HyperionDev

NumPy Basics

Task

Visit our website

Introduction

During the remainder of this bootcamp, you will learn many techniques needed for data engineering, data visualisation, and machine learning. It's essential to understand and be able to use Python packages to achieve these tasks. Here you will be briefly introduced to NumPy, a key data science package.

Before we explore NumPy, let's install it using the pip package manager. Open your terminal and run the following command:

```
pip install -U numpy
```

It may take some time for the package to install.

Import packages

As you know, to use packages or libraries in Python, you will need to import them at the top of your Python file. There are several ways to achieve this, and we will use the NumPy package to illustrate three methods.

1. The standard approach uses a simple import statement.

   ```
   import numpy
   ```

   This will create a reference to the NumPy module in the current namespace. To use functions or attributes provided by NumPy, you would use the dot notation `numpy.X`, where X is the specific function, method, or attribute. For example, `numpy.array()`:

   ```
   # Converts a list of float values into an array
   numpy_values = numpy.array([22.5, 24.9, 27.8, 27.1])
   ```

2. When several NumPy functions are used, we utilise a placeholder or nickname instead of `numpy` by adding `as "nickname"` to the end of the import statement. This is the most commonly used option. See an example in the following code:

   ```
   import numpy as np
   ```

When you need to invoke something from NumPy, use `nickname.X` instead of `numpy.X`. For example, `np.array()`:

```
# Converts a list of float values into an array
numpy_values = np.array([22.5, 24.9, 27.8, 27.1])
```

3. To avoid the dot notation – i.e., `np.X`, or any reference to NumPy itself – in your script, you can simply import all the methods (functions) from NumPy. The asterisk (*) denotes "everything", so this will import "everything" from NumPy. Most of the time, it's better to use one of the previous two options to keep a connection between the module and its functions as other modules may have functions with the same name.

```
from numpy import *
```

For example, `array()`:

```
# Converts a list of float values into an array
numpy_values = array([22.5, 24.9, 27.8, 27.1])
```

Extra resource

You can also import single functions or submodules. Explore **additional methods for importing third-party code**.

NumPy

NumPy stands for Numeric Python, which is an open-source extension module for Python and hence uses Python syntax. NumPy is a fundamental package for scientific computing with Python, providing common mathematical and numerical routines in the form of pre-compiled functions.

NumPy brings the power of data structures to Python. Due to its array-oriented programming, it's easy to use NumPy to work with large multidimensional arrays and matrices and to carry out scientific computations.

Here's a simple example comparing NumPy and core Python:

```
import numpy as np
# Convert a list of values in degrees Celsius to Fahrenheit
c_values = [22.5, 24.9, 27.8, 27.1]

# Create a NumPy array and use NumPy scalar multiplication
f_values_numpy = np.array(c_values) * 9 / 5 + 32
print(f_values_numpy)  # Outputs array: [ 72.5 76.82 82.04 80.78]


# Using core Python iterate through all items in c_values
# and multiply each value by 9, divide by 5, and add 32
f_values_core = [x * 9 / 5 + 32 for x in c_values]
print(f_values_core)  # Outputs list: [ 72.5, 76.82, 82.04, 80.78]
```

You might wonder what the difference is. The main benefits of using NumPy arrays are smaller memory consumption, better runtime, and greater ease of manipulation of data compared to Python. As you can see in the code above, you don't have to loop through a NumPy array like a list!

Extra resources

To learn more about Numpy you can look into additional examples in **Chapter 4** of *Python for Data Analysis* (2017) by Wes McKinney, or consult Chapter 2 of ***Python Data Science Handbook*** (2016) by Jake VanderPlas. You may also read the **NumPy documentation** to explore its functionality.

SciPy

SciPy is built on top of NumPy and provides a comprehensive collection of algorithms and functions for scientific computing. It offers powerful tools for optimisation, integration, interpolation, linear algebra, signal and image processing, and statistical analysis. With SciPy, you can tackle complex mathematical problems and perform in-depth data analysis, making it an essential library for researchers and data scientists. Learn more about the SciPy package through its **documentation**.

You can install it using the following command in your terminal:

```
pip install -U scipy
```

Take note

First, read and run the **example files** provided. Feel free to write and run your own example code before doing the task to become more comfortable with the concepts covered.

Practical task

- Create a new program named **numpy_task.py**.
- Use comments and code to answer the following questions:
    i. Why doesn't `np.array((1, 0, 0), (0, 1, 0), (0, 0, 1, dtype=float)` create a two-dimensional array? Rewrite it correctly (**Hint:** Get familiar with **NumPy arrays**).
    ii. What is the difference between `a = np.array([0, 0, 0])` and `a = np.array([[0, 0, 0]])`?
    iii. Create a 3 by 4 by 4 array using the following code:

```
array = np.linspace(1, 48, 48).reshape(3, 4, 4)
```

    iv. Index or slice this array to obtain the following:
        1. 20.0

2. [9. 10. 11. 12.]
3. [[33. 34. 35. 36.] [37. 38. 39. 40.] [41. 42. 43. 44.] [45. 46. 47. 48.]]
4. [[5. 6.] [21. 22.] [37. 38.]]
5. [[36. 35.] [40. 39.] [44. 43.] [48. 47.]]
6. [[13. 9. 5. 1.] [29. 25. 21. 17.] [45. 41. 37. 33.]]
7. [[1. 4.] [45. 48.]]
8. [[25. 26. 27. 28.] [29. 30. 31. 32.] [33. 34. 35. 36.] [37. 38. 39. 40.]]

v. **Hint:** Look into these NumPy tools.

1. **Array manipulation routines**
2. **numpy.linspace**
3. **numpy.reshape**
4. **numpy.ndarray.flatten**

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.