In [49]:

```
import math
from sympy import *
init_printing(use_unicode=True)
x =symbols('x')
```

In [50]:

```
from icecream import ic
import random
import pandas as pd
import colorsys
```

Uniwersalne funkcje do wyliczania Taylora oraz rysowania wykresow

In [51]:

```
def calculateTaylorComponents(func, Nn, x0):
    skladowe = []
    for n in range(0, Nn+1):
        pochodna = diff(func, x, n)
        pochodnaX0 = pochodna.subs(x, x0)
        skladowa = pochodnaX0 / math.factorial(n) * ((x - x0)**n)
        # ic(pochodna, pochodnaX0, skladowa)
        skladowe.append(skladowa)
    return skladowe

def getTaylorSeriesFromComponents(skladowe):
    return sum(skladowe)

def getTaylorSeries(func, Nn, x0):
    return getTaylorSeriesFromComponents(calculateTaylorComponents(func, Nn , x0))

def getColorsHex(numColors):
    colorsHex = []
    for i in range(numColors):
        hue = i / numColors
        saturation = 0.8
        value = 0.8
        rgbColor = colorsys.hsv_to_rgb(hue, saturation, value)
        rHex, gHex, bHex = int(rgbColor[0] * 255), int(rgbColor[1] * 255), int(rgbColor[2]
* 255)
        colorHex = "#{:02x}{:02x}{:02x}".format(rHex, gHex, bHex)
        colorsHex.append(colorHex)
    return colorsHex

def getPlots(funcsWithPlotParameters, title, rangeX=(-5, 5), yLimit=None, showFinalPlot=Tr
ue):
    plots = plot(show=False, xlabel='x', ylabel='y', legend=True, title=title, ylim=yLimit
)
    for fp in funcsWithPlotParameters:
        func = fp.get('func', None)
        if func:
            color = fp.get('color', 'blue')
            label = fp.get('label', '')
            p = plot(func, (x, rangeX[0], rangeX[1]), show=False, line_color=color, label=
label, ylim=yLimit)
            plots.append(p[0])
    if showFinalPlot:
        plots.show()
    return plots
```

Zadanie T1

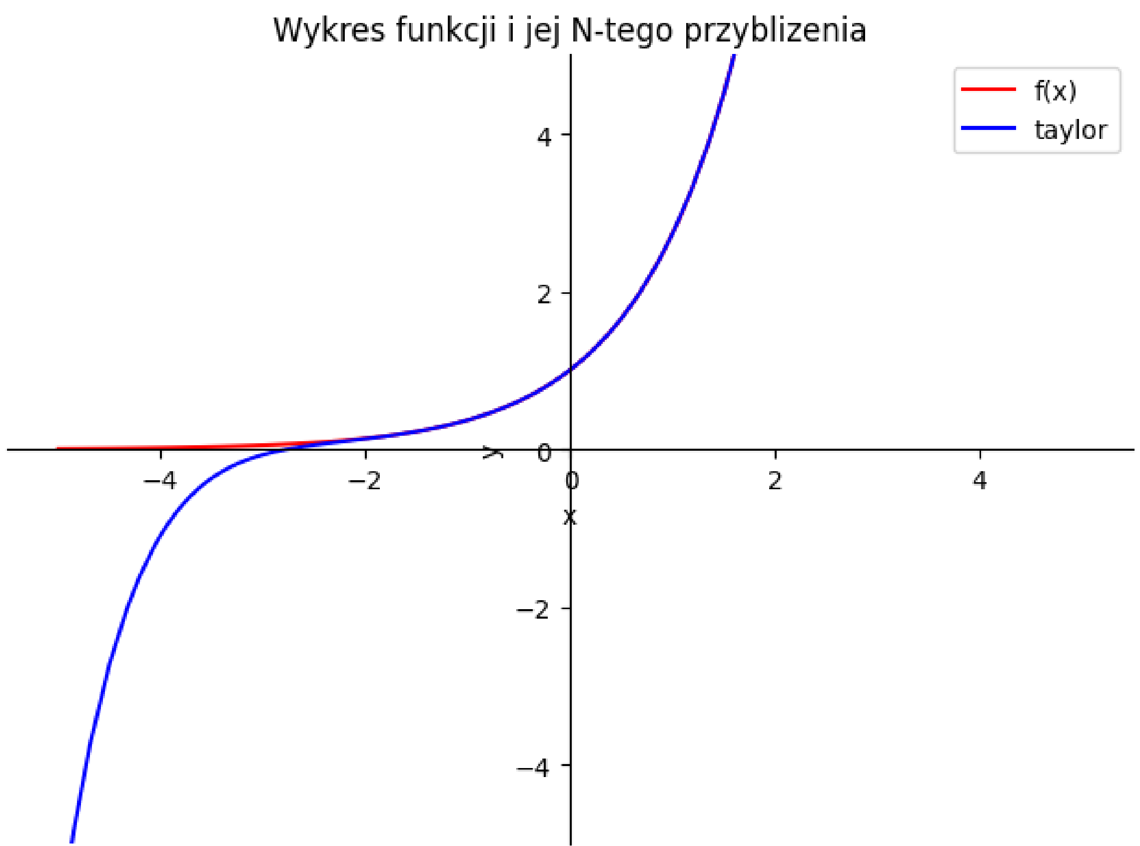
In [52]:

