Podczas wykonywania DFT za każdym razem dzielę przez N, a odwrotnej DFT mnożę przez N. Robię tak, żeby przyjąć konwencję stosowaną na wykładzie (w bibliotece `numpy` jest odmienna). Dzięki temu wszystkie wzory z wykładu mają tu zastosowanie.

1. Dane są dwa sygnały o okresie podstawowym $N = 4$: $s_1 = \{2,3,1,0\}$ i $s_2 = \{0,3,1,0\}$.

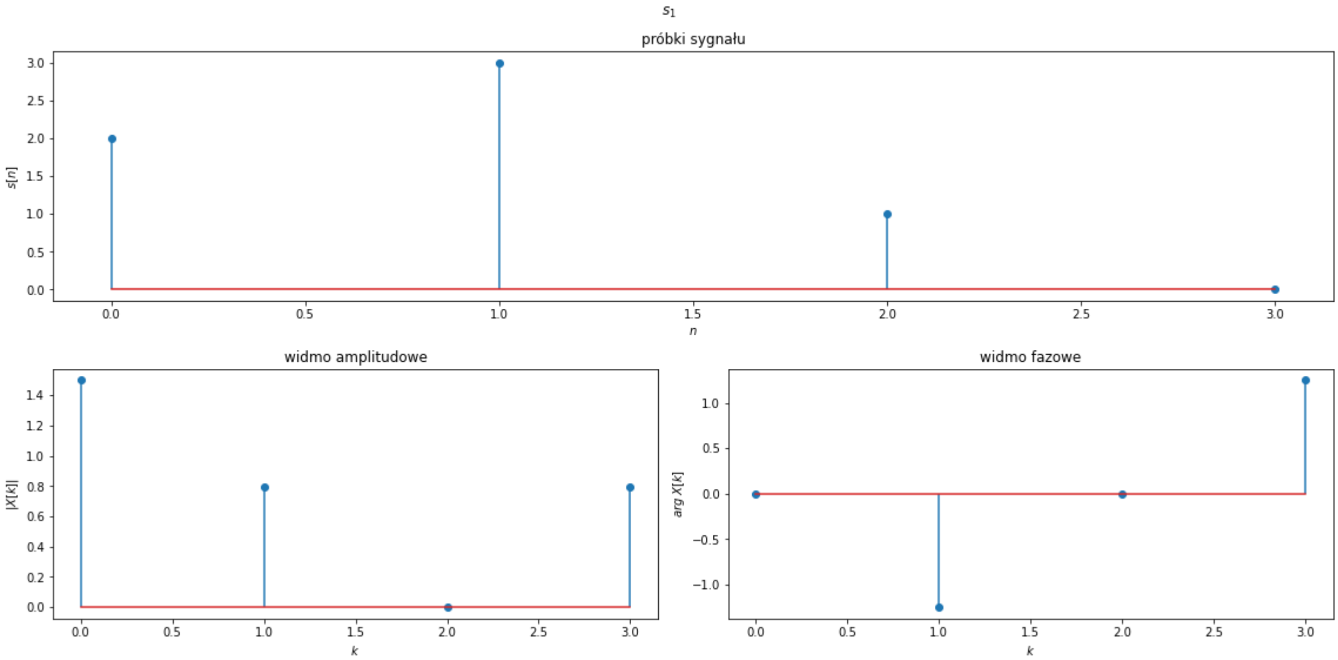
a) Dla każdego sygnału wyznaczyć i wykreślić widmo amplitudowe i fazowe, obliczyć moc sygnału i sprawdzić słuszność twierdzenia Parsevala.

b) Sprawdzić słuszność twierdzenia o dyskretnej transformacji Fouriera splotu kołowego sygnałów s_1 i s_2: wyznaczyć ręcznie splot kołowy sygnałów s_1 i s_2, a następnie wyznaczyć ten splot ponownie za pomocą dyskretnej transformacji Fouriera.

