TP3 : Reconstruction d'images tomographiques

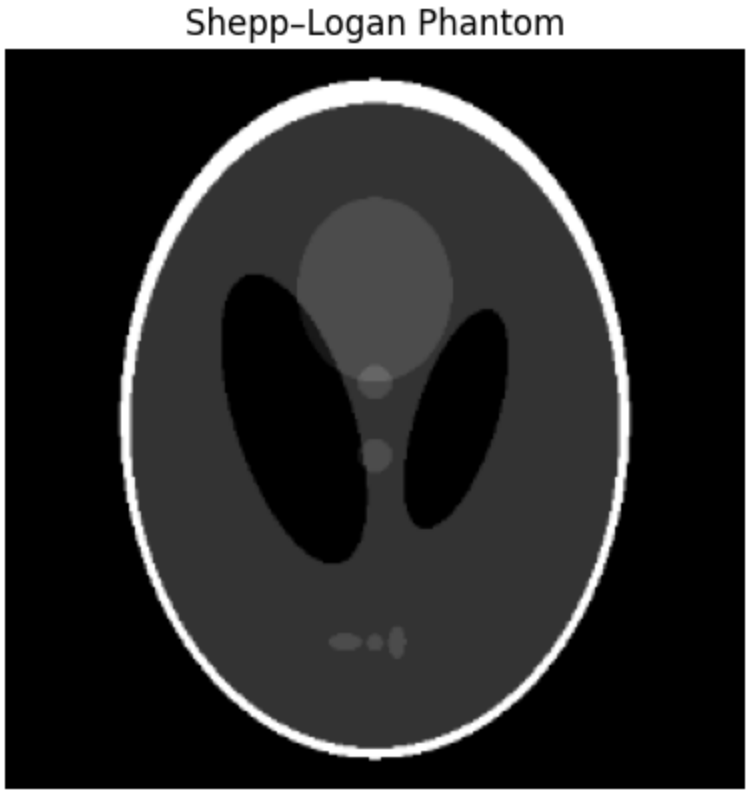
Nom et Prenom : [GHOMARI Alae]
Spécialité : [IAA]
Groupe : [02]

Cette cellule importe toutes les bibliothèques nécessaires au TP :