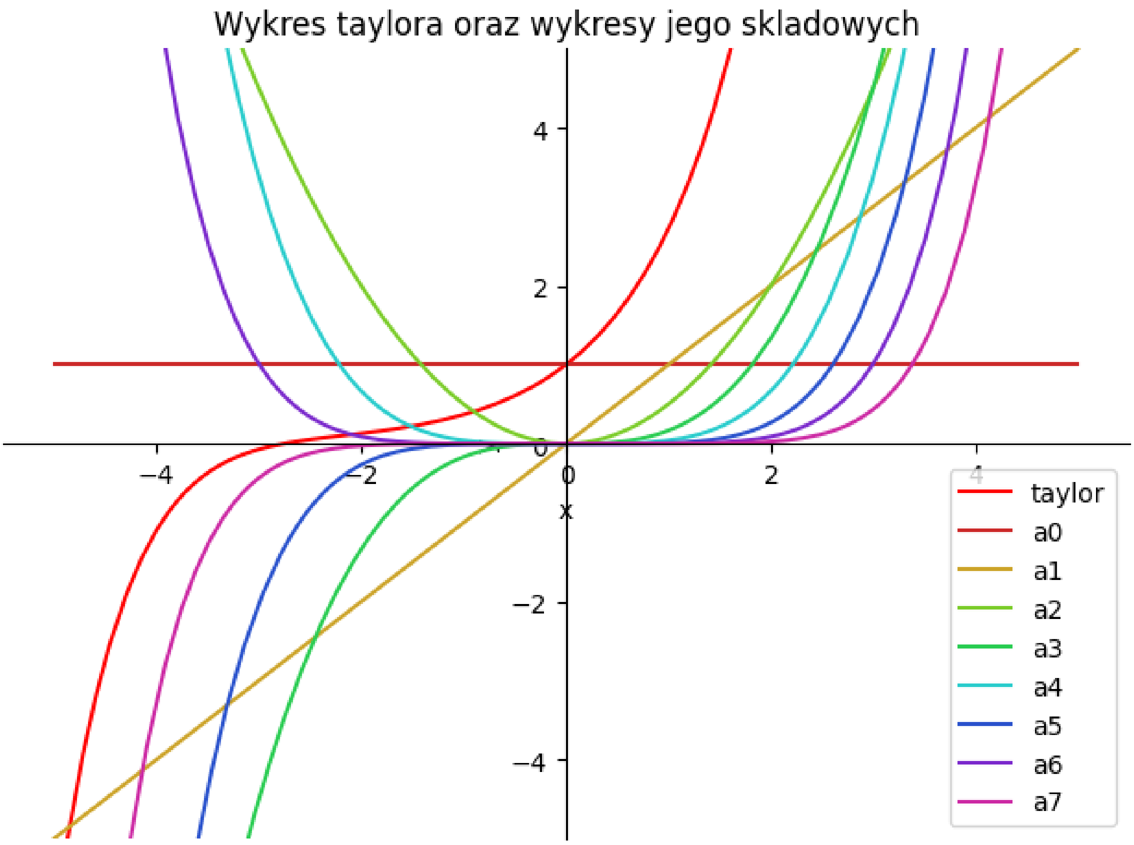
```
def t1(func, Nn, x0):
    ic(func, Nn, x0)
    skladowe = calculateTaylorComponents(func ,Nn, x0)
    szeregTaylora = getTaylorSeriesFromComponents(skladowe)
    ic(skladowe, szeregTaylora)
    # Pierwszy układ wykresów - wykres funkcji i jej N-tego przybliżenia
    wykresFunkcjiOrazTaylora = getPlots([    {'func':func, 'color':"red", 'label':"f(x)"},
                                             {'func':szeregTaylora, 'color':"blue", 'label
':"taylor"}],
                                             "Wykres funkcji i jej N-tego przyblizenia", (x
0-5, x0+5), (-5, 5))
    # Drugi układ wykresów - wykres tailora i wszystkich składowych
    taylorPlotData = [{'func': szeregTaylora, 'color': "red", 'label': "taylor"}]  # Dodan
ie danych wykresu Taylora
    colors = getColorsHex(len(skladowe))
    for i, skladowa in enumerate(skladowe):
        if skladowa:
            #randomColor = '#' + ''.join([random.choice('123456789ABCDE') for j in range(6
)])
            randomColor = colors[i]
            taylorPlotData.extend([{'func': skladowa, 'color': randomColor, 'label': f"a{i
}"}]) # Dodanie danych wykresow skladowych
    wykresTayloraOrazSkladowych = getPlots(taylorPlotData, "Wykres taylora oraz wykresy jeg
o skladowych", (x0-5, x0+5), (-5, 5))
    return [s for s in skladowe if s != 0], szeregTaylora, wykresFunkcjiOrazTaylora, wykre
sTayloraOrazSkladowych
```

In [53]:

```
# t1Func = sin(x)
t1Func = E**x
# t1Func = 1/x
t1N = 7
t1X0 = 0
# t1X0 = 1
w1, w2, w3, w4 = t1(t1Func, t1N, t1X0)
print(w1)
print(w2)
```

```
ic| func: exp(x), Nn: 7, x0: 0
ic| skladowe: [1, x, x**2/2, x**3/6, x**4/24, x**5/120, x**6/720, x**7/5040]
    szeregTaylora: x**7/5040 + x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1
```






```
[1, x, x**2/2, x**3/6, x**4/24, x**5/120, x**6/720, x**7/5040]
x**7/5040 + x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1
```

