print(unitVector)
```

```
[0.26726124 0.53452248 0.80178373]
```

Method 3: using linalg.norm() from numpy module

```
In [ ]: import numpy as np
        from numpy import*

        #Creating a numpy array
        data = np.array([1, 2, 3])
```

```
#Normalizing the array to unit vector
unitVector = data / linalg.norm(data)

#Printing the unit vector
print(unitVector)
```

```
[0.26726124 0.53452248 0.80178373]
```

Cartesian Vectors

In [ ]:

```
import itertools as it
import numpy as np

#Creating numpy arrays
array1 = np.array([1, 2, 3])
array2 = np.array([1, 2, 3])

#Creating cartesian vector
output = np.array(list(it.product(array1, array2)))

#Printing the cartesian vector
print(output)
```

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