

Questão 1

Correto Atingiu 1,00 de 1,00

Dado o código abaixo, assinale a única alternativa correta acerca do seu entendimento.

```
!pip install Pillow
```

```
from PIL import Image
```

```
# Function to convert a string message to binary
```

```
def message_to_binary(message):
```

```
    return ''.join([format(ord(char), '08b') for char in message])
```

```
# Function to convert binary data to string
```

```
def binary_to_message(binary_data):
```

```
    binary_chars = [binary_data[i:i + 8] for i in range(0, len(binary_data), 8)]
```

```
    return ''.join([chr(int(binary_char, 2)) for binary_char in binary_chars])
```

```
# Encode a message into an image
```

```
def encode_message(image_path, message, output_image_path):
```

```
    # Open image
```

```
    image = Image.open(image_path)
```

```
    image = image.convert('RGB') # Ensure it's in RGB mode
```

```
    pixels = image.load()
```

```
# Convert message to binary
```

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

```
data_index = 0
```

```
# Modify the pixels
```

```
for row in range(image.size[1]):
```

```
    for col in range(image.size[0]):
```

```
        if data_index < len(binary_message):
```

```
            # Get pixel RGB values
```

```
            r, g, b = pixels[col, row]
```

```
            # Modify LSB of each channel
```

```
            r = (r & 254) | int(binary_message[data_index])
```

```
            data_index += 1
```

```
        if data_index < len(binary_message):
```

```
            g = (g & 254) | int(binary_message[data_index])
```

```
        data_index += 1

    if data_index < len(binary_message):
        b = (b & 254) | int(binary_message[data_index])
        data_index += 1

    # Update pixel value
    pixels[col, row] = (r, g, b)

# Save the image
image.save(output_image_path)
print(f'Message encoded and saved to {output_image_path}')

# Decode a message from an image
def decode_message(image_path):
    # Open image
    image = Image.open(image_path)
    image = image.convert('RGB') # Ensure it's in RGB mode
    pixels = image.load()

    binary_message = ""
    for row in range(image.size[1]):
        for col in range(image.size[0]):
            r, g, b = pixels[col, row]
            binary_message += str(r & 1)
            binary_message += str(g & 1)
            binary_message += str(b & 1)

    # Split into chunks of 8 bits and convert to characters
    hidden_message = binary_to_message(binary_message)

    # Extract message before the termination sequence
    termination_index = hidden_message.find('p')
    if termination_index != -1:
        hidden_message = hidden_message[:termination_index]

    return hidden_message

# Example usage:
```

```
# Encode a message
```

```
encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')
```

```
# Decode the message
```

```
decoded_message = decode_message('encoded_natal.png')
```

```
print('Mensagem decodificada:', decoded_message)
```

- ☐ a. O código está criptografando a mensagem a ser embutida na imagem png, usando criptografia conhecida como chave de sessão.
- ☒ b. O código usa uma técnica conhecida como Steganography para embutir uma mensagem textual em uma imagem no formato png. Em seguida, esta mensagem é extraída da imagem gerada com o algoritmo Steganography . ✓
- ☐ c. O código calcula o hash das imagens original e alterada a fim de comprovar que a imagem original foi alterada pela técnica de Steganography.
- ☐ d. O código está criptografando a mensagem a ser embutida na imagem png, usando criptografia conhecida como chave pública e privada.
- ☐ e. O arquivo alterado pela técnica de Steganography é idêntico ao arquivo original e possui o mesmo hash pois não há alteração da imagem original em relação a imagem gerada pela técnica de Steganography.

A resposta correta é:

O código usa uma técnica conhecida como Steganography para embutir uma mensagem textual em uma imagem no formato png. Em seguida, esta mensagem é extraída da imagem gerada com o algoritmo Steganography .

