NumPy

Week 6

Nurul Afiqah Burhanuddin
nurul.afiqah@ukm.edu.my

Introduction

- fundamental library for scientific computing in Python
- provides a powerful multidimensional array object
- `ndarray`: n-dimensional arrays of homogeneous data types
- facilitates advanced mathematical operations for array by vectorization - eliminates explicit looping and indexing in code, relying on optimized, pre-compiled C code for performance

Creating Arrays

- From list: `np.array([1, 2, 3])`
- Using built-in functions:
  - ➢`np.zeros()` – Array of zeros
  - ➢`np.ones()` – Array of ones
  - ➢`np.arange(0, 10, 2)` – Array of evenly spaced values
  - ➢`np.linspace(0, 1, 5)` – Array of evenly spaced values over a specified interval

Array Attributes

- `ndarray.shape` - Dimensions of the array
- `ndarray.size` - Total number of elements
- `ndarray.dtype` - Data type of the array

Array Indexing

- to access individual elements or subsets of elements in an array.
- through square brackets
- using an integer index from 0 to n-1
- 1D array : one index
- 2D array : two indices

```
array[row,column]
```

- 3D array: three indices.

```
array[depth,row,column]
```

Array Advanced Indexing

- Fancy indexing: using integer arrays
- Boolean indexing: Access elements that satisfy a condition

Operator	Description
a==b	True if a equals b
a!=b	True if a is not equal to b
a<=b, a<b	True if a is less than (less than or equal) to b
a>b, a>=b	True if a is greater than (greater than or equal) to b
a&b	True if both a and b are True
a|b	True if either a or b is True

Array Slicing

- Basic slicing works using the : operator.
- Slicing syntax allows specifying start, stop, and step parameters for each axis.
- Multi-dimensional slicing is done by providing slices for each dimension (separated by commas).
- Negative indices allow for counting from the end of the array.

Array Slicing

Operation	Description
array[start:stop]	Extract from start to stop - 1
array[start:stop:step]	Extract with a step between elements
array[:stop]	Extract from the start to stop - 1
array[start:]	Extract from start to the end of the array
array[start:stop, :]	Slice rows in 2D array
array[:, start:stop]	Slice columns in 2D array
array[start:stop,:,:]	Slice depths in 3D array
array[:,start:stop,:]	Slice rows in 3D array
array[:,:,start:stop,:,:]	Slice columns in 3D array

Array Manipulations

- `np.reshape()` - reshaping array
- `np.concatenate()` - join arrays along an axis.
- `np.vstack(),np.hstack` - join arrays vertically (horizontally)
- `np.split()` - split apart an array into multiple arrays along an axis
- `np.vplit(),np.hsplit()` -- splits an array vertically (horizontally)

Ufunc

- Functions that operate on arrays in an element-wise fashion.

- unary ufunc: operates on a single input array

```
np.sqrt(), np.exp(), np.max(), np.argmax()
```

- binary ufunc: operates on two input arrays (or an array and a scalar)

```
np.add(), np.subtract(), np.multiply(),

np.divide()
```

View and Copy

- **View:** a new array object that shares the same data as the original array.
- Changes made to the view will affect the original array, and vice versa - both the view and the original array point to the same memory location.
- View is a shallow copy: It doesn't duplicate the data, only creates a new array object with a different shape or slice.
- Use when you need to create a new array with a different shape or slice, but you still want to share the data.
- Basic slicing generally returns a view.

View and Copy

- **Copy:** a completely independent array with its own data.
- Modifying the copy does not affect the original array, and vice versa.
- Copy is a deep copy: It duplicates the data and creates a new memory allocation.
- You get a copy when explicitly calling the copy() method.
- Some slicing operations (fancy indexing) result in a copy.