

## Домашнє завдання №24\_2

Виконати домашнє завдання №24\_1 повторно за допомогою Python.

\* зверніть увагу, що блоки коду в Python виділяються відступами, тому будьте уважні при редагуванні наведеного далі прикладу коду: Python 3 забороняє змішування табуляції з пробілами у відступах.

\* **коментар:** далі наводиться приклад повністю виконаного завдання; для компіляції і запуску можна використати <https://replit.com/>, нижче також показаний спосіб виконання наведеного прикладу коду за допомогою цього засобу.

### Вибір варіанту

$$(N_{\text{ж}} + N_{\text{г}} + 1) \% 10 + 1$$

де:  $N_{\text{ж}}$  – порядковий номер студента в групі, а  $N_{\text{г}}$  – номер групи (1,2,3,4,5,6,7,8 або 9)

### Варіанти завдань

Номер **варіанту** відповідає максимально допустимій кількості облич (MAX\_FACES\_COUNT), які можуть бути виявлені на фото.

Спосіб виконання наведеного прикладу коду за допомогою

<https://replit.com/@scottatreplit/Python-OpenCV>

Спочатку створюємо проект на основі шаблону (рис. 0.1 та рис. 0.1), для якого вже виконано налаштування для OpenCV.

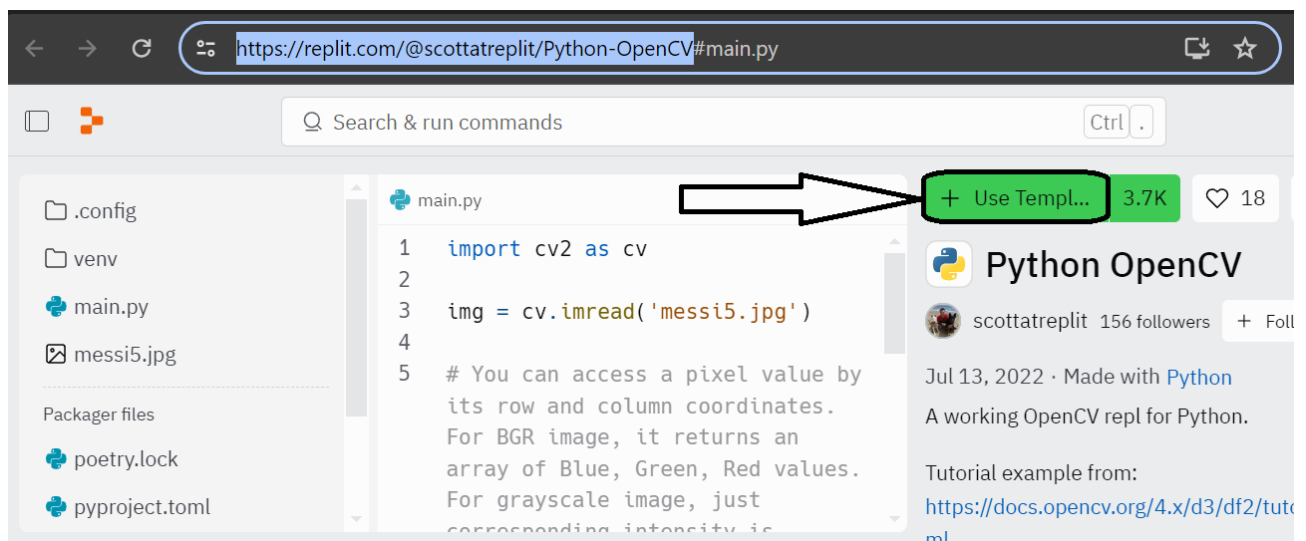


Рис. 0.1.