Zadanie T2

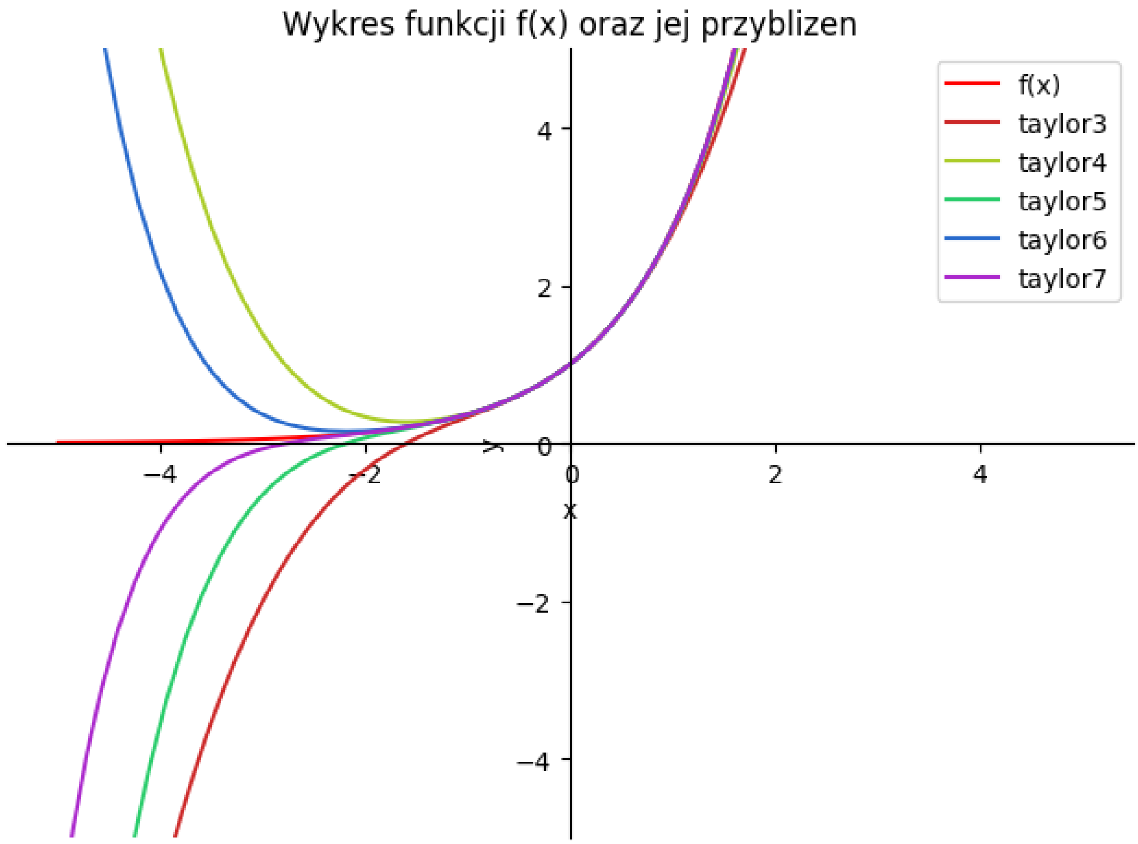
In [54]:

```
def t2(func, N1, N2, x0):
    ic(func, N1, N2, x0)
    przyblizenia = []
    skladowe = calculateTaylorComponents(func, N2, x0)
    for i in range(N1, N2+1):
        taylor = getTaylorSeriesFromComponents(skladowe[:i+1])
        ic(f"Taylor od i = {i}", taylor)
        przyblizenia.append(taylor)
    ic(przyblizenia)
    # Wykres funkcji i wszystkich przyblizen od N1 do N2
    functionPlotData = [{'func': func, 'color': "red", 'label': "f(x)"}]  # Dodanie danych
wykresu Funkcji
    colors = getColorsHex(len(przyblizenia))
    for i, przyblizenie in enumerate(przyblizenia):
        if przyblizenie:
            randomColor = colors[i]
            # randomColor = '#' + ''.join([random.choice('123456789ABCDE') for j in range(
6)])
            functionPlotData.extend([{'func': przyblizenie, 'color': randomColor, 'label':
f"taylor{N1+i}"}]) # Dodanie danych wykresow przyblizen
    wykresFunkcjiOrazJejPrzyblizen = getPlots(functionPlotData, "Wykres funkcji f(x) oraz
jej przyblizen", (x0-5, x0+5), (-5, 5))
    return przyblizenia, wykresFunkcjiOrazJejPrzyblizen
```

In [55]:

```
# t2Func = sin(x)
t2Func = t1Func
t2N1 = 3
t2N2 = 7
# t2X0 = 0
t2X0 = t1X0
w1, w2 = t2(t2Func, t2N1, t2N2, t2X0)
```

```
ic| func: exp(x), N1: 3, N2: 7, x0: 0
ic| f"Taylor od i = {i}": 'Taylor od i = 3'
    taylor: x**3/6 + x**2/2 + x + 1
ic| f"Taylor od i = {i}": 'Taylor od i = 4'
    taylor: x**4/24 + x**3/6 + x**2/2 + x + 1
ic| f"Taylor od i = {i}": 'Taylor od i = 5'
    taylor: x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1
ic| f"Taylor od i = {i}": 'Taylor od i = 6'
    taylor: x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1
ic| f"Taylor od i = {i}": 'Taylor od i = 7'
    taylor: x**7/5040 + x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1
ic| przyblizenia: [x**3/6 + x**2/2 + x + 1,
                   x**4/24 + x**3/6 + x**2/2 + x + 1,
                   x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1,
                   x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1,
                   x**7/5040 + x**6/720 + x**5/120 + x**4/24 + x**3/6 + x**2/2 + x + 1]
```



Zadanie T3

In [56]:

```
t3Func1 = cos(x)
t3Func2 = 1 / (1-x)
t3Func3 = log(1 / (1-x))

t3Maclaurin1 = getTaylorSeries(t3Func1, 5 ,0)
t3Maclaurin2 = getTaylorSeries(t3Func2, 5 ,0)
t3Maclaurin3 = getTaylorSeries(t3Func3, 5 ,0)

ic(t3Maclaurin1)
ic(t3Maclaurin2)
ic(t3Maclaurin3)

# t34 = log(1-x)
# t34r = getTaylorSeries(t34, 5 , 0)
# ic(t34r)
```

```
ic| t3Maclaurin1: x**4/24 - x**2/2 + 1
ic| t3Maclaurin2: x**5 + x**4 + x**3 + x**2 + x + 1
ic| t3Maclaurin3: x**5/5 + x**4/4 + x**3/3 + x**2/2 + x
```

