National University of Computer and Emerging Sciences

Lab Manual
CL461-Artificial Intelligence Lab

Department of Computer Science
FAST-NU, Lahore, Pakistan

In this lab, students will learn about classes, Inheritance in python, 2dimensational arrays and computer vision and image processing, real-world applications of matrices, the fundamentals of image processing, and how to use Python libraries to write computer vision code. They will also learn how to write **scratch code for image processing**.

Classes in python:

In Python, classes are a fundamental concept of object-oriented programming (OOP). They provide a way to structure and organize code by grouping related data and functionality together. Here's a basic overview of how classes work in Python:

Class Definition:

You define a class using the class keyword, followed by the class name. Inside the class, you can define attributes and methods.

#code

Example card

```
class MyClass:
  attribute1 = "value1"
  attribute2 = "value2"

obj1 = MyClass()
obj2 = MyClass()

print(obj1.attribute1)  # Output: value1
obj2.attribute1 = "new_value"
print(obj1.attribute1)  # Output: value1 (unchanged because obj2
created an instance attribute)
print(obj2.attribute1)  # Output: new_value (instance attribute
specific to obj2)
```

#output

value1

value1

new_value

Define and Initialize variable in constructor.

```
class MyClassWithConstructor:
    def __init__(self, instance_param):
```

```
        self.instance_attribute = instance_param

obj1 = MyClassWithConstructor("value1")
obj2 = MyClassWithConstructor("value2")

print(obj1.instance_attribute)  # Output: value1
print(obj2.instance_attribute)


#output

value1

value1

new_value
```

Differences between defining and initializing variables within a class versus inside the constructor in Python.

This creates class variables, shared by all instances of the class. They exist independently of objects.

You can change any class variable value using: ClassName.class_variable_name = “AI”

Inside the constructor (__init__):

This defines instance variables, unique to each instance. They become part of the instance's state.

Usage: object_name.instance_variable_name.

How to make variable private in class:

```
class BankAccount:
    def __init__(self, balance):
        self._balance = balance #private variable

    def get_balance(self):
        return self._balance

    def deposit(self, amount):
        self._balance += amount

    def withdraw(self, amount):
        if self._balance >= amount:
            self._balance -= amount
        else:
            print("Insufficient funds")
```

```
obj=BankAccount(2000)
print(obj.get_balance())
obj.deposit(500)
print(obj.get_balance())
obj.withdraw(200)
print(obj.get_balance())
```

```
#output
2000
2500
2300
```

Inheritance:

Inheritance is a fundamental concept in Object-Oriented Programming (OOP) that allows a class (called the child or derived class) to inherit attributes and behaviors from another class (called the parent or base class). This promotes code reuse, extensibility, and the creation of a hierarchy of classes.

```
# Parent class
class Animal:

  def __init__(self, name):
    self.name = name

  def speak(self):
    pass  # Abstract method, to be overridden in subclasses


# Child class inheriting from Animal
class Dog(Animal):

  def speak(self):
    return f"{self.name} says Woof!"


# Another child class inheriting from Animal
class Cat(Animal):

  def speak(self):
    return f"{self.name} says Meow!"


# Creating instances of the derived classes
dog = Dog("doggy")
cat = Cat("Catoo")

# Calling the speak method on instances
```

```
print(dog.speak())   # Output: doggy says Woof!
print(cat.speak())   # Output: catoo says Meow!
```

Image processing is an important field of study in AI because it enables computers to understand and manipulate visual data. Image processing is also important for AI because it can be used to pre-process the visual data for the improvement of accuracy.
Image processing is also able to generate new data. For example, image processing algorithms can be used to create synthetic images or to generate new views of existing images. This can be useful for training AI models or for creating new datasets for machine learning.

