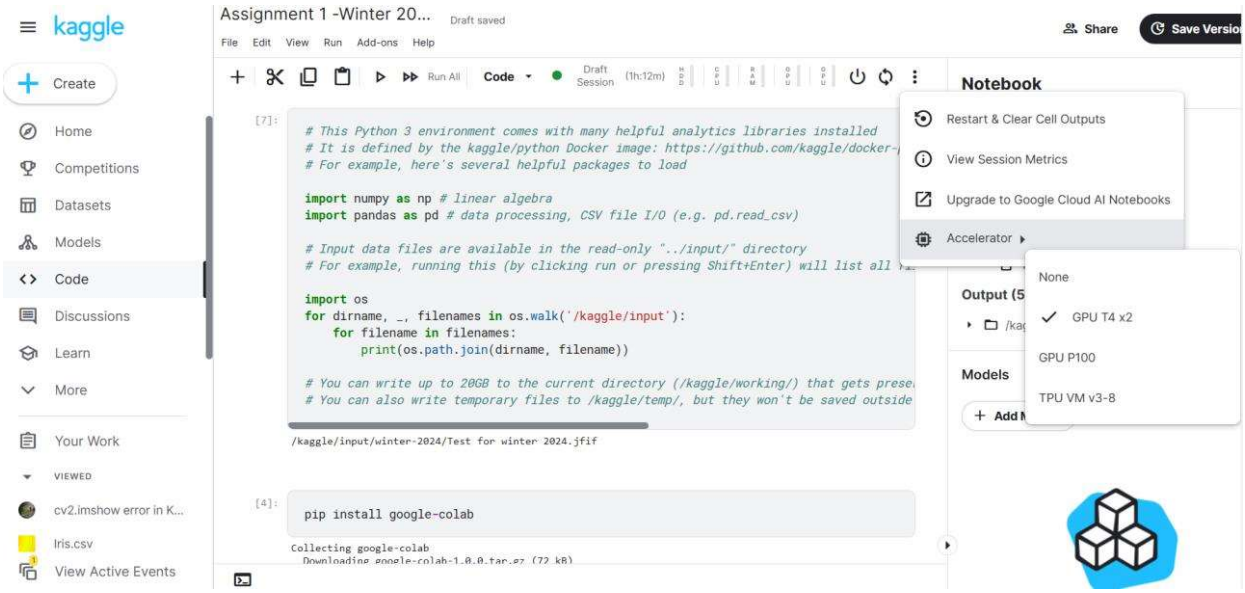
Assignment 1 –Key to completer by Kaggle

28-01-2024

Partial key for Module-1-Lab-processing-images

- Just to remind you that must experiencing many errors while using Kaggle/Colab
- Errors to me are not normal as I have already done with no issues in previous semesters.
- Images are on order as is in my notebook

However, here is all I have done to complete the most part of this section:

