Laboratory work № 1

“Data Analysis using Python Language”

Performed by:
student of group IM-14
Full name KULUBECİOGLU Mehmet

Compiled by:
Yulia Timofeeva

KYİV.2023

Lab work No.1

Working with NumPy arrays

Tasks:

Create a Python program that:

1. Generates random and non-random arrays in different ways specified in the theoretical information

In [10]:
```python
import numpy as np

# 1. Generate a random array with elements from 1 to 15 with step 2
random_array_1 = np.arange(3, 16, 3)
print("Random Array 1:")
print(random_array_1)

# 2. Generate an array of nine integer ones
non_random_array_1 = np.ones(9, dtype=int) * 2
print("Non-Random Array 1:")
print(non_random_array_1)

# 3. Generate a two-dimensional array with each row containing four zeros, with a floating-point data type
non_random_array_2 = np.zeros((2, 4), dtype=float) + 0.5
print("Non-Random Array 2:")
print(non_random_array_2)

# 4. Generate an array of five values spaced evenly between 0 and 2
random_array_2 = np.linspace(0, 2, 5)
print("Random Array 2:")
print(random_array_2)

# 5. Generate a 4x3 array of evenly distributed random numbers from 0 to 1
random_array_3 = np.random.random((4, 3)) * 10
print("Random Array 3:")
print(random_array_3)
```

```python
# 6. Generate a 4x4 array with random integers from 0 to 50
random_array_4 = np.random.randint(0, 50, (4, 4))
print("Random Array 4:")
print(random_array_4)

# 7. Generate an empty array with 5 elements (random content of memory cells)
empty_array = np.empty(5) * 3
print("Empty Array:")
print(empty_array)
```

```
Random Array 1:
[ 3  6  9 12 15]
Non-Random Array 1:
[2 2 2 2 2 2 2 2 2]
Non-Random Array 2:
[[0.5 0.5 0.5 0.5]
 [0.5 0.5 0.5 0.5]]
Random Array 2:
[0.  0.5 1.  1.5 2. ]
Random Array 3:
[[1.48442964 8.68516532 2.87690394]
 [1.32810386 1.40068923 3.20707493]
 [9.03162536 1.96064356 1.7393567 ]
 [6.16549209 5.76232823 4.35027025]]
Random Array 4:
[[42 18 14 37]
 [29 20 39  2]
 [23 10 12 27]
 [ 7 43 26  5]]
Empty Array:
[0.  1.5 3.  4.5 6. ]
```

2. Demonstrates access to array items using indexes, including negative ones; allocation of subarrays.

In [12]:
```python
import numpy as np

# 1. Generate a random array with elements from 1 to 15 with step 2
random_array = np.arange(1, 16, 2)
print("Original Random Array:")
print(random_array)

# 2. Demonstrates access to array items using indexes, including negative ones
print("\nAccessing Array Items:")
# First item
first_item = random_array[0]
print("First Item:", first_item)

# Second-to-last item
second_to_last_item = random_array[-2]
print("Second-to-last Item:", second_to_last_item)

# Last item
last_item = random_array[-1]
print("Last Item:", last_item)

# 3. Allocating subarrays
print("\nAllocating Subarrays:")
# Subarray 1: elements from index 2 to the end
subarray_1 = random_array[2:]
print("Subarray 1:")
print(subarray_1)
```

```python
# Subarray 2: elements up to index 5
subarray_2 = random_array[:5]
print("Subarray 2:")
print(subarray_2)

# Subarray 3: every other element
subarray_3 = random_array[::2]
print("Subarray 3:")
print(subarray_3)
```

```
Original Random Array:
[ 1  3  5  7  9 11 13 15]

Accessing Array Items:
First Item: 1
Second-to-last Item: 13
Last Item: 15

Allocating Subarrays:
Subarray 1:
[ 5  7  9 11 13 15]
Subarray 2:
[1 3 5 7 9]
Subarray 3:
[ 1  5  9 13]
```