Questão 2

Correto Atingiu 1,00 de 1,00

Dado o código abaixo, assinale a única alternativa correta acerca do seu entendimento.

```
!pip install Pillow

from PIL import Image

# Function to convert a string message to binary

def message_to_binary(message):

    return ".join([format(ord(char), '08b') for char in message])

# Function to convert binary data to string

def binary_to_message(binary_data):

    binary_chars = [binary_data[i:i + 8] for i in range(0, len(binary_data), 8)]

    return ".join([chr(int(binary_char, 2)) for binary_char in binary_chars])

# Encode a message into an image

def encode_message(image_path, message, output_image_path):

    # Open image

    image = Image.open(image_path)

    image = image.convert('RGB') # Ensure it's in RGB mode

    pixels = image.load()

    # Convert message to binary

    binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence

    data_index = 0

    # Modify the pixels

    for row in range(image.size[1]):

        for col in range(image.size[0]):

            if data_index < len(binary_message):

                # Get pixel RGB values

                r, g, b = pixels[col, row]

                # Modify LSB of each channel

                r = (r & 254) | int(binary_message[data_index])

                data_index += 1

            if data_index < len(binary_message):

                g = (g & 254) | int(binary_message[data_index])

                data_index += 1

            if data_index < len(binary_message):

                b = (b & 254) | int(binary_message[data_index])

                data_index += 1

            # Update pixel value

            pixels[col, row] = (r, g, b)
```

```
# Save the image

image.save(output_image_path)

print(f'Message encoded and saved to {output_image_path}')

# Decode a message from an image

def decode_message(image_path):

    # Open image

    image = Image.open(image_path)

    image = image.convert('RGB') # Ensure it's in RGB mode

    pixels = image.load()

    binary_message = ""

    for row in range(image.size[1]):

        for col in range(image.size[0]):

            r, g, b = pixels[col, row]

            binary_message += str(r & 1)

            binary_message += str(g & 1)

            binary_message += str(b & 1)

    # Split into chunks of 8 bits and convert to characters

    hidden_message = binary_to_message(binary_message)

    # Extract message before the termination sequence

    termination_index = hidden_message.find('b')

    if termination_index != -1:

        hidden_message = hidden_message[:termination_index]

    return hidden_message
```

Example usage:

Encode a message

```
encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')
```

Decode the message

```
decoded_message = decode_message('encoded_natal.png')
```

```
print('Mensagem decodificada:', decoded_message)
```

- ☐ a. A função `encode_message('natal.jpg', 'Feliz Natal!', 'encoded_natal.jpg')` está sendo usada no código para embutir na imagem `encoded_natal.jpg` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.jpg`
- ☐ b. A função `encode_message('natal.gif', 'Feliz Natal!', 'encoded_natal.gif')` está sendo usada no código para embutir na imagem `encoded_natal.gif` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.gif`
- ☐ c. A função `decode_message('natal.jpg', 'Feliz Natal!', 'encoded_natal.jpg')` está sendo usada no código para embutir na imagem `encoded_natal.jpg` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.jpg`

- ☒ d. A função `encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')` está sendo usada no código para embutir na imagem `encoded_natal.png` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.png` ✓
- ☐ e. A função `decode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')` está sendo usada no código para embutir na imagem `encoded_natal.png` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.png`

A resposta correta é:

A função `encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')` está sendo usada no código para embutir na imagem `encoded_natal.png` a mensagem 'Feliz Natal', utilizando como imagem de entrada, `natal.png`