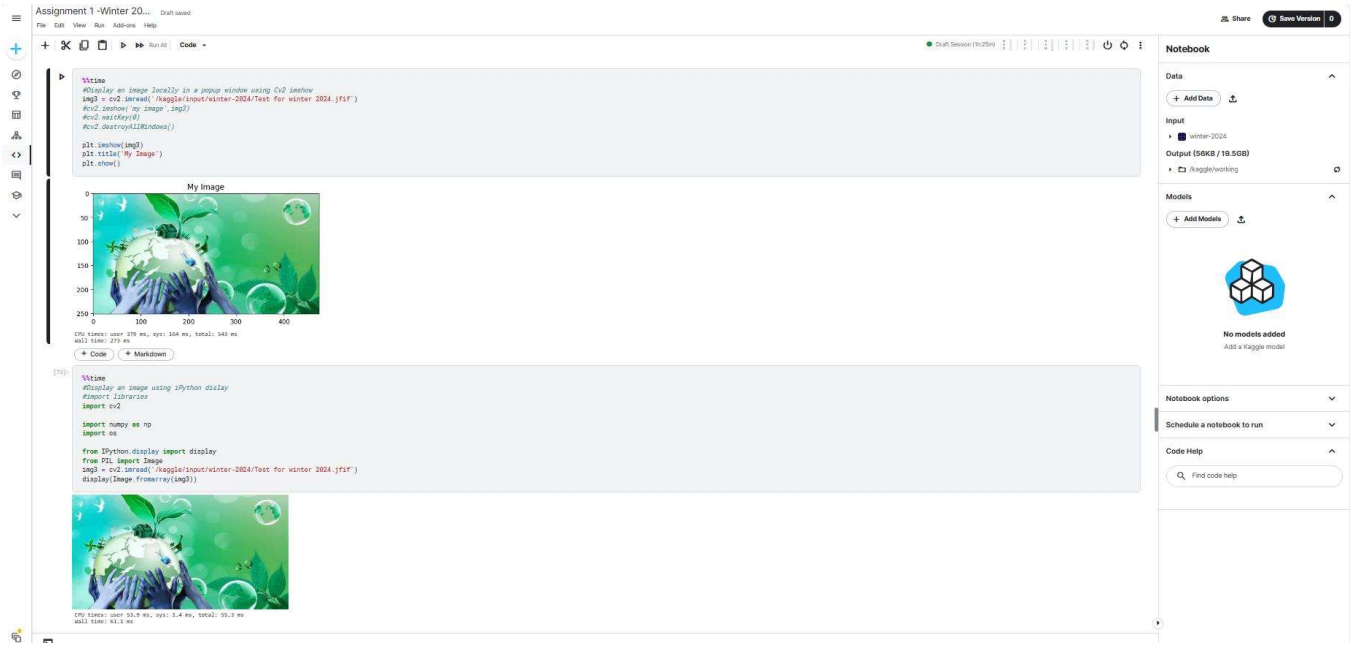


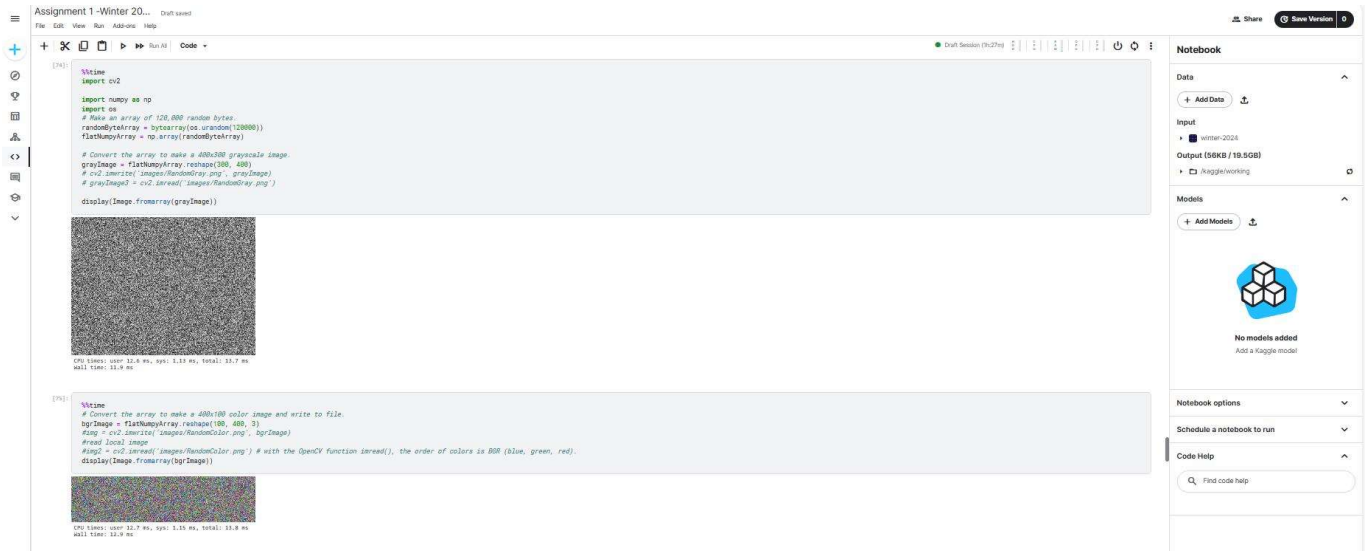
1- Make sure you are using GPU 4 *2

