Język Python

Zadanie 1

```
def pierwiastki(a, b = 0, c = 0):
    delta = b ** 2 - 4 * a * c
    if delta > 0:
        x1 = (-b - delta ** 0.5) / (2 * a)
        x2 = (-b + delta ** 0.5) / (2 * a)
        return x1, x2
    if delta == 0:
        x = -b / (2 * a)
        return x
    return None
```

```
pierwiastki(2, -5, 2)
```

```
(0.5, 2.0)
```

Zadanie 2

```
def guess_number():
    import random
    number = random.randint(1, 100)
    while True:
        guess = int(input("Podaj liczbę: "))
        if guess == number:
            print("Zgadłeś!")
            break
        elif guess < number:
            print("Za mała")
        else:
            print("Za duża")
```

```
guess_number()
```

```
Podaj liczbę: 50
Za duża
Podaj liczbę: 30
Za mała
Podaj liczbę: 40
Za duża
Podaj liczbę: 35
Za mała
Podaj liczbę: 37
Zgadłeś!
```

Zadanie 3

```
def dna_len(dna):
    return len(dna)

dna_len("TTAGTAGGGTTGTCCAT")
```

```
17
```

```
def dna_check(dna):
    for char in dna:
        if char not in "ATGC":
            return False
    return True

#dna_check("TTAGTAGGGTTGTCCAT")
dna_check("RTTAGTAGGGTTGTCCAT")
```

```
False
```

```
def dna_replace(dna):
    new_dna = ""
    for char in dna:
        if char == "A":
            new_dna += "T"
        elif char == "T":
            new_dna += "A"
        elif char == "G":
            new_dna += "C"
        elif char == "C":
            new_dna += "G"
    return new_dna


dna_replace("ATCGAA")
```

```
'TAGCTT'
```

```
def dna_reverse(dna):
    return dna[::-1]

dna_reverse("ATCGAT")
```

```
'TAGCTA'
```

```
dna = "AATTGGCCATGC"
dna[5:7]
```

```
'GC'
```

```
def dna_gg(dna):
    count = 0
    for i in range(len(dna) - 1):
        if dna[i:i+2] == "GG":
            count += 1
    return count

dna_gg("RTGTAGTAGGAAGGTGTCCATGG")
```

```
3
```

```
def dna_count(dna, str):
    count = 0
    for i in range(len(dna) - len(str) + 1):
        if dna[i:i+len(str)] == str:
            count += 1
    return count

dna_count("RTTAGTAGGATATATGTCCAT", "ATA")
```

```
2
```

Biblioteka Numpy

Zadanie 1

```
import numpy as np

matrix = np.random.randint(0,100,size = (10,10))
print(matrix)

print(np.max(matrix))
print(np.min(matrix))
```

```
[[86 80 69 39 18 83 51  7 58  8]
 [24 30 38 40 67 68 69 51  5 72]
 [ 3 49 30 38 57 56 68 99 83 18]
 [36 60 70 72  3 16 70 35 19 53]
 [24  7 20 65  3 10 48 12 84 14]
 [39 63  4 39 58 34 23 81 98 46]
 [56 84 51 55 57 42 81 62 83 65]
 [49 16 62  3 18 67 58 96 90 43]
 [73 22 70 42 60 78 51  0 56 82]
 [82  8  6 59  9 60 87 53 71 69]]
99
0
```

Zadanie 2

```
def max_in_row(matrix):
    return np.max(matrix, axis = 1)


def max_in_col(matrix):
    return np.max(matrix, axis = 0)


matrix = np.random.randint(0,100,size = (10,10))
print(matrix)
print("Max w wierszach: ", max_in_row(matrix))
print("Max w kolumnach: ", max_in_col(matrix))
```

```
[[90  3 79  4 33 69 48 64 95 51]
 [ 6 25 62 50 78 40 89 67 85 78]
 [43 55 42 96 55 14 25 51 39 31]
 [62 22 43 18  6 18 23 47 84 31]
 [47 50  7 10  6 47 45 73 24 63]
 [79 59 94 74 66 26  9 90 12  7]
 [91 95 46 61 18  4 10 21 17 32]
 [43 86 16 26 78 34 68 36 94 52]
 [79 21 92 94 45 17 34 88 79 38]
 [17 69 40 74 25 93 14 10 27 51]]
Max w wierszach:  [95 89 96 84 73 94 95 94 94 93]
Max w kolumnach:  [91 95 94 96 78 93 89 90 95 78]
```

Zadanie 3

```
def matrix_def():
    matrix = np.zeros((5,5))
    for i in range(5):
        for j in range(5):
            if i == 0 or i == 4 or j == 0 or j == 4:
                matrix[i,j] = 1
            else:
                matrix[i,j] = 0
    return matrix

def zero_to_one(matrix):
    for i in range(5):
        for j in range(5):
            if matrix[i,j] == 0:
                matrix[i,j] = 1
            else:
                matrix[i,j] = 0
    return matrix


print(matrix_def())

matrix = matrix_def()
print(zero_to_one(matrix))
```

```
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
[[0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0.]]
```

Zadanie 4

```
def matrix_sum(matrix1, matrix2):
    return matrix1 + matrix2

def matrix_sub(matrix1, matrix2):
    return matrix1 - matrix2

def matrix_mul(matrix1, matrix2):
    return np.dot(matrix1, matrix2)

def matrix_2_mul(matrix):
    return (matrix * 2)/ 10

def matrix_print_mid(matrix):
    print(matrix[1:4,1:4])

def matrix_to_vector(matrix):
    return matrix.flatten()


matrix1 = np.random.randint(0,100,size = (5,5))
matrix2 = np.random.randint(0,100,size = (5,5))
print(matrix1)
print(matrix2)
```

```
[[83  9 80 27 64]
 [ 7 38  4 55 21]
 [43 31 13 80 94]
 [57  0 64 51 25]
 [53 70 27 57 16]]
[[79 13 19 23 56]
 [38 78 51  0 22]
 [51  1 72 52 86]
 [93 31 54 17 48]
 [62 24  0 67 51]]
```

```
print(matrix_sum(matrix1, matrix2))
```

```
[[ 65 106  51 113 102]
 [ 55  87  59 118  50]
 [110 125  90 152  19]
 [ 72  81  55  58 122]
 [145  42  87  86  76]]
```

```
print(matrix_sub(matrix1, matrix2))
```

```
[[-17  64 -23 -71  96]
 [ 55 -75 -45 -40  -4]
 [ 20  37 -86 -38  17]
 [-44  17 -39 -42  68]
 [ 15 -38  65  36 -74]]
```

```
print(matrix_mul(matrix1, matrix2))
```

```
[[17458  4234  9254 10816 16286]
 [ 8618  5268  5329  2711  5283]
 [18506  7726  7654  9323 12842]
 [14060  2986  8445  7181 12419]
 [14517  8327  9599  4664 10382]]
```

```
print(matrix_2_mul(matrix1))
```

```
[[ 4.8 17.   2.8  4.2 19.8]
 [11.   1.2  1.4  7.8  4.6]
 [13.  16.2  0.4 11.4  3.6]
 [ 2.8  9.8  1.6  1.6 19. ]
 [16.   0.4 15.2 12.2  0.2]]
```

```
matrix_print_mid(matrix1)
```

```
[[38  4 55]
 [31 13 80]
 [ 0 64 51]]
```

```
print(matrix_to_vector(matrix1))
```

```
[83  9 80 27 64  7 38  4 55 21 43 31 13 80 94 57  0 64 51 25 53 70 27 57
 16]
```