# Câu hỏi ôn tập

**QS 1: Một điểm ảnh (pixel) có tọa độ (x, y) thể hiện điều gì đối với ảnh xám và ảnh màu?**

***Answer:*** Đối với cả ảnh xám và ảnh màu, một điểm ảnh (pixel) có tọa độ (x, y) thể hiện thông tin về màu sắc hoặc độ sáng tại vị trí cụ thể trong hình ảnh.

1. Trong trường hợp ảnh xám (grayscale image):
    - Mỗi điểm ảnh ở tọa độ (x, y) chứa giá trị độ sáng duy nhất tại điểm đó.
    - Giá trị này thường nằm trong khoảng từ 0 đến 255, trong đó 0 thường thể hiện màu đen hoàn toàn, 255 thể hiện màu trắng hoàn toàn và các giá trị giữa thể hiện các mức độ sáng khác nhau.
2. Trong trường hợp ảnh màu (color image):
    - Mỗi điểm ảnh ở tọa độ (x, y) chứa thông tin về màu sắc tại điểm đó.
    - Thông thường, một điểm ảnh màu RGB (Red, Green, Blue) sẽ có ba giá trị màu tương ứng với mức độ đỏ, xanh lá cây và xanh dương.
    - Giá trị của mỗi thành phần màu thường nằm trong khoảng từ 0 đến 255, trong đó 0 thường thể hiện màu tối nhất (ví dụ: đỏ tối, xanh lá cây tối, xanh dương tối) và 255 thể hiện màu sáng nhất (ví dụ: đỏ sáng, xanh lá cây sáng, xanh dương sáng).

**QS 2: Với 1 byte/pixel thì thể hiện được cường độ sáng trên 1 pixel của ảnh nào? Cụ thể cường độ sáng bằng bao nhiêu?**

***Answer:*** Nếu mỗi pixel trong ảnh được biểu diễn bằng 1 byte (8 bit), thì pixel đó có thể biểu diễn cường độ sáng trên ảnh xám (grayscale image), giá trị biểu diễn trong khoảng từ 0 đến 255. Cụ thể, giá trị cường độ sáng của pixel được biểu diễn như sau:

- 0 tương ứng với cường độ sáng tối nhất (đen hoàn toàn).
- 255 tương ứng với cường độ sáng sáng nhất (trắng hoàn toàn).
- Các giá trị khác trong khoảng từ 1 đến 254 biểu diễn các mức độ sáng khác nhau giữa đen và trắng.

**QS 3: Sự khác nhau giữa ảnh xám và ảnh màu?**

***Answer:*** Sự khác nhau chính giữa ảnh xám và ảnh màu là trong cách chúng lưu trữ và biểu diễn thông tin về màu sắc. Dưới đây là một số điểm quan trọng để thể hiện sự khác biệt giữa chúng:

1. Thông tin màu sắc:
    - **Ảnh xám (Grayscale Image):** Trong ảnh xám, mỗi pixel chứa thông tin về độ sáng tại vị trí đó. Ảnh xám không chứa thông tin về màu sắc và thường được biểu diễn bằng cường độ sáng duy nhất từ 0 (đen) đến 255 (trắng).
    - **Ảnh màu (Color Image):** Trong ảnh màu, mỗi pixel chứa thông tin về màu sắc tại vị trí đó. Thông thường, ảnh màu sử dụng các kênh màu (ví dụ: Red, Green, Blue - RGB) để biểu diễn màu sắc. Mỗi pixel có thể có giá trị riêng cho từng kênh màu, cho phép biểu diễn một loạt màu sắc khác nhau.
2. Kích thước dữ liệu:
    - **Ảnh xám:** Vì chỉ chứa thông tin về độ sáng, ảnh xám thường có kích thước dữ liệu nhỏ hơn so với ảnh màu. Mỗi pixel chỉ cần một byte để lưu trữ thông tin cường độ sáng (từ 0 đến 255).
    - **Ảnh màu:** Với ảnh màu, cần lưu trữ thông tin cho mỗi kênh màu (thường là RGB tức Red, Green, và Blue), do đó mỗi pixel trong ảnh màu thường cần ít nhất 3 byte để lưu trữ thông tin về màu sắc. Nếu sử dụng các không gian màu khác như CMYK hoặc HSV, sẽ cần nhiều hơn 3 byte cho mỗi pixel.
3. Hiệu suất và ứng dụng:

3. **Hiệu suất và ứng dụng:**

   - **Ảnh xám:** Thích hợp cho các ứng dụng yêu cầu ít thông tin màu sắc và tập trung vào độ sáng, như hình ảnh y tế, xử lý hình ảnh và nhận dạng chữ viết.
   - **Ảnh màu:** Được sử dụng rộng rãi trong hầu hết các ứng dụng hình ảnh, bao gồm nhiếp ảnh, video, đồ họa máy tính, và nhiều ứng dụng sáng tạo khác đòi hỏi sự tái hiện chính xác của màu sắc.