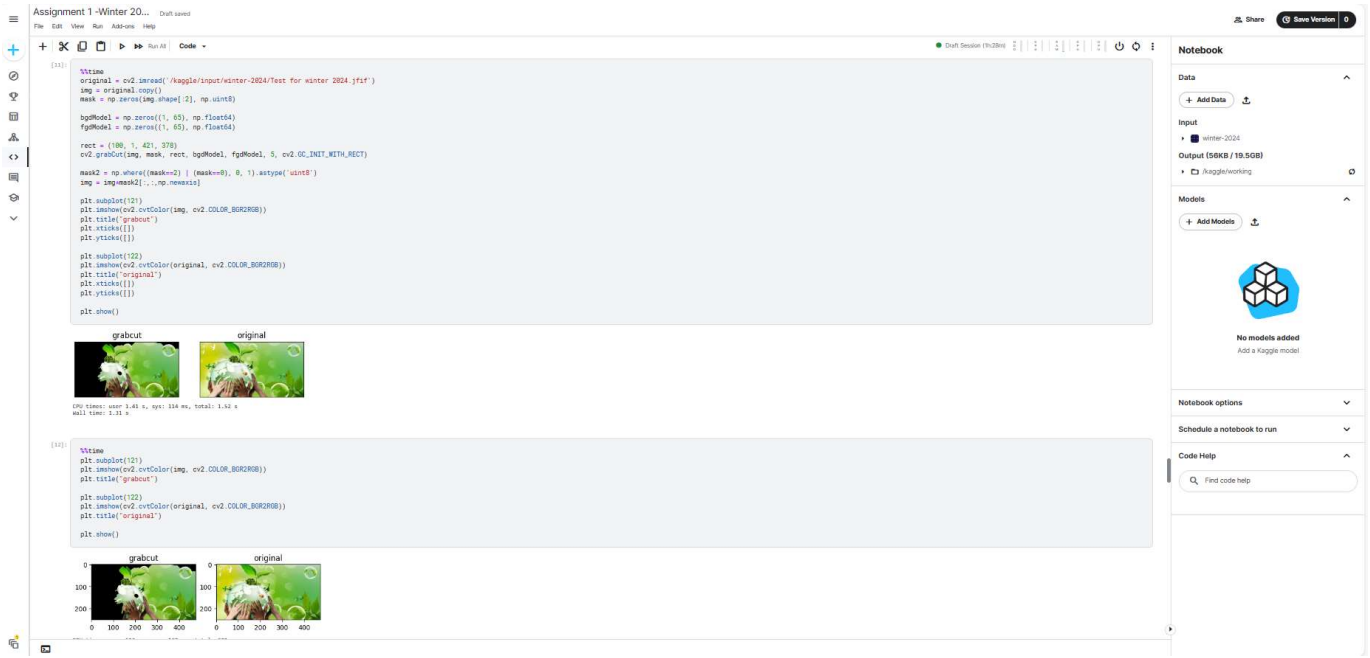
2- Make sure you run “pip install google-colab”

