# I. NumPy.

**При реалізації завдань потрібно використовувати методи з бібліотеки numpy.**

**Рішення з циклами будуть оцінені в 0 балів.**

1. **(2б)** Імпортувати **numpy** загальноприйнятим способом.

In [63]:

```
import numpy as np
```

1. **(3б)** Визначити список з оцінками студента(ки) за семестр. Використовуючи його, створити одновимірний масив **numpy**.

In [4]:

```
grades = [87, 96, 62, 92, 77, 93, 66]
grades_array = np.array(grades)
```

1. **(4б)** Визначити список зі списками, де кожен рядок списку формально означає магазин, а кожна колонка - певний продукт (батон, яйця першої категорії, молоко та ковбаса), а елементами на пересічі є ціни на ці продукти відповідно. Використовуючи цей список, створити матрицю (двовимірний масив) **numpy**.

In [68]:

```
prices = [
    [33,   12.12, 47,    6.20],
    [32,   11,    59.41, 7.44],
    [30.4, 13.6,  61,    4.23]
]
prices_matrix = np.array(prices)
print(prices_matrix)
```

```
[[33.   12.12 47.    6.2 ]
 [32.   11.   59.41  7.44]
 [30.4  13.6  61.    4.23]]
```

1. **(3б)** Отримайте тип даних значень з масивів завдань **2** та **3.** Виведіть їх на екран.

In [69]:

```
print("grades_array elem  is", grades_array.dtype)
print("prices_matrix elem is", prices_matrix.dtype)
```

```
grades_array elem  is int32
prices_matrix elem is float64
```

1. **(3б)** Отримайте форми (кортеж з кількістю рядків та колонок) масивів завдань **2** та **3.** Виведіть їх на екран.

In [12]:

```
print("Shape of grades_array  is", grades_array.shape)
print("Shape of prices_matrix is", prices_matrix.shape)
```

```
Shape of grades_array  is (7,)
Shape of prices_matrix is (3, 4)
```

1. **(5б)** Поясніть, чому для масиву із завдання **2** ми отримали саме такий результат. На основі того ж самого

масиву з цілими числами-оцінками створіть вектор-стовбець (не вектор-рядок!).

На вхід ми використали одновимірний список, його форма це **7**, але оскільки ***shape*** повертає кортеж, то виходить **(7,)**

In [62]:

```
column_vector = grades_array.reshape(-1, 1)
print("Column vector shape is", column_vector.shape)
print(column_vector)
```

```
Column vector shape is (7, 1)
[[87]
 [96]
 [62]
 [92]
 [77]
 [93]
 [66]]
```

1. **(6б)** Тезисно поясніть різницю між **Python** списком та **NumPy** масивом.

Стандартні списки можуть зберігати елементи різних типів, проте операції на них виконуються повільніше і вони займають більше пам'яті. Також над **NumPy** масивами можна виконувати матричні та векторні операції.

1. **(7б)** Створіть одновимірний масив за допомогою спеціальної функції **numpy**, який би відображав динаміку стабільно зростаючого (з рівними проміжками) прибутку з продажів за тиждень, де у перший день не було продажів, а в останній день тижня вдалося заробити **1000** грн **50** коп.

In [22]:

```
profits = np.linspace(0, 1000.50, 7)
profits = list(profits)
print(profits)
```

```
[0.0, 166.75, 333.5, 500.25, 667.0, 833.75, 1000.5]
```

1. **(8б)** Створіть два масиви. Використовуючи їх, продемонструйте відмінність вертикального та горизонтального об'єднання масивів.

In [24]:

```
array1 = np.array([0, 1, 2])
array2 = np.array([7, 8, 9])

vertical_join = np.vstack((array1, array2))
print("Vertical Join:\n", vertical_join)

horizontal_join = np.hstack((array1, array2))
print("Horizontal Join:\n", horizontal_join)
```

```
Vertical Join:
 [[0 1 2]
 [7 8 9]]
Horizontal Join:
 [0 1 2 7 8 9]
```

1. **(12б)** Визначити функцію, яка приймає на вході масив і транспонує його. Скористайтесь методом **reshape**. І не забудьте почати з **docstrings**.

In [40]:

```
def transpose_array(array):
```

```
    """
    This function transposes the input 1D array to a column.

    Parameters:
    array (numpy.array): The input array.

    Returns:
    numpy.array: The transpose of the input array.
    """
    return array.reshape(-1, 1)
print(transpose_array(array1))
```

```
[[0]
 [1]
 [2]]
```

1. **(7б)** Створіть два масиви. Використовучи ці масиви, продемонструйте **1)** операцію по-елементного додавання, **2)** операцію по-елементного віднімання, **3)** множення масиву на число, **4)** операцію по-елементного множення, і **5)** матричного множення.