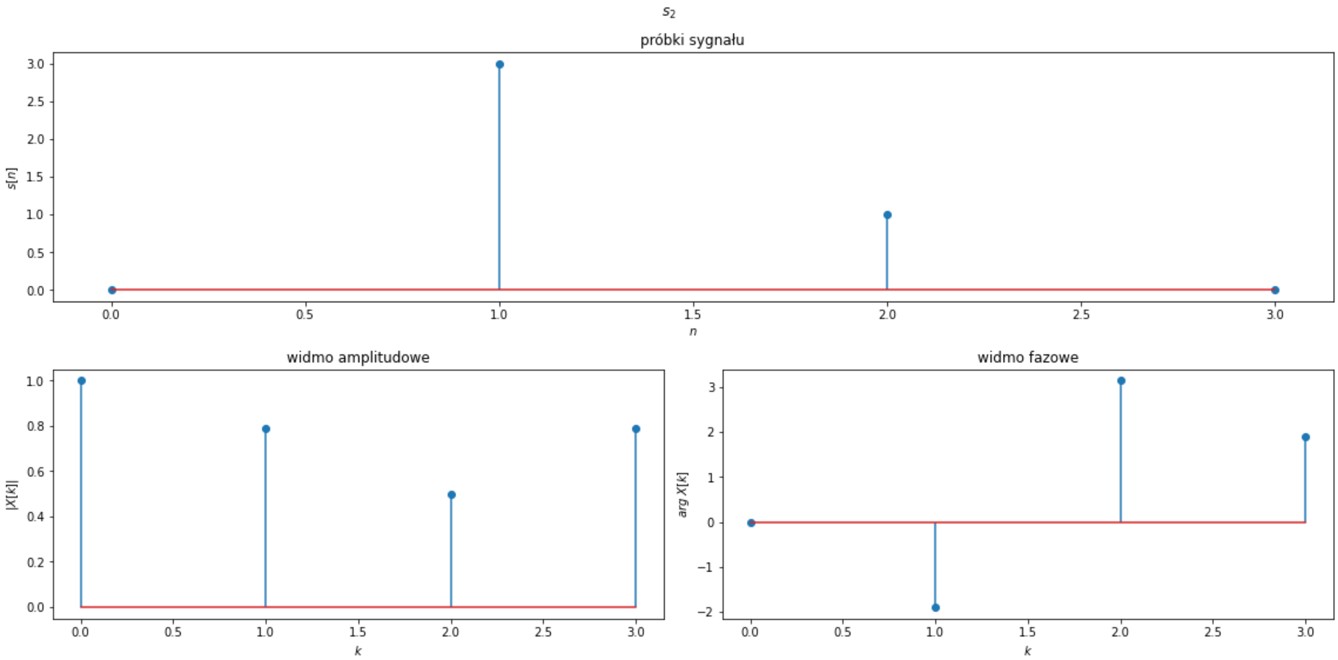
```
from matplotlib import pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np

N = 4

s1 = np.array([2, 3, 1, 0])
s2 = np.array([0, 3, 1, 0])

SIGNALS = [s1, s2]
SIGNAL_NAMES = ["$s_1$", "$s_2$"]
```

```
# 1 a)

for s, name in zip(SIGNALS, SIGNAL_NAMES):
    fig = plt.figure(figsize=(16, 8))
    fig.suptitle(f"{name}")
    gs = GridSpec(ncols=2, nrows=2, figure=fig)

    ax_s = fig.add_subplot(gs[0, :])
    ax_ampl = fig.add_subplot(gs[1, 0])
    ax_phase = fig.add_subplot(gs[1, 1])

    ax_s.stem(s)
    ax_s.set_title("próbki sygnału")
    ax_s.set_xlabel("$n$")
    ax_s.set_ylabel("$s[n]$")

    s_fft = np.fft.fft(s) / N

    s_amplitude_spectrum = np.abs(s_fft)
    s_phase_spectrum = np.angle(s_fft)

    ax_ampl.stem(s_amplitude_spectrum)
    ax_ampl.set_title("widmo amplitudowe")
    ax_ampl.set_xlabel("$k$")
    ax_ampl.set_ylabel("$|X[k]|$")

    ax_phase.stem(s_phase_spectrum)
    ax_phase.set_title("widmo fazowe")
    ax_phase.set_xlabel("$k$")
    ax_phase.set_ylabel("$arg \\; X[k]$")

    power = np.mean(np.square(s))
    power_parsival = np.sum(np.square(s_amplitude_spectrum))

    bot_text = f"Moc: {power}\nMoc z tw. Parsivala: {power_parsival}"
```

```
    fig.text(0.5, 0, bot_text, ha="center", va="top", fontsize=16)

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



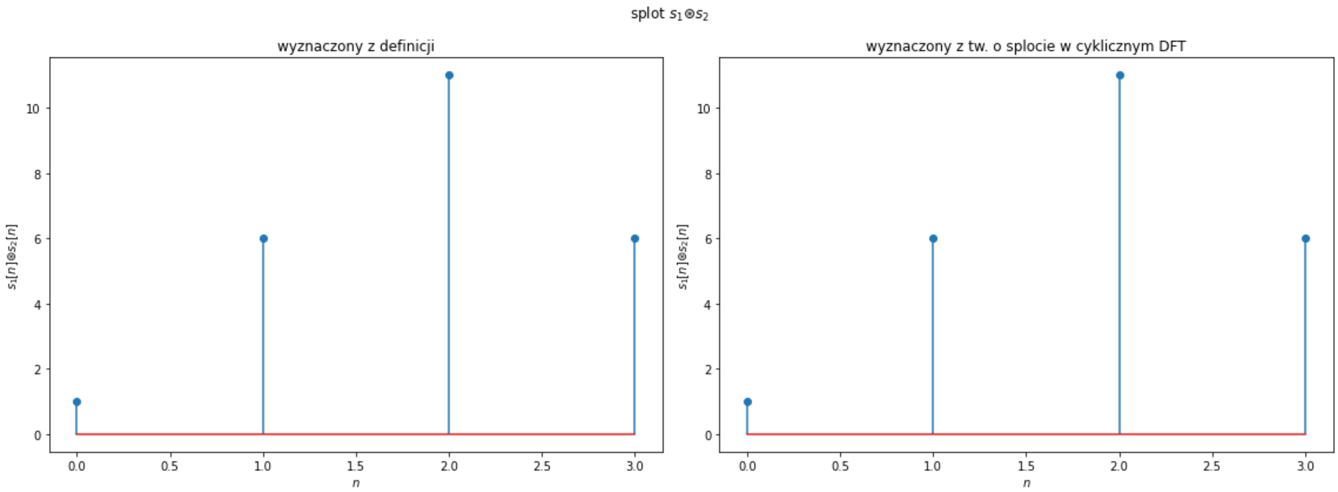
Moc: 3.5
Moc z tw. Parsivala: 3.5



Moc: 2.5
Moc z tw. Parsivala: 2.5

```
# 1) b

def circular_convolve(x1: np.ndarray, x2: np.ndarray):
    N = len(x1)
    convolution = [0] * N
    for n in range(N):
        for m in range(len(x1)):
            m2 = (n - m) % N
            convolution[n] += x1[m] * x2[m2]
    return np.array(convolution)


fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle("splot $s_1 \\circledast s_2$")
```

```
fig.set_size_inches(16, 6)

convolution = circular_convolve(s1, s2)
ax1.stem(convolution)
ax1.set_title("wyznaczony z definicji")

s1_fft = np.fft.fft(s1) / N
s2_fft = np.fft.fft(s2) / N
convolution_from_fft = np.real(np.fft.ifft(s1_fft * s2_fft) * N) * N

ax2.stem(convolution_from_fft)
ax2.set_title("wyznaczony z tw. o splocie w cyklicznym DFT")

for ax in [ax1, ax2]:
    ax.set_xlabel("$n$")
    ax.set_ylabel("$s_1[n] \\circledast s_2[n]$")

