# Module 5- Sequences

In Python programming, a **sequence** is collection of data values that are grouped together so that they can organized or manipulated by using the single name of the sequence.

## Arrays

An **Array** is a sequential collection of data values which include string and numeric data types among other data types, but all the elements in the array are of the same type.

### Examples of Arrays

1. high\_temperatures= [81, 85, 86, 89, 98, 91, 79]
2. cart\_prices= [12.99, 8.49, 5.25, 22.00, 3.75]
3. reviews= [“Great quality and fast shipping!”, “Product was okay, but packaging was damaged.”, “Excellent value for the price.”, “Not what I expected. Returning it.”, “Five stars! Highly recommended.”]
4. species= ["Jaguar", "Toucan", "Capuchin Monkey", "Toucan","Sloth", "Jaguar", "Macaw", "Capuchin Monkey", "Capybara", "Toucan"]

### Creating an Array in Python

|  |  |
| --- | --- |
| datascience Module | numpy Module |
| from datascience import\*  make\_array(collection of values separated by a comma)  high\_temperatures= make\_array(81, 85, 86, 89, 98, 91, 79) | import numpy as np  np.array([collection values separated by a comma])  high\_temperatures = np.array([81, 85, 86, 89, 98, 91, 79]) |

Practice 1- In the Module 5 Lecture Notebook, create an array for the b-d examples above under “Examples of Arrays”.

Practice 2- In the Module 5 Lecture Notebook, take the high\_temperatures array and evaluate the following expression:



What is the output?

What does the above expression represent?

### Array Methods (Functions)

#### Some Built-in Functions

my\_array.sum()- Finds the sum of all the elements in the array.

my\_array.mean()- Finds the mean of the elements in the array.

There are other built-in functions that we can use, but we have to be careful because the functions (methods) are really for what we call **list**. The difference between a list and an array is that a list can contain elements of different data types whereas array can only contain one data type. Instead of built-in functions, we use the functions(methods) in the Numpy module(package).

#### Numpy Functions on Arrays

Each of these functions takes an array as an argument and returns a single value.

| **Function** | **Description** |
| --- | --- |
| np.prod |  |
| np.sum |  |
| np.all | Test whether all elements are true values (non-zero numbers are true) |
| np.any | Test whether any elements are true values (non-zero numbers are true) |
| np.count\_nonzero |  |

Each of these functions takes an array as an argument and returns an array of values.

| **Function** | **Description** |
| --- | --- |
| np.diff |  |
| np.round |  |
| np.cumprod | A cumulative product: for each element, multiply all elements so far |
| np.cumsum | A cumulative sum: for each element, add all elements so far |
| np.exp | Exponentiate each element, |
| np.log |  |
| np.sqrt |  |
| np.sort |  |

Each of these functions takes an array of strings and returns an array.

| **Function** | **Description** |
| --- | --- |
| np.char.lower |  |
| np.char.upper |  |
| np.char.strip | Remove spaces at the beginning or end of each element |
| np.char.isalpha | Whether each element is only letters (no numbers or symbols) |
| np.char.isnumeric | Whether each element is only numeric (no letters) |

Each of these functions takes both an array of strings and a *search string*; each returns an array.

| **Function** | **Description** |
| --- | --- |
| np.char.count | Count the number of times a search string appears among the elements of an array |
| np.char.find | The position within each element that a search string is found first |
| np.char.rfind | The position within each element that a search string is found last |
| np.char.startswith | Whether each element starts with the search string |

[Full Numpy Reference](https://numpy.org/doc/stable/reference/) (https://numpy.org/doc/stable/reference/)

## Ranges

A **range** is an array numbers in increasing or decreasing order, each separated by a regular interval (same difference between adjacent elements).

We create a range by using the numpy function **arange**. Below are some different options:

np.arange(end point)

This creates a range starting at 0 and ending at the specified end point which increasing by 1. We would say that it has “Step 1”.

Pratice 3- In the Module 5 Lecture Notebook, type the following statement and write down the output:

[ ] np.arange(5)

[ ]

### np.arange(start, end)

For this option, the range start and ends at the specified values with Step 1.

Practice 4- In the Module 5 Lecture Notebook, type the following statement and write down the output:

[ ] np.arange(1,7)

[ ]

### np.arange(start, end, step)

For this option, you can now specify the start, end and step of your range.

Practice 5- In the Module 5 Lecture Notebook, type the following statement and write down the output:

[ ] np.arange(1,9,2)

[ ]

Practice 6- In the Module 5 Lecture Notebook, type the following statement and write down the output:

[ ] np.arange(1.5,-2,-0.5)

[ ]

Practice 7- Use a range to find the following sum:



## Calling an Item(element) in an Array

If you would like to pull a particular item in an array, we use the **item()** function as follows:

my\_array.item(index)

Each item in an array has index or location number on the array. The first item in an array has index 0.

Practice 8- Take the high\_temperatures array and complete the following:

1. Sort the array.
2. Find the difference between the smallest and largest item in the array using the **item()** function.

## Array Operations

Similar to numeric operations, we can perform operations with arrays of that have the same number of elements.

Practice 9- Create two arrays called my\_array1 and my\_array2 and complete the following table:

|  |  |
| --- | --- |
| Operation | Description |
| Addition, my\_array1+my\_array2 |  |
| Subtraction, my\_array1-my\_array2 |  |
| Multiplication, my\_array1\*my\_array2 |  |
| Division, my\_array1/my\_array2 |  |