**QS 4:**

```
print(img.shape) #img là ảnh đầu vào
```

**-> kết quả trả về :** `(720, 1280, 3)`

**Kết quả trên nói nên điều gì?**

***Answer:*** **Kết quả** `print(img.shape)` **cho biết thông tin về kích thước của ảnh** `img`**. Cụ thể:**

- **Số 720 là chiều cao của ảnh, tức là ảnh có 720 hàng pixel từ trên xuống dưới.**
- **Số 1280 là chiều rộng của ảnh, tức là ảnh có 1280 cột pixel từ trái sang phải.**
- **Số 3 là số kênh màu, cho biết ảnh có 3 kênh màu. Thông thường, khi số kênh màu là 3, đây là ảnh màu RGB (Red, Green, Blue), có nghĩa là mỗi pixel trong ảnh chứa thông tin về màu sắc thông qua ba thành phần màu: đỏ, xanh lá cây và xanh dương.**

**QS 5: Nêu sự khác nhau giữa Classical programing và machine learning?**

***Answer:*** **Sự khác nhau chính giữa lập trình cổ điển (Classical Programming) và học máy (Machine Learning) nằm ở cách chúng xử lý thông tin và tạo ra các giải pháp:**

1. **Lập Trình Cổ Điển (Classical Programming):**
   - **Lập trình cổ điển dựa trên quy tắc và thuật toán được xác định trước bởi người lập trình.**
   - **Người lập trình cung cấp các luật logic cụ thể để giải quyết vấn đề, và chương trình sẽ tuân theo những quy tắc này một cách tường minh.**
   - **Dữ liệu đầu vào và quy tắc xử lý dữ liệu được cung cấp trước, và chương trình thực hiện các thao tác dựa trên quy tắc này để tạo ra đầu ra.**
2. **Học Máy (Machine Learning):**
   - **Học máy là một phần của trí tuệ nhân tạo (AI) và tập trung vào việc xây dựng mô hình có khả năng "học" từ dữ liệu.**
   - **Thay vì cung cấp quy tắc cụ thể, học máy sử dụng dữ liệu để tự động học các quy tắc và mối quan hệ từ dữ liệu đầu vào.**
   - **Mô hình học máy sẽ phân tích dữ liệu, tìm ra các mẫu và quy luật ẩn sau dữ liệu, sau đó sử dụng mô hình này để đưa ra dự đoán hoặc giải quyết vấn đề.**

# Python

**2. Các kiểu dữ liệu cơ bản trong python: Int, Float, Bool, Tuple, Dictionary**

**2.1. Kiểu số: số nguyên (integer), số thực (float)**

In [117]:

```
x = 4
print(x)
print(type(x))
```

```
print(x + 1)
print(x - 1)
print(x * 2)
print(x ** 2)
y = 3.5
print(type(y))
print (y+1., y-1., y*2., y ** 2)
```

```
4
<class 'int'>
5
3
8
16
<class 'float'>
4.5 2.5 7.0 12.25
```

### 2.2. Kiểu dữ liệu logic (Boolean)

In [118]:

```
t = True
f = False
print(type(t))
print(t and f) #Logical AND
print(t or f) #Logical OR
print (not t) #Logical NOT
print(t != f) #Logical XOR
```

```
<class 'bool'>
False
True
False
True
```

### 2.3. Kiểu dữ liệu chuỗi (string)

In [119]:

```
string1 = 'Nguyen Trieu Vuong'
string2 = " - 20002182"
print(string1)
print(len(string1))
print(string1+ string2)
```

```
Nguyen Trieu Vuong
18
Nguyen Trieu Vuong - 20002182
```

### 2.4. Containers (list, dictionaries, sets, tuples)

**Một container (vùng chứa) là một lớp, một cấu trúc dữ liệu để lưu trữ các đối tượng một cách có tổ chức.**

#### 2.4.1. Danh sách (List)

In [120]:

```
xs = [1, 2, 3]
print(xs, xs[1], xs[-1])
xs[1]= 'abc'
print(xs)
xs.append('cde')
print(xs)
xs.remove('abc')
print(xs)
```

```
[1, 2, 3] 2 3
```

```
[1, 'abc', 3]
[1, 'abc', 3, 'cde']
[1, 3, 'cde']
```

In [121]:

```
name = ['Nguyen', 'Trieu', 'Vuong']
for name in name:
  print(name)
```

```
Nguyen
Trieu
Vuong
```

### 2.4.2. Dictionaries

**Một dictionary lưu trữ các cặp (khóa: giá trị)**

In [122]:

```
d = {'cat': 'cute', 'dog': 'furry'}
print(d['cat'])
print('cat' in d)
d['fish'] = 'wet'
print(d)
print (d.get('fish'))
print (d.get('monkey'))
del d['fish']
print(d.get('fish'))
```

```
cute
True
{'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
wet
None
None
```