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

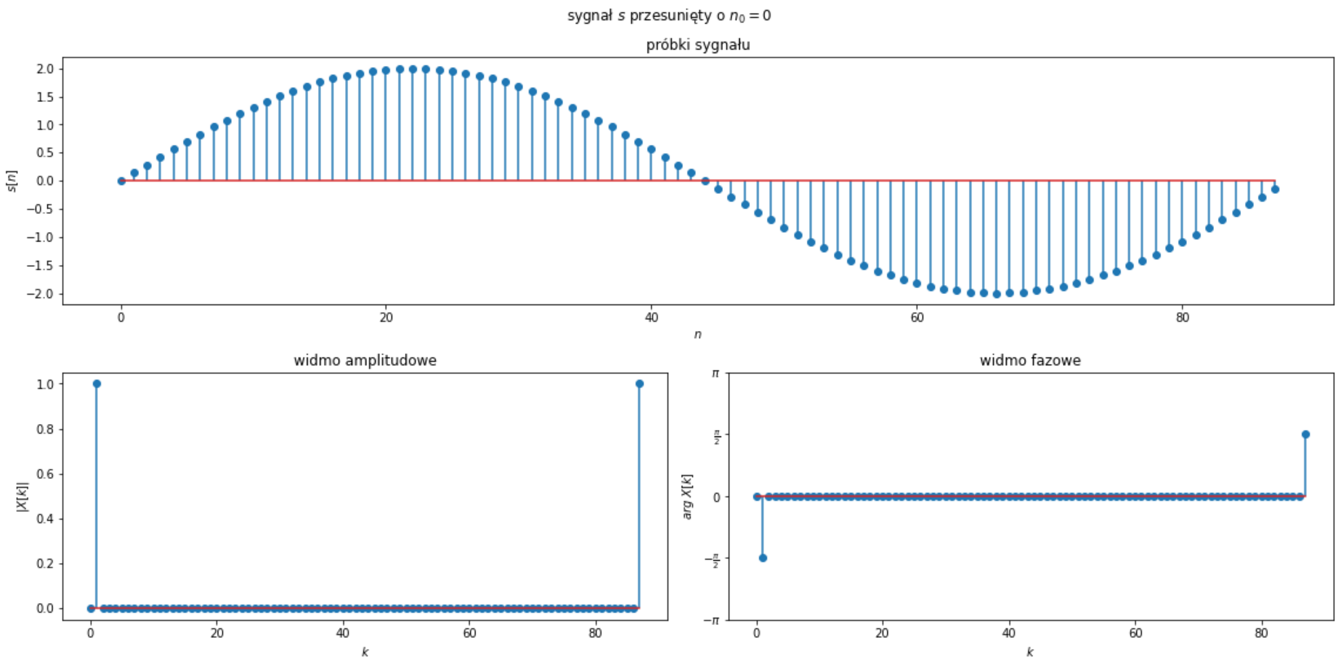


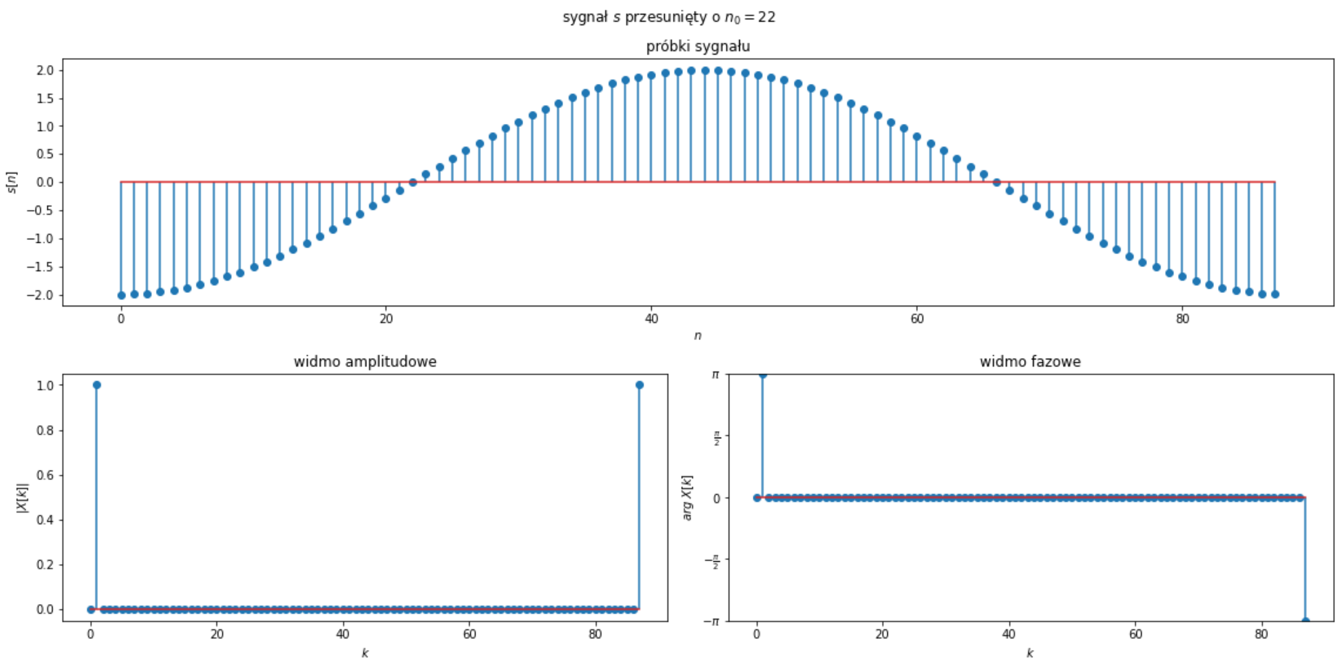
2. Zbadać wpływ przesunięcia w czasie na postać widma amplitudowego i widma fazowego dyskretnego sygnału harmonicznego $s[n] = A\sin\left(2\pi\frac{n}{N}\right)$ o amplitudzie $A = 2$ i okresie podstawowym $N = 88$. W tym celu dla każdej wartości $n_0 \in \left\{0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}\right\}$ wykreślić widmo amplitudowe i fazowe przesuniętego sygnału $s[n - n_0]$. Skomentować otrzymane wyniki.