Out[56]:

$$\frac{x^5}{5}+\frac{x^4}{4}+\frac{x^3}{3}+\frac{x^2}{2}+x$$

Uniwersalne funkcje do wyliczania Fouriera

In [57]:

```
def calculateFourierComponents(func, Nn):
    a0 = 1 / pi * integrate(func, (x, -1 * pi, pi)) / 2 # od razu podzielone na 2
    # ic(a0)
    skladowe = []
    for n in range(1, Nn+1):
        an = 1 / pi * integrate(func * cos(n*x), (x, -1 * pi, pi))
        bn = 1 / pi * integrate(func * sin(n*x), (x, -1 * pi, pi))
        skladowa = an * cos(n*x) + bn * sin(n*x)
        ic(n, an, bn, skladowa)
        skladowe.append(skladowa)
    return a0, skladowe

def getFourierSeriesFromComponents(a0, skladowe):
    return a0 + sum(skladowe)

def getFourierSeries(func, Nn):
    return getFourierSeriesFromComponents(*calculateFourierComponents(func, Nn))
```

Zadanie F1

In [58]:

```
def f1(func, Nn):
    ic(func, Nn)
    a0, skladowe = calculateFourierComponents(func ,Nn)
    szeregFouriera = getFourierSeriesFromComponents(a0, skladowe)
    ic(a0, skladowe, szeregFouriera)
    # Układ wykresów - wykres funkcji, wykres N-tego przybliżenia oraz wykres skladowych
    fourierPlotData = [{'func': func, 'color': "red", 'label': "f(x)"},
                       {'func': szeregFouriera, 'color': "blue", 'label': "fourier"},
                       {'func': a0, 'color': "yellow", 'label': "a0 /2 "}]  # Dodanie dany
ch wykresu Funkcji, Fouriera i a0
    colors = getColorsHex(len(skladowe))
    for i, skladowa in enumerate(skladowe):
        if skladowa:
            randomColor = colors[i]
            #randomColor = '#' + ''.join([random.choice('0123456789ABCDEF') for j in range
(6)])
            fourierPlotData.extend([{'func': skladowa, 'color': randomColor, 'label': f"sk
ladowa{i+1}"}]) # Dodanie danych wykresow skladowych
    wykresFunkcjiFurieraISkladowych = getPlots(fourierPlotData, "Wykres funkcji, fouriera
i skladowych", (-4, 4))
    return [s for s in skladowe if s != 0], szeregFouriera, wykresFunkcjiFurieraISkladowyc
h
```

In [59]:

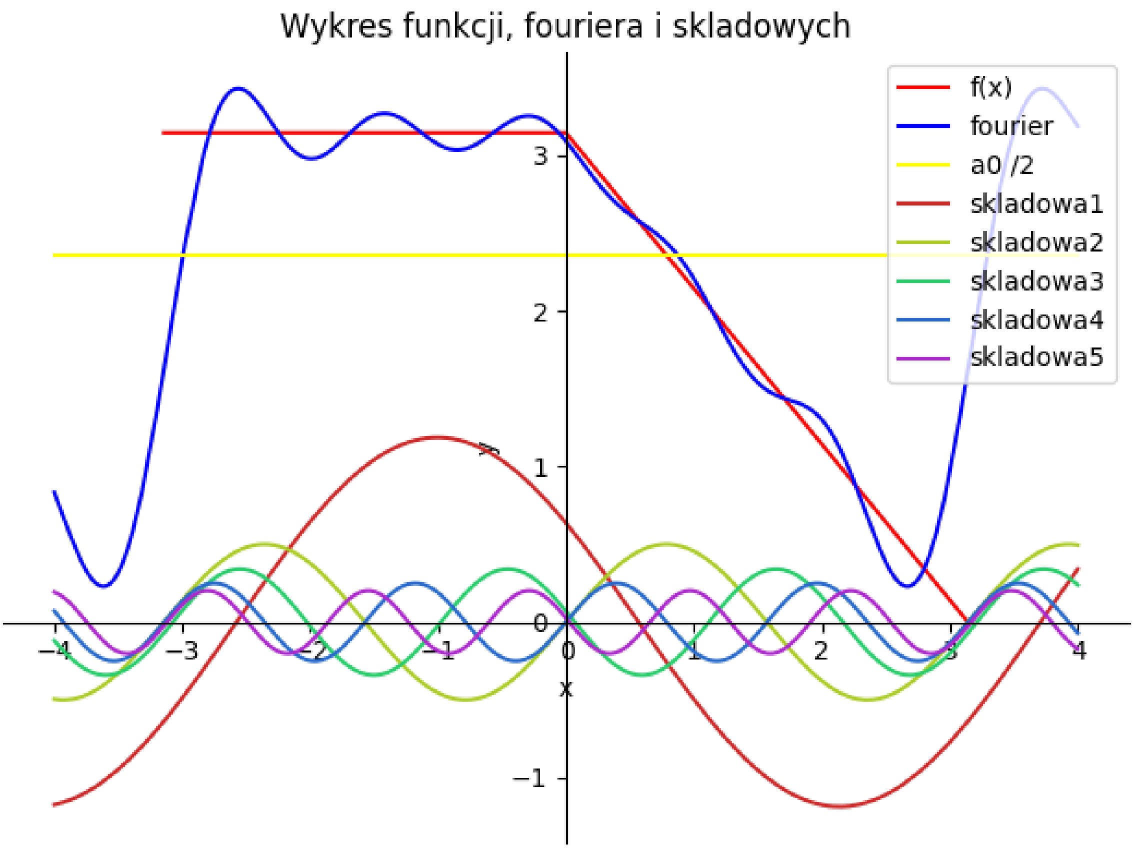
```
# f1Func = Piecewise(
#     (-1, (x > -pi) & (x < 0)),
#     (0, x == 0),
#     (1, (x > 0) & (x < pi))
# )
f1Func = Piecewise(
    (pi, (x > -pi) & (x < 0)),
    (pi, x == 0),
    (pi - x, (x > 0) & (x < pi))
)
# f1Func = Piecewise(
#     (x + pi, (x >= -pi) & (x < 0)),
#     (0, x == 0),
#     (x**2 - pi**2, (x > 0) & (x <= pi))
```

```
# )

f1N = 5


w1, w2, w3 = f1(f1Func, f1N)
print(w1)
print(w2)
```

```
ic| func: Piecewise((pi, (x < 0) & (x > -pi)), (pi - x, (x > 0) & (x < pi)))
    Nn: 5
ic| n: 1, an: 2/pi, bn: -1, skladowa: -sin(x) + 2*cos(x)/pi
ic| n: 2, an: 0, bn: 1/2, skladowa: sin(2*x)/2
ic| n: 3
    an: 2/(9*pi)
    bn: -1/3
    skladowa: -sin(3*x)/3 + 2*cos(3*x)/(9*pi)
ic| n: 4, an: 0, bn: 1/4, skladowa: sin(4*x)/4
ic| n: 5
    an: 2/(25*pi)
    bn: -1/5
    skladowa: -sin(5*x)/5 + 2*cos(5*x)/(25*pi)
ic| a0: 3*pi/4
    skladowe: [-sin(x) + 2*cos(x)/pi,
               sin(2*x)/2,
               -sin(3*x)/3 + 2*cos(3*x)/(9*pi),
               sin(4*x)/4,
               -sin(5*x)/5 + 2*cos(5*x)/(25*pi)]
    szeregFouriera: -sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 - sin(5*x)/5 + 2*cos(x)
/pi + 2*cos(3*x)/(9*pi) + 2*cos(5*x)/(25*pi) + 3*pi/4
```




```
[-sin(x) + 2*cos(x)/pi, sin(2*x)/2, -sin(3*x)/3 + 2*cos(3*x)/(9*pi), sin(4*x)/4, -sin(5*x)/
5 + 2*cos(5*x)/(25*pi)]
-sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 - sin(5*x)/5 + 2*cos(x)/pi + 2*cos(3*x)/(9*
pi) + 2*cos(5*x)/(25*pi) + 3*pi/4
```

