Name: Accessing and changing pixel

**Summary:**  
This program uses OpenCV and NumPy to read an image, display it, and retrieve information about its properties such as shape, size, and data type. It also accesses pixel values at specific coordinates, prints them, modifies certain pixel values, and displays the updated image.

**Input:**

* An image file (hazard10.jpg).
* Pixel positions (100,100) and (135,134) are accessed and modified.

**Output:**

* Prints the shape, size, and data type of the image.
* Prints the pixel value at (100,100) and its blue channel value.
* Shows the original image in one window and the modified image in another window

|  |
| --- |
| import cv2 as cv  import numpy as np  img = cv.imread('hazard10.jpg')  cv.imshow('image',img)  print('the shape of the image is')  print(img.shape)  print('the size of a image')  print(img.size)  print('the data type of the pixel values in the image is ')  print(img.dtype)  px = img[100,100]  print('the value of the pixel 100,100 is')  print(px)  blue = img[100,100,0]  print('the value of the blue colour in pixel at 100x100')  print(blue)  img[100,100] = [212,255,249]  img[135,134] = [255,255,255]  cv.imshow('image1',img)  cv.waitKey(0)  cv.destroyAllWindows() |

Capture video

**Summary :**  
This program uses OpenCV to capture live video from a webcam. It continuously reads frames, displays them in a window, and stops when the **Esc (27)** key is pressed.

**Input:**

* Live video feed from the default camera (VideoCapture(0)).
* User key press (Esc key to exit).

**Output:**

* Displays the live video frames in a window named **'frame'**.
* Stops capturing and closes the window when the user presses the **Esc** key.

|  |
| --- |
| import cv2 as cv  import numpy as np  cam = cv.VideoCapture(0)  while(True):  ret,frame = cam.read()  cv.imshow('frame',frame)  key = cv.waitKey(0)  if key == 27:  break    cam.release()  cv.destroyAllWindows |

LIST

**Summary :**  
This program demonstrates Python list operations such as creation, indexing, updating, appending, inserting, removing, slicing, and iterating. It shows how to work with mixed data types, nested lists, and perform basic manipulations.

**Input:**

* Predefined lists (number, mix\_list, nested\_list, value, mix\_value, nested\_value, valu\_list, list\_val, listl, friut).
* Operations: indexing, updating values, appending, inserting, removing elements, deleting items, slicing, and iteration with for loops.

**Output:**

* Prints the lists in different states after modifications.
* Displays specific list elements using indexing.
* Shows results of slicing operations (sub\_list, negative indexing, step slicing).
* Prints lengths of lists.
* Iterates through friut and listl, printing their contents.

|  |
| --- |
| number = [1,2,3,4,5]  mix\_list = [1,"myself",3.14,True]  nested\_list = [1,2,[3,4]]  value = [13,15,34,55,77,99,32,234,]  mix\_value = [1233,"yourself",4.13,True]  nested\_value = [13,14,[23,54]]  print(value)  print(mix\_value)  print(nested\_value)  print(value[0])  print(mix\_list[1])  print(nested\_value)  value[1]=89  print(value)  mix\_list.append(784)  print(mix\_list)  nested\_list.insert(2,5423)  print(nested\_list)  mix\_value.remove(4.13)  print(mix\_value)  del value[3]  print(value)  valu\_list = [1,2,3,4,5,5,66,7,47,57]  sub\_list = valu\_list[1:9]  print(sub\_list)  print(valu\_list[-3:-1])  print(valu\_list[::2])  list\_val = [1,2,3,4,5,6,7,8,9,10]  print(len(list\_val))  listl =[1,2,3,45,567]  friut = ['orange','apple','banana','guava']  for fruits in friut:  print(friut)  for l in listl:  print(listl) |

Functions

**Summary :**  
This program demonstrates different types of Python functions: simple functions, functions with parameters, default arguments, return values, variable-length arguments (\*args and \*\*kwargs), and lambda functions.

**Input:**

* Function calls with different arguments:
  + nmana() (prints a static message).
  + aB("rohith"), aB("ronith").
  + square(4) and lambda square(5).
  + greet() and greet("hhdhdhh").
  + introduce(age=14, name="Rohith.N").
  + add\_numbers(1,2,3,4,19876%3235^23423).
  + describe\_person(key="ss", value="rr").