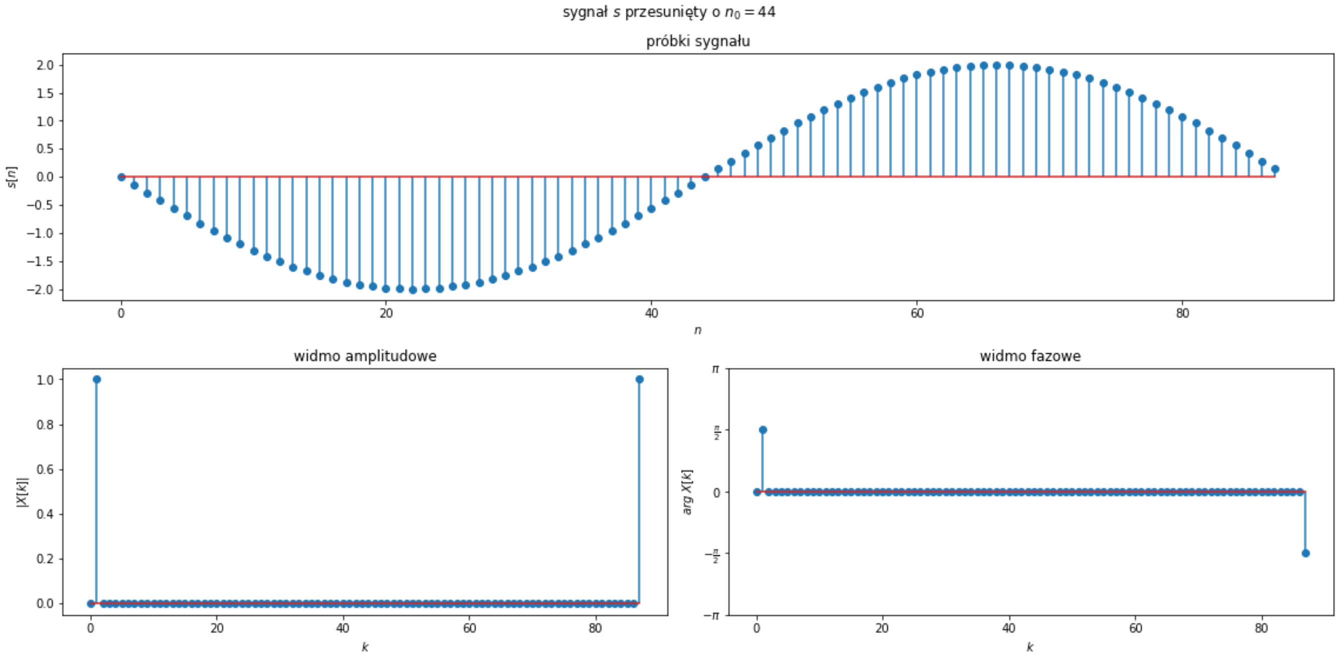
```python
from matplotlib import pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np

N = 88
A = 2
SHIFTS = np.arange(4) * N // 4
A_THRESHOLD = 1e-6
```

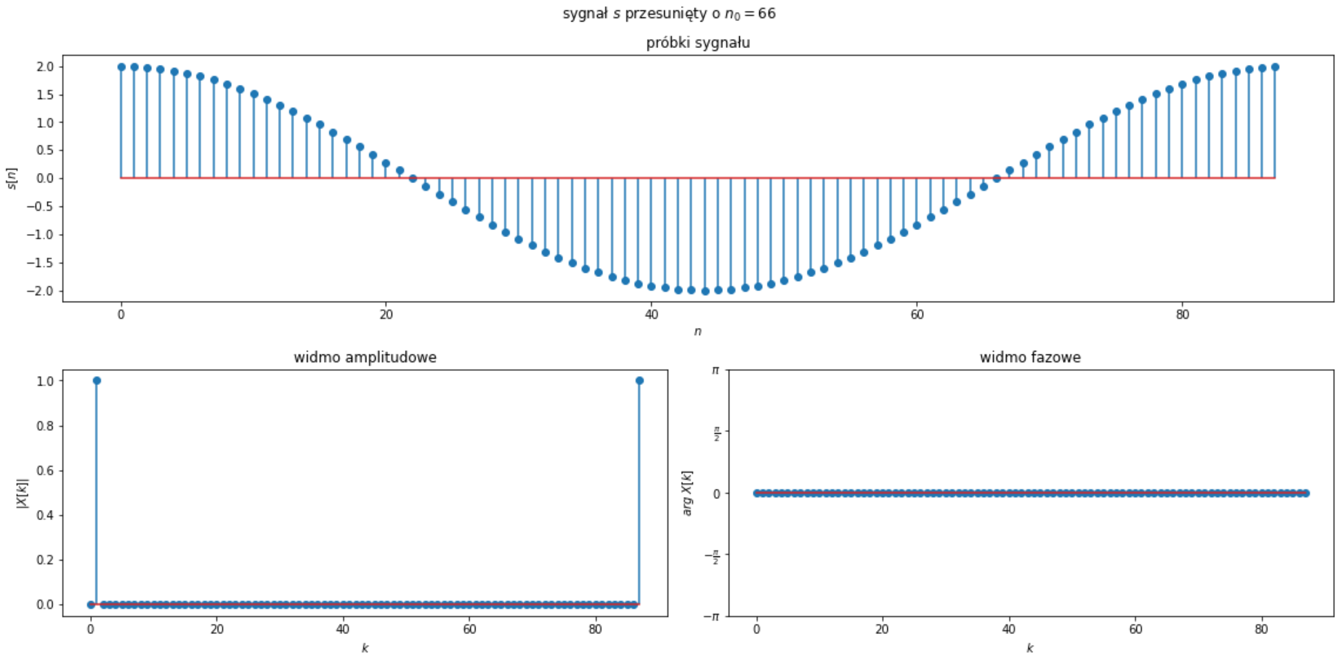
```python
for shift in SHIFTS:
    fig = plt.figure(figsize=(16, 8))
    fig.suptitle(f"sygnał $s$ przesunięty o $n_0={shift}$")
    gs = GridSpec(ncols=2, nrows=2, figure=fig)

    ax_s = fig.add_subplot(gs[0, :])
    ax_ampl = fig.add_subplot(gs[1, 0])
    ax_phase = fig.add_subplot(gs[1, 1])

    s = A * np.sin(2 * np.pi * (np.arange(N) - shift) / N)
    ax_s.stem(s)
    ax_s.set_title("próbki sygnału")
    ax_s.set_xlabel("$n$")
    ax_s.set_ylabel("$s[n]$")

    s_fft = np.fft.fft(s) / N
    s_amplitude_spc = np.abs(s_fft)

    # filter insignificant frequencies for clearer plots
    significance_mask = s_amplitude_spc <= A_THRESHOLD
    s_amplitude_spc[significance_mask] = 0
    s_fft[significance_mask] = 0

    s_phase_spc = np.angle(s_fft)

    ax_ampl.stem(s_amplitude_spc)
    ax_ampl.set_title("widmo amplitudowe")
    ax_phase.stem(s_phase_spc)
    ax_ampl.set_xlabel("$k$")
    ax_ampl.set_ylabel("$|X[k]|$")

    ax_phase.set_title("widmo fazowe")
    ax_phase.set_ylim((-np.pi, np.pi))
    ax_phase.set_xlabel("$k$")
    ax_phase.set_ylabel("$arg \\; X[k]$")
    yticks = [
        (-np.pi, '$-\\pi$'),
        (-np.pi / 2, '$-\\frac{\\pi}{2}$'),
        (0, '$0$'),
        (np.pi / 2, '$\\frac{\\pi}{2}$'),
        (np.pi, '$\\pi$')
    ]
    ax_phase.set_yticks([t for t, _ in yticks])
```

```
    ax_phase.set_yticklabels([l for _, l in yticks])

    fig.tight_layout()
    plt.show()
    plt.close()
```









Obserwacje:

- dla każdego przesunięcia widmo amplitudowe jest takie samo
- dla każdego kolejnego przesunięcia sygnału o $\frac{N}{4}$ zmienia się widmo fazowe:
  - $X[1]$ zmniejszane jest o $-\frac{\pi}{2}$
  - $X[N-1]$ zwiększane jest o $+\frac{\pi}{2}$

Wynika to właściwości przesunięcia w dzedzinie czasu DFT

$$x[n-n_0] \leftrightarrow X[k]e^{-j\frac{2\pi}{N}kn_0}$$

W naszym przypadku:

$$x[n-22] \leftrightarrow X[k]e^{-j\frac{\pi}{2}k}$$

$$x[n-44] \leftrightarrow X[k]e^{-j\pi k}$$

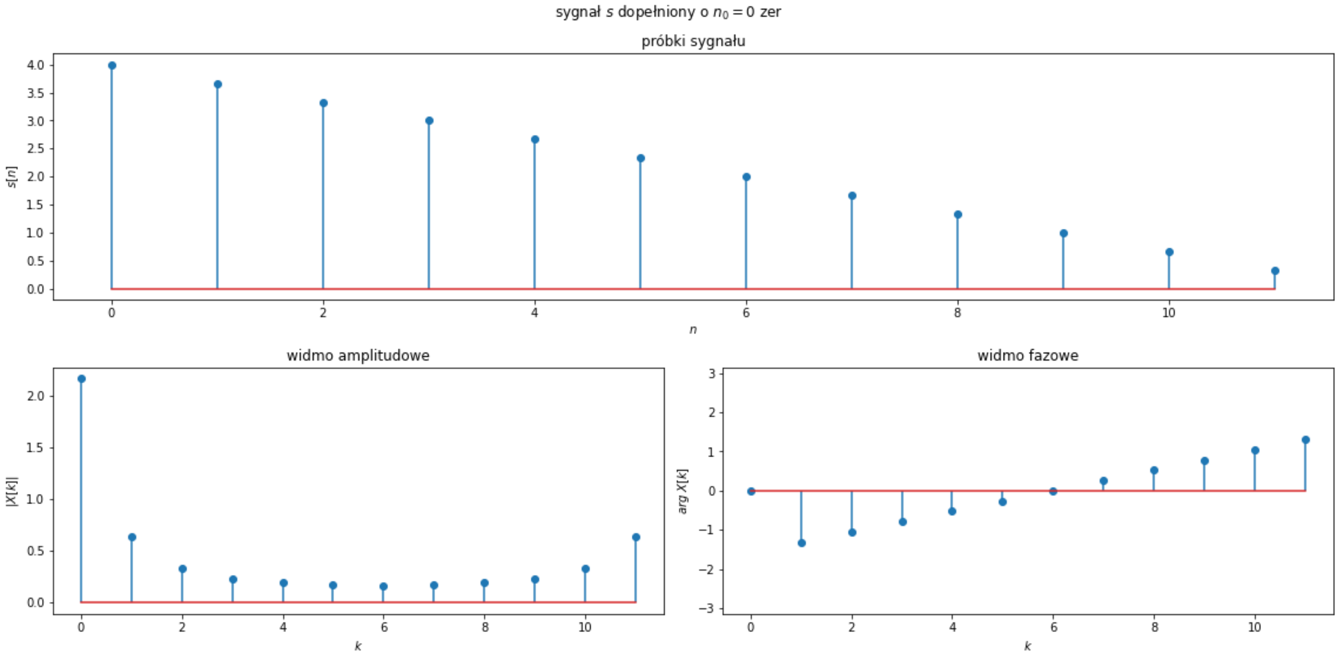
$$x[n-66] \leftrightarrow X[k]e^{-j\frac{3\pi}{2}k}$$

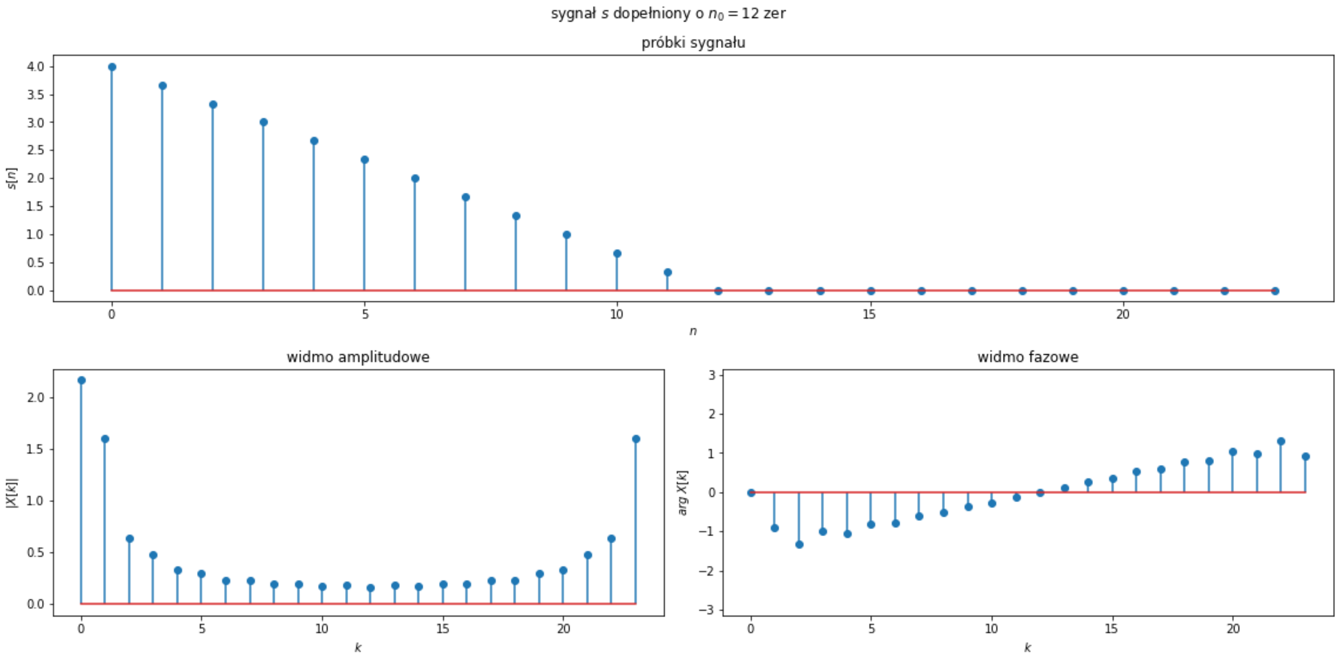
Po podstawieniu $k = 1$ lub $k = -1$ wyjdą zaobserwowane zmiany w widmie.

