Assignment 5: Complex numbers and Fourier transform

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import cv2
```

Question 1: Complex exponent

Complex exponent is the complex version of the library function `exp`.

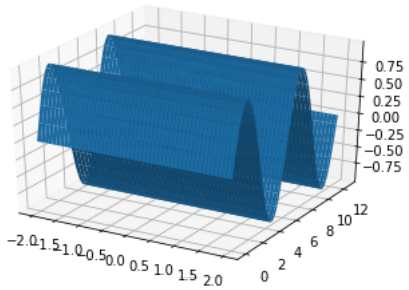
1. Implement complex exponent using real-valued functions from the `numpy` library. Use Euler's formula for the exponent of a complex number.

```
def cexp(z):
    """computes the  exponent of a complex number or a numpy array of numbers.
    """
    pass
```

2. Using Axes3d.plot_surface, plot the real and the imaginary parts of $\exp(z)$ for range $\mathcal{R}(z) \in [-2, 2], \mathcal{I}(z) \in [0, 2\pi]$.

```
# example - with 3d sin wave
x = np.linspace(-2,2,100)
y = np.linspace(0, 4*np.pi, 100)
x_1, y_1 = np.meshgrid(x, y)
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_surface(x_1, y_1, np.sin(y_1))
```

```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x7f7959d85290>
```



3. Similarly, plot the magnitude and the phase of $\exp(z)$, for the same range.
   We define the magnitude and phase of $z = a + i \cdot b$:

$$magnitude(z) = \sqrt{a^2 + b^2}$$

$$phase(z) = \arctan(\frac{b}{a})$$

   You can use np.abs and np.angle to compute those.

4. Find the maximum and minimum magnitude and phase of $exp(z - \bar{z})$ ($\bar{z}$ is the conjugate of z), for the same range.

5. Find the maximum and minimum of the real and imaginary parts of $\exp(z \cdot \bar{z})$, for the same range.

Question 2: Image processing with Fourier transform

After performing in 2022 student fest, the singer Lena-Lee tried to get back on stage for her encore. Unfortunately she got stuck behind a fence.
In this exercise we will help Lena-Lee by moving the fence using the Fourier transform.

1. Load and display the provided LenaLee.png

```
img = cv2.imread('LenaLee.png',0) # Reads the original grayscale image
if img is None:
    raise Exception("Couldnt load image, make sure you uploaded it.")
plt.imshow(img,cmap='gray')
```

2. Compute the image's DFT using the numpy.fft.fft2 function. Shift the zero-frequency component to the center of the spectrum using numpy.fft.fftshift.

Note - The fftshift usually used for visualization. Now we can see also the negative frequencies components, and at the center of the image(image.width//2, image.height//2), we can find the zero-frequency component.

3. Display the amplitude spectrum of the image.

Note - The output of numpy.fft.fftshift is complex array, to get the amplitude we use np.abs.
For visualization it is recommended to display $20 * \log(abs(output) + 1)$ instead.

4. Filter out the signal which creates the bars of the fence.

Hint 1 - Are the bars have low frequency or high frequency?
Hint 2 - A signal is represented with two symmetric points in the spectrum.
Hint 3 - As one can see, the sinusoidal noise is horizontal. That means the vertical component of the noise function is zero, and the two points represnting the noise located at the middle row of the image.

5. Apply the inverse DFT on the filtered Fourier spectrum using numpy.fft.ifft2 and numpy.fft.ifftshift. Display the output(for displaying use the absolute value of the image).