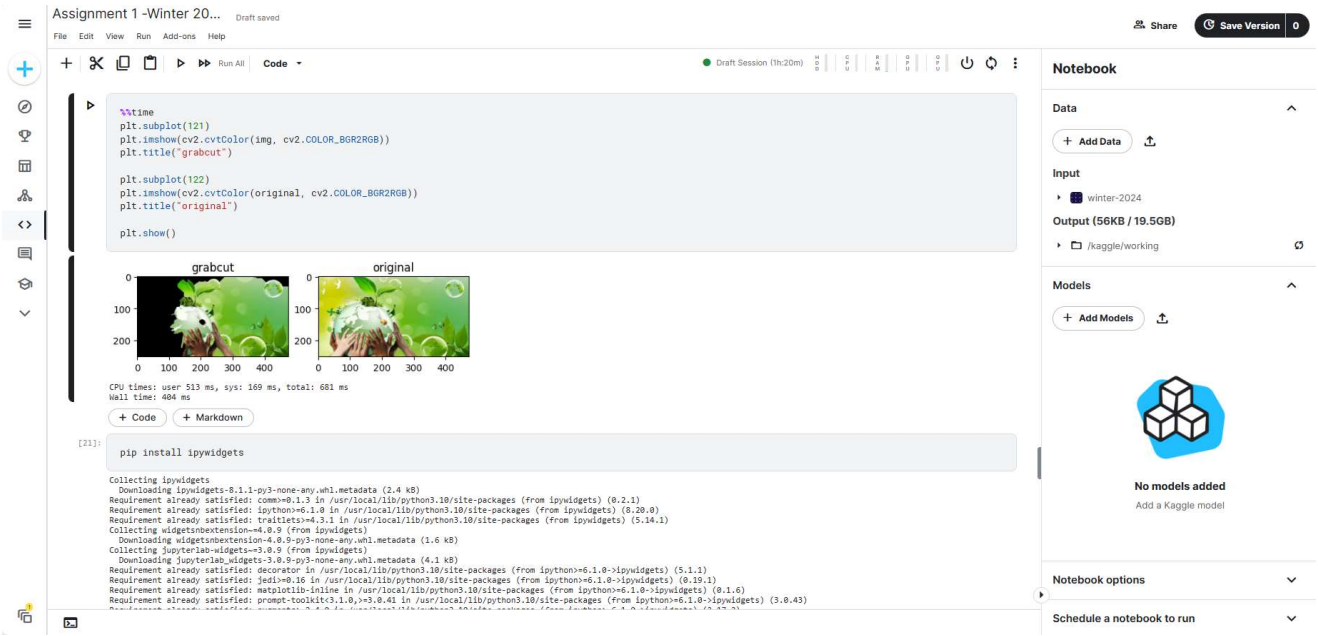
3- Run “pip install opencv-python”