In [41]:

```
addition = array1 + array2
print("Addition:\n", addition)

subtraction = array1 - array2
print("Subtraction:\n", subtraction)

multiplication_by_number = array1 * 2
print("Multiplication by a Number:\n", multiplication_by_number)

elementwise_multiplication = array1 * array2
print("Multiplication:\n", elementwise_multiplication)

array1 = array1.reshape(1,3)
array2 = array2.reshape(3,1)
matrix_multiplication = np.matmul(array1, array2)
print("Matrix Multiplication:\n", matrix_multiplication)
```

```
Addition:
 [ 7  9 11]
Subtraction:
 [-7 -7 -7]
Multiplication by a Number:
 [0 2 4]
Multiplication:
 [ 0  8 18]
Matrix Multiplication:
 [[26]]
```

1. **(7б)** Створіть двовимірний масив (матрицю) та розрахуйте: **1)** мінімальне число, **2)** максимальне число, **3)** суму чисел, **4)** мінімальні числа для кожного рядка, **5)** максимальні числа для кожного стовпчика.

In [44]:

```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])

min_number = np.min(matrix)
print("Minimum: ", min_number)

max_number = np.max(matrix)
print("Maximum: ", max_number)

sum_numbers = np.sum(matrix)
print("Sum: ", sum_numbers)

min_numbers_rows = np.min(matrix, axis=1)
print("Row Minimums: ", min_numbers_rows)
```

```
max_numbers_columns = np.max(matrix, axis=0)
print("Col Maximums: ", max_numbers_columns)
```

```
Minimum:  1
Maximum:  12
Sum:  78
Row Minimums:  [ 1  4  7 10]
Col Maximums:  [10 11 12]
```

1. **(6б)** Використовуючи двовимірний масив з попереднього завдання, отримайте значення першого та другого стовпчика всіх рядків, окрім першого та останнього.

In [45]:

```
subset = matrix[1:-1, :2]
print(subset)
```

```
[[4 5]
 [7 8]]
```

1. **(7б)** Створіть матрицю , де деякі елементи повторюються. Знайдіть унікальні значення елементів цієї матриці та їхні частоти.

In [50]:

```
matrix = np.array([[5, 2, 2], [3, 3, 4], [1, 4, 3], [4, 4, 5]])

unique_elements, frequencies = np.unique(matrix, return_counts=True)

print("Unique:      ", unique_elements)
print("Frequencies: ", frequencies)
```

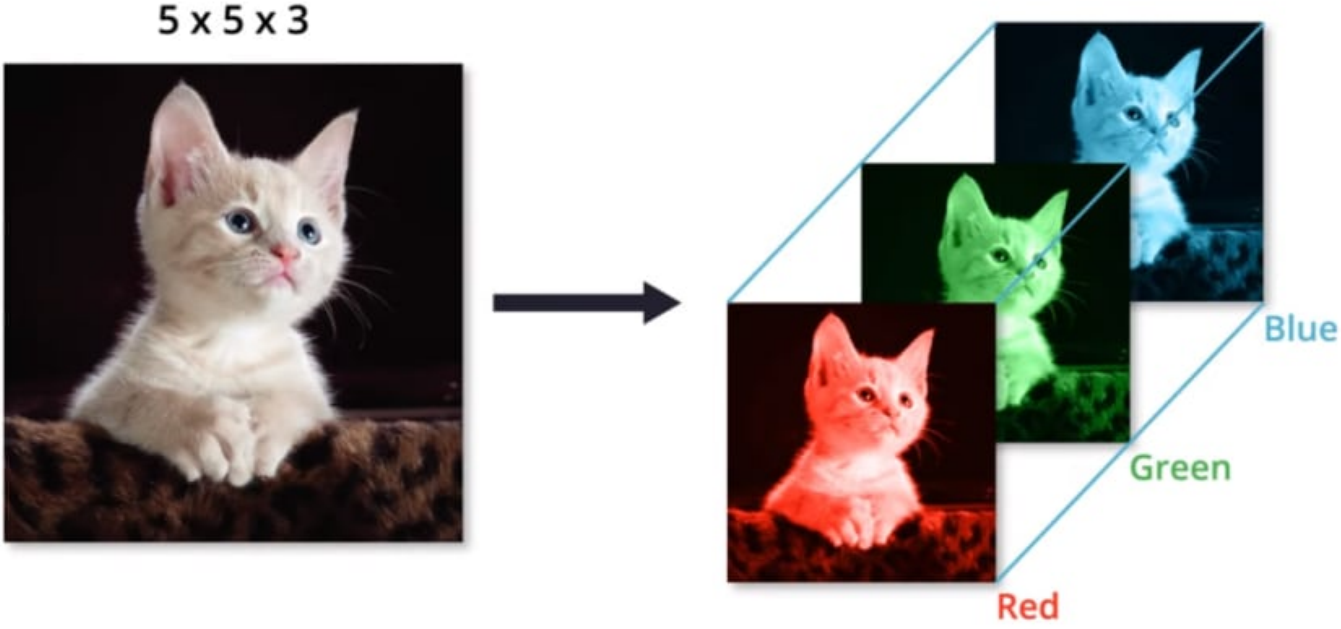
```
Unique:       [1 2 3 4 5]
Frequencies:  [1 2 3 4 2]
```

