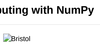
Introduction to Computer Programming

Week 8.3: Scientific computing with NumPy



Scientific computing

Many real-world problems are so complex that they do not have an exact solution.

Scientific computing is concerned with the development of algorithms to find **approximate** solutions to these problems.

Many of these algorithms involve calculations with large collections of numbers (vectors and matrices)

NumPy

NumPy is a Python library that enables large collections of numbers to be stored as **arrays**.

Arrays provide a way to store vectors, matrices, and other types of numerical data.

NumPy provides very fast mathematical functions that can operate on these arrays.

Advantages of using NumPy:

- Memory efficient and very fast
- Used in other libraries (e.g. data science, machine learning)
- Extensive built-in functionality (e.g. linear algebra, statistics)

Getting started

It is common to import NumPy using the command

In [16]:
```
import numpy as np
```

Defining arrays

Arrays are defined using the `array` function.

A vector (1D array) can be created by passing a list to `array`

Example: Create an array for the vector $a = (1, 2, 3)$

In [ ]:
```

```

Like lists, elements in arrays are accessed using square brackets.

In NumPy, the first element of an array has index 0.

In [1]:
```
# print the first entry in a
```

In [2]:
```
# print a and then change the third entry to 5
```

A matrix (2D array) can be created by passing `array` a nested list.

Each inner list will be a row of the matrix

Example: Define the matrix

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

In [ ]:
```

```

Elements in a 2D array can be accessed using square brackets with indices separated by a comma.

The first index is for the row and the second is for the column

Remember, indexing starts at 0.

In [22]:
```
# print the matrix A

# print the entry in the first row, second column
```

Exercise:

Create the array

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 9 & 8 & 7 \\ 2 & 4 & 8 \end{pmatrix}$$

Add the entry in the first row, second column to the entry in the third row, first column

Solution:

In [ ]:
```

```

Some useful functions for creating arrays

`linspace(a, b, N)` creates a 1D array with N uniformly spaced entries between a and b (inclusive)

In [3]:
```
# create an array with 5 entries between 0 and 1
```

`ones(dims)` creates arrays filled with ones, where `dims` is an integer or a tuple of integers that describes the dimensions of the array

In [4]:
```
# create a 1D array of length 3 filled with ones
```

In [5]:
```
# tuples are used to create multi-dimensional arrays of ones
```

`zeros(dims)` creates arrays filled with zeros

In [6]:
```
# create a 3 x 3 array of zeros by passing a tuple as an argument
```

`eye(N)` creates the $N \times N$ identity matrix

In [7]:
```
# create a 3 x 3 identity matrix
```

Arrays of random numbers

There are several NumPy functions for creating arrays of random numbers

`random.random(dims)` creates an array with random numbers between 0 and 1 from a uniform distribution

In [8]:
```
# tuples are used to create random matrices
```

`random.randint(a, b, dims)` creates an array with random integers between a and $b - 1$

In [9]:
```
# create a vector with three random integers between 1 and 9
```

In [10]:
```
# create a 3 x 2 matrix with random integers between 1 and 9
```

Operations on arrays

If we were using lists, then we'd have to use `for` loops or list comprehensions to carry out operations

In [16]:
```
l = [1, 2, 3, 4, 5, 6]
l2 = [e + 1 for e in l]
print(l2)
```

```
[2, 3, 4, 5, 6, 7]
```

With NumPy, such operations become trivial

In [17]:
```
l = np.array([1, 2, 3, 4, 5, 6])
l2 = l + 1
print(l2)
```

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

Example: Define the vectors $a = (1, 2, 3)$ and $b = (3, 2, 1)$. Compute $a + b$, $c = 0.5a$, and the dot product $a \cdot b$

In [19]:
```
# defining the vectors
```

In [12]:
```
# computing a + b and printing the result
```

In [13]:
```
# computing c = 0.5a and printing the result
```

In [14]:
```
# computing a.b
```

Question: What happens if we multiply the two vectors a and b?

In [20]:
```
# printing a and b
print('a =', a, '\nb =', b)
```

```
a = [1 2 3]
b = [3 2 1]
```

Answer: The * operator performs element-by-element multiplication. The vectors must be the same size for this to work correctly

In [23]:
```
a = np.array([1, 2, 3])
c = np.array([1, 1, 1, 1])
a * c
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-032aee7d0f56> in <module>
      1 a = np.array([1, 2, 3])
      2 c = np.array([1, 1, 1, 1])
----> 3 a * c

ValueError: operands could not be broadcast together with shapes (3,) (4,)
```

Exercise:

Create a vector of length 20 where each entry is a uniformly distributed random number between 3 and 4

Solution:

In [ ]:
```

```

Matrix operations

Matrices can be added using + and multiplied using @

Example: Consider the matrices

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 4 \\ 6 & 2 \end{pmatrix}$$

Compute $A + B$ and AB

In [24]:
```

```

In [ ]:
```

```

In [ ]:
```

```

Warning: It is very tempting to use * for matrix multiplication, but this computes the element-wise product

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 4 \\ 6 & 2 \end{pmatrix}$$

In [ ]:
```

```

Applying mathematical functions to arrays

NumPy comes with mathematical functions that can operate on arrays.

Example: compute $y = \sin(x)$ at 10 equally spaced points between 0 and 2π

In [ ]:
```

```

Other functions include cos, tan, arccos, arcsin, exp, log, and more

Linear algebra with NumPy

NumPy can perform some linear algebra calculations.

Example: Use `np.linalg.solve` to solve the system $Ax = b$ when

$$A = \begin{pmatrix} 1 & 2 \\ 4 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

In [17]:
```
# define A and b
A = np.array([[1, 2], [4, 1]])
b = np.array([3, 1])

# solve for the vector x and print the result
```

Summary

NumPy is a library for the creation and manipulation of arrays

It comes loaded with functions for operating on these arrays

It also has functions for linear algebra