CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Numpy

Other numpy functions

Split an array equally into multiple arrays.
Arrays are views so they are shallow copies of the main array.

```
import numpy as np

data = np.array([[1, 2, 3], [2, 4, 5], [4, 5, 7], [6, 2, 3]], float)


a,b,c,d = np.split(data, 4)

print(b)
```

```
[[2. 4. 5.]]
```

Other numpy functions

Split an array nearly equally into multiple arrays.
Arrays are views so they are shallow copies of the main array.

```
import numpy as np

data = np.array([[1, 2, 3], [2, 4, 5], [4, 5, 7], [6, 2, 3]], float)


a, b, c = np.array_split(data, 3)

print(a)

print(b)
```

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

[[4. 5. 7.]]
```

Other numpy functions

Split an array equally into multiple arrays vertically.
Arrays are views so they are shallow copies of the main array.

```
import numpy as np

data = np.array([[1, 2, 3], [2, 4, 5], [4, 5, 7], [6, 2, 3]], float)


a, b, c = np.hsplit(data, 3)


print(c)
```

```
[[3.]
 [5.]
 [7.]
 [3.]]
```

Other numpy functions

Swap the dimensions. It is also called transpose .
The transpose is a view so it is a shallow copy.

```
import numpy as np

data = np.array([[1, 2, 3], [2, 4, 5], [4, 5, 7], [6, 2, 3]], float)


a = np.transpose(data)

print(a)
```

```
[[1. 2. 4. 6.]
 [2. 4. 5. 2.]
 [3. 5. 7. 3.]]
```

Other numpy functions

Stack rows or columns at the end of an array.

```
import numpy as np

data = np.array([[1, 2, 3], [2, 4, 5], [4, 5, 7], [6, 2, 3]], float)

a = np.array([[21, 0, 2], [1, 4, 15], [3, 9, 2], [3, 4, 6]], float)

c = np.vstack((a,data))
print(c)
c = np.hstack((a,data))
print(c)
c = np.concatenate((a, data),axis=0)

print(c)

c = np.concatenate((a, data),axis=1)

print(c)
```

Wrong dimensions generate error

```
[[21.  0.  2.]
 [ 1.  4. 15.]
 [ 3.  9.  2.]
 [ 3.  4.  6.]
 [ 1.  2.  3.]
 [ 2.  4.  5.]
 [ 4.  5.  7.]
 [ 6.  2.  3.]]
[[21.  0.  2.  1.  2.  3.]
 [ 1.  4. 15.  2.  4.  5.]
 [ 3.  9.  2.  4.  5.  7.]
 [ 3.  4.  6.  6.  2.  3.]]
[[21.  0.  2.]
 [ 1.  4. 15.]
 [ 3.  9.  2.]
 [ 3.  4.  6.]
 [ 1.  2.  3.]
 [ 2.  4.  5.]
 [ 4.  5.  7.]
 [ 6.  2.  3.]]
[[21.  0.  2.  1.  2.  3.]
 [ 1.  4. 15.  2.  4.  5.]
 [ 3.  9.  2.  4.  5.  7.]
 [ 3.  4.  6.  6.  2.  3.]]
```

Matrix determinant

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

Matrix determinant

```
import numpy as np

b = np.array([[6,1,1], [4, -2, 5], [2,8,7]])

print (b)

print (np.linalg.det(b) )

print (6*(-2*7 - 5*8) - 1*(4*7 - 5*2) + 1*(4*8 - -2*2))
```

```
[[ 6  1
1] [ 4
-2  5] [
2  8  7]]

-306.0

-306
```

Exercise

Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.

Exercise

```
import numpy as np
a = np.array([[10,40],[30,20]])
print("Original array:")
print(a)
print("Sort the array along the first axis:")
print(np.sort(a, axis=0))
print("Sort the array along the last axis:")
print(np.sort(a))
print("Sort the flattened array:")
print(np.sort(a, axis=None))
```

Exercise

Write a NumPy program to create a structured array from given student name, height, class and their data types. Now sort the array on height.

Exercise

```
import numpy as np
data_type = [('name', 'S4'), ('class', int), ('height', float)]
students_details = [('James', 5, 48.5), ('Nail', 6, 52.5),('Paul', 5, 42.10), ('Pit', 5, 40.11)]
# create a structured array
students = np.array(students_details, dtype=data_type)
print("Original array:")
print(students)
print("Sort by height")
print(np.sort(students, order='height'))
print(students[0]['name'])
```

Exercise

Write a NumPy program to capitalize the first letter, lowercase, uppercase, swapcase, title-case of all the elements of a given array.

Exercise

```
import numpy as np
x = np.array(['python', 'PHP', 'java', 'C++'], dtype=np.str)
print("Original Array:")
print(x)
capitalized_case = np.char.capitalize(x)
lowered_case = np.char.lower(x)
uppered_case = np.char.upper(x)
swapcased_case = np.char.swapcase(x)
titlecased_case = np.char.title(x)
print("\nCapitalized: ", capitalized_case)
print("Lowered: ", lowered_case)
print("Uppered: ", uppered_case)
print("Swapcased: ", swapcased_case)
print("Titlecased: ", titlecased_case)
```

```
Capitalized:  ['Python' 'Php' 'Java' 'C++']
Lowered:  ['python' 'php' 'java' 'c++']
Uppered:  ['PYTHON' 'PHP' 'JAVA' 'C++']
Swapcased:  ['PYTHON' 'php' 'JAVA' 'c++']
Titlecased:  ['Python' 'Php' 'Java' 'C++']
```

Other numpy functions

datetime64

```
a = np.datetime64('2019-03-21T13:22:23')

b = np.datetime64(a,'D')

print(b)
```

2019-03-21

Code	Meaning
Y	year
M	month
W	week
D	day
h	hour
m	minute
s	second

Other numpy functions

Datetime64

Exercise

Write a NumPy program to display all the dates for the month of March, 2017

Other numpy functions

Datetime64
Exercise

```
import numpy as np
print("March, 2017")
print(np.arange('2017-03', '2017-04', dtype='datetime64[D]'))
```

```
['2017-03-01' '2017-03-02' '2017-03-03' '2017-
03-04' '2017-03-05' '2017-03-06' '2017-03-07'
'2017-03-08' '2017-03-09' '2017-03-10' '2017-
03-11' '2017-03-12' '2017-03-13' '2017-03-14'
'2017-03-15' '2017-03-16' '2017-03-17' '2017-
03-18' '2017-03-19' '2017-03-20' '2017-03-21'
'2017-03-22' '2017-03-23' '2017-03-24' '2017-
03-25' '2017-03-26' '2017-03-27' '2017-03-28'
'2017-03-29' '2017-03-30' '2017-03-31']
```

Other numpy functions

Datetime64
Exercise

Write a NumPy program to get the dates of yesterday, today and tomorrow.

Other numpy functions

Datetime64
Exercise

```
import numpy as np
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
print("Yestraday: ",yesterday)
today = np.datetime64('today', 'D')
print("Today: ",today)
tomorrow = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
print("Tomorrow: ",tomorrow)
```