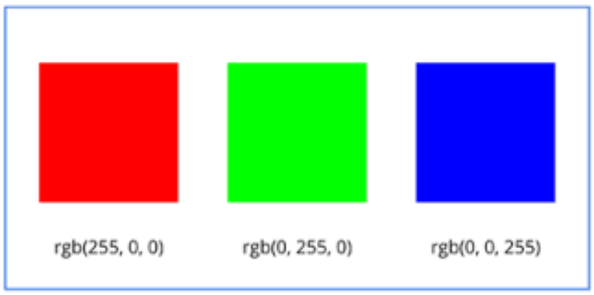
1. **(20б)** Тривимірні масиви (тензори) дуже широко використовуються для вирішення задач комп'ютерного зору, де один вимір відповідає за висоту зображення у пікселях, другий - за ширину зображення у пікселях, а третій - **RGB** шар. Значення такого масиву зазвичай знаходяться у проміжку від **0** до **255**, що є позначенням для інтенсивності того чи іншого кольору (див. зображення нижче). Пропоную вам, маючи знання з **numpy**, торкнутись до світу роботи з особливим типом даних - зображеннями. Сподіваюсь, когось з вас це зацікавить.

Завдання:

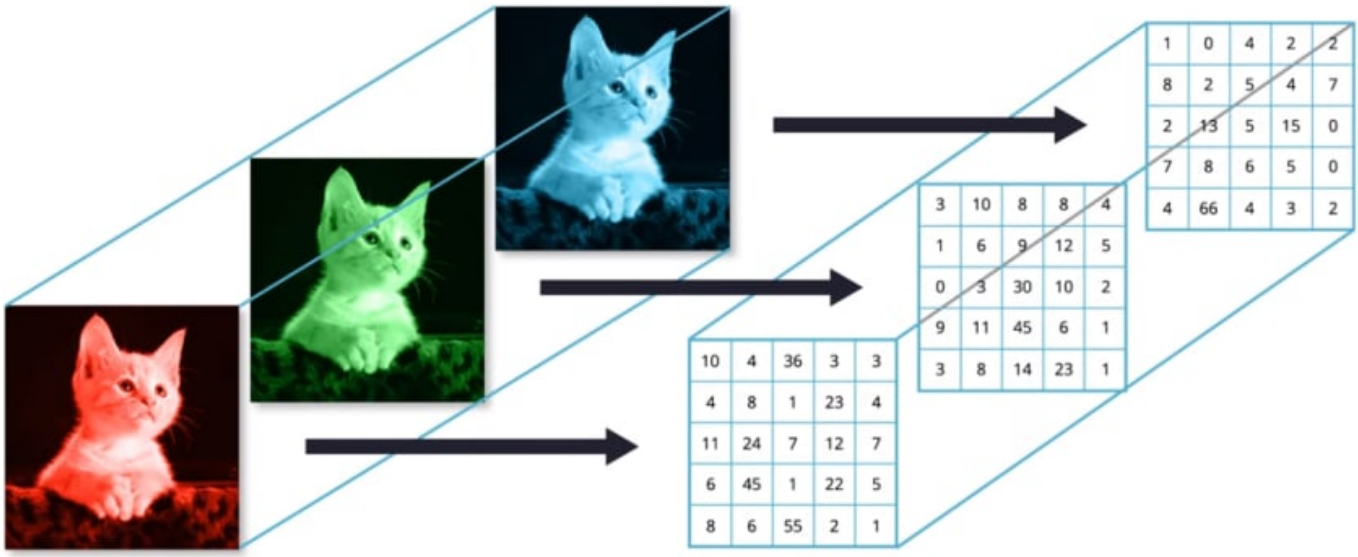
1. Визначити змінні для висоти та ширини майбутнього зображення. Присвоїти їм значення **480** та **720** відповідно.
2. Визначте змінну для зображення, яка прийматиме тривимірний масив відповідного розміру з пустими значеннями, які ми заповнимо пізніше. Тип даних масиву вкажіть **np.uint8** (це спеціальний тип даних для шкали пікселів зображень).
3. Розділимо зображення навпіл по висоті. Для цього обчисліть серединний піксель та збережіть його у змінну. Будьте уважні, це число буде використовуватися для слайсингу, де крайні значення не включаються у діапазон.
4. Розфарбуємо верхню частину зображення. За допомогою слайсингу на змінній з зображенням, виділіть першу половину пікселів по висоті, залишаючи всі пікселі в рядках по ширині та глибині (всі **3 RGB** шари). Та призначте їй кортеж **(0, 87, 184)**, що позначає колір у трьох **RGB** шарах відповідно.
5. Тепер розфарбуємо нижню частину зображення. За допомогою слайсингу на змінній з зображенням, виділіть другу половину пікселів по висоті, залишаючи всі пікселі в рядках по ширині та глибині (всі **3 RGB** шари). Та призначте їй кортеж **(255, 215, 0)**, що позначає колір у трьох **RGB** шарах відповідно. Вже здогадуєтесь, що за зображення ви створюєте**?)**
6. На останньому рядку комірки просто введіть назву змінної з зображенням.