In image processing, an **image** is defined as a two-dimensional function, F(x, y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x, y) is called the **intensity of that image at that point.**

Declaration of 2d array:

```
rows = 3
cols = 4
array = [[0 for j in range(cols)] for i in range(rows)]
print("Before")
print(array)
print("After")
array[0][0]=1
print(array)
```

#output
Before
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
After
[[1, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

Another method (not *update only one element)*

```
rows, cols = (5, 5)
arr = [[0] * cols] * rows
print("before")
print(arr)

arr[0][0] = 1  # not update only one element
print("After")
print(arr)

#output
before
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0],
[0, 0, 0, 0, 0]]
After
```

```
[[1, 0, 0, 0, 0], [1, 0, 0, 0, 0], [1, 0, 0, 0, 0], [1, 0, 0, 0, 0],
[1, 0, 0, 0, 0]]
```
I hope you note the difference between them.

The intensity of an image can be represented by a single value, such as a brightness level, or by multiple values, such as **the red, green, and blue** components of a color image.

Digital images are represented as a **two-dimensional array** of pixels, where each pixel contains a value representing the intensity of the image at that point. The size of the image is determined by the number of rows and columns in the array.

The following is an example of a two-dimensional array representing a digital image:

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

This image is a 3x3 image, with each pixel represented by a single value. The value **of 0 represents black, and the value of 255 represents white.** Therefore, this image is a simple black and white image with a white square in the center.

Image processing is the field of computer science that deals with the processing of digital images. Image processing algorithms can be used to perform a variety of tasks, such as:

- Image enhancement: improving the quality of an image by making it brighter, sharper, or more contrasty.
- Image restoration: removing noise or other artifacts from an image.
- Image segmentation: dividing an image into regions of interest.
- Feature extraction: identifying and extracting important features from an image.
- Object recognition: identifying objects in an image.

Image processing is a powerful tool that can be used to improve the quality of images and to extract information from them. It is used in a wide variety of applications, including medical imaging, security, and robotics.

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis.

How to Install OpenCV for Python on Windows?

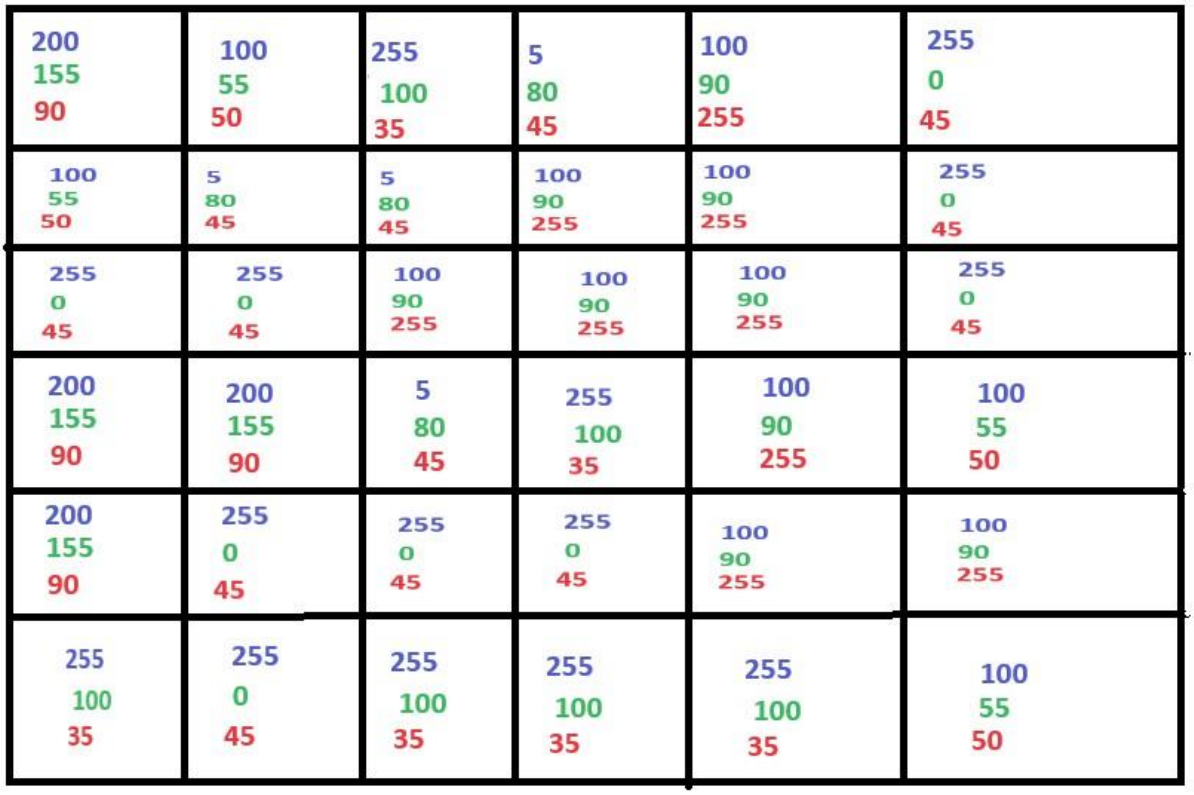
Use this command on anaconda tool terminal.
pip install opencv-python

