Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara de Boumerdès
Faculté des Sciences, Département : Informatique

# PW 1 ► TAI	**Year:** S1-Master I2A Pro.	GACEB

The objective of this practical work is to learn about some Python libraries specialized in computer vision and to discover the PyCharm IDE, which will be used to carry out the practical work for this module. The work carried out will introduce you to the representation of a digital image in memory and teach you how to open, modify (manipulate at pixel level), display and save an image.

Exercise 1 : Installing the Pycharm IDE and creating a new Python project

The code developed during all practical work will be created using the PyCharm Community Edition IDE, using specialised libraries for image manipulation. This IDE is a free, open-source integrated development environment designed specifically to meet the needs of Python developers of all levels (on Linux, Windows and MacOS). It is an excellent tool for prototyping, easy to use for beginners in development, while being robust enough to meet the requirements of experienced developers. The first step is to download and install PyCharm on your machine (Windows) by following these steps:

1. Download the installer from the PyCharm download section on the JetBrains website and select the Community version: https://download.jetbrains.com/python/pycharm-community-2025.2.2.exe
2. Launch the installer pycharm-community-2025.2.2.exe, click Next, choose the location where you want to install the software, and click Next again.
3. Follow the wizard, checking all the options (such as ‘Open Folder as Project’, adding to PATH, etc.) and click Next again, then Install. PyCharm will now install. Once the installation is complete, click Finish.

The second step is to create your first project on PyCharm:

1. Launch PyCharm on your computer.
2. Create a new project: On the welcome screen or from the menu, click ‘New Project’.
3. A new configuration window will open: In the ‘Location’ field, specify the path where the project will be saved and give your project a name (in your case, TP_TAI_Name_Surname_Group).
4. Python version: Ensure that the Python version selected in the Virtualenv options is the one you wish to use.
5. Start creation: Click the ‘Create’ button to generate the project. PyCharm will create the project folder and the virtual environment.
6. From the menu or by right-clicking on the project name (context menu), click on ‘New’ □ ‘Python File’, then enter ‘TP1_Exo1’ in Name and press Enter. The TP1_Exo1.py file is now added to your project. Once the project has been created, you can start writing your Python code in the newly created file.
7. Unzip the attached Images folder and copy it to your project folder.

Exercise 2 : Image manipulation and specialised libraries

There are several Python packages available for manipulating images. To begin with, we use the OpenCV-Python library. This is a Python wrapper for OpenCV (Open Computer Vision), an open-source software library originally developed by Intel for computer vision and machine learning. It provides a comprehensive set of tools for various computer vision tasks, leveraging the simplicity and readability of Python while benefiting from the performance of the underlying C++ implementation. In OpenCV-Python, images are fundamentally represented as NumPy arrays, where each element of the array corresponds to a pixel. It is essential to understand how to access and manipulate each pixel in order to perform various image processing tasks.

1. Before you begin, check that the NumPy and OpenCV-Python packages are installed in your environment: File > Settings > Interpreter for your project, searching in the list of packages

already installed. If the package does not exist, click on the + button, then search for and install the desired package.

2. In the same project, create a Python file named ‘TP1_Exo2.py’, then enter the following code.
3. Reading and displaying images: Images are read using cv2.imread() and can be displayed using cv2.imshow(). cv2.waitKey(0) is used to wait for a key press, and cv2.destroyAllWindows() closes all OpenCV windows. Enter and execute the following code in TP1_Exo2.py:

```
import cv2
# Reading an image
img = cv2.imread('Images/BoatsColor.bmp')

# Load a colour image with its conversion to greyscale
gray_img = cv2.imread('Images/BoatsColor.bmp', cv2.IMREAD_GRAYSCALE)
# Converting a colour image to a greyscale image
gray_img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Displaying an image
cv2.imshow('Input Image', img)
cv2.imshow(' Image loaded in greyscale ', gray_img)
cv2.imshow(' Image converted to greyscale ', gray_img2)
cv2.waitKey(0)

# Reading the colours of a pixel at position (100, 150)
pixel = img[100, 150]
print(f" The RGB values of Pixel at (100, 150) are: {pixel}")

""" Reading image dimensions: dimensions = img.shape
For a colour image, dimensions contains three values:
(height, width, channels).  height: number of rows of pixels in the image. width: number of
columns of pixels in the image. channels: number of colour channels (e.g. 3 for RGB, 1 for
greyscale). For a greyscale image, dimensions contains two values: (height, width """

height, width, channels = img.shape
print(f"Height: {height} pixels")
print(f"Width: {width} pixels")
print(f"Channels: {channels}")

#Changing the pixel colours of the centre line to red
for i in range(0, width-1):
    img[int (height/2), i] = [0, 0, 255] # [B,G,R] modified to red
 cv2.imshow('Converted Image', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise 3 : Fixed thresholding of an image

In a new file called ‘TP1_Exo3.py’, we want to code the binarisation of the image ‘counter.jpg’. All pixels with GL >80 are converted to white and pixels with GL≤80 are converted to black. To do this, develop a code that obtains the binary image in “imgb1” using the openCV functions and the binary image “imgb2” using your own code. Save imgb1 in the folder “Images/imgb1.jpg”