In [123]:

```
d = {'bird': 2, 'cat': 4, 'spider' : 8}
for animal in d:
  print('A %s has %d legs' % (animal, d[animal]))
```

```
A bird has 2 legs
A cat has 4 legs
A spider has 8 legs
```

### 2.4.3. Sets

**Một set được coi là một tập hợp không có thứ tự của các phần tử riêng biệt**

In [124]:

```
set1 = {'abc', 3, 4.5, True}
print (type (set1))
print(3 in set1)
set1.add('def')
print(set1)
set1.remove('abc')
print(set1)
animals = {'cat', 'dog', 'fish'}
for ind, animal in enumerate (animals):
  print('#%d: %s' % (ind+ 1, animal))
```

```
<class 'set'>
True
{'abc', True, 3, 4.5, 'def'}
```

```
{True, 3, 4.5, 'def'}
#1: fish
#2: dog
#3: cat
```

#### 2.4.4. Tuples

**Tuple là một collection các giá trị được sắp xếp có thứ tự ( bất biến)**

In [125]:

```
t = (2,3)
print(type(t))
d = {(x, x+1): x for x in range (10)}
print(d)
print(d[(1,2)])
print(d[t])
```

```
<class 'tuple'>
{(0, 1): 0, (1, 2): 1, (2, 3): 2, (3, 4): 3, (4, 5): 4, (5, 6): 5, (6, 7): 6, (7, 8): 7, (
8, 9): 8, (9, 10): 9}
1
2
```

### 2.5. Functions

**Các hàm Python được định nghĩa bằng def.**

In [126]:

```
def first(x):
  if x > 0: return 'Positive'
  elif x < 0: return 'Negative'
  else: return 'Zero'
a = [-2, 0, 2]
for i in a:
  print (first(i))
```

```
Negative
Zero
Positive
```

### 2.6. Lớp (Classes)

In [127]:

```
class Employee:
  empCount = 0
  def __init__(self, name, salary):
    self.name = name
    self.salary = salary
    Employee.empCount += 1
  def displayCount(self):
    print("Total Employee: %d" % Employee.empCount)
  def displayEmployee(self):
    print("Name: ", self.name, ", Salary: ", self.salary)
emp1 = Employee("Trieu", 2000)
emp2 = Employee("Vuong", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
Employee.displayCount(Employee.empCount)
```

```
Name:  Trieu , Salary:  2000
Name:  Vuong , Salary:  5000
Total Employee: 2
```

# Python and Numpy for Computer Vision

### 3. Numpy

**Numpy là thư viện cốt lõi cho tính toán khoa học bằng Python. Nó cung cấp một đối tượng mảng đa chiều hiệu xuất cao và các công cụ để làm việc với mảng này.**

### 3.1. Mảng (array)

In [128]:

```
# Import thư viện NumPy với bí danh là 'np'
import numpy as np

# Tạo một mảng NumPy 1 chiều (vector) có giá trị [1, 2, 3]
a = np.array([1, 2, 3])

# In ra kiểu dữ liệu của biến 'a', thường là 'numpy.ndarray'
print(type(a))

# In ra hình dạng (shape) của mảng 'a', trong trường hợp này, là (3,), tức là mảng có 3 p
hần tử trong chiều 1
print(a.shape)

# In ra các phần tử của mảng 'a' tại các vị trí cụ thể
print(a[0], a[1], a[2])

# Gán giá trị mới cho phần tử đầu tiên của mảng 'a'
a[0] = 5

# In ra mảng 'a' sau khi thay đổi giá trị
print(a)

# Tạo một mảng NumPy 2 chiều (ma trận) có giá trị [[1, 2, 3], [4, 5, 6]]
b = np.array([[1, 2, 3], [4, 5, 6]])

# In ra hình dạng (shape) của mảng 'b', trong trường hợp này, là (2, 3), tức là mảng có 2
hàng và 3 cột
print(b.shape)

# In ra các phần tử của mảng 'b' tại các vị trí cụ thể
print(b[0, 0], b[0, 1], b[1, 0])
```

```
<class 'numpy.ndarray'>
(3,)
1 2 3
[5 2 3]
(2, 3)
1 2 4
```

In [129]:

```
import numpy as np

# Tạo ma trận 2x2 chứa toàn giá trị 0 và lưu vào biến a
a = np.zeros((2,2))
print(a)
```

```
# Tạo ma trận 1x2 chứa toàn giá trị 1 và lưu vào biến b
b = np.ones((1,2))
print(b)

# Tạo ma trận 2x2 chứa toàn giá trị 7 và lưu vào biến c
c = np.full((2,2), 7)
print(c)

# Tạo ma trận đơn vị 2x2 (ma trận có đường chéo chính có giá trị 1, còn lại là 0) và lưu vào biến d
d = np.eye(2)
print(d)

# Tạo ma trận 2x2 chứa các giá trị ngẫu nhiên từ 0 đến 1 và lưu vào biến e
e = np.random.random((2,2))
print(e)
```

```
[[0. 0.]
 [0. 0.]]
[[1. 1.]]
[[7 7]
 [7 7]]
[[1. 0.]
 [0. 1.]]
[[0.68207636 0.7209271 ]
 [0.41715518 0.88810871]]
```