3. Demonstrates basic arithmetic operations on arrays, as well as the work of methods reduce, accumulate, outer.

In [17]:
```python
import numpy as np

# Define two arrays
array_a = np.array([1, 2, 3, 4])
array_b = np.array([5, 6, 7, 8])

# Print arrays
print("Array a:")
print(array_a)

print("\nArray b:")
print(array_b)

# Basic arithmetic operations
print("\nBasic Arithmetic Operations:")
sum_result = array_a + array_b
print("Sum:")
print(sum_result)

product_result = array_a * array_b
print("Product:")
print(product_result)

difference_result = array_a - array_b
print("Difference:")
print(difference_result)
```

```python
# Reduce method result (sum of array a)
reduce_result = np.add.reduce(array_a)
print("\nReduce Method Result (Sum of array a):")
print(reduce_result)

# Other method result (running sum of arrays a and b)
accumulate_result = np.cumsum(array_a + array_b)
print("\nOther Method Result (Running Sum of arrays a and b):")
print(accumulate_result)

# Outer method result (outer product of arrays a and b)
outer_result = np.outer(array_a, array_b)
print("\nOuter Method Result (Outer product of arrays a and b):")
print(outer_result)
```

```
Array a:
[1 2 3 4]

Array b:
[5 6 7 8]

Basic Arithmetic Operations:
Sum:
[ 6  8 10 12]
Product:
[ 5 12 21 32]
Difference:
[-4 -4 -4 -4]
Division:
[0.2        0.33333333 0.42857143 0.5       ]

Reduce Method Result (Sum of array a):
10

Other Method Result (Running Sum of arrays a and b):
[ 6 14 24 36]

Outer Method Result (Outer product of arrays a and b):
[[ 5  6  7  8]
 [10 12 14 16]
 [15 18 21 24]
 [20 24 28 32]]
```

4. Calculates statistical characteristics: minimum and maximum values, sample mean, variance, standard deviation, median and 25 and 75 percentiles for petal width values (petal_width) from the iris flower data set (iris.csv).

In [18]:
```
pip install numpy pandas
```

```
Requirement already satisfied: numpy in c:\users\mehme\anaconda3\lib\site-packages (1.24.3)
Requirement already satisfied: pandas in c:\users\mehme\anaconda3\lib\site-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\mehme\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mehme\anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\mehme\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.
16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [21]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\mehme\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\mehme\anaconda3\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\mehme\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\mehme\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mehme\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [22]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris

# Load the Iris dataset from scikit-learn
iris = load_iris()
iris_data = pd.DataFrame(data=np.c_[iris['data'], iris['target']], columns=iris['feature_names'] + ['target'])

# Extract petal width values
petal_width = iris_data['petal width (cm)'].values

# Calculate statistical characteristics
minimum_value = np.min(petal_width)
maximum_value = np.max(petal_width)
mean_value = np.mean(petal_width)
variance_value = np.var(petal_width)
std_deviation_value = np.std(petal_width)
median_value = np.median(petal_width)
percentile_25 = np.percentile(petal_width, 25)
percentile_75 = np.percentile(petal_width, 75)

# Print the results
print("Statistical Characteristics for Petal Width:")
print(f"Minimum Value: {minimum_value}")
print(f"Maximum Value: {maximum_value}")
print(f"Sample Mean: {mean_value}")
print(f"Variance: {variance_value}")
print(f"Standard Deviation: {std_deviation_value}")
print(f"Median: {median_value}")
print(f"25th Percentile: {percentile_25}")
print(f"75th Percentile: {percentile_75}")
```

```
Statistical Characteristics for Petal Width:
Minimum Value: 0.1
Maximum Value: 2.5
Sample Mean: 1.1993333333333336
Variance: 0.5771328888888888
Standard Deviation: 0.7596926279021594
Median: 1.3
25th Percentile: 0.3
75th Percentile: 1.8
```