Laboratorium
Metod Numerycznych
Aproksymacja

Instrukcja:
Na zajęciach należy wykonać poniższe zadania, dokonać testu na platformie github, a następnie sporządzić sprawozdanie zawierające odpowiedzi z komentarzami.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import splrep, splev, BSpline

##
import scipy
from typing import Union, List, Tuple

def linear_least_squares(x:np.ndarray,y:np.ndarray)-> np.ndarray:
    if len(x) != len(y) or len(x) < 2:
            return None

        #Tworzy macierz A, gdzie pierwszy wiersz to wartości x,
        #a drugi wiersz to jedynki. vstack łączy te dwie tablice w pionie, d
    A = np.vstack([x, np.ones(len(x))]).T

        #parametr(rcond=None) - wszystkie wartości singularne są brane pod u
        #[0] - wybiera pierwszy element z tej krotki, który zawiera współczy
        #interesują nas tylko współczynniki aproksymacji, a reszta informacj
    m, c = np.linalg.lstsq(A, y, rcond=None)[0]

    return np.array([m, c])



def chebyshev_nodes(n):
    if n <= 0:
        return None
    nodes = np.cos(np.pi*np.arange(n+1)/n)
    return nodes
```

Aproksymacja funkcji jest fundamentalnym zagadnieniem w analizie numerycznej, które polega na przybliżaniu funkcji bardziej skomplikowanej przez funkcję prostszą. Jest to szczególnie użyteczne w przypadkach, kiedy mamy do czynienia z funkcjami trudnymi do analizy lub gdy potrzebujemy szybkich obliczeń numerycznych.

[Metoda Najmniejszych Kwadratów (https://en.wikipedia.org/wiki/Least_squares)](https://en.wikipedia.org/wiki/Least_squares)

Metoda najmniejszych kwadratów jest jedną z najczęściej stosowanych technik aproksymacji. Polega ona na minimalizacji sumy kwadratów różnic między wartościami obserwowanymi a wartościami przewidywanymi przez model aproksymacyjny.