Reading an image in OpenCV using Python

```
import cv2 #import library. # To read
image from disk, we use # cv2.imread
function, in below method, img =
cv2.imread("apple.png)
cv2.imshow("image", img)
cv2.waitKey(0)

# It is for removing/deleting created GUI window from screen
# and memory
cv2.destroyAllWindows()
```

What is behind img variable?



How to save image using cv2 function

```
cv2.imwrite("..path\\MyImage.jpg", img);
```

How to resize an image using cv2 function

```
resizedimage = cv2.resize(image, (1050, 1610))
```

How to convert color image to gray scale image

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

What is behind gray_image variable?
If the image is grayscale, then on each index of 2d array there is only only value which is representing gray scale intensity.

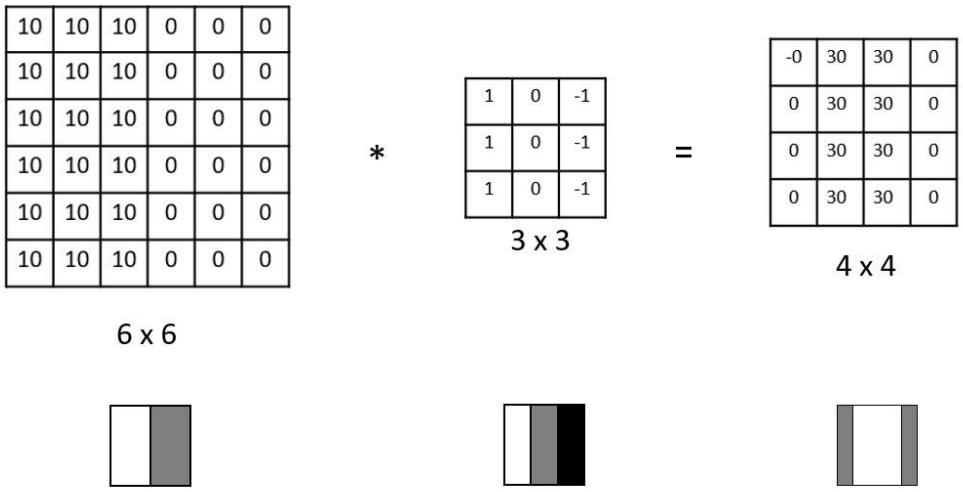
How to access these variables values for manipulation.
For colored image variable
Print(img[0][0][0]) # this will print blue color value which are on 1st index
Print(img[0][0][1]) # this will print green color value which are on 1st index

Print(img[0][0][2]) # this will print red color value which are on 1st index

How to see the image dimensations and their number of channels?

```
dimensions = input_image.shape
height = input_image.shape[0] width
= input_image.shape[1]
number_of_channels = input_image.shape[2]
```

How edge detection filter work in image processing.
Prewitt filter



Apply Prewitt filter on image using cv2 import
cv2

```
# Read the image
img = cv2.imread('image.jpg'
```

```
# Convert the image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
# Define the Prewitt kernels
```

```
prewitt_y = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])

# Apply the Prewitt filter
prewitt_filtered_img = cv2.filter2D(gray_img, cv2.CV_64F, prewitt_x)

# Display the filtered image
cv2.imshow( 'Prewitt filtered image', prewitt_filtered_img)
cv2.waitKey
                (0)
cv2.destroyAllWindows()
```