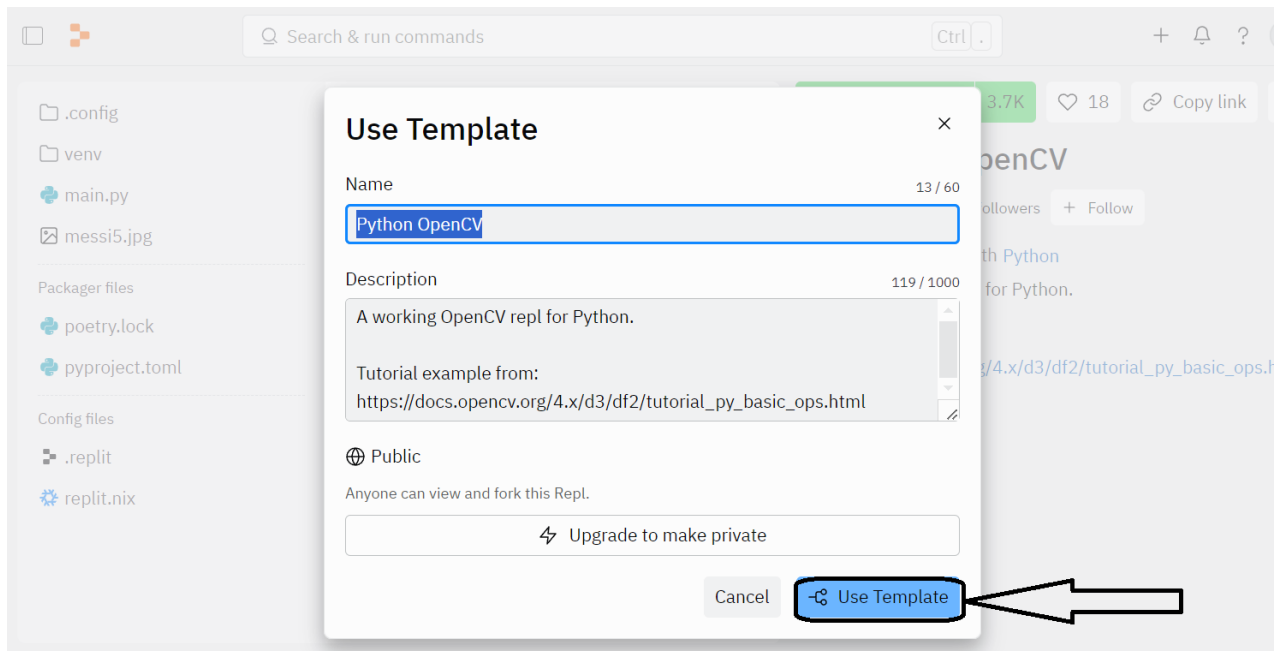


Рис. 0.2.

На рисунках 1, 2 та 5 послідовно показаний спосіб виконання наведеного прикладу коду. При запуску коду можливе оновлення конфігурації пакетів, що займе деякий час. На рисунку 6 показаний результат виконання програми.