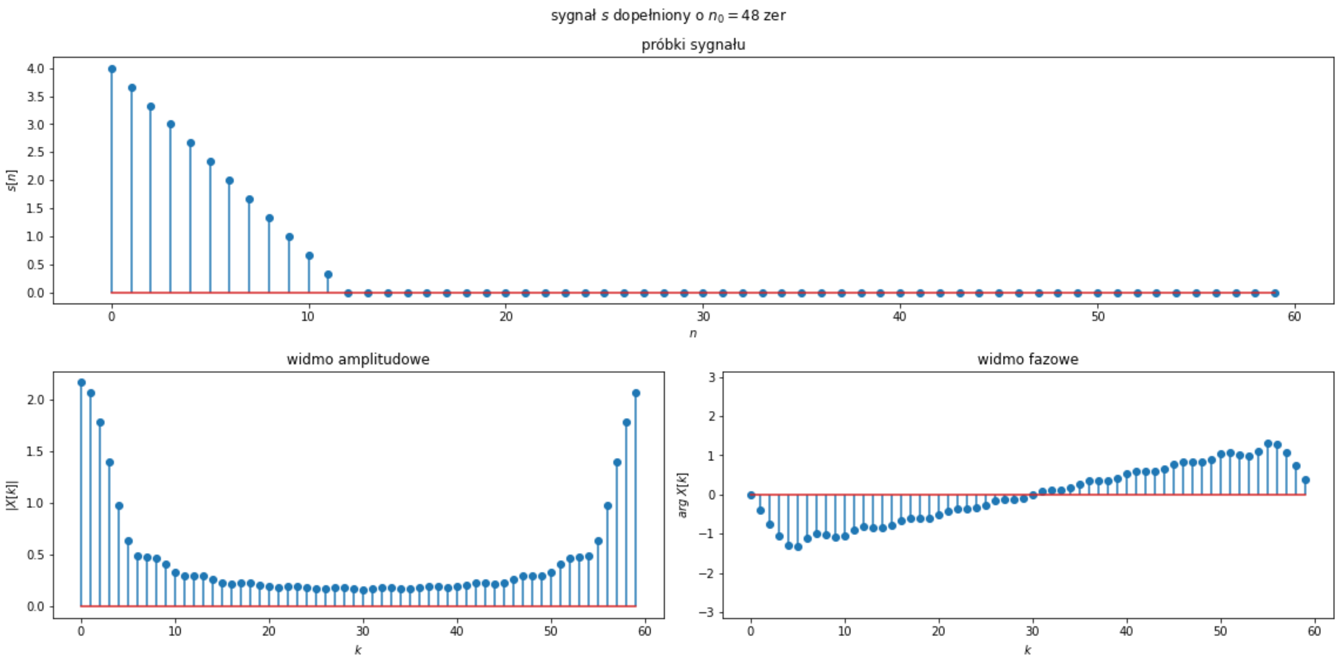
3. Zbadać wpływ dopełnienia zerami na postać widma amplitudowego i widma fazowego dyskretnego sygnału $s[n] = A\left(1 - \frac{n \bmod N}{N}\right)$ o amplitudzie $A = 4$ i okresie podstawowym $N = 12$. W tym celu dla każdej wartości $N_0 \in \{0, 1N, 4N, 9N\}$ wykreślić widmo amplitudowe i fazowe sygnału $s[n]$ dopełnionego N_0 zerami. Skomentować otrzymane wyniki.

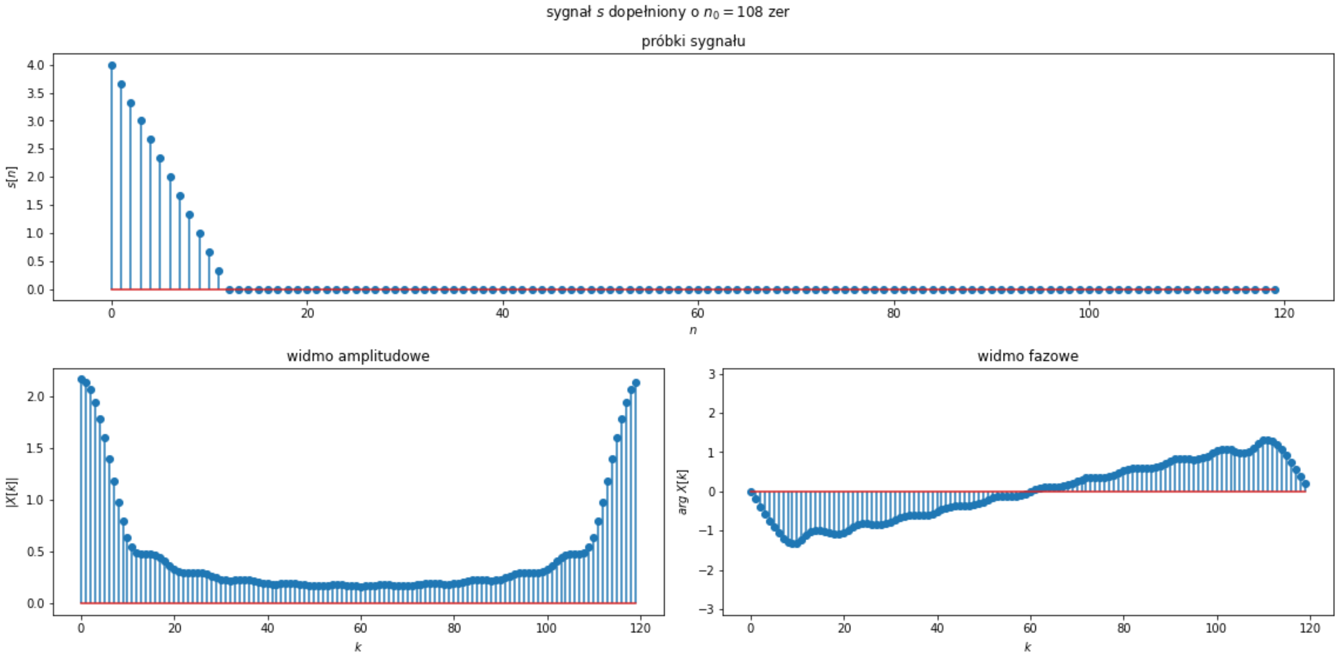
```
from matplotlib import pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np

N = 12
A = 4
N0 = np.square(np.arange(4)) * N

s0 = A * (1 - (np.arange(N) % N) / N)
```

```
for n0 in N0:
    fig = plt.figure(figsize=(16, 8))
    fig.suptitle(f"sygnał $s$ dopełniony o $n_0={n0}$ zer")
    gs = GridSpec(ncols=2, nrows=2, figure=fig)

    ax_s = fig.add_subplot(gs[0, :])
    ax_ampl = fig.add_subplot(gs[1, 0])
    ax_phase = fig.add_subplot(gs[1, 1])

    s = np.concatenate((s0, np.zeros(n0)))

    ax_s.stem(s)
    ax_s.set_title("próbki sygnału")
    ax_s.set_xlabel("$n$")
    ax_s.set_ylabel("$s[n]$")

    s_fft = np.fft.fft(s) / N
    s_amplitude_spectrum = np.abs(s_fft)
    s_phase_spectrum = np.angle(s_fft)

    ax_ampl.stem(s_amplitude_spectrum)
    ax_ampl.set_title("widmo amplitudowe")
    ax_ampl.set_xlabel("$k$")
    ax_ampl.set_ylabel("$|X[k]|$")

    ax_phase.stem(s_phase_spectrum)
    ax_phase.set_title("widmo fazowe")
    ax_phase.set_xlabel("$k$")
    ax_phase.set_ylabel("$arg \\; X[k]$")
    ax_phase.set_ylim((-np.pi, np.pi))

    fig.tight_layout()
    plt.show()
    plt.close()
```