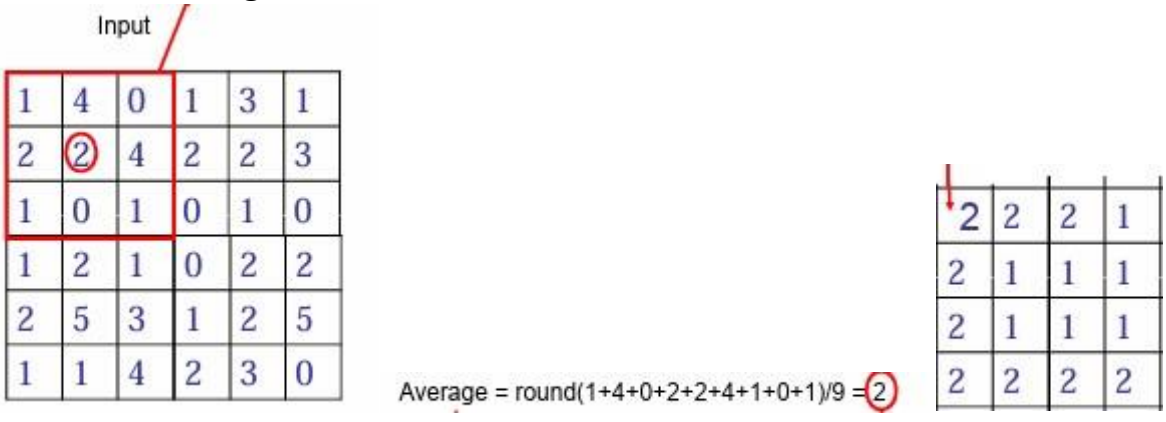
How can we transform color image into gray scale image using math.?

