Lab 2 - Recap Numpy

Python list

```
daily_returns = [3, 4, 5, 6, 7] #lista
daily_returns[0] = 5 # modificarea valorii de pe o poziție
print(daily_returns)
print(daily_returns[0]) # acces la valoarea de pe o poziție
```

```
[5, 4, 5, 6, 7]
5
```

```
daily_returns_adjusted = daily_returns + daily_returns#* 2
print(daily_returns_adjusted)
```

```
[5, 4, 5, 6, 7, 5, 4, 5, 6, 7]
```

NumPy

https://numpy.org/learn/ https://numpy.org/doc/stable/

```
!pip install numpy
# !pip install {numele_librariei}
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
```

```
import numpy as np

# Creare vector numpy pornind de la o listă
lista = [3, 4, 5, 6, 7] # lista satndard Python
a = np.array(lista) # array din NumPy
a[0] = 5
print(a)
b = a + a
print(b)
```

```
[5 4 5 6 7]
[10  8 10 12 14]
```

`arange()`: https://numpy.org/doc/stable/reference/generated/numpy.arange.html#numpy-arange

```
#np.arange(10)
#np.arange(5, 10)
#np.arange(5, 10, 2)
a = np.arange(10, dtype=float)
print(a)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

```
a = np.arange(10, dtype=float)
#lista.sum()
#a.sum()
print(a)
print(a.sum())
print(a.max())
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
45.0
9.0
```

```
a = np.arange(10, dtype=float)
print(np.sqrt(a))
# SQRT dintr-un număr
import math
math.sqrt(9)
```

```
[0.         1.         1.41421356 1.73205081 2.         2.23606798
 2.44948974 2.64575131 2.82842712 3.        ]
3.0
```

`np.ones()`: construiește un vector cu toate elementle 1 `np.zeros()`: construiește un vector cu toate elementle 0

```
#a = np.ones(5)
a = np.zeros(5)
print(a)
```

```
[0. 0. 0. 0. 0.]
```

Other methods of defining a matrix: https://numpy.org/doc/stable/reference/routines.array-creation.html#routines-array-creation

Matrici

```
# Pornind de liste Python
lista_de_liste = [[1, 2, 3], [4, 5, 6]]
print(lista_de_liste)
# lista_de_liste + 2 # operația nu este suportată
m = np.array(lista_de_liste)
print(m + 2)
```

```
[[1, 2, 3], [4, 5, 6]]
[[3 4 5]
 [6 7 8]]
```

```
#m = np.zeros((2, 3))
m = np.full((2,3), 6)
print(m)
print(m.sum())
```

```
[[6 6 6]
 [6 6 6]]
36
```

```
lista_de_liste = [[1, 2, 3], [4, 5, 6]]
#print(lista_de_liste)
m = np.array(lista_de_liste)
print(m)
print(m.sum(axis=0)) # suma pe coloane
print(m.sum(axis=1)) # suma pe linii
```

```
[[1 2 3]
 [4 5 6]]
[5 7 9]
[ 6 15]
```

```
m.transpose()
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

```
m.flatten()
```

```
array([1, 2, 3, 4, 5, 6])
```

```
print(m)
# identificam elementele din matrice care respecta o conditie
print(np.where(m > 3, 1, 0))
# număr de elemnte din matrice mai marei decât 3
print(np.sum(np.where(m > 3, 1, 0)))
np.count_nonzero(np.where(m > 3, 1, 0))

m > 3
```

```
[[1 2 3]
 [4 5 6]]
[[0 0 0]
 [1 1 1]]
3
array([[False, False, False],
       [ True,  True,  True]])
```

Exercises

1. Compute the Average Stock Price

   Given an array of stock prices, compute the average price.

```
import numpy as np
prices = [100, 101, 102, 103, 104] # lista standard
prices_array = np.array(prices) # array din NumPy
print(prices_array.mean())
```

```
102.0
```

2. Count the Number of Days Stock Price Increased. Given an array of daily stock prices, count the number of days the stock price increased compared to the previous day.

```
import numpy as np

prices = np.array([100, 101, 102, 101, 105, 107])
#[100, 101, 102, 101, 105]
#[101, 102, 101, 105, 107]
prices_current_day = prices[:-1] #prices[0:-1] #prices[0:4]
prices_next_day = prices[1:]
print(prices_current_day)
print(prices_next_day)
comparison = prices_next_day > prices_current_day
print(comparison)
days_with_increase = np.sum(comparison)
print(days_with_increase)
difeference = prices_next_day - prices_current_day
print(difeference)
#
```

```
[100 101 102 101 105]
[101 102 101 105 107]
[ True  True False  True  True]
4
[ 1  1 -1  4  2]
```

3. Count the Number of Days Above a Threshold Given an array of prices, count how many days the price was above 103.

```
import numpy as np
prices = np.array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109])
days_over_103 = np.sum(prices > 103)
print(days_over_103)
```

```
6
```

7. Portfolio Value Given arrays of stock prices and the number of shares held, calculate the total portfolio value.

```
import numpy as np
prices_list = [100, 102, 102] # lista
prices = np.array(prices_list) # vector creat pe baza listei
#print(type(prices_list))
#print(type(prices))
shares = np.array([10, 20, 30])
shares_value = prices * shares
print(shares_value)
portfolio_value = np.sum(shares_value)
print(portfolio_value)
```

```
[1000 2040 3060]
6100
```

8. Correlation Between Two Stocks Given two arrays of daily returns for two stocks, compute their correlation coefficient.

   Hint: Use the `np.corrcoef` function to compute the correlation coefficient.

```
import numpy as np

stock_a_returns = np.array([0.05, 0.03, 0.02, -0.01, 0.04])
stock_b_returns = np.array([0.03, 0.02, -0.01, 0.01, 0.02])

correlation_coefficient = np.corrcoef(stock_a_returns, stock_b_returns)[1, 0]

print(correlation_coefficient)
```

```
0.5585123213730495
```

9. Calculate Daily Returns Given an array of closing prices for a stock, calculate the daily returns.

   Hint: To compute the **daily return** for a stock, use the following formula:

[\text{Daily Return} = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1]

Where:

- (P_t) is the **closing price** of the stock on day (t)
- (P_{t-1}) is the **closing price** on the previous trading day

Example: if a stock closed at **105 ron** today and at **100 ron** yesterday:

[\text{Daily Return} = \frac{105 - 100}{100} = \frac{5}{100} = 0.05 \text{ or } 5%]

```
import numpy as np

daily_prices = np.array([100, 101, 102, 103])
previous_day_prices = daily_prices[:-1]
next_day_prices = daily_prices[1:]
daily_returns = (next_day_prices - previous_day_prices) / previous_day_prices
print(daily_returns)
#
```

```
[0.01       0.00990099 0.00980392]
```