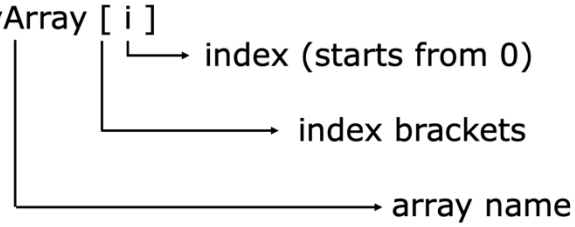
Information Analysis and Visualisation

Laboratory Session 2

The tasks in this session are focused on one-dimensional arrays. By definition, an array is a collection of elements of the same type. Each element can be accessed by its index, which is an integer value starting from zero. The Python library NumPy provides the necessary functionality to initialise and process arrays. As with other data-related problems, there are more than one way to solve these tasks.

- Accessing individual elements
  - MyArray [i]



Note:

NumPy arrays follow similar syntax as C/C++ and Java

Examples:

- Define one-dimensional array of 50 integers using the 'empty' method from the NumPy library
- This method does not initialise the elements
- The elements will have "random" values found in the memory locations at the time the 'empty' method is called

```
import numpy as np
# Define number of elements
NumberOfElements = 50
# Allocate memory
MyArray1 = np.empty(shape=NumberOfElements, dtype=int)
```

- Initialise an array with 10 random (pseudo-random) integer values
- The values will be within the interval [-5..5]

```
MyArray2 = np.random.randint(-5, 5, 10, dtype=int)
```

- In most cases, we use loops when we work with arrays

- We will use the following loops:
  - FOR
  - WHILE

- Use **FOR loop** to visualise all elements alongside their indices:

```
for i in range(0, 10):
    print(i, MyArray2[i])
```

- Use **WHILE loop** to achieve the same:

```
i = int(0)
while i < 10:
    print(i, MyArray2[i])
    i = i + 1
```

Tasks:

1. Initialise one-dimensional array A with 10 random integer numbers from the interval [0..100] and visualise the elements alongside their indices. Have in mind that NumPy arrays start from index zero.

2. Visualise the sum and the average value of the array

3. Visualise the minimum and maximum values alongside their indices. This can be achieved by using the NumPy functions minimum, maximum, argmax and argmin.

4. Initialise arrays B and C with values from array A arranged in ascending and descending order, respectively. Visualise arrays B and C.

5. Implement at least one of the Tasks 2 to 4 without using functions from the NumPy library, apart from initialising the array.

6. Visualise arrays A, B and C as a table. Display the indices of the arrays in the first column of the table.

7. Initialise array D with 10 integer values, such that the first value is 10 and every consecutive value is equal to the previous value plus 4. Display the array alongside its indices.

8. Guess the number (game)

(Extra task if there is available time left)

Write a scrip which generates a random integer value between 1 and 10. Use the 'random' and 'datetime' libraries in order to guarantee different random number every time the script is run.

In another cell, write a script which reads an integer value entered by the user and then compares it with the random value from the previous cell. If the entered value is larger or smaller than the random value, display an appropriate message. If the user entered the same value, congratulate them with a message. You can hide the generated random value from the user by scrolling down the Jupyter Notebook page.

For more information about the NumPy library:

1. NumPy official website:
https://numpy.org

2. NumPy Tutorial on W3 School:
https://www.w3schools.com/python/numpy/