Matematycznie, jeśli mamy zbiór danych (x_i, y_i), szukamy funkcji $f(x)$, która minimalizuje: $S = \sum_{i=1}^{n} (y_i - f(x_i))^2$

[Funkcje Sklejane (Splines) (https://en.wikipedia.org/wiki/Spline_(mathematics))](https://en.wikipedia.org/wiki/Spline_(mathematics))

Funkcje sklejane, znane również jako splines, to kolejna popularna technika aproksymacji. Splines to ciągłe i gładkie funkcje, które są definiowane kawałkami przez wielomiany. W przeciwieństwie do globalnych metod aproksymacji, jak wielomiany wysokiego stopnia, splines unikają problemu oscylacji, oferując lepszą kontrolę nad kształtem krzywej aproksymacyjnej.

[B-Splines (https://en.wikipedia.org/wiki/B-spline)](https://en.wikipedia.org/wiki/B-spline)

B-Splines, czyli bazowe splines, stanowią rozszerzenie idei funkcji sklejanych. Są to specjalne rodzaje splines, które zapewniają dodatkową elastyczność i kontrolę nad

Zadanie 1.

Napisz skrypt w języku Python, który dokona aproksymacji liniowej funkcji sinus:

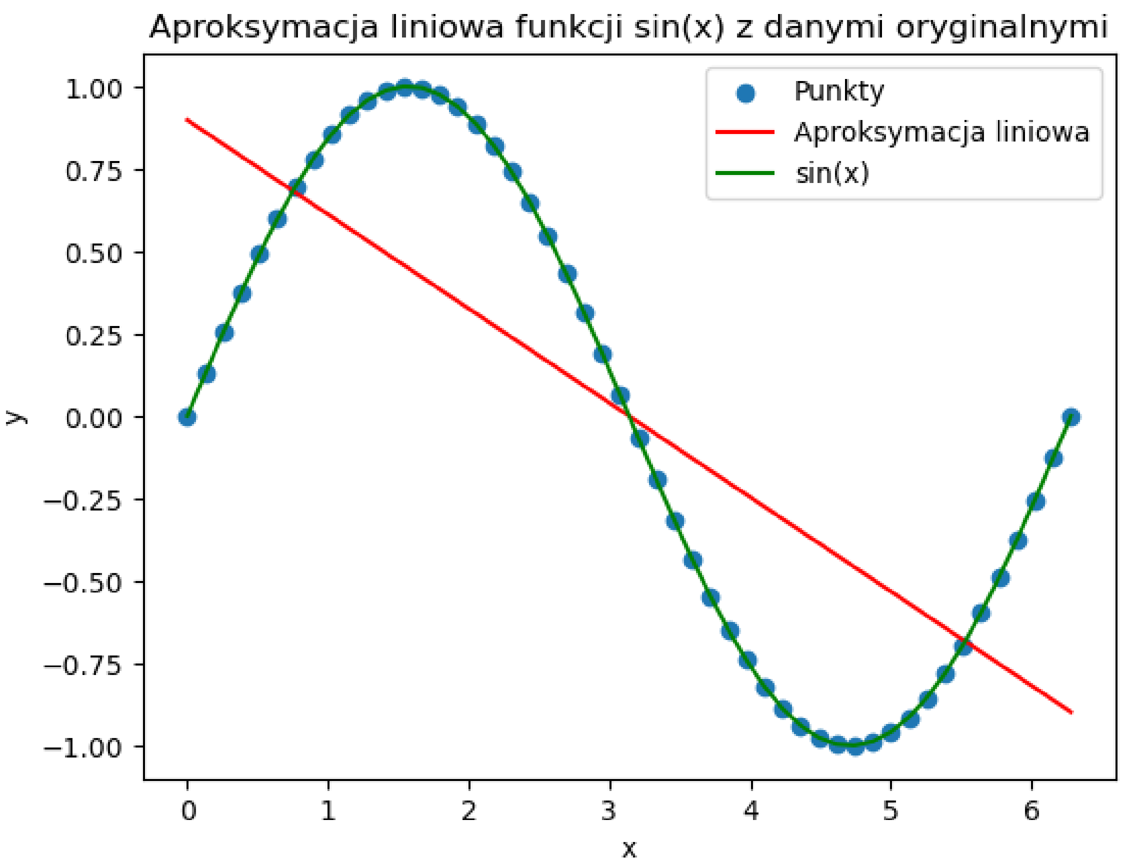
$$y = sin(x)$$

na przedziale od 0 do 2π używając metody najmniejszych kwadratów. Wygeneruj zbiór danych składający się z minimum 50 punktów z równymi odstępami w tym przedziale. Twoje zadanie polega na znalezieniu najlepiej dopasowanej linii prostej do tych danych.

Rozwiązanie powinno zawierać nasstępujące punkty:

1. Wygeneruj zbiór danych (x, y), gdzie y to wartości sin(x).
2. Napisz funkcję wykonującą aproksymację liniową metodą najmniejszych kwadratów (funkcja *linear_least_squares* z main.py).
3. Wyświetl oryginalne dane i dopasowaną linię na wykresie.

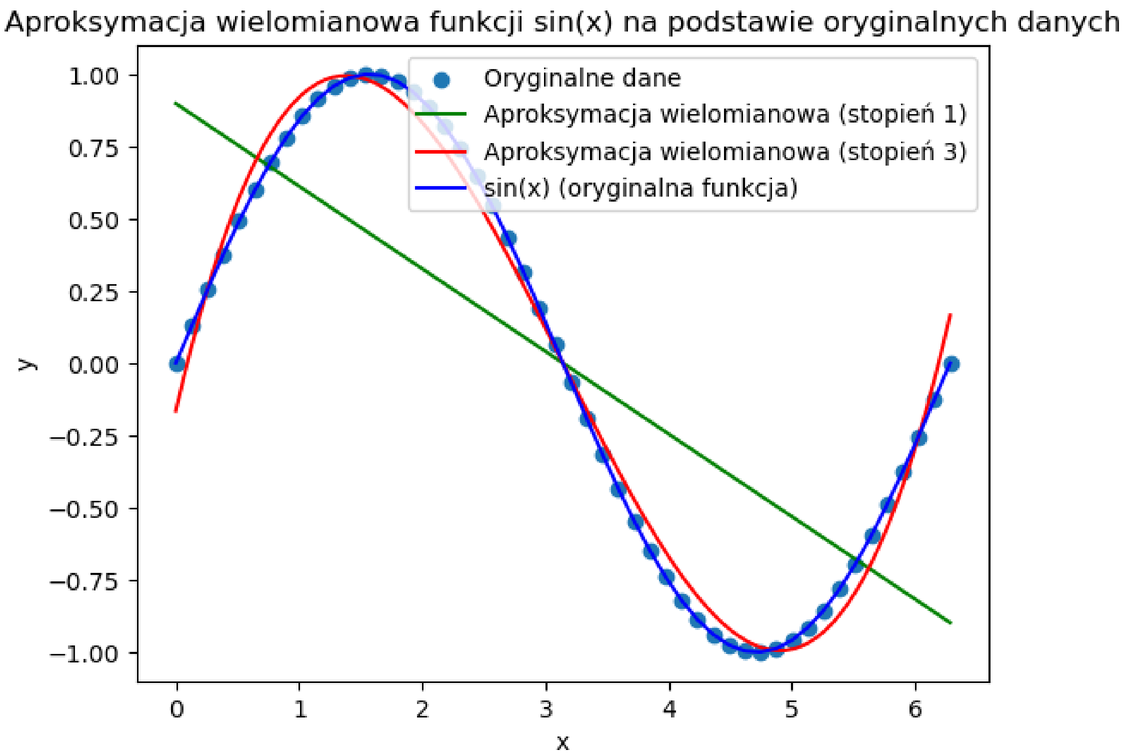
In [2]:
```
# Generowanie zbioru danych
liczba_punktow = 50
wartosci_x = np.linspace(0, 2 * np.pi, liczba_punktow)
wartosci_y = np.sin(wartosci_x)