### 3.2. Array indexing

In [130]:

```
import numpy as np

a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])  # Tạo một mảng NumPy a với các giá trị đã cho.
print(a)  # In ra mảng a.

b = a[:2, 1:3]  # Tạo một mảng b là một phần của mảng a, gồm các hàng từ 0 đến 1 và cột từ 1 đến 2.
print(b)  # In ra mảng b.

print(a[0, 1])  # In ra giá trị tại hàng 0, cột 1 của mảng a.

b[0, 0] = 77  # Gán giá trị 77 vào vị trí hàng 0, cột 0 của mảng b.
print(a[0, 1])  # In ra giá trị tại hàng 0, cột 1 của mảng a sau khi mảng b đã thay đổi.
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
2
77
```

### 3.3. Loại dữ liệu (datatypes)

In [131]:

```
import numpy as np

x = np.array([1, 2])  # Tạo một mảng NumPy từ danh sách [1, 2]
print(x.dtype)  # In ra kiểu dữ liệu của các phần tử trong mảng x

x = np.array([1.0, 2.0])  # Tạo một mảng NumPy từ danh sách [1.0, 2.0]
print(x.dtype)  # In ra kiểu dữ liệu của các phần tử trong mảng x

x = np.array([1, 2], dtype=np.int64)  # Tạo một mảng NumPy từ danh sách [1, 2] với kiểu dữ liệu là int64
print(x.dtype)  # In ra kiểu dữ liệu của các phần tử trong mảng x
```

```
int64
```

```
float64
int64
```

### 3.4. Broadcasting

**Broadcasting là một cơ chế mạnh mẽ cho phép numpy làm việc với các mảng có hình dạng khác nhau khi thực hiện các phép toán số học.**

In [132]:

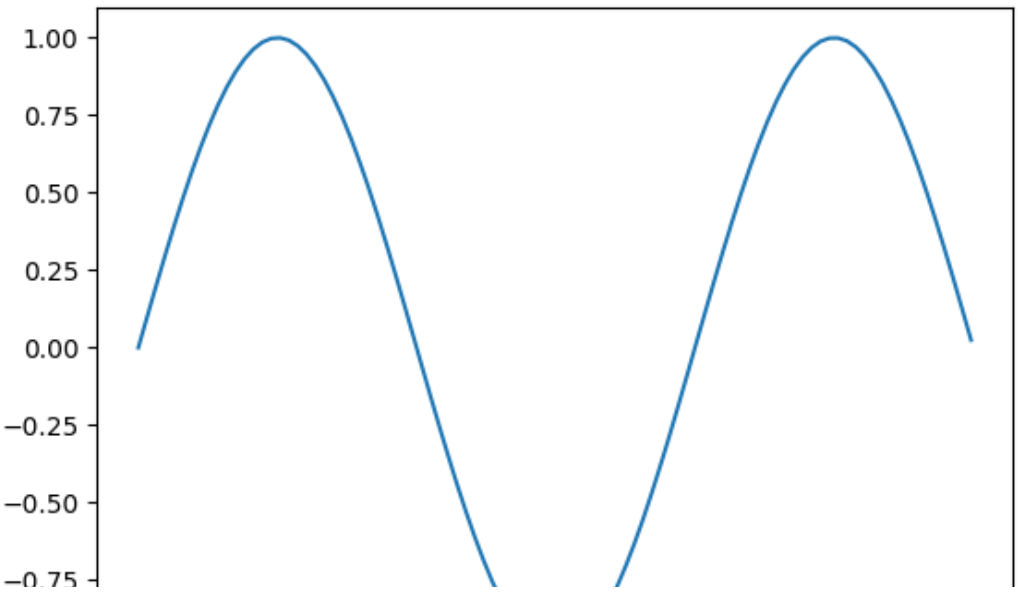
```
import numpy as np  # Import thư viện numpy để làm việc với mảng đa chiều.

x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])  # Tạo một mảng numpy 2D (ma trận) x.
v = np.array([1, 0, 1])  # Tạo một mảng numpy 1D v.

# Sử dụng hàm tile của numpy để sao chép mảng v thành một mảng 2D vv với 4 hàng và 1 cột.
vv = np.tile(v, (4, 1))

# In ra giá trị của mảng vv.
print(vv)

# Thực hiện phép cộng giữa ma trận x và ma trận vv. Mỗi hàng của x sẽ được cộng với v.
y = x + vv

# In ra giá trị của mảng y sau khi thực hiện phép cộng.
print(y)
```