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.data import shepp_logan_phantom
from skimage.transform import radon, iradon, iradon_sart, resize
from skimage.metrics import mean_squared_error, peak_signal_noise_ratio, structural
```

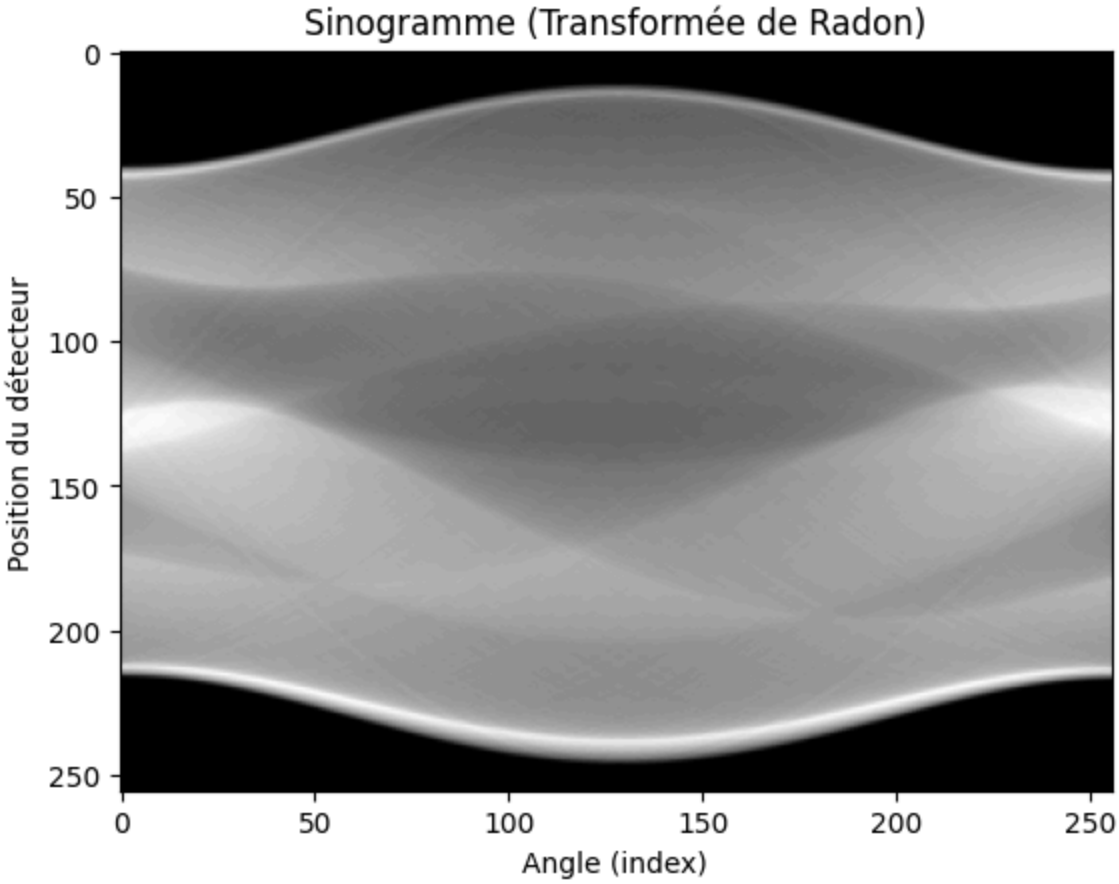
On charge l'image *Shepp–Logan Phantom* depuis la bibliothèque prédéfinie `scikit-image` , on la redimensionne en 256×256 et on l'affiche en niveaux de gris.

In [7]:
```
# Charger le phantom
phantom = shepp_logan_phantom()

# Redimensionner pour un format carré utilisable
phantom = resize(phantom, (256, 256), anti_aliasing=True)

# Affichage
plt.imshow(phantom, cmap='gray')
plt.title("Shepp-Logan Phantom")
plt.axis('off')
plt.show()
```



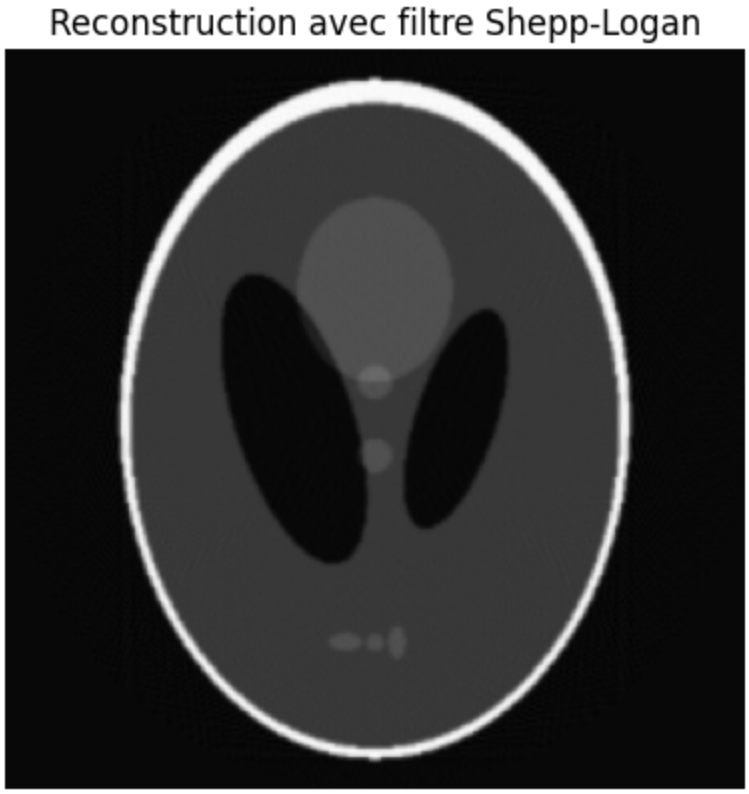
On définit les angles de projection entre 0° et 180°, puis on applique la **transformée de Radon** pour obtenir le **sinogramme**, qui représente les projections de l'image sous différents angles. Et on l'affiche en niveaux de gris.

In [8]:
```
theta = np.linspace(0., 180., max(phantom.shape), endpoint=False)