# Aproksymacja liniowa
wspolczynniki = linear_least_squares(wartosci_x, wartosci_y)

# Wygenerowanie linii aproksymacyjnej
aproks_x = np.linspace(0, 2 * np.pi, 100)
aproks_y = wspolczynniki[0] * aproks_x + wspolczynniki[1]

# Wykres
plt.scatter(wartosci_x, wartosci_y, label='Punkty')
plt.plot(aproks_x, aproks_y, color='red', label='Aproksymacja liniowa')
plt.plot(wartosci_x, wartosci_y, color='green', label='sin(x)')

plt.legend()
plt.title('Aproksymacja liniowa funkcji sin(x) z danymi oryginalnymi')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Zadanie 2.

Napisz skrypt w języku Python, który dokona aproksymacji wielmianowej stopnia pierwszego oraz trzeciego funkcji z poprzedniego zadania równiez na przedziale od 0 do 2π, używając metody najmniejszych kwadratów. Wygeneruj zbiór danych składający się z minimum 50 punktów z równymi odstępami w tym przedziale. Do wykonania obu aproksymacji uzyj funkcji *np.polyfit*. oraz *np.poly1d*. Zobrazuj obie aproksymacji w zestawieniu z oryginalną funkcją na jednym wykresie.

In [3]:
```
# Aproksymacja wielomianowa stopnia pierwszego
wspolczynniki_poly1 = np.polyfit(wartosci_x, wartosci_y, deg=1)
aproks_poly1 = np.poly1d(wspolczynniki_poly1)

# Aproksymacja wielomianowa stopnia trzeciego
wspolczynniki_poly3 = np.polyfit(wartosci_x, wartosci_y, deg=3)
aproks_poly3 = np.poly1d(wspolczynniki_poly3)

# Wygenerowanie danych aproksymacyjnych
x_aproks = np.linspace(0, 2 * np.pi, 100)
y_aproks_poly1 = aproks_poly1(x_aproks)
y_aproks_poly3 = aproks_poly3(x_aproks)

# Wykres
plt.scatter(wartosci_x, wartosci_y, label='Oryginalne dane')