Questão 3

Correto Atingiu 1,00 de 1,00

Dado o código abaixo, assinale a única alternativa correta acerca do seu entendimento.

```
!pip install Pillow
```

```
from PIL import Image
```

```
# Function to convert a string message to binary
```

```
def message_to_binary(message):
```

```
    return ".join([format(ord(char), '08b') for char in message])
```

```
# Function to convert binary data to string
```

```
def binary_to_message(binary_data):
```

```
    binary_chars = [binary_data[i:i + 8] for i in range(0, len(binary_data), 8)]
```

```
    return ".join([chr(int(binary_char, 2)) for binary_char in binary_chars])
```

```
# Encode a message into an image
```

```
def encode_message(image_path, message, output_image_path):
```

```
    # Open image
```

```
    image = Image.open(image_path)
```

```
image = image.convert('RGB') # Ensure it's in RGB mode

pixels = image.load()

# Convert message to binary

binary_message = message_to_binary(message) + '111111111111110' # Special
termination sequence

data_index = 0

# Modify the pixels

for row in range(image.size[1]):

    for col in range(image.size[0]):

        if data_index < len(binary_message):

            # Get pixel RGB values

            r, g, b = pixels[col, row]

            # Modify LSB of each channel

            r = (r & 254) | int(binary_message[data_index])

            data_index += 1

        if data_index < len(binary_message):
```



```
g = (g & 254) | int(binary_message[data_index])

data_index += 1


if data_index < len(binary_message):

    b = (b & 254) | int(binary_message[data_index])

    data_index += 1


# Update pixel value

pixels[col, row] = (r, g, b)


# Save the image

image.save(output_image_path)

print(f'Message encoded and saved to {output_image_path}')


# Decode a message from an image

def decode_message(image_path):

    # Open image

    image = Image.open(image_path)

    image = image.convert('RGB') # Ensure it's in RGB mode

    pixels = image.load()
```

```
binary_message = ""

for row in range(image.size[1]):

    for col in range(image.size[0]):

        r, g, b = pixels[col, row]

        binary_message += str(r & 1)

        binary_message += str(g & 1)

        binary_message += str(b & 1)

# Split into chunks of 8 bits and convert to characters

hidden_message = binary_to_message(binary_message)

# Extract message before the termination sequence

termination_index = hidden_message.find('p')

if termination_index != -1:

    hidden_message = hidden_message[:termination_index]

return hidden_message
```

Example usage:

Encode a message

```
encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')
```

Decode the message

```
decoded_message = decode_message('encoded_natal.png')
```

```
print('Mensagem decodificada:', decoded_message)
```

- ☐ a. O código abaixo extraído do código dado, está convertendo o texto original para o correspondente binário, acrescido de um flag ou padding identificando o final da mensagem a ser inserida no arquivo de entrada (.jpg) . O flag será utilizado no processo de codificação da imagem alterada pela técnica de Steganography , a fim de identificar o final da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento binário a fim de que não haja necessidade de extrair o flag da mensagem decodificada.

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

- ☒ b. O código abaixo extraído do código dado, está convertendo o texto original para o correspondente binário, acrescido de um flag ou padding identificando o final da mensagem a ser inserida no arquivo de entrada (.png) . O flag será utilizado no processo de decodificação da imagem alterada pela técnica de Steganography , a fim de identificar o final da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento de string a fim de extrair o flag da mensagem decodificada. ✓

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

- ☐ c. O código abaixo extraído do código dado, está convertendo a imagem original para a correspondente imagem binária, acrescido de um flag ou padding identificando o final da imagem a ser inserida no arquivo de entrada (.png) . O flag será utilizado no processo de decodificação da imagem alterada pela técnica de Steganography , a fim de identificar o final da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento de string a fim de extrair o flag da mensagem decodificada.

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

- ☐ d. O código abaixo extraído do código dado, está convertendo o texto original para o correspondente binário, acrescido de um flag ou padding identificando o final da mensagem a ser inserida no arquivo de entrada (.png) . O flag será utilizado no processo de codificação da imagem alterada pela técnica de Steganography , a fim de identificar o início da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento de string a fim de extrair o flag da mensagem decodificada.

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

- ☐ e. O código abaixo extraído do código dado, está convertendo o texto binário para o correspondente texto em um sistema de codificação que possa ser recuperado pelo Sistema Operacional, acrescido de um flag ou padding identificando o final da mensagem a ser inserida no arquivo de entrada (.png) . O flag será utilizado no processo de decodificação da imagem alterada pela técnica de Steganography , a fim de identificar o final da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento de string a fim de extrair o flag da mensagem decodificada.