Running CPU

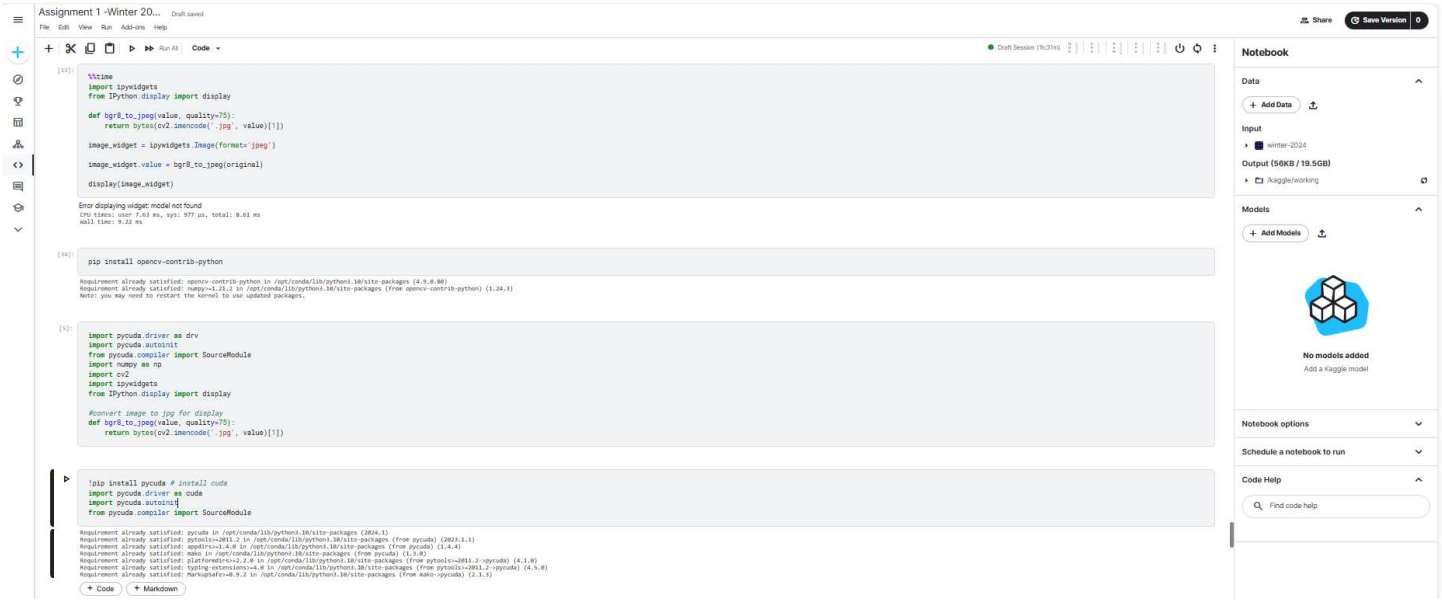

Assignment 1 -Winter 20...
Share
Save Version
Notebook
Data
Add Data
Input
winter-2024
Output (56KB / 19.5GB)
/kaggle/working
Models
Add Models
No models added
Add a Kaggle model
Notebook options
Schedule a notebook to run
Code Help
Find code help
%%time
#Display an image locally in a popup window using Cv2 imshow
img3 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif')
#cv2.imshow('my image',img3)
#cv2.waitKey(0)
#cv2.destroyAllWindows()
plt.imshow(img3)
plt.title('My Image')
plt.show()
My Image
CPU times: user 179 ms, sys: 164 ms, total: 343 ms
wall time: 275 ms
Code
Markdown
%%time
#Display an image using iPython diplay
#import libraries
import cv2
import numpy as np
import os
from IPython.display import display
from PIL import Image
img3 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif')
display(Image.fromarray(img3))
CPU times: user 53.9 ms, sys: 3.4 ms, total: 57.3 ms
wall time: 61.3 ms


Assignment 1 -Winter 20...
Share
Save Version
Notebook
Data
Add Data
Input
winter-2024
Output (56KB / 19.5GB)
/kaggle/working
Models
Add Models
No models added
Add a Kaggle model
Notebook options
Schedule a notebook to run
Code Help
Find code help
%%time
import cv2
import numpy as np
import os
# Make an array of 120,000 random bytes.
randomByteArray = bytearray(os.urandom(120000))
flatNumpyArray = np.array(randomByteArray)
# Convert the array to make a 400x300 grayscale image.
grayImage = flatNumpyArray.reshape(300, 400)
# cv2.imwrite('images/RandomGray.png', grayImage)
# grayImage2 = cv2.imread('images/RandomGray.png')
display(Image.fromarray(grayImage))
CPU times: user 12.6 ms, sys: 1.13 ms, total: 13.7 ms
wall time: 11.9 ms
%%time
# Convert the array to make a 400x100 color image and write to file.
bgrImage = flatNumpyArray.reshape(100, 400, 3)
#img = cv2.imwrite('images/RandomColor.png', bgrImage)
#read local image
#img2 = cv2.imread('images/RandomColor.png') # with the OpenCV function imread(), the order of colors is BGR (blue, green, red).
display(Image.fromarray(bgrImage))
CPU times: user 12.7 ms, sys: 1.15 ms, total: 13.8 ms
wall time: 11.9 ms





4- You may need to run “ipywidgets”



5- Run “pip install opencv-contrib-python"

6- You may need to run “pip install --upgrade pip”

Running GPU

```
!pip install pycuda # install cuda
import pycuda.driver as cuda
import pycuda.autoinit
from pycuda.compiler import SourceModule
```

7- Make sure you are running following in one cell:
   a. !pip install pycuda # install cuda
   b. import pycuda.driver as cuda
   c. import pycuda.autoinit
   d. from pycuda.compiler import SourceModule

```
%%time
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 13 12:15:39 2018

@author: bhaumik
"""

#pass c code to perform work on GPU

mod = SourceModule \
    (
        '''
#include<stdio.h>
#define INDEX(a, b) a*256+b

__global__ void bgr2gray(float *d_result,float *b_img, float *g_img, float *r_img)
{
unsigned int idx = threadIdx.x+(blockIdx.x*(blockDim.x*blockDim.y));

unsigned int a = idx/256;
unsigned int b = idx%256;
d_result[INDEX(a, b)] = (0.299*r_img[INDEX(a, b)]+0.587*g_img[INDEX(a, b)]+0.114*b_img[INDEX(a, b)]);

}

  '''
      )
```

```
CPU times: user 9.14 ms, sys: 2.72 ms, total: 11.9 ms
Wall time: 137 ms
```