plt.plot(x_aproks, y_aproks_poly1, label='Aproksymacja wielomianowa (stopien
plt.plot(x_aproks, y_aproks_poly3, label='Aproksymacja wielomianowa (stopien
plt.plot(wartosci_x, np.sin(wartosci_x), label='sin(x) (oryginalna funkcja)'

plt.legend()
plt.title('Aproksymacja wielomianowa funkcji sin(x) na podstawie oryginalnyc
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Zadanie 3.

Napisz skrypt w języku Python, który dokona aproksymacji funckjami sklejanymi stopnia trzeciego funkcji na przedziale $[-1; 1]$:

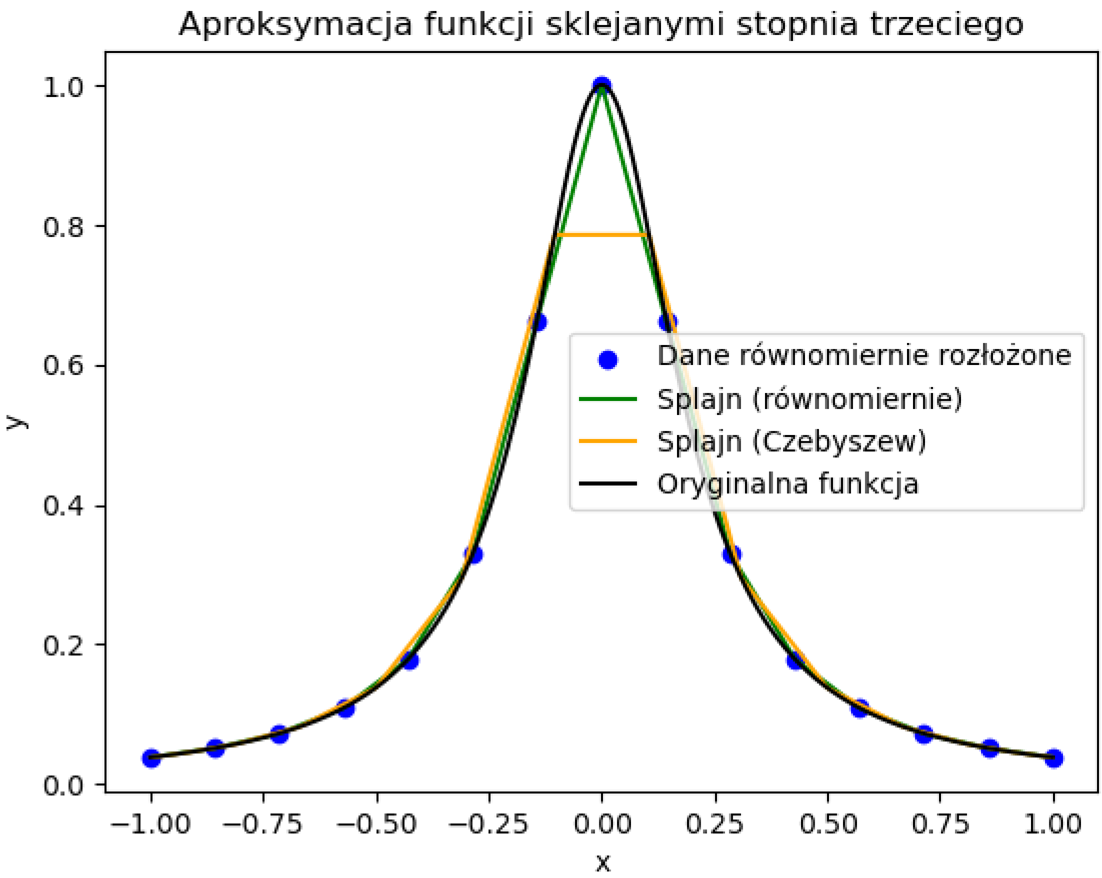
$$f(x) = \frac{1}{1 + 25x^2}$$

Wygeneruj dwa warianty danych dla funkcji: Pierwszy zestaw z 15 równomiernie rozłożonymi punktami. Drugi zestaw z 15 punktami węzłów Czebyszewa (za pomoca funkcji *cheb_roots* z pliku main.py).

Następnie dokonaj aproksymacji tych danych przy użyciu funkcji sklejanych (splajnów) trzeciego stopnia dla obu zestawów punktów. Wyświetl na wykresie zarówno wygenerowane dane, krzywe aproksymacji dla obu zestawów punktów, jak i oryginalny przebieg funkcji. Wykorzystaj funkcje scipy.interpolate.splrep i scipy.interpolate.splev z biblioteki SciPy do stworzenia i ewaluacji splajnów. Do wizualizacji wyników użyj matplotlib.

Porównaj jakość dopasowania dla obu przypadków i zwróć uwagę na efekt Rungego. *Czy wykorzystanie węzłów Czebysheva jest dobrym rozwiązaniem eliminacji problem efektu Rungego w tym przypadku? Uzasadnij odpowiedź.*

In [5]:
```
#Generowanie danych
liczba_punktow = 15

#Generowanie punktów równomiernie rozłożonych
x_rownomiernie = np.linspace(-1, 1, liczba_punktow)
y_rownomiernie = 1 / (1 + 25 * x_rownomiernie**2)

