In [ ]:
```
Nurshanov Dias
Python for data analysis 3 Lab
IT3-2208
```

In [20]:
```
# 1 Exercise)
import numpy as np
from copy import copy
from pprint import pprint
zeros = np.zeros(10)
vowels = np.array(list('aeiou'))
ones = np.ones([2,5])
myarray1 = np.array([
    [2.7, -2, -19],
    [0, 3.4, 99.9],
    [10.6, 0, 13],
])
myarray2 = np.arange(4, 4 * 15 + 1, 4, dtype=float).reshape(3, 5)

print("a) ones divided by 3:\n", ones/3)

# Task b) Add the arrays myarray1 and myarray2 (shape mismatch, so slice myarray2)
myarray2_sliced = myarray2[:, :3]
myarray_add = myarray1 + myarray2_sliced
print("\nb) Addition of myarray1 and myarray2_sliced:\n", myarray_add)

# Task c) Subtract myarray1 from myarray2 (shape mismatch, so slice myarray2)
myarray_subtract = myarray2_sliced - myarray1
print("\nc) Subtraction of myarray1 from myarray2_sliced:\n", myarray_subtract)

# Task d) Multiply myarray1 and myarray2 element-wise (shape mismatch, so slice myarray2)
myarray_multiply = myarray1 * myarray2_sliced
print("\nd) Element-wise multiplication of myarray1 and myarray2_sliced:\n", myarray_multiply)

# Task e) Matrix multiplication of myarray1 and myarray2 (requires matching inner dimensions)
# We'll slice the necessary part of myarray2 to match for matrix multiplication
myarray2_sliced_for_dot = myarray2[:, :3]
myarray3 = np.dot(myarray1, myarray2_sliced_for_dot.T)
print("\ne) Matrix multiplication of myarray1 and myarray2_sliced_for_dot.T:\n", myarray3)

# Task f) Divide myarray1 by myarray2 (shape mismatch, so slice myarray2)
myarray_divide = myarray1 / myarray2_sliced
print("\nf) Division of myarray1 by myarray2_sliced:\n", myarray_divide)

# Task g) Find the cube of all elements of myarray1 and divide by 2.
myarray1_cube_div_2 = (myarray1 ** 3) / 2
print("\ng) Cube of myarray1 divided by 2:\n", myarray1_cube_div_2)

# Task h) Find the square root of all elements of myarray2 and divide by 2, rounded to 2 decimal places.
myarray2_sqrt_div_2 = np.around(np.sqrt(myarray2) / 2, decimals=2)
print("\nh) Square root of myarray2 divided by 2 and rounded to 2 decimals:\n", myarray2_sqrt_div_2)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] 10
['a' 'e' 'i' 'o' 'u'] 5
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]] 2
[[  2.7  -2.  -19. ]
 [  0.    3.4  99.9]
 [ 10.6   0.   13. ]] 3
[[ 4.  8. 12. 16. 20.]
 [24. 28. 32. 36. 40.]
 [44. 48. 52. 56. 60.]] 3
a) ones divided by 3:
 [[0.33333333 0.33333333 0.33333333 0.33333333 0.33333333]
 [0.33333333 0.33333333 0.33333333 0.33333333 0.33333333]]

b) Addition of myarray1 and myarray2_sliced:
 [[  6.7   6.   -7. ]
 [ 24.   31.4 131.9]
 [ 54.6  48.   65. ]]

c) Subtraction of myarray1 from myarray2_sliced:
 [[  1.3  10.   31. ]
 [ 24.   24.6 -67.9]
 [ 33.4  48.   39. ]]

d) Element-wise multiplication of myarray1 and myarray2_sliced:
 [[  10.8  -16.  -228. ]
 [   0.    95.2 3196.8]
 [ 466.4    0.   676. ]]

e) Matrix multiplication of myarray1 and myarray2_sliced_for_dot.T:
 [[-233.2 -599.2 -965.2]
 [1226.  3292.  5358. ]
 [ 198.4  670.4 1142.4]]

f) Division of myarray1 by myarray2_sliced:
 [[ 0.675      -0.25       -1.58333333]
 [ 0.          0.12142857  3.121875  ]
 [ 0.24090909  0.          0.25      ]]

g) Cube of myarray1 divided by 2:
 [[ 9.841500e+00 -4.000000e+00 -3.429500e+03]
 [ 0.000000e+00  1.965200e+01  4.985015e+05]
 [ 5.955080e+02  0.000000e+00  1.098500e+03]]

h) Square root of myarray2 divided by 2 and rounded to 2 decimals:
 [[1.   1.41 1.73 2.   2.24]
 [2.45 2.65 2.83 3.   3.16]
 [3.32 3.46 3.61 3.74 3.87]]
```

In [46]:

```python
import numpy as np

# 2 Exercise)
myarray4 = np.arange(-1, -1 + 14 * 3 * 0.25, 0.25).reshape(14, 3)
print("Original myarray4:\n", myarray4)

# Split the array into 3 parts
myarray4s = np.array_split(myarray4, 3)
print("\nSplit parts of myarray4:")
for idx, part in enumerate(myarray4s):
    print(f"\nPart {idx + 1}:\n", part)