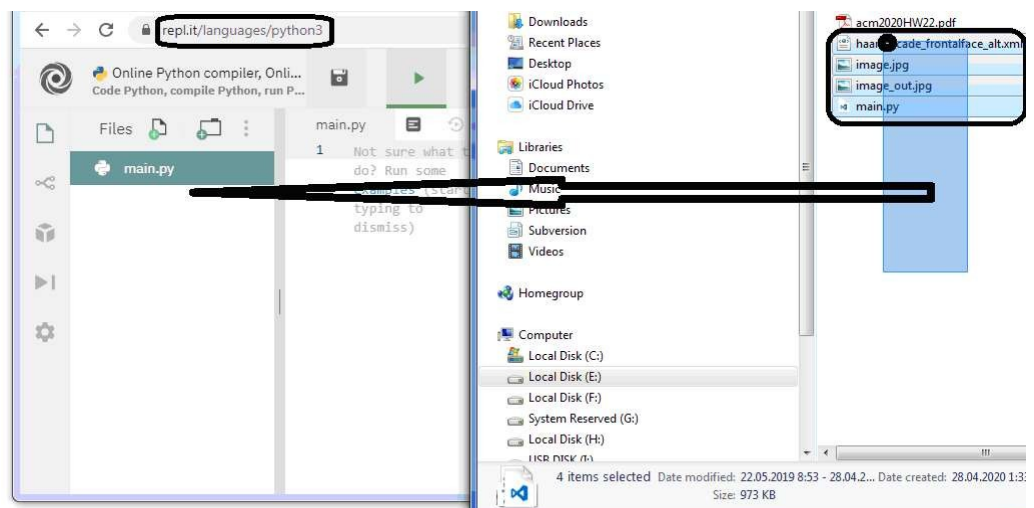


Рис. 1. Перетягування файлів

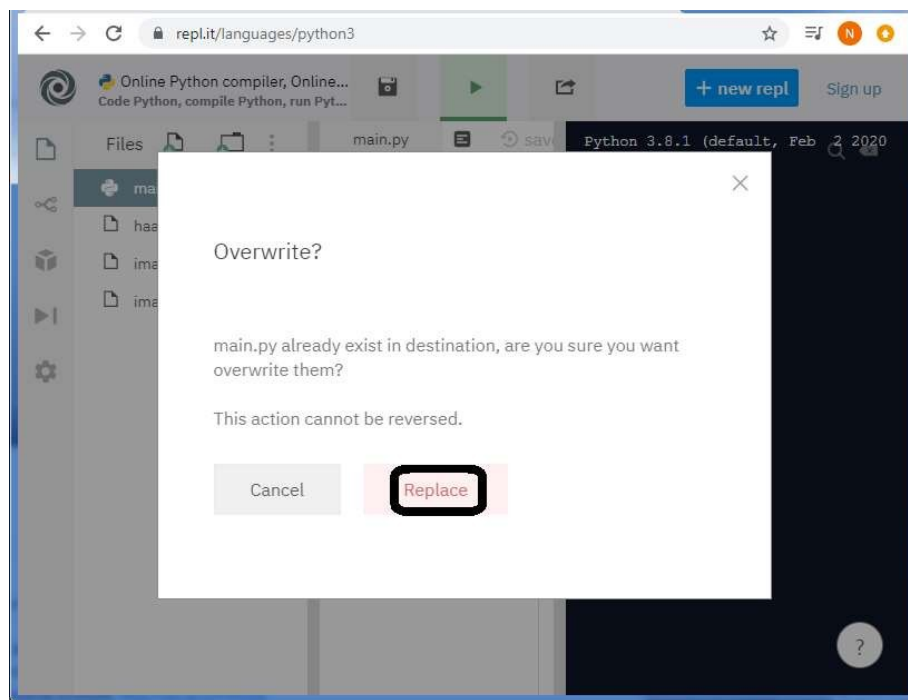


Рис. 2. Підтвердження перезапису файлу `main.py`

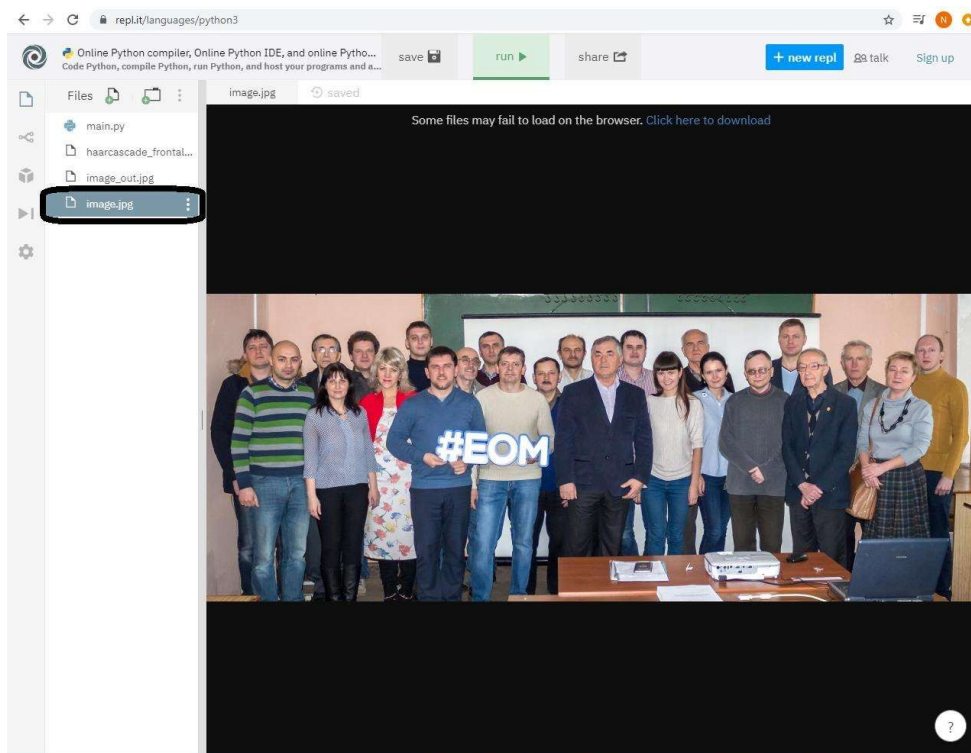


Рис. 3. Відображення оригінального зображення

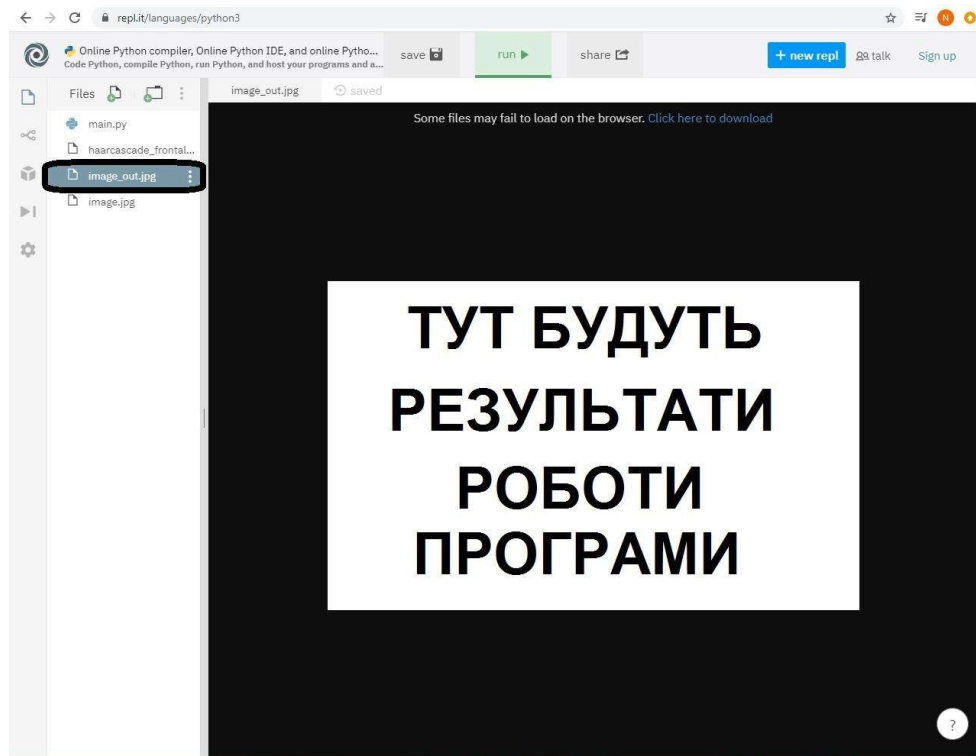


Рис. 4. Файл для збереження результатів виконання програми

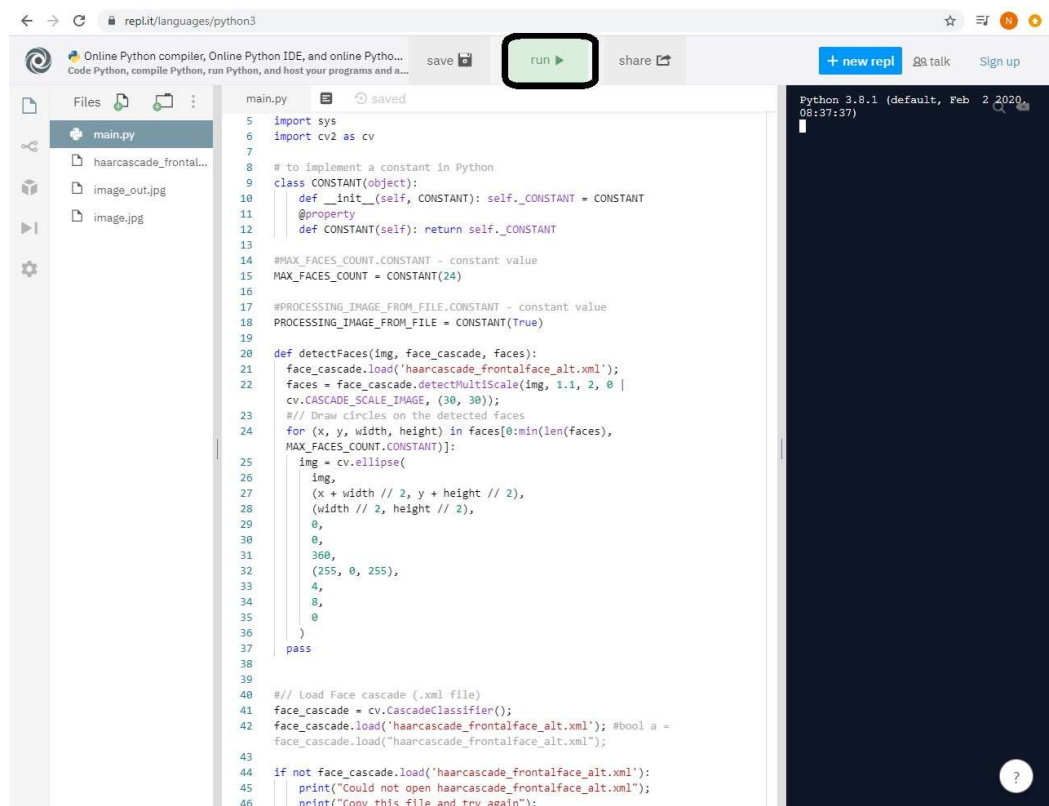


Рис. 5. Запуск програми

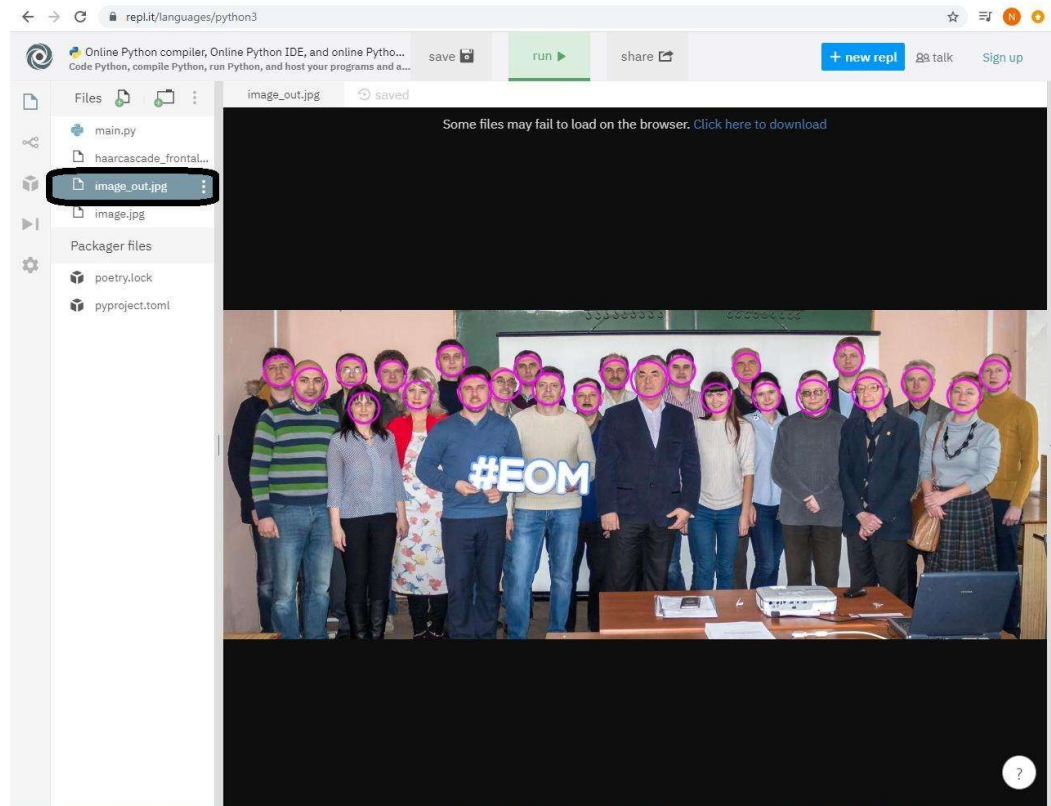


Рис. 6. Оновлене вмістиме файлу для збереження результатів виконання програми

## Приклад коду

\* до прикладу коду додається файл “*haarcascade\_frontalface\_alt.xml*”, який знаходиться на репозиторії з кодом, або ж його