#Generowanie punktów na podstawie węzłów Czebyszewa
wezly_chebyszewa = chebyshev_nodes(liczba_punktow)
x_chebyszew = wezly_chebyszewa
y_chebyszew = 1 / (1 + 25 * x_chebyszew**2)

#Wykres danych rownomiernie rozlozonych
plt.scatter(x_rownomiernie, y_rownomiernie, label='Dane równomiernie rozłożo

##Aproksymacja funkcji sklejanymi dla danych równomiernie rozłożonych
#Sortowanie indeksów równomiernie rozłożonych danych
indeksy_posortowane_rownomiernie = np.argsort(x_rownomiernie)

#Posortowane wartości x, y równomiernie rozłożonych danych
x_rownomiernie_posortowane = x_rownomiernie[indeksy_posortowane_rownomiernie
y_rownomiernie_posortowane = y_rownomiernie[indeksy_posortowane_rownomiernie

#Przeprowadzenie aproksymacji sklejanej dla posortowanych danych równomierni
tck_rownomiernie = splrep(x_rownomiernie_posortowane, y_rownomiernie_posorto

#Ewaluacja funkcji sklejanej dla posortowanych danych równomiernie rozłożony
y_spline_rownomiernie = splev(x_rownomiernie_posortowane, tck_rownomiernie)

#Rysowanie krzywej aproksymacyjnej na wykresie dla danych równomiernie rozło
plt.plot(x_rownomiernie_posortowane, y_spline_rownomiernie, label='Splajn (r

##Aproksymacja funkcji sklejanymi dla danych na podstawie węzłów Czebyszewa
#Sortowanie indeksów węzłów Czebyszewa
indeksy_posortowane_chebyszew = np.argsort(x_chebyszew)

#Posortowane wartości x, y na podstawie węzłów Czebyszewa
x_chebyszew_posortowane = x_chebyszew[indeksy_posortowane_chebyszew]
y_chebyszew_posortowane = y_chebyszew[indeksy_posortowane_chebyszew]

#Przeprowadzenie aproksymacji sklejanej dla posortowanych danych na podstawi
tck_chebyszew = splrep(x_chebyszew_posortowane, y_chebyszew_posortowane, k=3

#Ewaluacja funkcji sklejanej dla posortowanych danych na podstawie węzłów Cz
y_spline_chebyszew = splev(x_chebyszew_posortowane, tck_chebyszew)

#Rysowanie krzywej aproksymacyjnej na wykresie dla wezłów Chebyszewa
plt.plot(x_chebyszew_posortowane, y_spline_chebyszew, label='Splajn (Czebysz

#Oryginalny przebieg funkcji
x_original = np.linspace(-1, 1, 1000)
y_original = 1 / (1 + 25 * x_original**2)
plt.plot(x_original, y_original, label='Oryginalna funkcja', color='black')

plt.legend()
plt.title('Aproksymacja funkcji sklejanymi stopnia trzeciego')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Mozemy zwrocic uwage, że metoda Czebyszewa nie znajduje punktu 0.

Zadanie 4.