# a) Find the sum of all elements.
total_sum = np.sum(myarray4)
print(f'\na) Find the sum of all elements: {total_sum}')

# b) Find the sum of all elements row wise.
sum_row_wise = [np.sum(row) for part in myarray4s for row in part]
print(f'\nb) Find the sum of all elements row wise:')
print(*sum_row_wise, sep='\n')

# c) Find the sum of all elements column wise.
sum_col_wise = np.sum([np.sum(part, axis=0) for part in myarray4s], axis=0)
print(f'\nc) Find the sum of all elements column wise:')
print(sum_col_wise)

# d) Find the max of all elements.
max_value = np.max(myarray4)
print(f'\nd) Max of all elements: {max_value}')

# e) Find the min of all elements in each row.
min_row_wise = [np.min(row) for part in myarray4s for row in part]
print(f'\ne) Min of all elements in each row:')
print(*min_row_wise, sep='\n')
```

```python
# f) Find the mean of all elements in each row.
mean_row_wise = [np.mean(row) for part in myarray4s for row in part]
print(f'\nf) Mean of all elements in each row:')
print(*mean_row_wise, sep='\n')

# g) Find the standard deviation column wise.
std_col_wise = np.std(myarray4, axis=0)
print(f'\ng) Standard deviation column wise:')
print(std_col_wise)
```

```
Original myarray4:
 [[-1.   -0.75 -0.5 ]
 [-0.25  0.    0.25]
 [ 0.5   0.75  1.  ]
 [ 1.25  1.5   1.75]
 [ 2.    2.25  2.5 ]
 [ 2.75  3.    3.25]
 [ 3.5   3.75  4.  ]
 [ 4.25  4.5   4.75]
 [ 5.    5.25  5.5 ]
 [ 5.75  6.    6.25]
 [ 6.5   6.75  7.  ]
 [ 7.25  7.5   7.75]
 [ 8.    8.25  8.5 ]
 [ 8.75  9.    9.25]]

Split parts of myarray4:

Part 1:
 [[-1.   -0.75 -0.5 ]
 [-0.25  0.    0.25]
 [ 0.5   0.75  1.  ]
 [ 1.25  1.5   1.75]
 [ 2.    2.25  2.5 ]]

Part 2:
 [[2.75 3.   3.25]
 [3.5  3.75 4.  ]
 [4.25 4.5  4.75]
 [5.   5.25 5.5 ]
 [5.75 6.   6.25]]

Part 3:
 [[6.5  6.75 7.  ]
 [7.25 7.5  7.75]
 [8.   8.25 8.5 ]
 [8.75 9.   9.25]]

a) Find the sum of all elements: 173.25

b) Find the sum of all elements row wise:
-2.25
0.0
2.25
4.5
6.75
9.0
11.25
13.5
15.75
18.0
20.25
22.5
24.75
27.0

c) Find the sum of all elements column wise:
[54.25 57.75 61.25]

d) Max of all elements: 9.25

e) Min of all elements in each row:
-1.0
-0.25
0.5
1.25
2.0
2.75
3.5
4.25
5.0
5.75
6.5
```

```
7.25
8.0
8.75

f) Mean of all elements in each row:
-0.75
0.0
0.75
1.5
2.25
3.0
3.75
4.5
5.25
6.0
6.75
7.5
8.25
9.0

g) Standard deviation column wise:
[3.02334666 3.02334666 3.02334666]
```

In [62]:

```python
import numpy as np

data = np.genfromtxt('iris.txt', delimiter=',', dtype=[('f0', 'f4'), ('f1', 'f4'), ('f2', 'f4'), ('f3', 'f4'),

numeric_data = np.array([list(row)[:4] for row in data])

iris_max = np.around(np.max(numeric_data, axis=0), 2)  # Maximum values column-wise
iris_min = np.around(np.min(numeric_data, axis=0), 2)  # Minimum values column-wise
iris_avg = np.around(np.mean(numeric_data, axis=0), 2)  # Mean values column-wise
iris_std = np.around(np.std(numeric_data, axis=0), 2)  # Standard deviation column-wise
iris_var = np.around(iris_std ** 2, 2)  # Variance = square of the standard deviation

print("Overall statistics:\n")