# Calcul du sinogramme
sinogram = radon(phantom, theta=theta, circle=True)

# Affichage
plt.imshow(sinogram, cmap='gray', aspect='auto')
plt.title("Sinogramme (Transformée de Radon)")
plt.xlabel("Angle (index)")
plt.ylabel("Position du détecteur")
plt.show()
```



On applique la fonction `iradon` pour reconstruire l'image à partir du sinogramme à l'aide du **filtre Shepp–Logan (méthode FBP)**, puis on affiche le résultat reconstruit.

In [9]:
```
from skimage.transform import iradon
import matplotlib.pyplot as plt

recon_shepp = iradon(sinogram, theta=theta, filter_name='shepp-logan', circle=True)

plt.imshow(recon_shepp, cmap='gray')
plt.title("Reconstruction avec filtre Shepp-Logan")
plt.axis('off')
plt.show()
```



On utilise la méthode **SART** pour reconstruire l'image de manière **itérative** à partir du sinogramme.
Chaque itération améliore progressivement la qualité de la reconstruction.

In [10]:
```
recon_sart = iradon_sart(sinogram, theta=theta)  # première itération
for i in range(10):  # itérations supplémentaires
    recon_sart = iradon_sart(sinogram, theta=theta, image=recon_sart)

plt.imshow(recon_sart, cmap='gray')
plt.title("Reconstruction par SART (iradon_sart)")
plt.axis('off')
plt.show()
```

On compare les deux méthodes (**FBP** et **SART**) à l'aide de trois métriques :
**MSE ** : mesure l'erreur moyenne entre l'image originale et reconstruite.
**PSNR ** : évalue la qualité de la reconstruction.
**SSIM ** : mesure la similarité visuelle entre les images.
Les valeurs sont affichées pour juger laquelle des deux reconstructions est la plus fidèle à l'image originale.

In [11]:
```
# Calcul et affichage direct des métriques
for name, img in {"FBP": recon_shepp, "SART": recon_sart}.items():
    img = np.clip(img, 0, 1)
    mse = mean_squared_error(phantom, img)
    psnr = peak_signal_noise_ratio(phantom, img, data_range=1.0)
    ssim = structural_similarity(phantom, img, data_range=1.0)
    print(f"{name} - MSE: {mse:.6f}, PSNR: {psnr:.2f}, SSIM: {ssim:.4f}")
```