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

A resposta correta é:

O código abaixo extraído do código dado, está convertendo o texto original para o correspondente binário, acrescido de um flag ou padding identificando o final da mensagem a ser inserida no arquivo de entrada (.png) . O flag será utilizado no processo de decodificação da imagem alterada pela técnica de Steganography , a fim de identificar o final da mensagem inserida na imagem. Apesar do flag ser mostrado na mensagem recuperada, idealmente a mensagem recuperada precisa ser tratada com funções de processamento de string a fim de extrair o flag da mensagem decodificada.

```
binary_message = message_to_binary(message) + '111111111111110' # Special termination sequence
```

Questão 4

Correto Atingiu 1,00 de 1,00

Dado o código abaixo, assinale a única alternativa correta acerca do seu entendimento.

```
!pip install Pillow
```

```
from PIL import Image
```

```
# Function to convert a string message to binary
```

```
def message_to_binary(message):
```

```
    return ".join([format(ord(char), '08b') for char in message])
```

```
# Function to convert binary data to string
```

```
def binary_to_message(binary_data):
```

```
    binary_chars = [binary_data[i:i + 8] for i in range(0, len(binary_data), 8)]
```

```
    return ".join([chr(int(binary_char, 2)) for binary_char in binary_chars])
```

```
# Encode a message into an image
```

```
def encode_message(image_path, message, output_image_path):
```

```
    # Open image
```

```
    image = Image.open(image_path)
```

```
image = image.convert('RGB') # Ensure it's in RGB mode

pixels = image.load()

# Convert message to binary

binary_message = message_to_binary(message) + '111111111111110' # Special
termination sequence

data_index = 0

# Modify the pixels

for row in range(image.size[1]):

    for col in range(image.size[0]):

        if data_index < len(binary_message):

            # Get pixel RGB values

            r, g, b = pixels[col, row]

            # Modify LSB of each channel

            r = (r & 254) | int(binary_message[data_index])

            data_index += 1

        if data_index < len(binary_message):
```

```
g = (g & 254) | int(binary_message[data_index])

data_index += 1


if data_index < len(binary_message):

    b = (b & 254) | int(binary_message[data_index])

    data_index += 1


# Update pixel value

pixels[col, row] = (r, g, b)


# Save the image

image.save(output_image_path)

print(f'Message encoded and saved to {output_image_path}')


# Decode a message from an image

def decode_message(image_path):

    # Open image

    image = Image.open(image_path)

    image = image.convert('RGB') # Ensure it's in RGB mode

    pixels = image.load()
```

```
binary_message = ""

for row in range(image.size[1]):

    for col in range(image.size[0]):

        r, g, b = pixels[col, row]

        binary_message += str(r & 1)

        binary_message += str(g & 1)

        binary_message += str(b & 1)

# Split into chunks of 8 bits and convert to characters

hidden_message = binary_to_message(binary_message)

# Extract message before the termination sequence

termination_index = hidden_message.find('p')

if termination_index != -1:

    hidden_message = hidden_message[:termination_index]

return hidden_message
```



```
# Example usage:
```

```
# Encode a message
```

```
encode_message('natal.png', 'Feliz Natal!', 'encoded_natal.png')
```

```
# Decode the message
```

```
decoded_message = decode_message('encoded_natal.png')
```

```
print('Mensagem decodificada:', decoded_message)
```

☐ a. As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography
2. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida
3. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
4. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada

☐ b. As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
2. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography
3. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida
4. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada

☒ c.



As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography
2. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
3. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida
4. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada

☐ d. As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
2. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida
3. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada
4. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography

☐ e. As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography
2. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
3. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada
4. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida

A resposta correta é:

As funções:

1. message_to_binary(message)
2. binary_to_message(binary_data)
3. encode_message(image_path, message, output_image_path)
4. decode_message(image_path)

Fazem respectivamente:

1. Converte a mensagem textual para a corresponde mensagem binária, a fim de ser inserida na imagem a ser alterada pela técnica de Steganography
2. Converte a mensagem recuperada da imagem alterada pela técnica de Steganography, do formato binário para o texto plano a fim de ser mostrada para o usuário
3. Codifica a mensagem na imagem de entrada no formato png , gerando uma imagem alterada com a mensagem embutida
4. Decodifica a imagem alterada pela técnica de Steganography, a fim de extrair a mensagem embutida na imagem alterada

◀ APS11 - Atividade Prática Supervisionada

Seguir para...

APS13 - Atividade Prática Supervisionada ▶

...