Zadanie F2

In [60]:

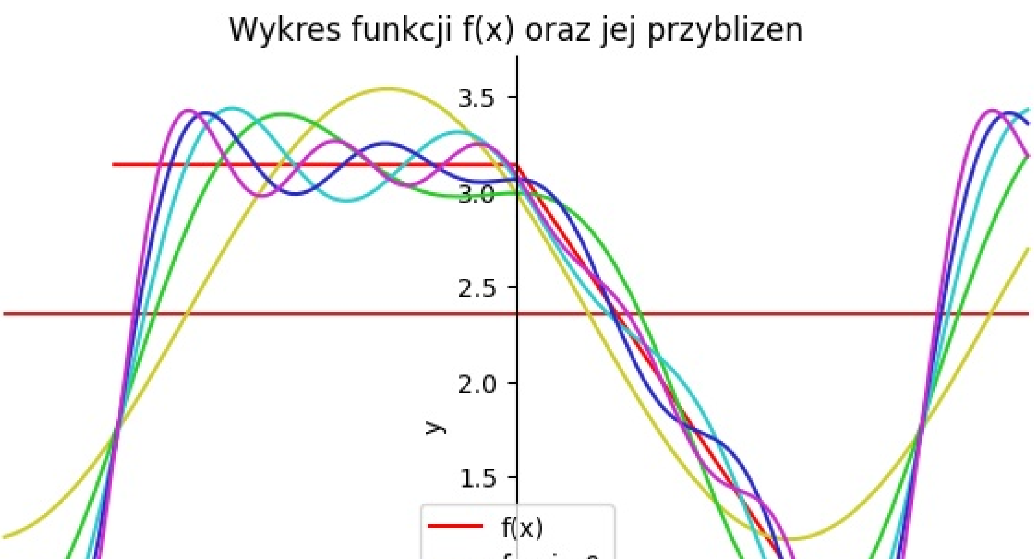
```
def f2(func, N1, N2):
    ic(func, N1, N2)
```

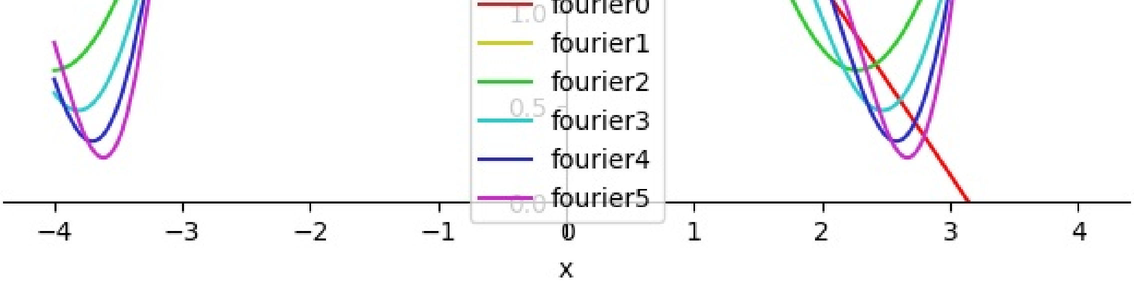
```
    przyblizenia = []
    a0, skladowe = calculateFourierComponents(func, N2)
    for i in range(N1, N2+1):
        fourier = getFourierSeriesFromComponents(a0, skladowe[:i])
        # ic(f"Fourier od i = {i}", fourier)
        przyblizenia.append(fourier)
    ic(przyblizenia)
    # Wykres funkcji i wszystkich przyblizen od N1 do N2
    functionPlotData = [{'func': func, 'color': "red", 'label': "f(x)"}]  # Dodanie danych
wykresu Funkcji
    colors = getColorsHex(len(przyblizenia))
    for i, przyblizenie in enumerate(przyblizenia):
        if przyblizenie:
            randomColor = colors[i]
            # randomColor = '#' + ''.join([random.choice('0123456789ABCDEF') for j in rang
e(6)])
            functionPlotData.extend([{'func': przyblizenie, 'color': randomColor, 'label':
f"fourier{N1+i}"}]) # Dodanie danych wykresow przyblizen
    wykresFunkcjiOrazJejPrzyblizen = getPlots(functionPlotData, "Wykres funkcji f(x) oraz
jej przyblizen", (-4, 4))
    return przyblizenia, wykresFunkcjiOrazJejPrzyblizen
```