```
FBP - MSE: 0.000999, PSNR: 30.01, SSIM: 0.9504
SART - MSE: 0.000289, PSNR: 35.38, SSIM: 0.8975
```

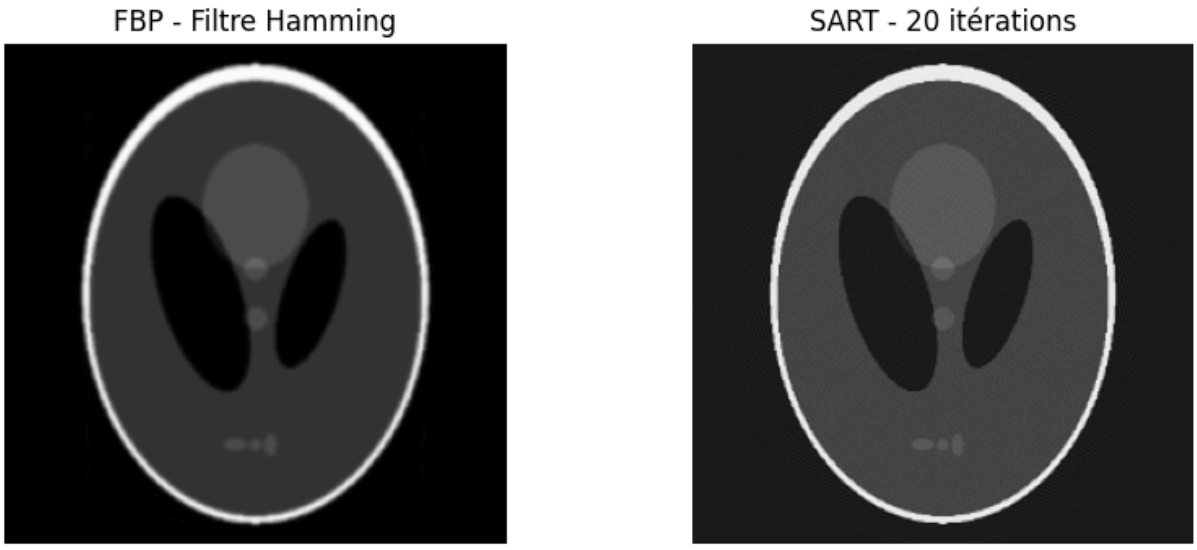
Interprétation des résultats

Avec le **filtre Shepp–Logan** et **10 itérations pour SART**,
SART donne une image plus lisse et plus propre(MSE plus faible , PSNR plus élevé),
FBP préserve mieux les détails mais ajoute un peu de bruit(SSIM légèrement supérieur). ==> SART est meilleure

Experience (2)

On change le filtre FBP en "hamming" , et on teste un nombre d'itérations différent pour SART (exemple : 20)

In [14]:
```
# Reconstruction FBP avec filtre Hamming
recon_fbp_hamming = iradon(sinogram, theta=theta, filter_name='hamming', circle=Tru

# Reconstruction SART avec plus d'itérations (ex : 20)
recon_sart_iter = iradon_sart(sinogram, theta=theta)
for i in range(20):  # nombre d'itérations ajusté
    recon_sart_iter = iradon_sart(sinogram, theta=theta, image=recon_sart_iter)

# Affichage des reconstructions
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
axes[0].imshow(recon_fbp_hamming, cmap='gray')
axes[0].set_title("FBP - Filtre Hamming")
axes[0].axis('off')

axes[1].imshow(recon_sart_iter, cmap='gray')
axes[1].set_title("SART - 20 itérations")
axes[1].axis('off')

plt.show()

# Comparaison des métriques
for name, img in {"FBP (Hamming)": recon_fbp_hamming, "SART (20 itérations)": recon
    img = np.clip(img, 0, 1)
    mse = mean_squared_error(phantom, img)
    psnr = peak_signal_noise_ratio(phantom, img, data_range=1.0)
    ssim = structural_similarity(phantom, img, data_range=1.0)
    print(f"{name} - MSE: {mse:.6f}, PSNR: {psnr:.2f}, SSIM: {ssim:.4f}")
```




```
FBP (Hamming) - MSE: 0.001931, PSNR: 27.14, SSIM: 0.9600
SART (20 itérations) - MSE: 0.000260, PSNR: 35.85, SSIM: 0.8791
```

Interprétation des résultats Experience (2)

SART avec 20 itérations donne une image plus nette et de meilleure qualité (PSNR plus élevé),
tandis que **FBP (Hamming)** garde bien les détails mais reste un peu plus bruité.

les résultats montrent que SART reste la meilleure approche ,une meilleure qualité d'image globale(PSNR plus élevé). FBP (Hamming) garde un SSIM légèrement supérieur, ce qui veut dire qu'il préserve un peu mieux la structure, mais avec plus de bruit.