sygnał s dopełniony o $n_0 = 0$ zer
próbki sygnału
$s[n]$
n
widmo amplitudowe
$|X[k]|$
k
widmo fazowe
$\arg X[k]$
k


sygnał s dopełniony o $n_0 = 12$ zer
próbki sygnału
$s[n]$
n
widmo amplitudowe
$|X[k]|$
k
widmo fazowe
$\arg X[k]$
k


sygnał s dopełniony o $n_0 = 48$ zer
próbki sygnału
$s[n]$
n
widmo amplitudowe
$|X[k]|$
k
widmo fazowe
$\arg X[k]$
k



Obserwacje:

- dopełnienie sygnału zerami nie zmienia ogólnego kształtu obu widm, a jedynie zwiększa rozdzielczość w dziedzinie częstotliwości, dodając wartości pośrednie

Zachowanie to wynika ze wzoru na odległość między kolejnymi wyrazami DFT:

$$\Delta f_N = \frac{f_s}{N}$$

Jak widać można ją zmniejszyć poprzez zwiększenie ilości próbek.

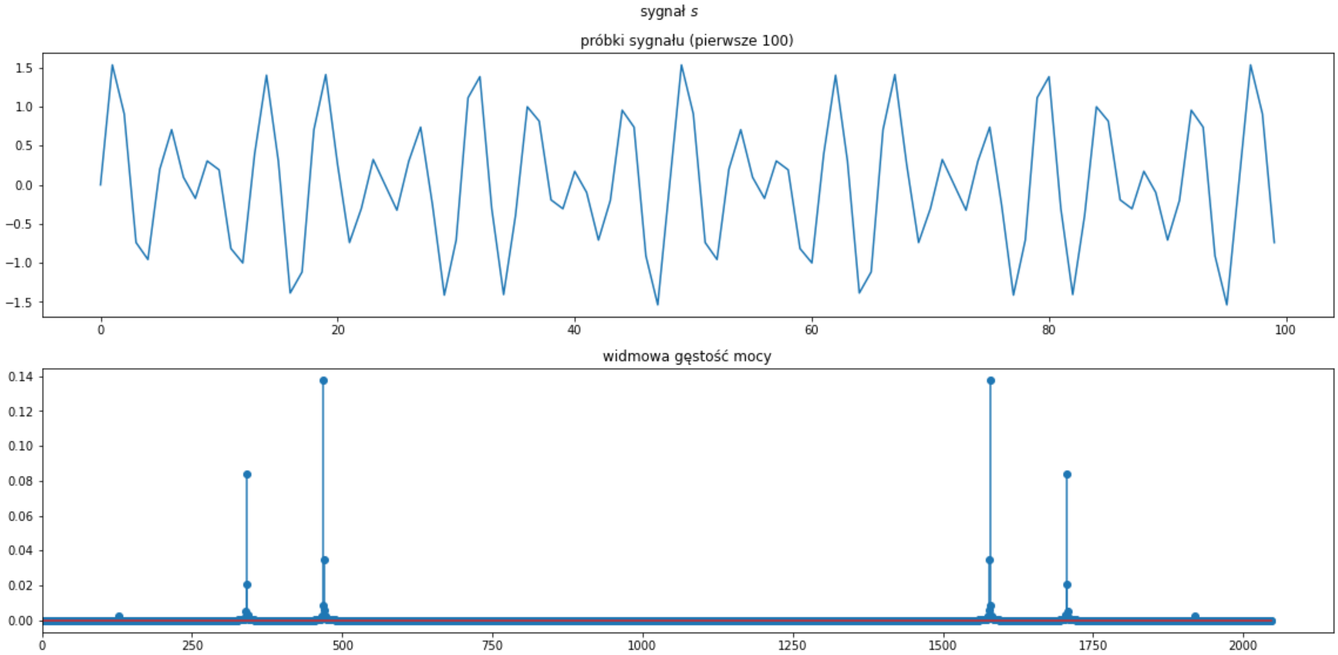
4. Dany jest sygnał rzeczywisty $s(t) = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_3 \sin(2\pi f_3 t)$, gdzie $A_1 = 0.1$, $f_1 = 3000$ Hz, $A_2 = 0.7$, $f_2 = 8000$ Hz, $A_3 = 0.9$, $f_3 = 11000$ Hz. Przy założeniu, że liczba próbek sygnału wynosi $N_1 = 2048$, przedstawić wykres widmowej gęstości mocy sygnału $s(t)$. Czy dla podanej liczby próbek mamy do czynienia ze zjawiskiem przecieku widma? Czy sytuacja uległaby zmianie dla liczby próbek $N_2 = \frac{3}{2} N_1$? Odpowiedź uzasadnić.

```python
# zgodnie z zaleceniami przyjmuję częstotliwość próbkowania 48 kHZ

from matplotlib import pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np
from typing import Iterable

N = 2048
SAMPLE_FREQUENCY = 48_000

COMPONENT_AMPLITUDES = [0.1, 0.7, 0.9]
COMPONENT_FREQUENCIES = [3e3, 8e3, 11e3]

A_THRESHOLD = 1e-6

def sin_samples(amplitude: float, frequency: float,
                n_samples: int, sample_frequency: float):
    time_samples = 1 / sample_frequency * np.arange(n_samples)
    return amplitude * np.sin(2 * np.pi * frequency * time_samples)


def sin_sum_samples(amplitudes: Iterable[float], frequencies: Iterable[float],
                    n_samples: int, sample_frequency: float):
    return sum(
        sin_samples(amplitude, frequency, n_samples, sample_frequency)
        for amplitude, frequency in zip(amplitudes, frequencies)
    )
```

```python
def plot(signal: np.ndarray):
    fig, (ax_1, ax_2) = plt.subplots(2, 1)
    fig.set_size_inches((16, 8))
    fig.suptitle(f"sygnał $s$")

    ax_1.plot(signal[:100])
    ax_1.set_title("próbki sygnału (pierwsze 100)")

    s_fft = np.fft.fft(signal) / N

    # filter values close to 0
    s_amplitude_spc = np.abs(s_fft)
    siginificance_mask = s_amplitude_spc > A_THRESHOLD
    s_amplitude_spc_filtered = s_amplitude_spc[siginificance_mask]
    x = np.arange(len(s_amplitude_spc))[siginificance_mask]

    s_power_spc = s_amplitude_spc_filtered ** 2

    ax_2.stem(x, s_power_spc)
    ax_2.set_title("widmowa gęstość mocy")
    ax_2.set_xlim((0, ax_2.get_xlim()[1]))

    fig.tight_layout()
```

```
    plt.show()
    plt.close()
```

```
s = sin_sum_samples(COMPONENT_AMPLITUDES, COMPONENT_FREQUENCIES,
                    N, SAMPLE_FREQUENCY)

plot(s)
```