можна взяти з бібліотеки OpenCV: “*opencv/sources/data/haarcascades/haarcascade\_frontalface\_alt.xml*”; цей файл і файли з зображеннями потрібно не забути скопіювати перед виконанням програми

Наведений зразок коду реалізовує завдання з виконання умови виявлення на фото не більше 24 облич.

Максимальна кількість облич, які можуть бути виявлені наведеним прикладом коду	24
Оголошення в коді	MAX_FACES_COUNT = CONSTANT(24)

```
import sys
import cv2 as cv

# to implement a constant in Python
class CONSTANT(object):
    def __init__(self, CONSTANT): self._CONSTANT = CONSTANT
    @property
    def CONSTANT(self): return self._CONSTANT

#MAX_FACES_COUNT.CONSTANT - constant value
MAX_FACES_COUNT = CONSTANT(24)

#PROCESSING_IMAGE_FROM_FILE.CONSTANT - constant value
PROCESSING_IMAGE_FROM_FILE = CONSTANT(True)

def detectFaces(image, face_cascade):
    faces= face_cascade.detectMultiScale(image, 1.1, 2, 0 | cv.CASCADE_SCALE_IMAGE, (30, 30));
    for (x, y, width, height) in faces[0:min(len(faces), MAX_FACES_COUNT.CONSTANT)]:
        image = cv.ellipse(
            image,
            (x + width // 2, y + height // 2),
            (width // 2, height // 2),
            0,
            0,
            360,
            (255, 0, 255),
            4,
            8,
            0
        )
    pass

face_cascade = cv.CascadeClassifier();
if not face_cascade.load('haarcascade_frontalface_alt.xml'):
    print("Could not open haarcascade_frontalface_alt.xml");
    print("Copy this file and try again");
    sys.exit(-1);

if PROCESSING_IMAGE_FROM_FILE.CONSTANT == True:
    image = cv.imread("image.jpg");
    if (image is None) :
        print("Could not open or find the image")
        print("Copy the image file and try again")
        sys.exit(-1);

    detectFaces(image, face_cascade)

    #cv.imshow("Detected Face", image);
    cv.imwrite("image_out.jpg", image);
else:
    cap = cv.VideoCapture(0)
    while True:
        _, image = cap.read()
        detectFaces(image, face_cascade)
        cv.imshow("Detected Face", image);
        cv.waitKey(1);
    #sys.exit();
```