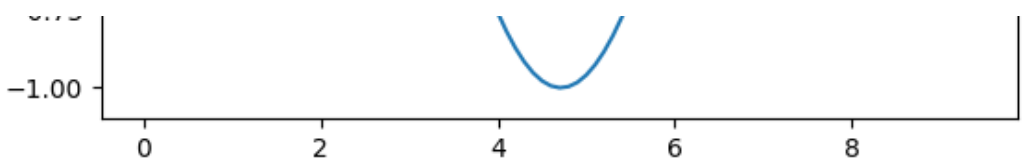
```
[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

### 4. Matplotlib

In [133]:

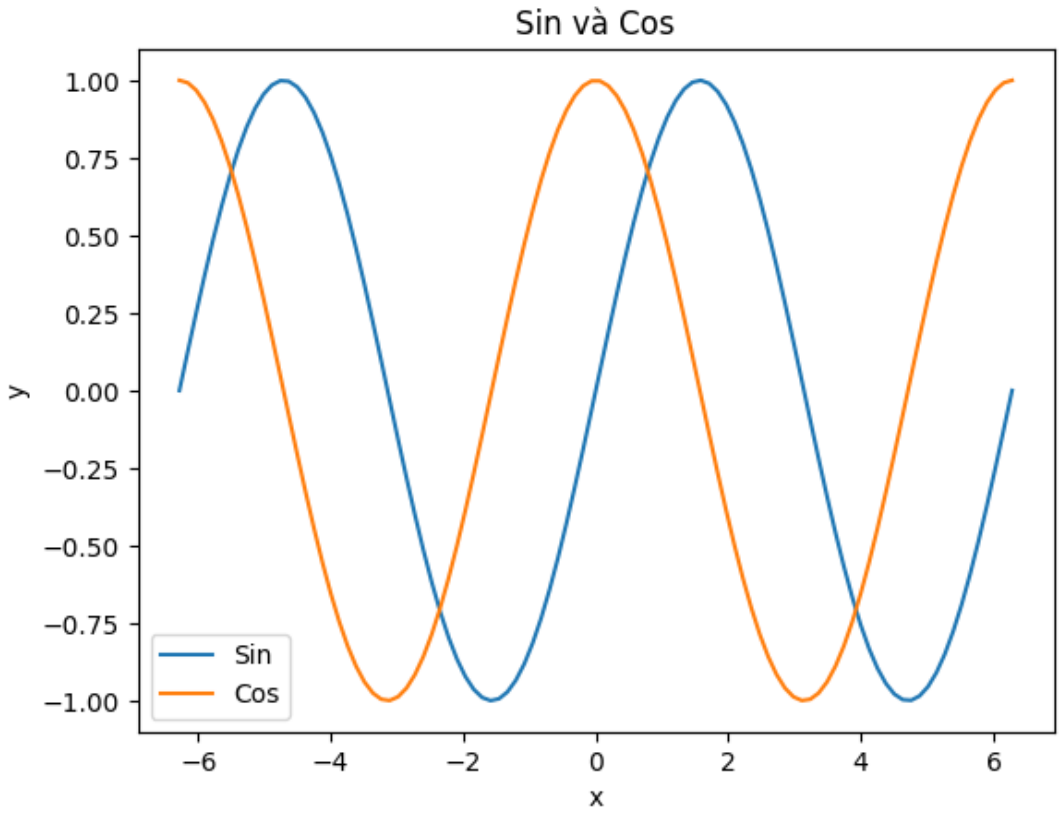
```
import numpy as np
import matplotlib.pyplot as plt  # Import thư viện Matplotlib để vẽ biểu đồ

x = np.arange(0, 3 * np.pi, 0.1)  # Tạo mảng x chứa các giá trị từ 0 đến 3*pi với bước nhảy là 0.1
y = np.sin(x)  # Tính giá trị sin của mảng x và lưu vào mảng y

plt.plot(x, y)  # Vẽ biểu đồ dựa trên các giá trị trong mảng x và y
plt.show()  # Hiển thị biểu đồ
```





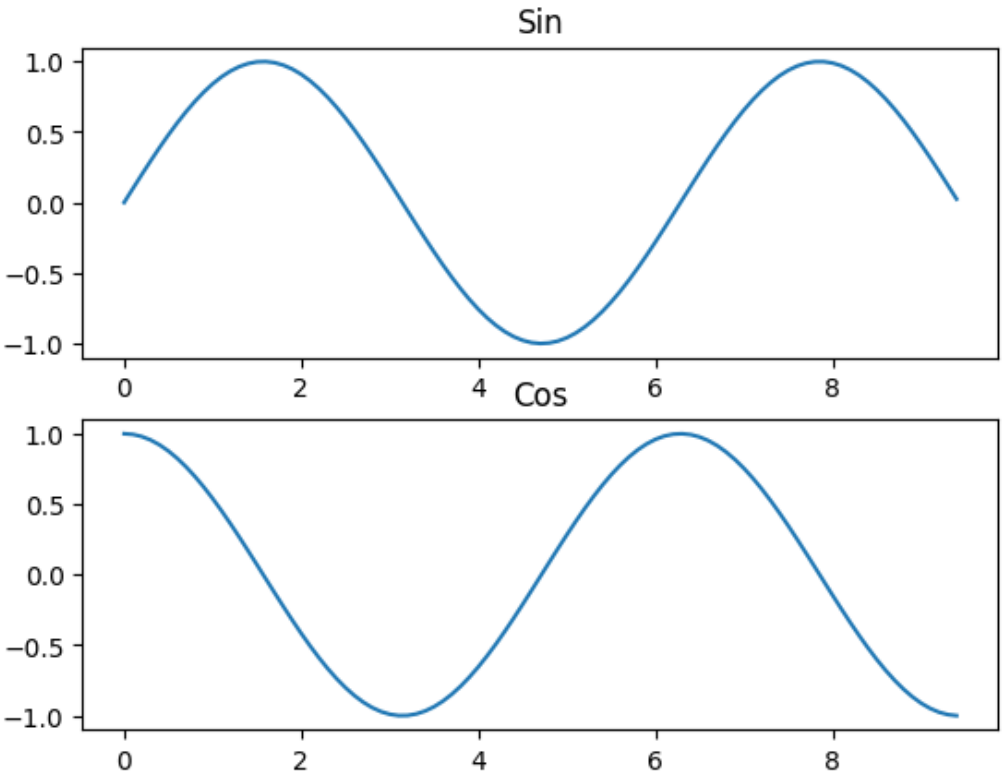

In [134]:

```
import numpy as np
import matplotlib.pyplot as plt

# Tạo dãy số x từ -2π đến 2π bằng numpy.linspace, với 100 điểm dữ liệu
x = np.linspace(-2 * np.pi, 2 * np.pi, 100)

# Tính giá trị sin(x) và lưu vào biến y_sin
y_sin = np.sin(x)

# Tính giá trị cos(x) và lưu vào biến y_cos
y_cos = np.cos(x)

# Vẽ đồ thị sin(x) với x trên trục hoành và y_sin trên trục tung
plt.plot(x, y_sin)

# Vẽ đồ thị cos(x) với x trên trục hoành và y_cos trên trục tung
plt.plot(x, y_cos)

# Đặt nhãn cho trục hoành
plt.xlabel('x')

# Đặt nhãn cho trục tung
plt.ylabel('y')

# Đặt tiêu đề cho đồ thị
plt.title('Sin và Cos')

# Đặt chú thích cho đường vẽ, ở đây là 'Sin' và 'Cos'
plt.legend(['Sin', 'Cos'])

# Hiển thị đồ thị lên màn hình
plt.show()
```




In [135]:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 3* np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.subplot(2, 1, 1)
plt.plot(x, y_sin)
plt.title('Sin')
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cos')
plt.show()
```



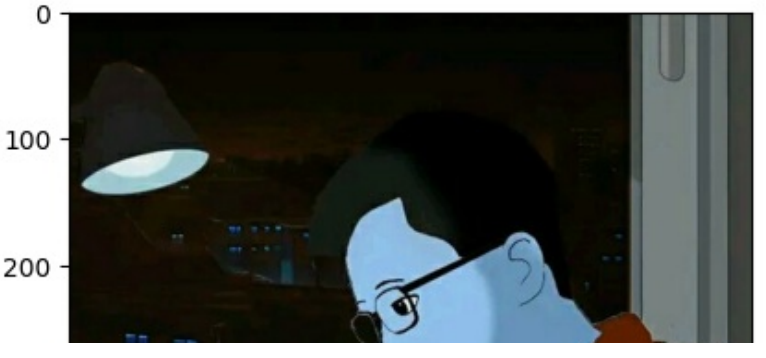

> **4.1. Matplotlib với Opencv (Open Source Computer Vision Library)**
>
> - **Khai báo thư viện opencv và matplotlib**
> - **Đọc ảnh vào bằng** `cv2.imread()`

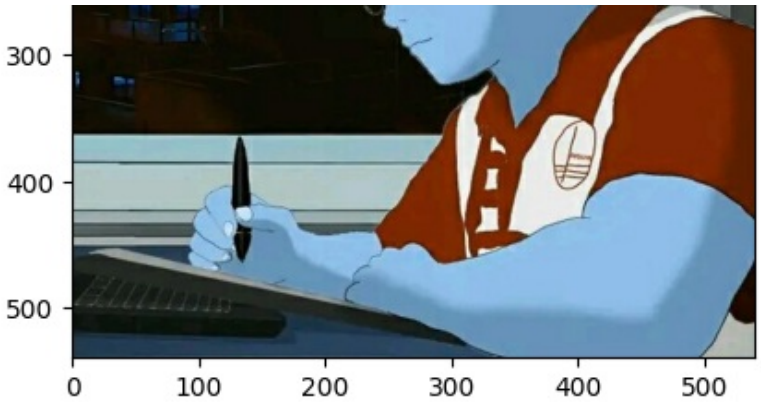
In [136]:

```
import cv2
from matplotlib import pyplot as plt
img= cv2.imread('/content/image.png')
plt.imshow(img)
```