In [61]:

```
f2Func = f1Func
f2N1 = 0
f2N2 = 5
w1, w2 = f2(f2Func, f2N1, f2N2)
print(w1)
print(w2)
```

```
ic| func: Piecewise((pi, (x < 0) & (x > -pi)), (pi - x, (x > 0) & (x < pi)))
    N1: 0
    N2: 5
ic| n: 1, an: 2/pi, bn: -1, skladowa: -sin(x) + 2*cos(x)/pi
ic| n: 2, an: 0, bn: 1/2, skladowa: sin(2*x)/2
ic| n: 3
    an: 2/(9*pi)
    bn: -1/3
    skladowa: -sin(3*x)/3 + 2*cos(3*x)/(9*pi)
ic| n: 4, an: 0, bn: 1/4, skladowa: sin(4*x)/4
ic| n: 5
    an: 2/(25*pi)
    bn: -1/5
    skladowa: -sin(5*x)/5 + 2*cos(5*x)/(25*pi)
ic| przyblizenia: [3*pi/4,
                   -sin(x) + 2*cos(x)/pi + 3*pi/4,
                   -sin(x) + sin(2*x)/2 + 2*cos(x)/pi + 3*pi/4,
                   -sin(x) + sin(2*x)/2 - sin(3*x)/3 + 2*cos(x)/pi + 2*cos(3*x)/(9*pi) + 3
*pi/4,
                   -sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 + 2*cos(x)/pi + 2*cos(3*
x)/(9*pi) + 3*pi/4,
                   -sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 - sin(5*x)/5 + 2*cos(x)/
pi + 2*cos(3*x)/(9*pi) + 2*cos(5*x)/(25*pi) + 3*pi/4]
```






```
[3*pi/4, -sin(x) + 2*cos(x)/pi + 3*pi/4, -sin(x) + sin(2*x)/2 + 2*cos(x)/pi + 3*pi/4, -sin(
x) + sin(2*x)/2 - sin(3*x)/3 + 2*cos(x)/pi + 2*cos(3*x)/(9*pi) + 3*pi/4, -sin(x) + sin(2*x)
/2 - sin(3*x)/3 + sin(4*x)/4 + 2*cos(x)/pi + 2*cos(3*x)/(9*pi) + 3*pi/4, -sin(x) + sin(2*x)
/2 - sin(3*x)/3 + sin(4*x)/4 - sin(5*x)/5 + 2*cos(x)/pi + 2*cos(3*x)/(9*pi) + 2*cos(5*x)/(2
5*pi) + 3*pi/4]
Plot object containing:
[0]: cartesian line: Piecewise((pi, (x < 0) & (x > -pi)), (pi - x, (x > 0) & (x < pi))) for
x over (-4.0, 4.0)
[1]: cartesian line: 3*pi/4 for x over (-4.0, 4.0)
[2]: cartesian line: -sin(x) + 2*cos(x)/pi + 3*pi/4 for x over (-4.0, 4.0)
[3]: cartesian line: -sin(x) + sin(2*x)/2 + 2*cos(x)/pi + 3*pi/4 for x over (-4.0, 4.0)
[4]: cartesian line: -sin(x) + sin(2*x)/2 - sin(3*x)/3 + 2*cos(x)/pi + 2*cos(3*x)/(9*pi) +
3*pi/4 for x over (-4.0, 4.0)
[5]: cartesian line: -sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 + 2*cos(x)/pi + 2*cos(3
*x)/(9*pi) + 3*pi/4 for x over (-4.0, 4.0)
[6]: cartesian line: -sin(x) + sin(2*x)/2 - sin(3*x)/3 + sin(4*x)/4 - sin(5*x)/5 + 2*cos(x)
/pi + 2*cos(3*x)/(9*pi) + 2*cos(5*x)/(25*pi) + 3*pi/4 for x over (-4.0, 4.0)
```