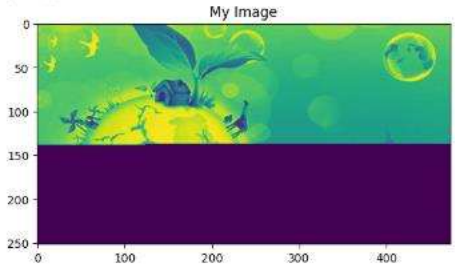
[68]:

```
%%time
import numpy as np
import ipywidgets
from IPython.display import display
from IPython.display import display
from PIL import Image

import matplotlib.pyplot as plt
%matplotlib inline

from time import time
#convert image to jpg for display
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])

h_img = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif',1)
print(h_img.shape)

h_result=r_img

h_gray=cv2.cvtColor(h_img,cv2.COLOR_BGR2GRAY)
#print a
b_img = h_img[:, :, 0].reshape(119448).astype(np.float32)
g_img = h_img[:, :, 1].reshape(119448).astype(np.float32)
r_img = h_img[:, :, 2].reshape(119448).astype(np.float32)
h_result=r_img
bgr2gray = mod.get_function("bgr2gray")




#create instance of cuda function
#bgr2gray = mod.get_function("bgr2gray")
print(h_result.shape)

#call pycuda function
bgr2gray(cuda.Out(h_result),cuda.In(b_img),cuda.In(g_img),cuda.In(r_img),block=(1024, 1, 1),grid=(64, 1, 1))
h_result=np.reshape(h_result,(252,474)).astype(np.uint8)
#using Cv2
#cv2.imshow('Grayscale Image',h_result)
#cv2.waitKey(0)
#cv2.destroyAllWindows()

#using plt
plt.imshow(h_result)
plt.title('My Image')
plt.show()
#display transformed image and using widget
image_widget = ipywidgets.Image(format='jpeg')
image_widget.value = bgr8_to_jpeg(h_result)
display(image_widget)
#original image
orgimage_widget = ipywidgets.Image(format='jpeg')
orgimage_widget.value = bgr8_to_jpeg(h_img)
display(orgimage_widget)
```

```
(252, 474, 3)
(119448,)
```

```
#using plt
plt.imshow(h_result)
plt.title('My Image')
plt.show()
#display transformed image and using widget
image_widget = ipywidgets.Image(format='jpeg')
image_widget.value = bgr8_to_jpeg(h_result)
display(image_widget)
#original image
orgimage_widget = ipywidgets.Image(format='jpeg')
orgimage_widget.value = bgr8_to_jpeg(h_img)
display(orgimage_widget)
```

```
(252, 474, 3)
(119448,)
```



```
CPU times: user 429 ms, sys: 153 ms, total: 582 ms
Wall time: 315 ms
```

[69]:
```
%%time
mod = SourceModule \
    (
        """

__global__ void add_num(float *d_result, float *d_a, float *d_b,int N)
{
 int tid = threadIdx.x + blockIdx.x * blockDim.x;
while (tid < N)
    {
 d_result[tid] = d_a[tid] + d_b[tid];
 if(d_result[tid]>255)
 {
 d_result[tid]=255;
 }
 tid = tid + blockDim.x * gridDim.x;
}
    }

  """
      )
```

```
CPU times: user 1.95 ms, sys: 0 ns, total: 1.95 ms
Wall time: 1.52 ms
```

[48]:
```
img1 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif',0)
img2 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif',0)
print(img1.shape)
print(img2.shape)
```

```
(252, 474)
(252, 474)
```

[51]:
```
h_img1 = img1.reshape(119448).astype(np.float32)
h_img2 = img2.reshape(119448).astype(np.float32)
print(h_img1.shape)
print(h_img2.shape)
```

```
(119448,)
(119448,)
```

[54]:
```
N = h_img1.size
h_result=h_img1
print(N)
print(h_result)
```

```
119448
[229. 224. 217. ...  69.  72.  82.]
```

[64]:
```
add_img = mod.get_function("add_num")
add_img(cuda.Out(h_result), cuda.In(h_img1), cuda.In(h_img2),np.uint32(N),block=(1024, 1, 1), grid=(64, 1, 1))
h_result=np.reshape(h_result,(252,474)).astype(np.uint8)
```

[48]:
```
img1 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif',0)
img2 = cv2.imread('/kaggle/input/winter-2024/Test for winter 2024.jfif',0)
print(img1.shape)
print(img2.shape)
```

```
(252, 474)
(252, 474)
```

[51]:
```
h_img1 = img1.reshape(119448).astype(np.float32)
h_img2 = img2.reshape(119448).astype(np.float32)
print(h_img1.shape)
print(h_img2.shape)
```

```
(119448,)
(119448,)
```

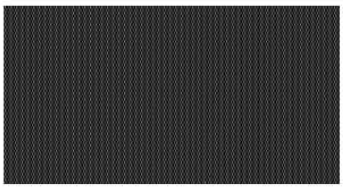
[54]:
```
N = h_img1.size
h_result=h_img1
print(N)
print(h_result)
```

```
119448
[229. 224. 217. ...  69.  72.  82.]
```

[64]:
```
add_img = mod.get_function("add_num")
add_img(cuda.Out(h_result), cuda.In(h_img1), cuda.In(h_img2),np.uint32(N),block=(1024, 1, 1), grid=(64, 1, 1))
h_result=np.reshape(h_result,(252,474)).astype(np.uint8)
```

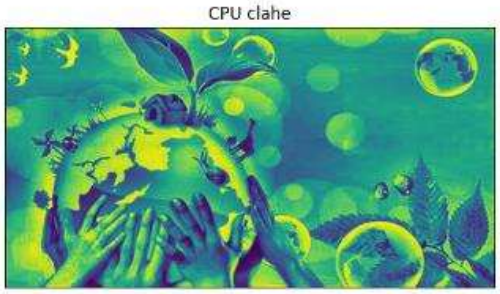
+ Code + Markdown

[65]:
```
image_widget = ipywidgets.Image(format='jpeg')
image_widget.value = bgr8_to_jpeg(h_result)
display(image_widget)
```



```
%%time
#CPU version
img = cv2.imread("/kaggle/input/winter-2024/Test for winter 2024.jfif", cv2.IMREAD_GRAYSCALE)

def plotImages(imgr):
    plt.imshow(imgr)
    plt.title("CPU clahe")
    plt.xticks([])
    plt.yticks([])



#contrast Limited Adaptive Histogram Equalization
clahe = cv2.createCLAHE(clipLimit=5.0, tileGridSize=(8, 8))
imgr = clahe.apply(img)

plotImages(imgr)
```

```
CPU times: user 32.5 ms, sys: 2.37 ms, total: 34.9 ms
Wall time: 28.6 ms
```



+ Code + Markdown

[71]:

```
%%time
import pycuda.gpuarray as gpuarray
import pycuda.driver as cuda
import pycuda.autoinit
import numpy

a_gpu = gpuarray.to_gpu(numpy.random.randn(4,4).astype(numpy.float32))
a_doubled = (2*a_gpu).get()
print(a_doubled)
print(a_gpu)
```

```
[[-0.26158872  0.4961721  -0.83486335  1.9181659 ]
 [ 0.33141446  1.6056442   1.8213003  -2.345274  ]
 [ 1.0849017  -2.7049644   1.4376756  -0.14914745]
 [ 1.6721809   1.285488   -3.562288   -0.01868289]]
[[-0.13079436  0.24808605 -0.41703168  0.95908296]
 [ 0.16570723  0.8028221   0.91065013 -1.172637  ]
 [ 0.54245085 -1.3524822   0.7188378  -0.07457373]
 [ 0.83609045  0.642744   -1.781144   -0.00934144]]
CPU times: user 19 ms, sys: 3.91 ms, total: 22.9 ms
Wall time: 639 ms
```