Object Oriented Programming (Using Python)

UNIT- III

Python Libraries:

- Numpy
  - Introduction
  - Installing Numpy
  - Arrays with Numpy
  - Create an array

Prof. R. MADANA MOHANA
Professor, Artificial Intelligence & Data Science
http://rmadanamohana.com/

Numpy

- **NumPy** (**Numerical Python**) is an open source Python library that's used in almost every field of science and engineering.
- It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems (PyData - A community for developers and users of open source data tools).
- NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development.
- The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

Numpy

- The **NumPy** library contains multidimensional array and matrix data structures.
- It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it.
- NumPy can be used to perform a wide variety of mathematical operations on arrays.
- It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Numpy

- **NumPy** can be easily interfaced with other Python packages and provides tools for integrating with other programming languages like C, C++ etc.

Advantages of numpy

- Fundamental package for scientific computing with Python.
- N-dimensional array object.
- Linear algebra, Fourier transform, random number capabilities.
- Building block for other packages (e.g. Scipy)
- Open source

Installing NumPy

To install NumPy, we strongly recommend using a scientific Python distribution.

If you already have Python, you can install NumPy with:

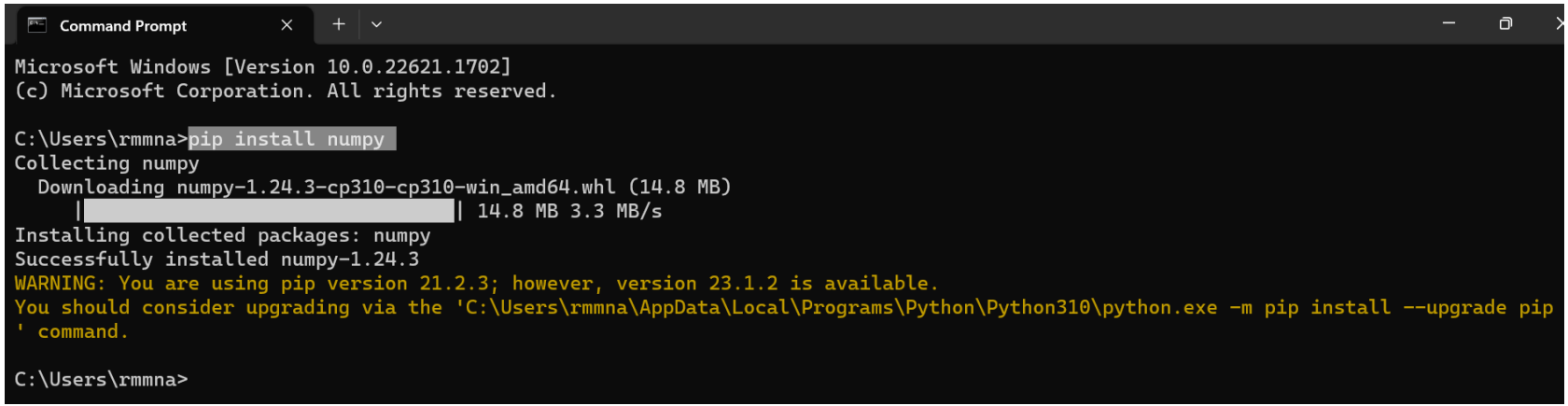
`pip install numpy # pip` is a package manager for Python packages, or modules

or

`conda install numpy # Conda` is a cross platform package and environment manager that installs and manages conda packages from the Anaconda repository as well as from the Anaconda Cloud.

How to install NumPy using official python IDLE?

- Press the Windows key on your keyboard.
- Type CMD and open Command Prompt. A black terminal should open up.
- Type `pip install numpy` and hit enter.
- It should start the installation. After you see the "Successfully Installed" message, go back to your IDLE and try importing numpy, it should work.



How to import NumPy

To access numpy and its functions import it in your Python code like this:

```
import numpy as np
```

We shorten the imported name to np for better readability of code using numpy. This is a widely adopted convention that you should follow so that anyone working with your code can easily understand it.

Arrays with numpy

- A NumPy array would require much less memory to store the same amount of data compared to a Python list, which helps in reading and writing from the array in a faster manner.

Attributes of an ndarray object

- NumPy's array class is called ndarray.
- It is also known by the alias array.
- Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality.

Attributes of an ndarray object

The more important attributes of an ndarray object are:

`ndarray.ndim`

- the number of axes (dimensions) of the array.

`ndarray.shape`

- the dimensions of the array.
- This is a tuple of integers indicating the size of the array in each dimension.
- For a matrix with `n` rows and `m` columns, shape will be (`n,m`).
- The length of the shape tuple is therefore the number of axes, `ndim`.

Attributes of an ndarray object

`ndarray.size`

- the total number of elements of the array.
- This is equal to the product of the elements of shape.

`ndarray.dtype`

- an object describing the type of the elements in the array.
- One can create or specify dtype's using standard Python types.
- Additionally **`NumPy`** provides types of its own.
- **`numpy.int32`**, **`numpy.int16`**, and **`numpy.float64`** are some examples.