Uniwersalne funkcje do zadan 3, 4, 5

In [62]:

```
def getFourierSubcomponents(func, Nn):
    a0 = 1 / pi * integrate(func, (x, -1 * pi, pi))
    elementyAiB = [{"a0":a0, "b0":0}]
    for n in range(1, Nn+1):
        an = 1 / pi * integrate(func * cos(n*x), (x, -1 * pi, pi))
        bn = 1 / pi * integrate(func * sin(n*x), (x, -1 * pi, pi))
        elementyAiB.append({f"a{n}":an, f"b{n}":bn})
    return elementyAiB

def displayFourierSubcomponents(func, Nn):
    elements = getFourierSubcomponents(func, Nn)
    for i, n in enumerate(elements):
        print(f"a{i} = {n[f'a{i}']}\nb{i} = {n[f'b{i}']}")

def createPandasTable(tableValues, columns, indexValues):
    table = pd.DataFrame(tableValues, columns=columns)
    newIndexes = [f"f({n})" for n in indexValues]
    newIndexes.append("f(x)")
    table.index = newIndexes
    return table

def countFourierFunctions(pFunction, pValues, numbers):
    resultTable = []
    rowCounter = 0
    for number in numbers:
        resultTable.append([])
        resultFourierFunction = getFourierSeries(pFunction, number).simplify()
        # resultFourierFunctionString = str(resultFourierFunction)[:120] + ('...' if len(s
tr(resultFourierFunction)) > 120 else '')
        for pValue in pValues:
            resultValue = round(resultFourierFunction.subs(x, pValue), 3)
            #print(f"Wartość: {str(pValue):<10} N:{number} wynosi: {resultFourierFunctionSt
ring:<125} = {resultValue}")
            resultTable[rowCounter].append(resultValue)
        rowCounter += 1
    resultTable.append([])
    for pValue in pValues:
        functionValue = round(pFunction.subs(x, pValue), 3)
```

```
        resultTable[rowCounter].append(functionValue)
        #print(f"Funkcja {pFunction} dla wartości: {str(pValue):<10} = {functionValue}")
    return createPandasTable(resultTable, pValues, numbers)
```

Zadanie F3

In [63]:

```
f3Func = Piecewise(
    (x + pi, (x >= -pi) & (x < 0)),
    (0, x == 0),
    (x**2 - pi**2, (x > 0) & (x <= pi))
)
```

In [64]:

```
displayFourierSubcomponents(f3Func, 9)
```

```
a0 = (-2*pi**3/3 + pi**2/2)/pi
b0 = 0
a1 = (2 - 2*pi)/pi
b1 = (-pi**2 - 4 - pi)/pi
a2 = 1/2
b2 = (-pi**2/2 - pi/2)/pi
a3 = (2/9 - 2*pi/9)/pi
b3 = (-pi**2/3 - pi/3 - 4/27)/pi
a4 = 1/8
b4 = (-pi**2/4 - pi/4)/pi
a5 = (2/25 - 2*pi/25)/pi
b5 = (-pi**2/5 - pi/5 - 4/125)/pi
a6 = 1/18
b6 = (-pi**2/6 - pi/6)/pi
a7 = (2/49 - 2*pi/49)/pi
b7 = (-pi**2/7 - pi/7 - 4/343)/pi
a8 = 1/32
b8 = (-pi**2/8 - pi/8)/pi
a9 = (2/81 - 2*pi/81)/pi
b9 = (-pi**2/9 - pi/9 - 4/729)/pi
```

In [65]:

```
exampleFunctions = (-3*pi/4, -1*pi/2, -1*pi/4, pi/4, pi/2, 3*pi/4)
example3Ns = (2, 5, 8)
f3resultTable = countFourierFunctions(f3Func, exampleFunctions, example3Ns)
f3resultTable
```

```
ic| n: 1
    an: (2 - 2*pi)/pi
    bn: (-pi**2 - 4 - pi)/pi
    skladowa: (-pi**2 - 4 - pi)*sin(x)/pi + (2 - 2*pi)*cos(x)/pi
ic| n: 2
    an: 1/2
    bn: (-pi**2/2 - pi/2)/pi
    skladowa: (-pi**2/2 - pi/2)*sin(2*x)/pi + cos(2*x)/2
ic| n: 1
    an: (2 - 2*pi)/pi
    bn: (-pi**2 - 4 - pi)/pi
    skladowa: (-pi**2 - 4 - pi)*sin(x)/pi + (2 - 2*pi)*cos(x)/pi
ic| n: 2
    an: 1/2
    bn: (-pi**2/2 - pi/2)/pi
    skladowa: (-pi**2/2 - pi/2)*sin(2*x)/pi + cos(2*x)/2
ic| n: 3
    an: (2/9 - 2*pi/9)/pi
    bn: (-pi**2/3 - pi/3 - 4/27)/pi
    skladowa: (-pi**2/3 - pi/3 - 4/27)*sin(3*x)/pi + (2/9 - 2*pi/9)*cos(3*x)/pi
ic| n: 4
    an: 1/8
    bn: (-pi**2/4 - pi/4)/pi
    skladowa: (-pi**2/4 - pi/4)*sin(4*x)/pi + cos(4*x)/8
ic| n: 5
```

```
    an: (2/25 - 2*pi/25)/pi
    bn: (-pi**2/5 - pi/5 - 4/125)/pi
    skladowa: (-pi**2/5 - pi/5 - 4/125)*sin(5*x)/pi + (2/25 - 2*pi/25)*cos(5*x)/pi
ic| n: 1
    an: (2 - 2*pi)/pi
    bn: (-pi**2 - 4 - pi)/pi
    skladowa: (-pi**2 - 4 - pi)*sin(x)/pi + (2 - 2*pi)*cos(x)/pi
ic| n: 2
    an: 1/2
    bn: (-pi**2/2 - pi/2)/pi
    skladowa: (-pi**2/2 - pi/2)*sin(2*x)/pi + cos(2*x)/2
ic| n: 3
    an: (2/9 - 2*pi/9)/pi
    bn: (-pi**2/3 - pi/3 - 4/27)/pi
    skladowa: (-pi**2/3 - pi/3 - 4/27)*sin(3*x)/pi + (2/9 - 2*pi/9)*cos(3*x)/pi
ic| n: 4
    an: 1/8
    bn: (-pi**2/4 - pi/4)/pi
    skladowa: (-pi**2/4 - pi/4)*sin(4*x)/pi + cos(4*x)/8
ic| n: 5
    an: (2/25 - 2*pi/25)/pi
    bn: (-pi**2/5 - pi/5 - 4/125)/pi
    skladowa: (-pi**2/5 - pi/5 - 4/125)*sin(5*x)/pi + (2/25 - 2*pi/25)*cos(5*x)/pi
ic| n: 6
    an: 1/18
    bn: (-pi**2/6 - pi/6)/pi
    skladowa: (-pi**2/6 - pi/6)*sin(6*x)/pi + cos(6*x)/18
ic| n: 7
    an: (2/49 - 2*pi/49)/pi
    bn: (-pi**2/7 - pi/7 - 4/343)/pi
    skladowa: (-pi**2/7 - pi/7 - 4/343)*sin(7*x)/pi + (2/49 - 2*pi/49)*cos(7*x)/pi
ic| n: 8
    an: 1/32
    bn: (-pi**2/8 - pi/8)/pi
    skladowa: (-pi**2/8 - pi/8)*sin(8*x)/pi + cos(8*x)/32
```