Out[136]:

```
<matplotlib.image.AxesImage at 0x7f569e98c2b0>
```





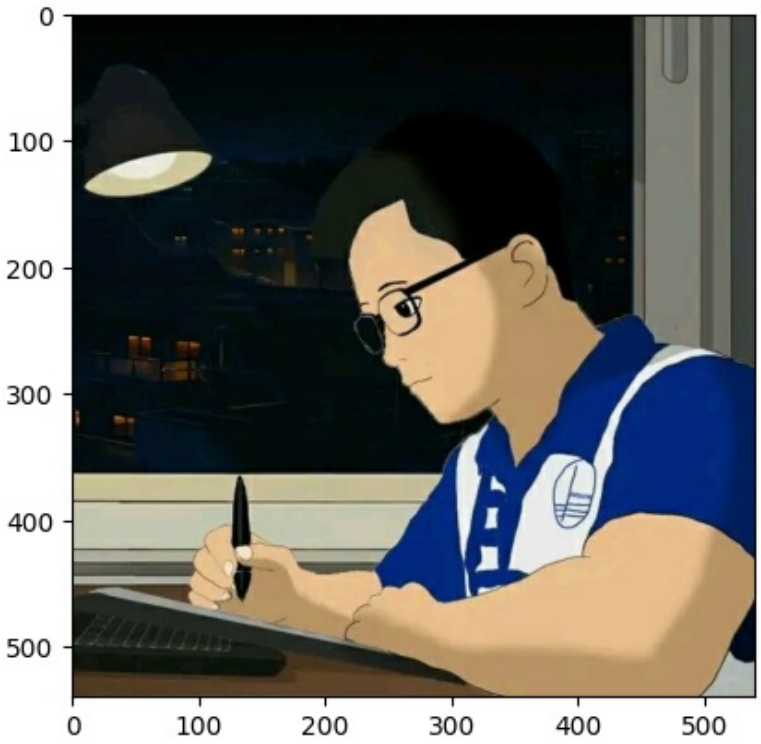

- **Convert màu ảnh từ BGR thành RGB**

In [137]:

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[137]:

```
<matplotlib.image.AxesImage at 0x7f569e759300>
```



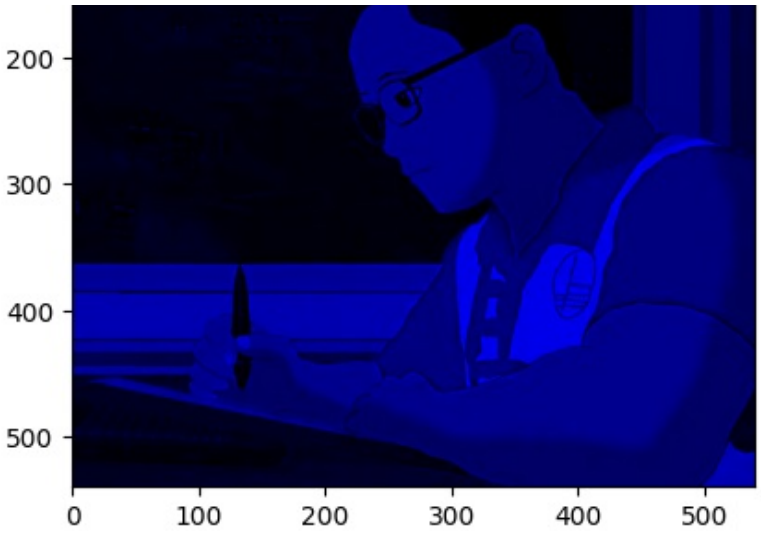

- **Lấy ra một kênh màu của ảnh đầu vào**

In [138]:

```
a = img.copy()
a[:,:,[0,1]] = 0
plt.imshow(a)
```

Out[138]:

```
<matplotlib.image.AxesImage at 0x7f569e7b3c70>
```




- **In ra shape của ảnh**
- **In ra độ sáng trên một điểm ảnh (pixel)**

In [139]:

```
print(img.shape)
print(img[0,0])
```

```
(540, 540, 3)
[12 16 19]
```