Attributes of an ndarray object

`ndarray.itemsize`

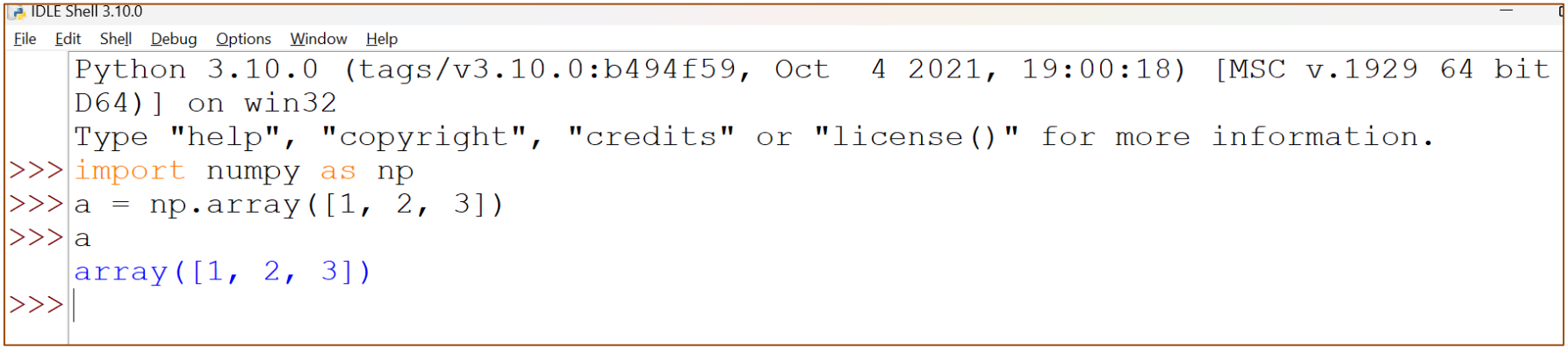
- the size in bytes of each element of the array.
- For example, an array of elements of type float64 has itemsize **`8`** (**`=64/8`**), while one of type complex32 has itemsize **`4`** (**`=32/8`**). It is equivalent to **`ndarray.dtype.itemsize`**.

`ndarray.data`

- the buffer containing the actual elements of the array.
- Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

Creating an array

- To create a NumPy array, you can use the function np.array().
- All you need to do to create a simple array is pass a list to it.
- If you choose to, you can also specify the type of data in your list.



Creating an array

Besides creating an array from a sequence of elements, you can easily create an array filled with 0's:

```
>>>np.zeros(2)
array([0., 0.])
>>>np.zeros((3, 4))
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Creating an array

An array filled with 1's:

```
>>> np.ones(2)
array([1., 1.])
>>> np.ones((2, 3, 4), dtype=np.int16)
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
```

Creating an array

Create an empty array :

- The function empty creates an array whose initial content is random and depends on the state of the memory.
- The reason to use empty over zeros (or something similar) is speed - just make sure to fill every element afterwards!

```
>>>import numpy as np
>>> np.empty(2)
array([-4.03016855e-113,  1.22438294e+210])
```

Creating an array

Create an empty array : Ex-2

```
>>>import numpy as np
>>> np.empty((2, 3))
array([[1.5, 2.5, 3. ],
       [4. , 5. , 6. ]])
```

Creating an array

Create an array with a range of elements:

To create sequences of numbers, NumPy provides the arange function which is analogous to the Python built-in range, but returns an array.

```
>>>import numpy as np
>>> np.arange(4)
array([0, 1, 2, 3])
>>> np.arange(10, 30, 5)
array([10, 15, 20, 25])
>>> np.arange(0, 2, 0.3)# it accepts float arguments
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])
```

Creating an array

Create an array that contains a range of evenly spaced intervals:

To do this, you will specify the first number, last number, and the step size.

```
>>>import numpy as np
>>> np.arange(2, 9, 2)
array([2, 4, 6, 8])
```

Creating an array

Create an array that contains a range of evenly spaced intervals:

You can also use np.linspace() to create an array with values that are spaced linearly in a specified interval.

```
>>>import numpy as np
>>> np.linspace(0, 10, num=5)
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
>>> np.linspace(0, 2, 9)
array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 ,
1.75, 2.  ])
```

Creating an array

Specifying your data type:

While the default data type is floating point (np.float64), you can explicitly specify which data type you want using the dtype keyword.

```
>>>import numpy as np
>>> x = np.ones(2, dtype=np.int64)
>>> x
array([1, 1], dtype=int64)
>>> y = np.zeros(2, dtype=np.int64)
>>> y
array([0, 0], dtype=int64)
```

Creating an array

Specifying your data type: Ex-1

While the default data type is floating point (np.float64), you can explicitly specify which data type you want using the dtype keyword.

```
>>>import numpy as np
>>> x = np.ones(2, dtype=np.int64)
>>> x
array([1, 1], dtype=int64)
>>> y = np.zeros(2, dtype=np.int64)
>>> y
array([0, 0], dtype=int64)
```

Creating an array

Specifying your data type: Ex-2

```
>>>import numpy as np
>>> a = np.array([[1, 2], [3, 4]], dtype=complex)
>>> a
array([[1.+0.j, 2.+0.j],
       [3.+0.j, 4.+0.j]])
```

Creating an array

Specifying your data type: cont'd.. Ex-3

```
>>>import numpy as np
>>> x = np.array([1, 2, 3])
>>> x
array([1, 2, 3])
>>> x.dtype
dtype('int32')
>>> y = np.array([1.0, 2.5, 3.5])
>>> y
array([1. , 2.5, 3.5])
>>> y.dtype
dtype('float64')
```

Creating an array

Transforming sequences of sequences into two-dimensional arrays, and three-dimensional arrays:

Array transforms sequences of sequences into two-dimensional arrays, sequences of sequences of sequences into three-dimensional arrays, and so on.

```
>>>import numpy as np
>>> z = np.array([(1.5, 2.5, 3), (4, 5, 6)])
>>> z
array([[1.5, 2.5, 3. ],
       [4. , 5. , 6. ]])
```