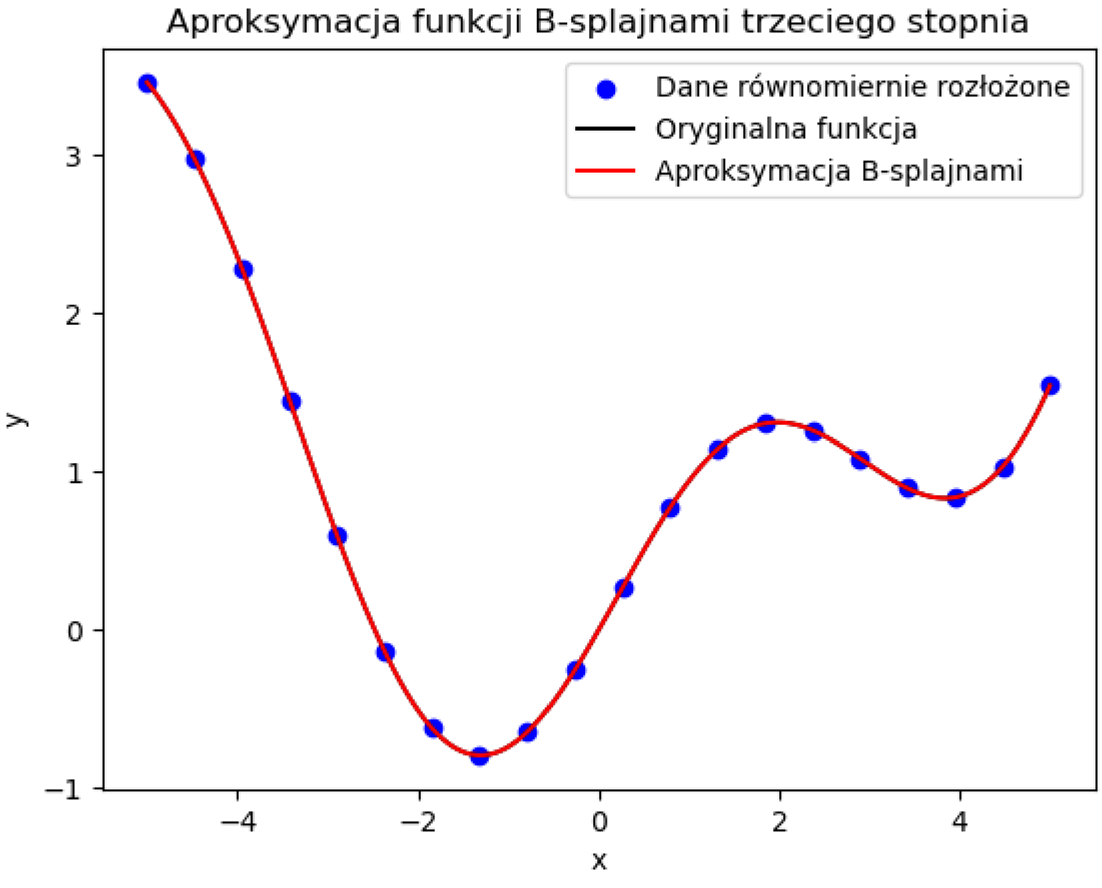
Napisz skrypt w języku Python, który dokona aproksymacji funckjami Bspline na przedziale $[-5; 5]$:

$$y = \sin(x) + 0.1x^2$$

używając 20 równomiernie rozłożonych punktów jako danych.Użyj B-splajnów do aproksymacji tych danych. Zastosuj B-splajny trzeciego stopnia. Porównaj krzywą aproksymacji z oryginalnymi danymi na wykresie. Podaj krótką analizę jakości dopasowania B-splajnów do danych.

Do utworzenia tej aproksymacji przydatne będą funkcje: *scipy.interpolate.splrep* oraz *scipy.interpolate.BSpline*

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import splrep, BSpline

def f(x):
    return np.sin(x) + 0.1 * x**2

# Generowanie danych
num_punkty = 20
dane_x= np.linspace(-5, 5, num_punkty)
dane_y = f(dane_x)

# Aproksymacja funkcji B-splajnami trzeciego stopnia

tck = splrep(dane_x, dane_y, k=3, s=0) #k okresla stopien, s okresla ze nie
bspline = BSpline(tck[0], tck[1], tck[2], extrapolate=False) #extrapolate=Fa

# Oryginalny przebieg funkcji
x_originalne = np.linspace(-5, 5, 1000)
y_originalne = f(x_originalne)

# Wykres
plt.scatter(dane_x, dane_y, label='Dane równomiernie rozłożone', color='blue
plt.plot(x_originalne, y_originalne, label='Oryginalna funkcja', color='blac
plt.plot(x_originalne, bspline(x_originalne), label='Aproksymacja B-splajnam

plt.legend()
plt.title('Aproksymacja funkcji B-splajnami trzeciego stopnia')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Materiały uzupełniające:

- Scipy Lecture Notes (http://www.scipy-lectures.org/index.html)
- NumPy for Matlab users (https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html#numpy-for-matlab-users)
- Python Tutorial - W3Schools (https://www.w3schools.com/python/default.asp)
- NumPy (https://www.numpy.org)
- Matplotlib (https://matplotlib.org/)
- Anaconda (https://www.anaconda.com/)
- Learn Python for Data Science (https://www.datacamp.com/learn-python-with-anaconda?utm_source=Anaconda_download&utm_campaign=datacamp_training&utm_medium=bar
- Learn Python (https://www.learnpython.org/)
- Wujek Google (https://google.pl) i Ciocia Wikipedia (https://pl.wikipedia.org/wiki/Wikipedia:Strona_g%C5%82%C3%B3wna)