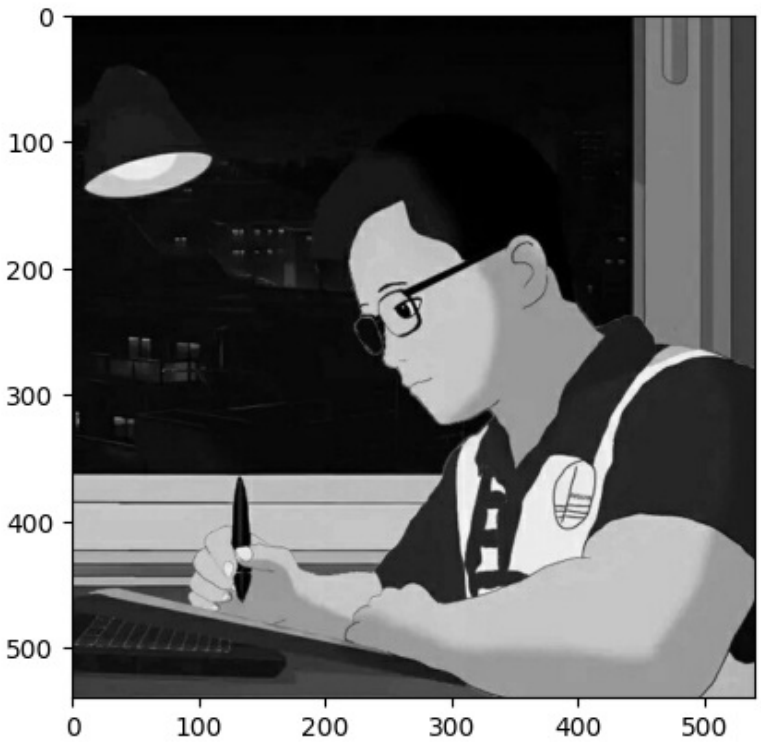
- **Convert ảnh từ RGB sang Gray**

In [140]:

```
gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
plt.imshow(gray_img, cmap = 'gray')
```

Out[140]:

```
<matplotlib.image.AxesImage at 0x7f569e60f1c0>
```



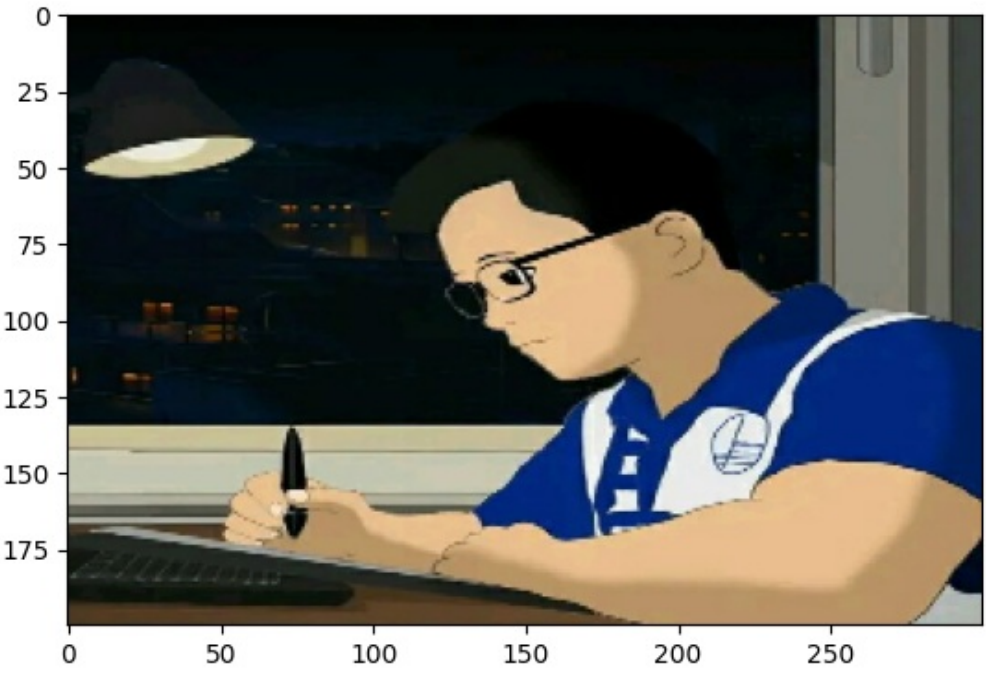

- **Thay đổi kích thước của ảnh sử dụng hàm** `cv2.resize()`

In [141]:

```
new_img = cv2.resize(img, (300,200))
plt.imshow(new_img)
```

Out[141]:

```
<matplotlib.image.AxesImage at 0x7f569e6adfc0>
```




## Mini QS

In [142]:

```
abc = [1,2,3,4,5,6,7,8,9]
sum = 0
for i in abc:
  if i % 2 == 0:
    print(i)
    sum += i
print(sum)
```

```
2
4
6
8
20
```

In [143]:

```
import random
strl = 'xin chao, con so may man cua ban la:'
list_num = list(range(1, 100))
lucky_num = random. choice (list_num)
random.choice
print(strl, lucky_num)
```

```
xin chao, con so may man cua ban la: 42
```

Xin chào, con số may mắn của bạn là: 42

In [144]:

```
import random
s=''
for i in range(0, 21, 2):
  s += str(i)

b = random.sample(range(len(s)), 2)
b.sort()

new = s[b[0]: b[1]]
print(new)
```

```
468101214161
```

In [145]:

```
def greet (*names):
  print('Hello,', names[0], names[1], names [2])

greet('Steve', 'Bill', 'Yash')
```

```
Hello, Steve Bill Yash
```

In [146]:

```
def greet (**person):
  print('Hello', person['a'],  person['b'])

greet (a='Steve', b='Jobs')
greet (a='Jobs', b='Steve', c = '1')
```

```
Hello Steve Jobs
Hello Jobs Steve
```