5 x 5 x 3



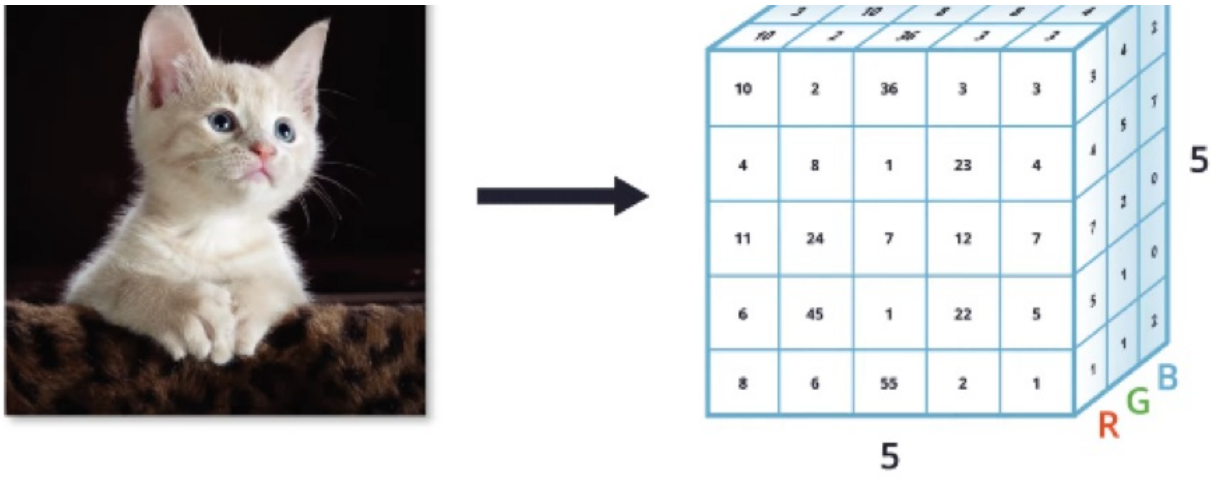

RGB Image




TensorFlow

5 x 5 x 3

3D Array




(c) Андрей Шмиг

In [67]:

```
height = 480
width = 720

image = np.zeros((height, width, 3), dtype = np.uint8)

center = height // 2

image[:center] = (0, 87, 184)

image[center:] = (255, 215, 0)

image
```

Out[67]:

```
array([[[  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184],
        ...,
        [  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184]],

       [[  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184],
        ...,
        [  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184]],

       [[  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184],
        ...,
        [  0,  87, 184],
        [  0,  87, 184],
        [  0,  87, 184]],

       ...,

       [[255, 215,   0],
        [255, 215,   0],
        [255, 215,   0],
        ...,
        [255, 215,   0],
```

```
        [255, 215,   0],
        [255, 215,   0]],

       [[255, 215,   0],
        [255, 215,   0],
        [255, 215,   0],
        ...,
        [255, 215,   0],
        [255, 215,   0],
        [255, 215,   0]],

       [[255, 215,   0],
        [255, 215,   0],
        [255, 215,   0],
        ...,
        [255, 215,   0],
        [255, 215,   0],
        [255, 215,   0]]], dtype=uint8)
```

## Бонус для тих, хто виконав останнє завдання.

Слава Україні! 

## Вітаю! Ви велика(ий) молодець, що впоралась(вся). Похваліть себе та побалуйте чимось приємним. Я Вами пишаюся.