Out[65]:

	-3*pi/4	-pi/2	-pi/4	pi/4	pi/2	3*pi/4
f(2)	0.218	2.410	2.431	-9.368	-8.419	-3.298
f(5)	0.364	1.946	2.868	-9.764	-7.705	-3.986
f(8)	0.684	1.327	1.769	-8.641	-7.134	-4.204
f(x)	0.785	1.571	2.356	-9.253	-7.402	-4.318

In [66]:

```
# # Tabela pomocnicza z różnicami względem f(x) dla sprawdzenia poprawnosci
# fxValues = f3resultTable.loc["f(x)"]
# differenceTable = f3resultTable.subtract(fxValues, axis=1).abs().round(3)
# differenceTable
```

Zadanie F4

In [67]:

```
f4Func = Piecewise(
    (x + pi, (x >= -pi) & (x < 0)),
    (0, x==0),
    (x - pi, (x > 0) & (x <= pi))
)
```

In [68]:

```
displayFourierSubcomponents(f4Func, 6)
```

```
a0 = 0
```

```
b0 = 0
a1 = 0
b1 = -2
a2 = 0
b2 = -1
a3 = 0
b3 = -2/3
a4 = 0
b4 = -1/2
a5 = 0
b5 = -2/5
a6 = 0
b6 = -1/3
```

In [69]:

```
example4Ns = (2, 4, 6)
f4resultTable = countFourierFunctions(f4Func, exampleFunctions, example4Ns)
f4resultTable
```

```
ic| n: 1, an: 0, bn: -2, skladowa: -2*sin(x)
ic| n: 2, an: 0, bn: -1, skladowa: -sin(2*x)
ic| n: 1, an: 0, bn: -2, skladowa: -2*sin(x)
ic| n: 2, an: 0, bn: -1, skladowa: -sin(2*x)
ic| n: 3, an: 0, bn: -2/3, skladowa: -2*sin(3*x)/3
ic| n: 4, an: 0, bn: -1/2, skladowa: -sin(4*x)/2
ic| n: 1, an: 0, bn: -2, skladowa: -2*sin(x)
ic| n: 2, an: 0, bn: -1, skladowa: -sin(2*x)
ic| n: 3, an: 0, bn: -2/3, skladowa: -2*sin(3*x)/3
ic| n: 4, an: 0, bn: -1/2, skladowa: -sin(4*x)/2
ic| n: 5, an: 0, bn: -2/5, skladowa: -2*sin(5*x)/5
ic| n: 6, an: 0, bn: -1/3, skladowa: -sin(6*x)/3
```

Out[69]:

	-3*pi/4	-pi/2	-pi/4	pi/4	pi/2	3*pi/4
f(2)	0.414	2	2.414	-2.414	-2	-0.414
f(4)	0.886	1.333	2.886	-2.886	-1.333	-0.886
f(6)	0.936	1.733	2.269	-2.269	-1.733	-0.936
f(x)	0.785	1.571	2.356	-2.356	-1.571	-0.785

Zadanie F5

In [70]:

```
f5Func = Piecewise(
    (-1*x - pi, (x >= -pi) & (x <= 0)),
    (x - pi, (x > 0) & (x <= pi))
)
```

In [71]:

```
displayFourierSubcomponents(f5Func, 6)
```

```
a0 = -pi
b0 = 0
a1 = -4/pi
b1 = 0
a2 = 0
b2 = 0
a3 = -4/(9*pi)
b3 = 0
a4 = 0
b4 = 0
a5 = -4/(25*pi)
b5 = 0
a6 = 0
b6 = 0
```

In [72]:

```
example5Ns = (1, 3, 5)
f5resultTable = countFourierFunctions(f5Func, exampleFunctions, example5Ns)
f5resultTable
```

```
ic| n: 1, an: -4/pi, bn: 0, skladowa: -4*cos(x)/pi
ic| n: 1, an: -4/pi, bn: 0, skladowa: -4*cos(x)/pi
ic| n: 2, an: 0, bn: 0, skladowa: 0
ic| n: 3, an: -4/(9*pi), bn: 0, skladowa: -4*cos(3*x)/(9*pi)
ic| n: 1, an: -4/pi, bn: 0, skladowa: -4*cos(x)/pi
ic| n: 2, an: 0, bn: 0, skladowa: 0
ic| n: 3, an: -4/(9*pi), bn: 0, skladowa: -4*cos(3*x)/(9*pi)
ic| n: 4, an: 0, bn: 0, skladowa: 0
ic| n: 5, an: -4/(25*pi), bn: 0, skladowa: -4*cos(5*x)/(25*pi)
```

Out[72]:

	-3*pi/4	-pi/2	-pi/4	pi/4	pi/2	3*pi/4
f(1)	-0.670	-1.571	-2.471	-2.471	-1.571	-0.670
f(3)	-0.771	-1.571	-2.371	-2.371	-1.571	-0.771
f(5)	-0.807	-1.571	-2.335	-2.335	-1.571	-0.807
f(x)	-0.785	-1.571	-2.356	-2.356	-1.571	-0.785