Blue=Blue*0.30
Green=Green*0.59
Red=Red*0.21

Number=Blue+Green+Red//3

Blue=Number
Green=Number
Red=Number

Blur filter or Average filter



How to implement Average filter and Sobel filter through scratch

```
import cv2 import math
cat=cv2.imread('cat.jpeg')
```

```
catx=cv2.imread('cat.jpeg')
caty=cv2.imread('cat.jpeg')
for i in range(331):     for j
in range(500):
cat[i][j][0]=round((cat[i][j][
0]*0.30))         cat[i][j][1] =
round((cat[i][j][1] * 0.59))
cat[i][j][2] =
round((cat[i][j][2] * 0.21))
#
cat[i][j][0]=round(cat[i][j]
[0]*0.07)
#       cat[i][j][0]=round(cat[i][j][1]*0.72) #
cat[i][j][0]=round(cat[i][j][2]*0.21)
        b=(int(cat[i][j][0]))
c= (int(cat[i][j][1]))         d=
(int(cat[i][j][2]))
e=int(int(b)+int(c)+int(d))
f=int(int(e)//3)
cat[i][j][0]=f         cat[i][j][1] =
f         cat[i][j][2] = f
catx[i][j][0] = f
catx[i][j][1] = f
catx[i][j][2] = f
caty[i][j][0] = f
caty[i][j][1] = f
caty[i][j][2] = f
dog=cat
cv2.imshow("image of cat",cat)

cv2.waitKey()
#catx=cv2.GaussianBlur(catx,(5,5),0)
#caty=cv2.GaussianBlur(caty,(5,5),0)

sfilter =[[0 for i in range(3)]for j in range(3)]
cv2.imread("cat.jpeg")
s1,s2,s3,s4,s5,s6,s7,s8,s9,s10=0,0,0,0,0,0,0,0,0,0
for i in range(331):     for j in range(500):
        s1, s2, s3, s4, s5, s6, s7, s8, s9, s10 = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
if i<330 and j<499:                 s1 = sfilter[1][1]=catx[i][j][0]//9#4
s2     =     sfilter[1][0]=catx[i][j-1][0]//9#*2                    s3   =
sfilter[1][2]=catx[i][j+1][0]//9#*2           s4 = sfilter[0][0]=catx[i-
1][j-1][0]//9#*1                  s5 = sfilter[0][1]=catx[i-1][j][0]//9#*2
s6   =   sfilter[0][2]=catx[i-1][j+1][0]//9#*1                    s7   =
sfilter[2][0]=catx[i+1][j-1][0]//9#*1                             s8   =
sfilter[2][1]=catx[i+1][j][0]//9#*2                               s9   =
sfilter[2][2]=catx[i+1][j+1][0]//9#*1
          s10=s1+s2+s3+s4+s5+s6+s7+s8+s9
catx[i][j][0]=s10             catx[i][j][1] = s10
catx[i][j][2] = s10             caty[i][j][0] =
s10             caty[i][j][1] = s10
caty[i][j][2] = s10




cv2.imshow('image of blur cat',catx)
cv2.waitKey()

pfilter =[[0 for i in range(3)]for j in range(3)]
i,j,k=0,0,0 h,m,n=0,0,0 for i in range(331):
```

```
for j in range(500):        if i<=328 and
j<=497:             pfilter[0][0] = cat[i][j]
          pfilter[0][2] = cat[i][j + 2]
pfilter[1][0] = cat[i + 1][j]            pfilter[1][2] =
cat[i + 1][j + 2]            pfilter[2][0] = cat[i + 2][j]
pfilter[2][2] = cat[i + 2][j + 2]            h = (1 *
pfilter[0][0]) + (pfilter[0][2] * -1)            m = (2 *
pfilter[1][0]) + (pfilter[1][2] * -2)            n = (1 *
pfilter[2][0]) + (pfilter[2][2] * -1)
total=h+m+n            cat[i][j]=abs(total)
catx[i][j]=cat[i][j] #          caty[i][j] = cat[i][j]
cv2.imshow("sobel filter with vertical blurring",catx) cv2.waitKey()
##########################################################################################################
######################## for i in range(331):
for j in range(500):        if i<=328 and j<=497:
pfilter[0][0] = caty[i][j]            pfilter[2][0] =
caty[i+2][j]            pfilter[0][1] = caty[i][j+1]
pfilter[2][1] = caty[i + 2][j + 1]
pfilter[0][2] = caty[i][j+2]            pfilter[2][2] =
caty[i + 2][j + 2]            h = (1 * pfilter[0][0]) +
(pfilter[2][0] * -1)            m = (2 * pfilter[0][1]) +
(pfilter[2][1] * -2)            n = (1 * pfilter[0][2]) +
(pfilter[2][2] * -1)            total=h+m+n
cat[i][j]=abs(total)            caty[i][j]=cat[i][j]
cv2.imshow("sobel apply with horizontal sobel ",caty)
cv2.waitKey()

i,j=0,0 for i in range(331):     for j in range(500):
mag=(math.sqrt(int(catx[i][j][0])**2 + int(caty[i][j][0])**2))
dog[i][j] = mag
cv2.imshow("magnitude formula",dog)
cv2.waitKey() cv2.destroyAllWindows()
cv2.destroyAllWindows()
```

Task 1

```
# Student class:
class Student:
    def __init__(self, matrix):
        self.matrix = matrix  # The matrix has the student's
information with indices: [0][0] - name, [0][1] - roll number, [0][2] -
subject name, [0][3] - total marks

# Derived class named 'Portal' inheriting from the base class 'Student'
class Portal(Student):
    def assign_grade(self):
        # Implement logic to assign grades based on total score

    def display_result(self,rnum):
        # Implement logic to display the result
        # search the record with student rollnumber
```

Task 2

Use horizontal filters to apply Sobel and Prewitt filters to an image from scratch.

Task 3

Calculate the frequency of each color and make a histogram graph. You are not allowed to calculate the frequency using any library.

Task 4
Calculate the gray level co-occurrences matrix.