**Output:**

* Prints **"er"**.
* Prints greetings with names: **"hello, rohith"**, **"hello, ronith"**.
* Prints **16** (square of 4).
* Prints greetings with default and custom names: **"hello, guest"**, **"hello, hhdhdhh"**.
* Prints introduction:
* Name: Rohith.N
* Age: 14
* Prints the result of add\_numbers(...).
* Prints key-value pairs from \*\*kwargs: **"key:ss"**, **"value:rr"**.
* Prints **25** (square of 5 using lambda).

|  |
| --- |
| def nmana():  print("er")  ##print (nmana)  nmana ()  def aB (name):  print (f"hello,{name}")  aB("rohith")  aB("ronith")  def square (x):  return x \* x  result = square (4)  print (result)  def greet (name = "guest"):  print (f"hello,{name}")  greet()  greet ("hhdhdhh")  def introduce (name,age):  print(f"Name:{name}")  print(f"Age:{age}")  introduce(age = 14,name = "Rohith.N")  def add\_numbers(\*args):  return sum (args)  result = add\_numbers(1,2,3,4,19876%3235^23423)  print(result)  def describe\_person(\*\*kwargs):  for key,value in kwargs.items():  print(f"{key}:{value}")  describe\_person(key = "ss" , value = "rr" )  square = lambda x:x\*x  result = square (5)  print (result) |

While loop

**Summary :**  
This program demonstrates the use of Python while loops with incrementing counters and the else clause. It shows how loops execute until their condition becomes false.

**Input:**

* Loop counters: i = 1 and count = 0.
* Conditions: i <= 10 and count < 3.

**Output:**

* First loop: Prints 1 (since i starts at 1 and then increases by 10, exiting immediately after).
* Second loop: Prints 0, 1, 2, and then prints **"loop finished"** after the loop ends because of the else clause.

|  |
| --- |
| i =1  while i <= 10:  print(i)  i += 10  count = 0  while count < 3:  print (count)  count += 1  else:  print("loop finished") |

For loop

**Summary :**  
This program demonstrates different uses of Python for loops, including iterating over ranges, lists, and strings. It also shows the use of continue, the else clause in loops, and custom step values in ranges.

**Input:**

* A range of numbers (range(5), range(5,11,3)).
* A list of fruits (['orange','apple','banana','guava']).
* A string ("my name is yipeeeee").

**Output:**

* Prints numbers from 0 to 4 using range(5).
* Prints each fruit from the fruits list.
* Prints each character from the string "my name is yipeeeee".
* Prints numbers 0, 1, 2, 4 (skipping 3 because of continue).
* Prints numbers 0 to 4, followed by **"loop finished"** due to the else clause.
* Prints numbers 5 8 11 in the same line, separated by spaces.

|  |
| --- |
| for i in range(5):  print(i)  fruits = ['orange','apple','banana','guava']  for f in fruits:  print(f)  text = "my name is yipeeeee"  for b in text:  print(b)  for i in range(5):  if i == 3:  continue  print(i)    for i in range(5):  print(i)  else:  print("loop finished")  for i in range(5,11,3):  print(i,end=" " ) |

If else

**Summary :**  
This program uses OpenCV to load and display an image. It includes error handling to check if the image was loaded successfully.

**Input:**

* An image file (hazard10.jpg).

**Output:**

* If the image loads successfully:
  + Prints **"image loaded sucssesfully"**.
  + Displays the image in a window named **'loaded image'** until a key is pressed.
* If the image fails to load:
  + Prints **"error error error"**.

|  |
| --- |
| import cv2 as cv  import numpy as np  image = cv.imread('hazard10.jpg')  if image is not None :  print (" image loaded sucssesfully")  cv.imshow('loaded image',image)  cv.waitKey(0)  cv.destroyAllWindows()  else:  print("error error error") |

If else ladder

**Summary :**  
This program uses OpenCV to load an image, convert it to grayscale, calculate its average brightness, and classify the image as **dark**, **medium**, or **bright**.

**Input:**

* An image file (gradient.png).

**Output:**

* If the image fails to load:
  + Prints **"error could not load the image"**.
* If the image loads successfully:
  + Displays the grayscale version of the image in a window named **'gray'**.
  + Calculates the average brightness of the image.
  + Prints whether the image is **dark** (avg < 50), **medium** (50–149), or **bright** (≥150).

|  |
| --- |
| import cv2 as cv  import numpy as np  image = cv.imread("gradient.png")  if image is None:  print("error could not load the image")  else:  gray\_ing = cv.cvtColor(image,cv.COLOR\_BGR2GRAY)  # avg\_brightness = gray\_ing.mean()  avg\_brightness = image.mean()  cv.imshow('gray',gray\_ing)  if avg\_brightness < 50 :  print ("the image is dark")  elif 50 <= avg\_brightness < 150:  print("the image is medium")  else:  print("the image is bright") |

Reading images

**Summary :**  
This program uses OpenCV to load and display an image in a window.

**Input:**

* An image file (hazard10.jpg).

**Output:**

* Opens a window titled **'football'** and displays the loaded image.
* The window remains open until any key is pressed, after which it closes.

|  |
| --- |
| import cv2 as cv  import numpy as np  image = cv.imread('hazard10.jpg')  cv.imshow('football',image)  cv.waitKey(0)  cv.destroyAllWindows() |