In [ ]:
```
BINARY FILE HANDLING IN PYTHON
Binary file handling in Python involves working with files in binary mode,
which is useful for non-textual data such as images, audio, and videos.

### 1. **Understanding Binary Files:**
   - Binary files store data in a format that is not human-readable.
   - Data is represented in binary (0s and 1s), organized in a specific structure.
   - Unlike text files, binary files don't have a specific character encoding like ASCII or UTF-8.

### 2. **Common Binary File Types:**
   - **Executable Files:** Programs compiled into binary form, ready for execution by the computer's processor.
   - **Image Files:** Store graphical information in binary format, such as JPEG, PNG, BMP, etc.
   - **Audio Files:** Contain sound data in binary format, like WAV, MP3, FLAC, etc.
   - **Video Files:** Store video data in binary form, like MP4, AVI, MOV, etc.
   - **Database Files:** Often store data in binary format for efficient storage and retrieval.

### 3. **Opening a Binary File:**
    To read from or write to a binary file, you first need to open it using the `open()` function
    with the appropriate mode:
    - `'rb'` for reading a binary file.
    - `'wb'` for writing to a binary file.

### 4. **Reading from a Binary File:**
    Once a file is opened in `'rb'` mode, you can read its contents using the `read()` method,
    which returns the data as a byte object.

    For example:

    # Opening a binary file in 'rb' mode
    with open('example.bin', 'rb') as file:
        binary_data = file.read()

### 5. **Writing to a Binary File:**
    To write data to a binary file, open the file in `'wb'` mode and use the `write()` method.
    The data must be provided as a bytes object.

    For example:

    # Writing to a binary file in 'wb' mode
    with open('output.bin', 'wb') as file:
        file.write(b'Some binary data')
```

In [4]:
```python
#Example 1
with open('data1.dat', 'wb') as f:
    text="hello world"
    bytestream = bytes(text, encoding="ASCII")
    try:
        f.write(bytestream)
    except Exception as err:
        print('Error: ',err)
    else:
        print('File saved')
```

```
File saved
```

In [9]:
```python
#Example 2
with open('data1.dat', 'rb') as f:
    data=f.read(2)
    print(data)
```

```
b'he'
```

In [11]:
```python
#Example 3
with open('data2.dat', 'wb') as f:
    text="བཀྲིས་བདེ་ལེགས།"
    bytestream = bytes(text, encoding="utf-8")
    try:
        f.write(bytestream)
    except Exception as err:
        print('Error: ',err)
    else:
        print('File saved')
```

```
File saved
```

In [16]:
```python
#Example 4
with open('data2.dat', 'rb') as f:
    data=f.read(64)
    print(data.decode(encoding="UTF-8"))
```

```
བཀྲིས་བདེ་ལེགས།
```

In [ ]:
```python
#Example 5
with open(r'C:\Users\TCV Computer Lab\Pictures\Screenshots\a.png', 'rb') as f:
    data = f.read()
    print(data)
    #we are getting the image data in encoded format
```

In [18]:
```
#Example 6
#inorder to decode image file, we can use Pillow package
from PIL import Image
import io

# Read the binary data from the image file
with open(r'C:\Users\TCV Computer Lab\Pictures\Screenshots\a.png', 'rb') as f:
    data = f.read()

# Decode the binary data using Pillow
image = Image.open(io.BytesIO(data))

# Now you can work with the image object
# For example, you can display it
image.show()
```

In [ ]:
```
Exercise 1:
Implement a Python program that copies the contents of one binary file ("source.jpg")
to another binary file ("destination.jpg").
Ensure that the copying process is done in binary mode to preserve the original file's contents.
```