print("Max values (column-wise):", iris_max)
print("Min values (column-wise):", iris_min)
print("Mean values (column-wise):", iris_avg)
print("Standard Deviation (column-wise):", iris_std)
print("Variance (column-wise):", iris_var)

print('2 exercise')
iris1 = np.array([row for row in data if row['species'] == 'Iris-setosa'], dtype=data.dtype)
iris2 = np.array([row for row in data if row['species'] == 'Iris-versicolor'], dtype=data.dtype)
iris3 = np.array([row for row in data if row['species'] == 'Iris-virginica'], dtype=data.dtype)

def compute_stats(iris_data):
    numeric_data = np.array([list(row)[:4] for row in iris_data])

    iris_max = np.around(np.max(numeric_data, axis=0), 2)
    iris_min = np.around(np.min(numeric_data, axis=0), 2)
    iris_mean = np.around(np.mean(numeric_data, axis=0), 2)
    iris_std = np.around(np.std(numeric_data, axis=0), 2)

    return iris_max, iris_min, iris_mean, iris_std

iris1_max, iris1_min, iris1_mean, iris1_std = compute_stats(iris1)
iris2_max, iris2_min, iris2_mean, iris2_std = compute_stats(iris2)
iris3_max, iris3_min, iris3_mean, iris3_std = compute_stats(iris3)

print(f"\nIris-setosa stats:\n Max: {iris1_max}\n Min: {iris1_min}\n Mean: {iris1_mean}\n Std: {iris1_std}")
print(f"\nIris-versicolor stats:\n Max: {iris2_max}\n Min: {iris2_min}\n Mean: {iris2_mean}\n Std: {iris2_std}"
print(f"\nIris-virginica stats:\n Max: {iris3_max}\n Min: {iris3_min}\n Mean: {iris3_mean}\n Std: {iris3_std}")

print('3 exercise')

features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

comparison_table = {}

comparison_table['Iris-setosa'] = iris1_min > iris_min
comparison_table['Iris-versicolor'] = iris2_min > iris_min
comparison_table['Iris-virginica'] = iris3_min > iris_min

# Output the comparison table
print("\nComparison of species' minimum values against the dataset's minimum values:")
for species, comparisons in comparison_table.items():
    print(f"\n{species}:")
    for i, feature in enumerate(features):
        print(f"{feature}: {comparisons[i]}")
```

```
Overall statistics:

Max values (column-wise): [7.9 4.4 6.9 2.5]
Min values (column-wise): [4.3 2.  1.  0.1]
Mean values (column-wise): [5.84 3.05 3.76 1.2 ]
Standard Deviation (column-wise): [0.83 0.43 1.76 0.76]
Variance (column-wise): [0.69 0.18 3.1  0.58]

Iris-setosa stats:
 Max: [5.8 4.4 1.9 0.6]
 Min: [4.3 2.3 1.  0.1]
 Mean: [5.01 3.42 1.46 0.24]
 Std: [0.35 0.38 0.17 0.11]

Iris-versicolor stats:
 Max: [7.  3.4 5.1 1.8]
 Min: [4.9 2.  3.  1. ]
 Mean: [5.94 2.77 4.26 1.33]
 Std: [0.51 0.31 0.47 0.2 ]

Iris-virginica stats:
 Max: [7.9 3.8 6.9 2.5]
 Min: [4.9 2.2 4.5 1.4]
 Mean: [6.59 2.97 5.55 2.03]
 Std: [0.63 0.32 0.55 0.27]

Comparison of species' minimum values against the dataset's minimum values:

Iris-setosa:
sepal_length: False
sepal_width: True
petal_length: False
petal_width: False

Iris-versicolor:
sepal_length: True
sepal_width: False
petal_length: True
petal_width: True

Iris-virginica:
sepal_length: True
sepal_width: True
petal_length: True
petal_width: True
```

In [ ]: