

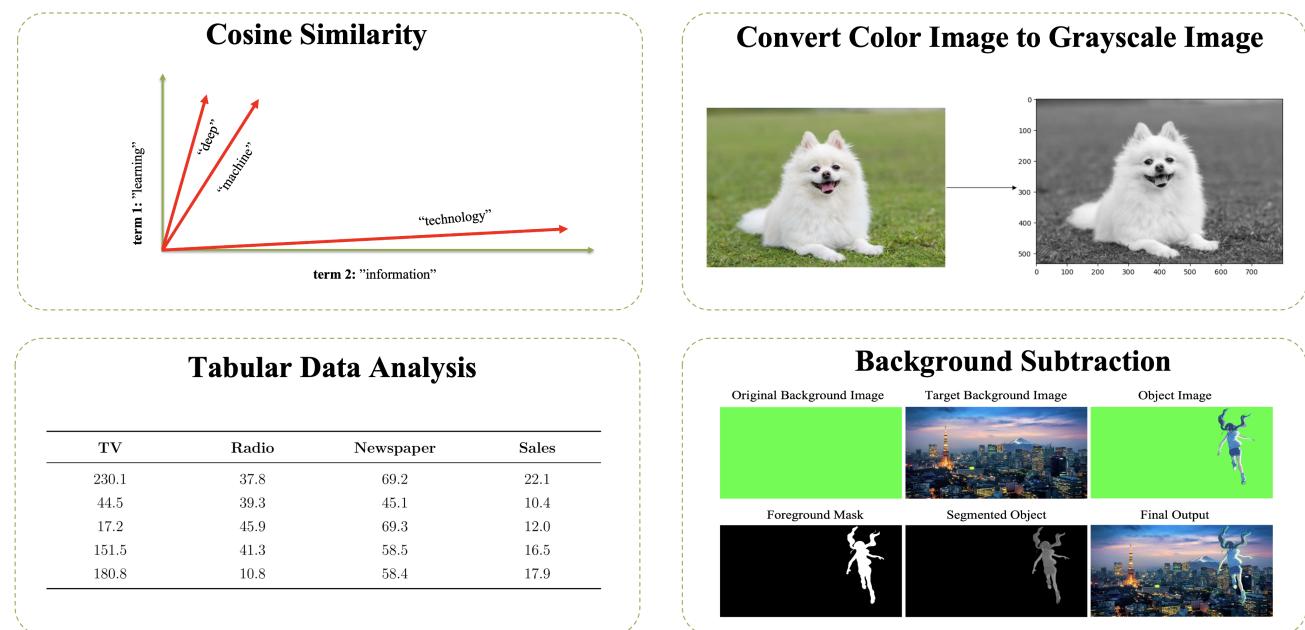
AI VIET NAM – AI COURSE 2025

# Exercise: Linear Algebra and Numpy

Quoc-Thai Nguyen, Tri-Quang Pham, Quang-Vinh Dinh

## I. Giới thiệu

**Giới thiệu về bài tập:** Ở phần bài tập này, ta sẽ học cách sử dụng thư viện numpy và ôn lại kiến thức đại số tuyến tính cơ bản



Hình 1: Minh họa nội dung bài tập.

# Mục lục

I.	Giới thiệu . . . . .	1
II.	Câu hỏi tự luận . . . . .	3
II.1.	Các phép toán trên vector và ma trận . . . . .	3
II.2.	Cosine Similarity . . . . .	5
II.3.	Background subtraction (tách nền) . . . . .	5
III.	Câu hỏi trắc nghiệm . . . . .	8
III.1.	Numpy . . . . .	8
III.2.	Xử lý ảnh . . . . .	12
III.3.	Phân tích dữ liệu dạng bảng . . . . .	14
	Phụ lục . . . . .	17

## II. Câu hỏi tự luận

### II.1. Các phép toán trên vector và ma trận

#### 1. Độ dài của vector:

Length of a vector

- Vector:  $\mathbf{v} = [v_1, v_2, \dots, v_n]^T$
- Length of a vector:  $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$

Hãy hoàn thiện hàm **compute\_vector\_length()** để tính độ dài của vector sử dụng thư viện numpy:

```

1 import numpy as np
2 def compute_vector_length(vector):
3     # **** Your code here ****
4
5     return len_of_vector

```

#### 2. Phép tích vô hướng:

Dot product

- Vector:  $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix}$
- Dot Product:  $\mathbf{v} \cdot \mathbf{u} = v_1 * u_1 + v_2 * u_2 + \dots + v_n * u_n$

Hãy hoàn thiện hàm **compute\_dot\_product()** sử dụng thư viện numpy:

```

1 def compute_dot_product(vector1, vector2):
2     # **** Your code here ****
3
4     return result

```

#### 3. Nhân vector với ma trận:

## Multiplying a vector by a matrix

- Matrix:  $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \mathbf{A} \in R^{m*n}$

- Vector:  $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}, \mathbf{v} \in R^n$

- $\mathbf{c} = \mathbf{Av} = \begin{bmatrix} a_{11} * v_1 + \dots + a_{1n} * v_n \\ \dots \\ a_{m1} * v_1 + \dots + a_{mn} * v_n \end{bmatrix}, \mathbf{c} \in R^n$

Hãy hoàn thiện hàm **matrix\_multi\_vector()** sử dụng thư viện numpy:

```

1 def matrix_multi_vector(matrix, vector):
2     # **** Your code here ****
3
4     return result

```

## 4. Nhân ma trận với ma trận:

## Multiplying a matrix by a matrix

- Matrix A:  $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \mathbf{A} \in R^{m*n}$

- Matrix B:  $\mathbf{B} = \begin{bmatrix} b_{11} & \dots & b_{1k} \\ \dots & \dots & \dots \\ b_{n1} & \dots & b_{nk} \end{bmatrix}, \mathbf{B} \in R^{n*k}$

- $\mathbf{C} = \mathbf{AB} = \begin{bmatrix} a_{11} * b_{11} + \dots + a_{1n} * b_{n1} & \dots & a_{11} * b_{1k} + a_{1n} * b_{nk} \\ \dots & \dots & \dots \\ a_{m1} * b_{11} + \dots + a_{mn} * b_{n1} & \dots & a_{m1} * b_{1k} + a_{mn} * b_{nk} \end{bmatrix}, \mathbf{C} \in R^{m*k}$

Hãy hoàn thiện hàm **matrix\_multi\_matrix()** sử dụng thư viện numpy:

```

1 import numpy as np
2 def matrix_multi_matrix(matrix1, matrix2):
3     # **** Your code here ****
4
5     return len_of_vector

```

## II.2. Cosine Similarity

### Cosine Similarity

- Data (vector  $\mathbf{x}, \mathbf{y}$ ):  $\mathbf{x} = \{x_1, \dots, x_N\}$   $\mathbf{y} = \{y_1, \dots, y_N\}$
- Cosine Similarity:  $cs(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_1^n x_i y_i}{\sqrt{\sum_1^n x_i^2} \sqrt{\sum_1^n y_i^2}}$

**Cho trước**  $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$   $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$  Tìm Cosine similarity  $cs(\mathbf{x}, \mathbf{y})$ .

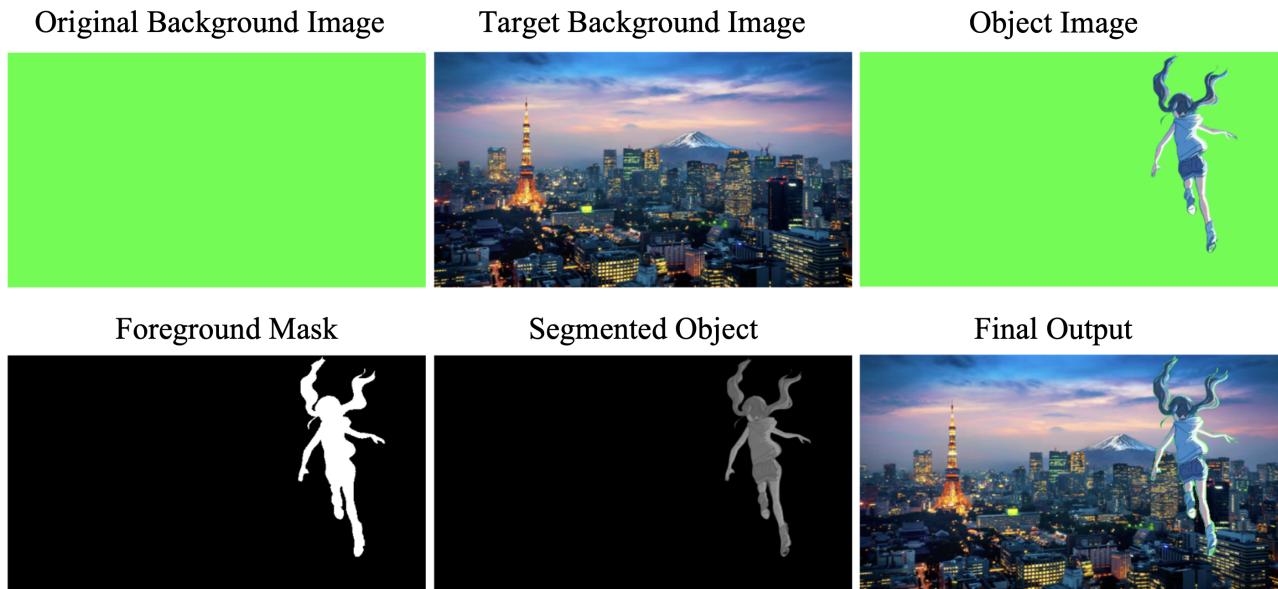
Dựa vào trên quả tính tay ở trên, hãy hoàn thiện hàm **compute\_cosine()** sử dụng thư viện numpy:

```

1 def compute_cosine(v1, v2):
2     # **** Your code here ****
3
4     return cos_sim

```

## II.3. Background subtraction (tách nền)



Hình 2: Kết quả trích xuất foreground (object mong muốn) và dán vào background mới

Viết chương trình sử dụng kỹ thuật background subtraction để trích xuất foreground (object mong muốn) và dán vào background mới. Input/Output của chương trình như sau:

- **Input:** Gồm 3 ảnh là Original Background Image (Greenscreen), Target Background Image và Object Image.
- **Output:** Ảnh mới khi trích xuất object từ Object Image và dán vào Target Background Image.

### Gợi ý:

1. Đưa cả 3 ảnh về cùng kích thước.
2. Dùng kỹ thuật background subtraction với Object Image và Original Background Image để lấy mask của object.
3. Mask object là một ảnh binary (Foreground Mask), sẽ gồm 2 giá trị: 0 là background, 1 các vùng pixel chứa object.
4. Tạo ra ảnh output bằng cách: tại vị trí pixel nào = 1 thì lấy giá trị của Object Image và vị trí nào = 0 thì lấy giá trị của Target Background Image.

**Hướng dẫn:** Để hoàn thành bài tập trên, bạn cần hoàn thiện các hàm sau đây (các bạn chú ý chỉnh lại đường dẫn đến ảnh cho phù hợp):

1. **Resize các ảnh đầu vào về cùng kích thước:**

```

1 import numpy as np
2 from google.colab.patches import cv2_imshow
3 import cv2
4
5 bg1_image = cv2.imread("GreenBackground.png", 1)
6 bg1_image = cv2.resize(bg1_image, (678, 381))
7
8 ob_image = cv2.imread("Object.png", 1)
9 ob_image = cv2.resize(ob_image, (678, 381))
10
11 bg2_image = cv2.imread("NewBackground.jpg", 1)
12 bg2_image = cv2.resize(bg2_image, (678, 381))

```

2. **Xây dựng hàm compute\_difference():**

```

1 def compute_difference(bg_img, input_img):
2     # **** Your code here ****
3
4     return difference_single_channel

```

Các bạn có thể sử dụng đoạn code bên dưới để hiển thị kết quả (hình 2) của hàm `compute_difference()` như sau:

```
1 difference_single_channel = compute_difference(bg1_image, ob_image)
2 cv2_imshow(difference_single_channel)
```

### 3. Xây dựng hàm compute\_binary\_mask():

```
1 def compute_binary_mask(difference_single_channel):
2     # ***** Your code here *****
3
4     return difference_binary
```

Các bạn có thể sử dụng đoạn code bên dưới để hiển thị kết quả (hình 3) của hàm **compute\_binary\_mask()** như sau:

```
1 binary_mask = compute_binary_mask(difference_single_channel)
2 cv2_imshow(binary_mask)
```

### 4. Xây dựng hàm replace\_background():

```
1 def replace_background(bg1_image, bg2_image, ob_image):
2     difference_single_channel = compute_difference(
3         bg1_image,
4         ob_image
5     )
6
7     binary_mask = compute_binary_mask(difference_single_channel)
8
9     output = np.where(binary_mask==255, ob_image, bg2_image)
10
11    return output
```

# III. Câu hỏi trắc nghiệm

## III.1. Numpy

Câu hỏi 1: Tính độ dài của vector: Cho đoạn code sau:

```

1 import numpy as np
2
3 def compute_vector_length(vector):
4     return # Your code here
5
6 vector = np.array([-2, 4, 9, 21])
7 result = compute_vector_length(vector)
8 print(round(result, 2))

```

Kết quả là gì?

- 1 (a) 43.28
- 2 (b) 33.28
- 3 (c) 13.28
- 4 (d) 23.28

Câu hỏi 2: Tính tích vô hướng: Cho đoạn code sau:

```

1 import numpy as np
2
3 def compute_dot_product(vector1, vector2):
4     return # Your code here
5
6 v1 = np.array([0, 1, -1, 2])
7 v2 = np.array([2, 5, 1, 0])
8 result = compute_dot_product(v1, v2)
9 print(round(result, 2))

```

Kết quả là gì?

- 1 (a) 3
- 2 (b) 4
- 3 (c) 5
- 4 (d) 6

Câu hỏi 3: Nhân vector với ma trận: Cho đoạn code sau:

```

1 import numpy as np
2
3 def matrix_multi_vector(matrix, vector):
4     return # Your code here
5
6 matrix = np.array([[1, 2, 3], [4, 5, 6]])
7 vector = np.array([1, 2, 3])

```

```

8 result = matrix_multi_vector(matrix, vector)
9 print(result)

```

Kết quả là gì?

- 1 (a) [3 1]
- 2 (b) [1 3]
- 3 (c) [2 3]
- 4 (d) [3 2]

Câu hỏi 4: Nhân ma trận với ma trận: Cho đoạn code sau:

```

1 import numpy as np
2
3 def matrix_multi_matrix(matrix1, matrix2):
4     return # Your code here
5
6 m1 = np.array([[0, 1, 2], [2, -3, 1]])
7 m2 = np.array([[1, -3], [6, 1], [0, -1]])
8 result = matrix_multi_matrix(m1, m2)
9 print(result)

```

Kết quả là gì?

- 1 (a) [[9 -1], [-16 -10]]
- 2 (b) [[8 -1], [-16 -10]]
- 3 (c) [[6 -1], [-16 -10]]
- 4 (d) [[7 -1], [-16 -10]]

Câu hỏi 5: Cosine Similarity: Cho đoạn code sau:

```

1 import numpy as np
2
3 def compute_cosine(v1, v2):
4     dot_product = np.dot(v1, v2)
5     norm_v1 = # **** Your code here ****
6     norm_v2 = # **** Your code here ****
7     return dot_product / (norm_v1 * norm_v2)
8
9 x = np.array([1, 2, 3, 4])
10 y = np.array([1, 0, 3, 0])
11 result = compute_cosine(x, y)
12 print(round(result, 3))

```

Kết quả là gì?

- 1 (a) 1.577
- 2 (b) 2.577
- 3 (c) 0.577
- 4 (d) 3.577

Câu hỏi 6: Lọc phần tử lẻ: Cho đoạn code sau:

```

1 import numpy as np
2 arr = np.arange(0, 10)
3 # Your code here

```

Kết quả là gì?

- 1 (a) [1 3 5 7 9]
- 2 (b) [1 2 3 4 5]
- 3 (c) [2 4 6 8 9]
- 4 (d) [1 7 5 7 9]

**Câu hỏi 7:** Thay thế phần tử lẻ bằng giá trị -1: Cho đoạn code sau:

```

1 import numpy as np
2 arr = np.arange(0, 10)
3 # Your code here = -1
4 print(arr)

```

Kết quả là gì?

- 1 (a) [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
- 2 (b) [0 -1 2 -1 4 -1 6 -1 8 -1]
- 3 (c) [-1 0 -1 0 -1 0 -1 0 -1 0]
- 4 (d) [-1 1 -1 1 -1 1 -1 1 -1 1]

**Câu hỏi 8:** Gộp mảng theo trục axis=0: Cho đoạn code sau:

```

1 import numpy as np
2 arr1 = np.arange(10).reshape(2, -1)
3 arr2 = np.ones((2, 5), dtype=int)
4 c = # Your code here
5 print(c)

```

Kết quả là gì?

- 1 (a) [[0 1 2 3 4] [5 6 7 8 9] [1 1 1 1 1] [1 1 1 1 1]]
- 2 (b) [[0 1 2 3 4] [5 6 7 8 9]]
- 3 (c) [[0 1 2 3 4] [5 6 7 8 9] [1 1 1 1 1]]
- 4 (d) [[0 1 2 3 4]]

**Câu hỏi 9:** Gộp mảng theo trục axis=1: Cho đoạn code sau:

```

1 import numpy as np
2 arr1 = np.arange(10).reshape(2, -1)
3 arr2 = np.ones((2, 5), dtype=int)
4 c = # Your code here
5 print(c)

```

Kết quả là gì?

- 1 (a) [[0 1 2 3 4 1 1 1 1 1] [5 6 7 8 9 1 1 1 1 2]]  
2 (b) [[1 1 1 1 1 1 1 1 1] [5 6 7 8 9 5 6 7 8 9]]  
3 (c) [[0 1 2 3 4 1 1 1 1 1] [5 6 7 8 9 1 1 1 1 1]]  
4 (d) [[0 2 2 2 2 1 1 1 1 1] [2 2 2 2 2 1 1 1 1 1]]

Câu hỏi 10: Lọc giá trị theo điều kiện: Cho đoạn code sau:

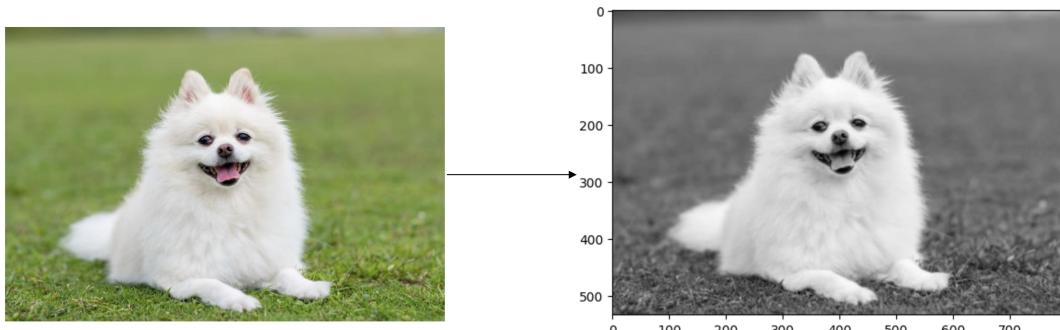
```
1 import numpy as np
2 a = np.array([2, 6, 1, 9, 10, 3, 27])
3 index = # Your code here
4 # điều kiện phần tử >= 5 và <= 10
5 print(a[index])
```

Kết quả là gì?

- 1 (a) [3 9 10]
2 (b) [1 9 10]
3 (c) [6 9 10]
4 (d) [2 1 10]

### III.2. Xử lý ảnh

Các câu hỏi phần này giải quyết bài toán chuyển ảnh màu về ảnh xám. Ví dụ được mô tả như hình sau: Ảnh sẽ được biểu diễn bởi tập các điểm ảnh (Pixel) có giá trị từ 0 đến 255. Ảnh màu



Hình 3: Chuyển ảnh màu thành ảnh xám

sẽ được biểu diễn thành ma trận có kích thước (Height, Width, Channel); trong đó số Channel là 3 tương ứng là giá trị biểu diễn cho các kênh màu R-Red, G-Green, B-Blue. Ảnh xám sẽ được biểu diễn thành ma trận có kích thước (Height, Width). Vì vậy, để chuyển ảnh màu thành ảnh xám, chúng ta sẽ tổng hợp từ 3 kênh màu RGB thành 1 giá trị duy nhất trong khoảng 0 đến 255. Có 3 phương pháp chính để tính giá trị chuyển đổi:

- Lightness: Tính giá trị trung bình của giá trị lớn nhất và nhỏ nhất cho các kênh màu:  $(\max(R,G,B) + \min(R,G,B))/2$
- Average: Tính giá trị trung bình của 3 kênh màu:  $(R+G+B)/3$
- Luminosity: Nhân hệ số tương ứng của 3 kênh màu như sau:  $0.21*R + 0.72*G + 0.07*B$

**Tải và đọc ảnh:**

```

1 !gdown 1i9dqan21DjQoG5Q_VEvm0LrVwAlXD0vB
2 import matplotlib.image as mpimg
3 import numpy as np
4 img = mpimg.imread("/dog.jpeg")

```

**Câu hỏi 11:** Chuyển ảnh màu sang xám (Lightness): Hoàn thiện đoạn code sau để chuyển đổi ảnh màu thành ảnh xám theo phương pháp Lightness:

```

1 # **** Your code here ****
2 # Sử dụng công thức Lightness:  $(\max(R,G,B) + \min(R,G,B))/2$ 
3 gray_img_01 = # Your code here
4
5 print(gray_img_01[0, 0])

```

- 1 a) [102.5]
- 2 b) [92.5]
- 3 c) [82.5]
- 4 d) [72.5]

**Câu hỏi 12:** Chuyển ảnh màu sang xám (Average): Hoàn thiện đoạn code sau để chuyển đổi ảnh màu thành ảnh xám theo phương pháp Average:

```

1 # **** Your code here ****
2 # Sử dụng công thức Average: (R+G+B)/3
3 gray_img_02 = # Your code here
4
5 print(gray_img_02[0, 0])

```

- 1 a) [107.7]
- 2 b) [97.7]
- 3 c) [87.7]
- 4 d) [77.7]

**Câu hỏi 13:** Chuyển ảnh màu sang xám (Luminosity): Hoàn thiện đoạn code sau để chuyển đổi ảnh màu thành ảnh xám theo phương pháp Luminosity:

```

1 # **** Your code here ****
2 # Sử dụng công thức Luminosity: 0.21*R + 0.72*G + 0.07*B
3 gray_img_03 = # Your code here
4
5 print(gray_img_03[0, 0])

```

- 1 a) [96.2]
- 2 b) [116.2]
- 3 c) [126.2]
- 4 d) [136.2]

**Câu hỏi 14:** Background Subtraction (Tính độ chênh lệch): Cho đoạn code sau:

```

1 import numpy as np
2 import cv2
3 bg1_image = cv2.imread("GreenBackground.png", 1)
4 bg1_image = cv2.resize(bg1_image, (678, 381))
5 ob_image = cv2.imread("Object.png", 1)
6 ob_image = cv2.resize(ob_image, (678, 381))
7 def compute_difference(bg_img, input_img):
8     return # Your code here
9 difference_single_channel = compute_difference(bg1_image, ob_image)
10 print(difference_single_channel.shape)

```

Kết quả là gì?

- 1 (a) (678, 381)
- 2 (b) (678, 381, 3)

- 3 (c) (381, 678)  
 4 (d) (381, 678, 3)

**Câu hỏi 15:** Background Subtraction (Tạo binary mask): Cho đoạn code sau:

```

1 import numpy as np
2 import cv2
3 def compute_binary_mask(difference_single_channel):
4     binary_mask = (difference_single_channel > 50).astype(np.uint8) * 255
5     return binary_mask
6 difference_single_channel = np.random.rand(678, 381) * 100
7 binary_mask = # Your code here
8 print(binary_mask.dtype)

```

Kết quả là gì?

- 1 (a) uint8  
 2 (b) float64  
 3 (c) int32  
 4 (d) float32

### III.3. Phân tích dữ liệu dạng bảng

Table 0: Dữ liệu Advertising dạng bảng

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12.0
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

**Câu hỏi 16:** Tìm giá trị lớn nhất cột Sales: Cho đoạn code sau:

```

1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv("/content/advertising.csv")
4 data = df.to_numpy()
5 sales = data[:, -1]
6 max_value = # Your code here
7 max_index = # Your code here
8 print(f"Max: {max_value} - Index: {max_index}")

```

Kết quả là gì?

- 1 (a) Max: 25 - Index: 30  
 2 (b) Max: 27 - Index: 30  
 3 (c) Max: 27 - Index: 175  
 4 (d) Max: 25 - Index: 175

**Câu hỏi 17:** Tính trung bình cột TV: Cho đoạn code sau:

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv("/content/advertising.csv")
4 data = df.to_numpy()
5 tv = data[:, 0]
6 mean_tv = # Your code here
7 print(mean_tv)
```

Kết quả là gì?

- 1 (a) 146.0  
 2 (b) 147.0  
 3 (c) 148.0  
 4 (d) 149.0

**Câu hỏi 18:** Đếm số bản ghi Sales  $\geq 20$ : Cho đoạn code sau:

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv("/content/advertising.csv")
4 data = df.to_numpy()
5 sales = data[:, -1]
6 count = # Your code here
7 print(count)
```

Kết quả là gì?

- 1 (a) 40  
 2 (b) 41  
 3 (c) 42  
 4 (d) 43

**Câu hỏi 19:** Trung bình Radio với điều kiện Sales  $\geq 15$ : Cho đoạn code sau:

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv("/content/advertising.csv")
4 data = df.to_numpy()
5 sales = data[:, -1]
6 radio = data[:, 1]
7 mean_radio = # Your code here
8 print(mean_radio)
```

Kết quả là gì?

- 1 (a) 25.2
- 2 (b) 26.2
- 3 (c) 27.2
- 4 (d) 28.2

Câu hỏi 20: Tổng Sales với điều kiện Newspaper > trung bình: Cho đoạn code sau:

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv("/content/advertising.csv")
4 data = df.to_numpy()
5 sales = data[:, -1]
6 newspaper = data[:, 2]
7 mean_newspaper = np.mean(newspaper)
8 sum_sales = # Your code here
9 print(sum_sales)
```

Kết quả là gì?

- 1 (a) 1403.1
- 2 (b) 1404.1
- 3 (c) 1405.1
- 4 (d) 1406.1

# Phụ lục

**1. Hint:** Các file code gợi ý có thể được tải [tại đây](#).

**2. Solution:** Các file code cài đặt hoàn chỉnh và phần trả lời nội dung trắc nghiệm có thể được tải [tại đây](#) (**Lưu ý:** Sáng thứ 3 khi hết deadline phần project, ad mới copy các nội dung bài giải nêu trên vào đường dẫn).

**3. Rubric:**

Mục	Kiến Thức	Đánh Giá
1	<ul style="list-style-type: none"> <li>- Thực hành các thao tác với numpy</li> <li>+ Khởi tạo array (1D, 2D tạo mảng với phần tử lặp lại, random, ...)</li> <li>+ Tìm element hoặc giá trị theo điều kiện</li> <li>+ Lấy hoặc replace các giá trị trong array theo điều kiện (slicing, indexing, ...)</li> <li>+ Thay đổi shape của array (1D-2D, 2D-1D, ...)</li> <li>+ Xếp chòng array theo các chiều</li> <li>+ Thao tác với 2 array (tìm giá trị giống, khác, max, min, ...)</li> <li>+ Hoán đổi hàng, cột, đảo ngược array</li> <li>+ Thực hiện các phép toán trên array (tìm max, min, mean theo chiều nhất định, tính khoảng cách 2 array, tìm cực đại cục bộ)</li> </ul>	<ul style="list-style-type: none"> <li>- Có thể hiểu và thực hiện được các thao tác chính và quan trọng trong thư viện numpy</li> <li>- Có nền tảng cơ bản để học và thực hiện các thao tác với mảng nhiều chiều</li> <li>- Nền tảng cơ bản để thực hiện các bài toán Linear Algebra, xử lý data cơ bản trong machine learning và deeplearning</li> </ul>
2	<ul style="list-style-type: none"> <li>- Hiểu cơ bản về các bài toán xử lý ảnh như chuyển ảnh màu thành ảnh xám</li> <li>- Cách biểu diễn và tính giá trị cho các kênh màu</li> <li>- Cách áp dụng phương pháp thay nền cho ảnh cơ bản</li> </ul>	<ul style="list-style-type: none"> <li>- Áp dụng các phép tính trên ma trận của numpy để chuyển đổi ảnh màu sang ảnh xám và thay nền cho đối tượng ảnh</li> </ul>
3	<ul style="list-style-type: none"> <li>- Một số kỹ thuật phân tích dữ liệu dạng bảng</li> <li>- Cách sử dụng pandas để load file csv</li> </ul>	<ul style="list-style-type: none"> <li>- Ứng dụng numpy để phân tích dữ liệu dạng bảng</li> </ul>