Sprawozdanie z programu do interpolacji Lagrange’a

Wykonał: Jakub Świderski

1. Import bibliotek do obliczeń i rysowania wykresów funkcji.

```
from sympy.abc import *
from sympy import *
import matplotlib.pyplot as plt
```

2. Pobranie od użytkownika ilości węzłów i utworzenie tablic do przetrzymywania ich.

```
try:
    n=(int(input("Podaj ilość węzłów:")))
    if n<2:
        exit("To nie jest poprawna wartość!")
except:
    exit("To nie jest poprawna wartość!")
punkt_x=[]
punkt_y=[]
```

3. Pobranie od użytkownika wartości poszczególnych punktów za pomocą pętli oraz zmiana ich typu z tekstowego na int lub float w zależności od podanej liczby (za pomocą operacji modulo 1).

```
try:
    for i in range(n):
        punkt_x.append(float(input("Podaj punkt x" + str(i) + ":")))
        if punkt_x[-1] % 1 == 0:
            punkt_x[-1] = int(punkt_x[-1])
        punkt_y.append(float(input("Podaj punkt y" + str(i) + ":")))
        if punkt_y[-1] % 1 == 0:
            punkt_y[-1] = int(punkt_y[-1])
except:
    exit("To nie jest poprawna wartość!")
```

4. Sprawdzenie, czy któryś z punktów x nie został powtórzony.

```
if len(punkt_x) > len(set(punkt_x)):
    exit("Elementy x muszą się różnić!")
```

5. Inicjalizacja funkcji f(x) do tymczasowego przetrzymywania poszczególnych wielomianów bazowych, oraz funkcji g(x) do przetrzymywania szukanego wielomianu interpolacyjnego.

```
f=1
g=0
```

6. Obliczanie przy pomocy podwójnej pętli. Pętla wewnętrzna odpowiada za przemnażanie funkcji f(x) przez (x-x[j])/(x[i]-x[j]), kiedy indeksy i oraz j są różne. Po zakończeniu wszystkich iteracji pętli wewnętrznej otrzymujemy wielomian, który po pomnożeniu przez y[i] tworzy wielomian bazowy. Następnie jest on dodawany do funkcji g(x). Aby przygotować miejsce do obliczenia kolejnego wielomianu przywracamy f(x) do wartości początkowej. Czynności powtarzamy w pętli zewnętrznej dla każdego punktu y, aż do uzyskania w g(x) wielomianu Lagrange'a.

```
for i in range(len(punkt_y)):
    for j in range(len(punkt_x)):
        if(i!=j):
            f*=(x-punkt_x[j])/(punkt_x[i]-punkt_x[j])
    f*=punkt_y[i]
    g+=f
    f=1
```

7. Uproszczenie funkcji g(x) oraz wypisanie jej na 2 różne sposoby.

```
g=expand(g)
print(g)
print(" ")
pprint(g)
```

8. Wyliczenie różnicy najbardziej odległych wartości x i pomnożenie przez 0.2 (w celu wyznaczania przedziału do rysowania wykresu.) Następnie utworzenie samego wykresu na podstawie danych zawartych w g(x), oraz naniesienie na wykres wszystkich podanych węzłów. Na koniec następuje dopisanie w pętli współrzędnych poszczególnych punktów do wykresu oraz wyświetlenie samego wykresu.

```
pr=(max(punkt_x)-min(punkt_x))*0.2
plot(g,(x,min(punkt_x)-pr,max(punkt_x)+pr),show=True)
plt.plot(punkt_x, punkt_y, 'o')
for i, j in zip(punkt_x, punkt_y):
    plt.annotate('(%s, %s)' % (i, j), xy=(i, j), textcoords='offset points', xytext=(0,10), ha='center')
plt.show()
```

Wynik działania programu:

```
Podaj ilość węzłów: 5
Podaj punkt x0: 1
Podaj punkt y0: 4
Podaj punkt x1: 2
Podaj punkt y1: -5
Podaj punkt x2: 3
Podaj punkt y2: 7
Podaj punkt x3: 4
Podaj punkt y3: 20
Podaj punkt x4: 5
Podaj punkt y4: -15
-29*x**4/24 + 35*x**3/4 - 283*x**2/24 - 67*x/4 + 25
```

$$-\frac{29 \cdot x^4}{24} + \frac{35 \cdot x^3}{4} - \frac{283 \cdot x^2}{24} - \frac{67 \cdot x}{4} + 25$$