Obserwacje:

- widać efekt zjawiska przecieku widma
- pomimo zastosowania filtracji składowych widma o małej aplitudzie, prawie każda składowa jest widoczna na wykresie

Efekt ten wynika z tego, że nie wszystkie składowe częstotliwości sygnału są podzielne przez rozdzielczość dyskretyzacji częstotliwości transformaty DFT.

$$\Delta f = \frac{f_s}{N} = \frac{48000}{2048} = 23.4375$$

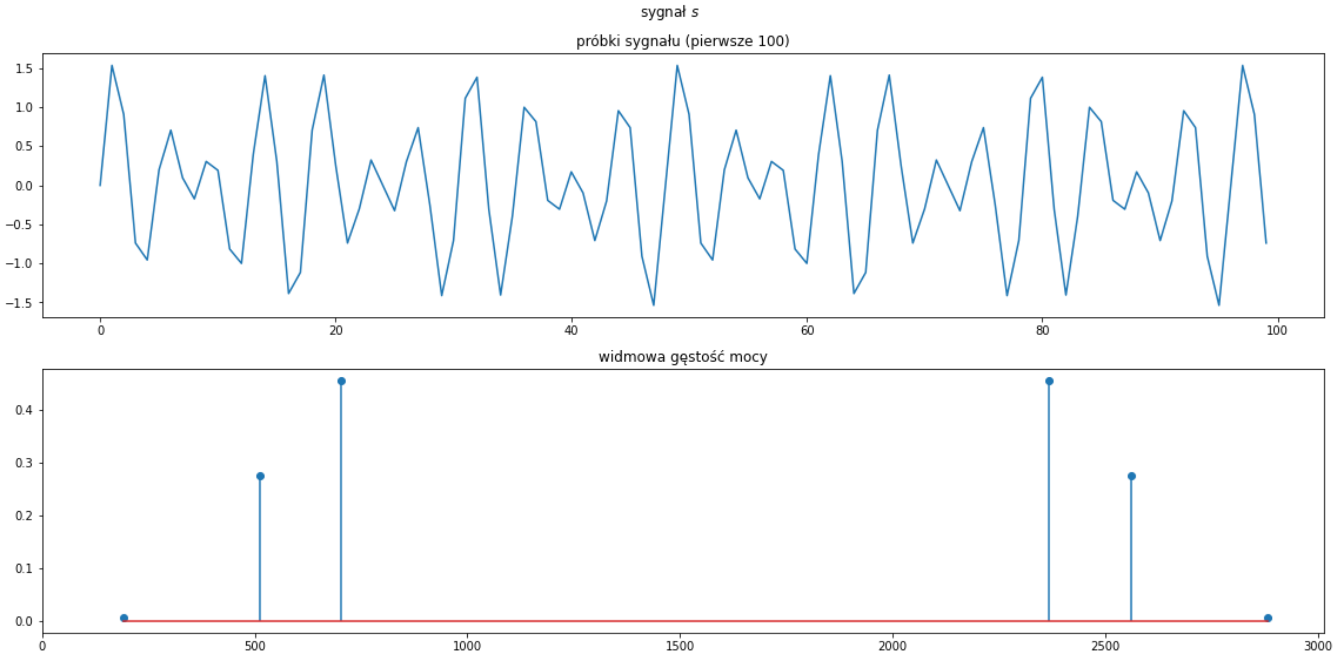
$$\frac{f_1}{\Delta f} = 128$$

$$\frac{f_2}{\Delta f} = 341\frac{1}{3}$$

$$\frac{f_3}{\Delta f} = 369\frac{1}{3}$$

Jak widać częstotliwości f_2 i f_3 nie dzielą się przez Δf. Oznacza to że nie można ich zareprezentować w dokładny sposób stosując taką dyskretyzację częstotliwości.

```
s = sin_sum_samples(COMPONENT_AMPLITUDES, COMPONENT_FREQUENCIES,
                3 / 2 * N, SAMPLE_FREQUENCY)

plot(s)
```



Obserwacje:

- tym razem widać tylko 3 prążki - dokładnie tyle ile składowych częstotliwości w sygnale
- w związku z tym nie ma efektu przecieku widma

Różnica w stosunku do poprzedniego przykładu jest w tym, że tym razem częstotliwości składowe sygnału dzielą się przez rozdzielczość dyskretyzacji częstotliwości transformaty DFT.

$$\Delta f = \frac{f_s}{\frac{3}{2}N} = \frac{48000}{3072} = 15.625$$

$$\frac{f_1}{\Delta f} = 192$$

$$\frac{f_2}{\Delta f} = 512$$

$$\frac{f_3}{\Delta f} = 704$$

Częstotliwości te dokładnie odpowiadają wartościom k, dla których prążki na wykresie widmowej gęstości mocy są widoczne.