Voici un programme Python qui simule l'action d'un filtre passe-bas et passe-haut sur un signal périodique dont le spectre est fourni.
Il utilise NumPy et Matplotlib pour générer et visualiser le signal et son spectre avant et après filtrage.

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

# 1. Paramètres du signal
fs = 1000  # Fréquence d'échantillonnage (Hz)
T = 1      # Durée du signal (s)
t = np.linspace(0, T, fs, endpoint=False)  # Axe temporel

# 2. Génération du signal périodique (somme de sinusoïdes)
freqs = [50, 150, 300]  # Fréquences en Hz
amplitudes = [1, 0.5, 0.2]  # Amplitudes des composantes
signal = sum(A * np.sin(2 * np.pi * f * t) for A, f in
zip(amplitudes, freqs))

# 3. Calcul du spectre du signal (FFT)
fft_signal = np.fft.fft(signal)
freqs_fft = np.fft.fftfreq(len(signal), 1/fs)

# 4. Définition des filtres
def filtre_passe_bas(signal_fft, freqs, fc):
    """Atténue les fréquences supérieures à fc"""
    filtré = signal_fft.copy()
    filtré[np.abs(freqs) > fc] = 0
    return filtré

def filtre_passe_haut(signal_fft, freqs, fc):
    """Atténue les fréquences inférieures à fc"""
    filtré = signal_fft.copy()
    filtré[np.abs(freqs) < fc] = 0
    return filtré

# 5. Application des filtres
fc = 100  # Fréquence de coupure du filtre
fft_passe_bas = filtre_passe_bas(fft_signal, freqs_fft, fc)
fft_passe_haut = filtre_passe_haut(fft_signal, freqs_fft, fc)

# 6. Reconstruction du signal après filtrage
signal_passe_bas = np.fft.ifft(fft_passe_bas).real
signal_passe_haut = np.fft.ifft(fft_passe_haut).real

# 7. Affichage des résultats
fig, axs = plt.subplots(3, 2, figsize=(12, 8))

# Signal initial
axs[0, 0].plot(t, signal)
axs[0, 0].set_title("Signal initial")
axs[0, 1].stem(freqs_fft[:fs//2], np.abs(fft_signal[:fs//2]),
use_line_collection=True)
axs[0, 1].set_title("Spectre du signal initial")

# Signal passe-bas
axs[1, 0].plot(t, signal_passe_bas, 'g')
axs[1, 0].set_title(f"Signal après filtre passe-bas (fc={fc} Hz)")
axs[1, 1].stem(freqs_fft[:fs//2], np.abs(fft_passe_bas[:fs//2]),
```

```
use_line_collection=True, linefmt='g', markerfmt='go')
axs[1, 1].set_title(f"Spectre après filtre passe-bas (fc={fc} Hz)")

# Signal passe-haut
axs[2, 0].plot(t, signal_passe_haut, 'r')
axs[2, 0].set_title(f"Signal après filtre passe-haut (fc={fc} Hz)")
axs[2, 1].stem(freqs_fft[:fs//2], np.abs(fft_passe_haut[:fs//2]),
use_line_collection=True, linefmt='r', markerfmt='ro')
axs[2, 1].set_title(f"Spectre après filtre passe-haut (fc={fc} Hz)")

for ax in axs.flat:
    ax.set_xlabel("Temps (s)" if ax in axs[:, 0] else "Fréquence
(Hz)")

plt.tight_layout()
plt.show()
```

Explication du programme :

1.Génération d'un signal périodique : somme de plusieurs sinusoïdes de différentes fréquences.

2.Calcul et affichage du spectre par Transformée de Fourier Discrète (FFT).

3.Application de filtres (passe-bas et passe-haut) en supprimant les fréquences indésirables.

4.Visualisation des résultats : signal filtré et son spectre.

Analyse des résultats :

Signal initial : Composé de 3 fréquences (50 Hz, 150 Hz, 300 Hz).

Filtrage passe-bas (fc = 100 Hz) :
Seules les fréquences < 100 Hz sont conservées (ici 50 Hz).
Le signal devient plus lisse.

Filtrage passe-haut (fc = 100 Hz) :
Seules les fréquences > 100 Hz sont conservées (150 Hz et 300 Hz).
Le signal devient plus complexe.