Zespół R - Zadanie 3

Nicolas Wielocha
Matusz Redzimski
Maciej Cymański

30.03.2022

Spis treści

1 Zadanie

Obliczyć $\log_2 22$ za pomocą wielomianów interpolujących funkcję z tabeli

x	1	2	4	8	16	32	64	128	256	512	1024	2048
$f(x)$	0	1	2	3	4	5	6	7	8	9	10	11

Sprawdzić eksperymentalnie jaki podzbiór danych z tabelki daje najlepsze przybliżenie dokładnej wartości logarytmu (czyli dla jakiego zestawu tych węzłów wielomian Lagrange'a przebiega najbliżej punktu $(22, \log_2 22)$.

2 Wstęp teoretyczny

2.1 Wielomian interpolacyjny w posatci Lagrange'a

Dla parami różnych węzłów $x_0, ..., x_n$ i danych wartości $y_0, ..., y_n$ wielomian P stopnia co najwyżej n, dany wzorem

$$P(x) = \sum_{i=0}^{n} y_i l_i(x),$$

gdzie

$$l_i(x) = \prod_{k=0, k\neq i}^{n} \frac{x - x_k}{x_i - x_k}, i = 0, 1, ..., n,$$

spełnia warunki

$$P(x_i) = y_i, i = 0, 1, ..., n.$$

Wielomiany $l_i = l_i(x)$ nazywamy **mnożnikami Lagrange'a**

Przy pomocy interpolacji wielomianów obliczymy $f(22)$, a następnie przy pomocy języka programowania Python przeprowadzimy serie obliczeń dla każdego możliwego podzbioru i ustalimy który przy użyciu wielomianu Lagrange'a najlepiej oddaje estymacje położenia punktu $(22, \log 22)$. Do obliczeń użyjemy biblioteki scipy.

3 Rozważmy funkcje dane tabelką:

Za pomocą metody zaprezentowanej w poprzednim podpukncie interpolujemy wielomian na podstawie wybranych węzłów.

x	8	16	32	64
$f(x)$	3	4	5	6

Niech $x_0 = 8, x_1 = 16, x_2 = 32, x_3 = 64$. Liczymy współczynniki $l_i(x)$, wtedy

$$l_0(x) = \frac{(x-16)}{(8-16)}\frac{(x-32)}{(8-32)}\frac{(x-64)}{(8-64)} = \frac{(16-x)(32-x)(64-x)}{10752}$$

$$l_1(x) = \frac{(x-8)}{(16-8)}\frac{(x-32)}{(16-32)}\frac{(x-64)}{(16-64)} = \frac{(32-x)(64-x)(-8+x)}{6144}$$

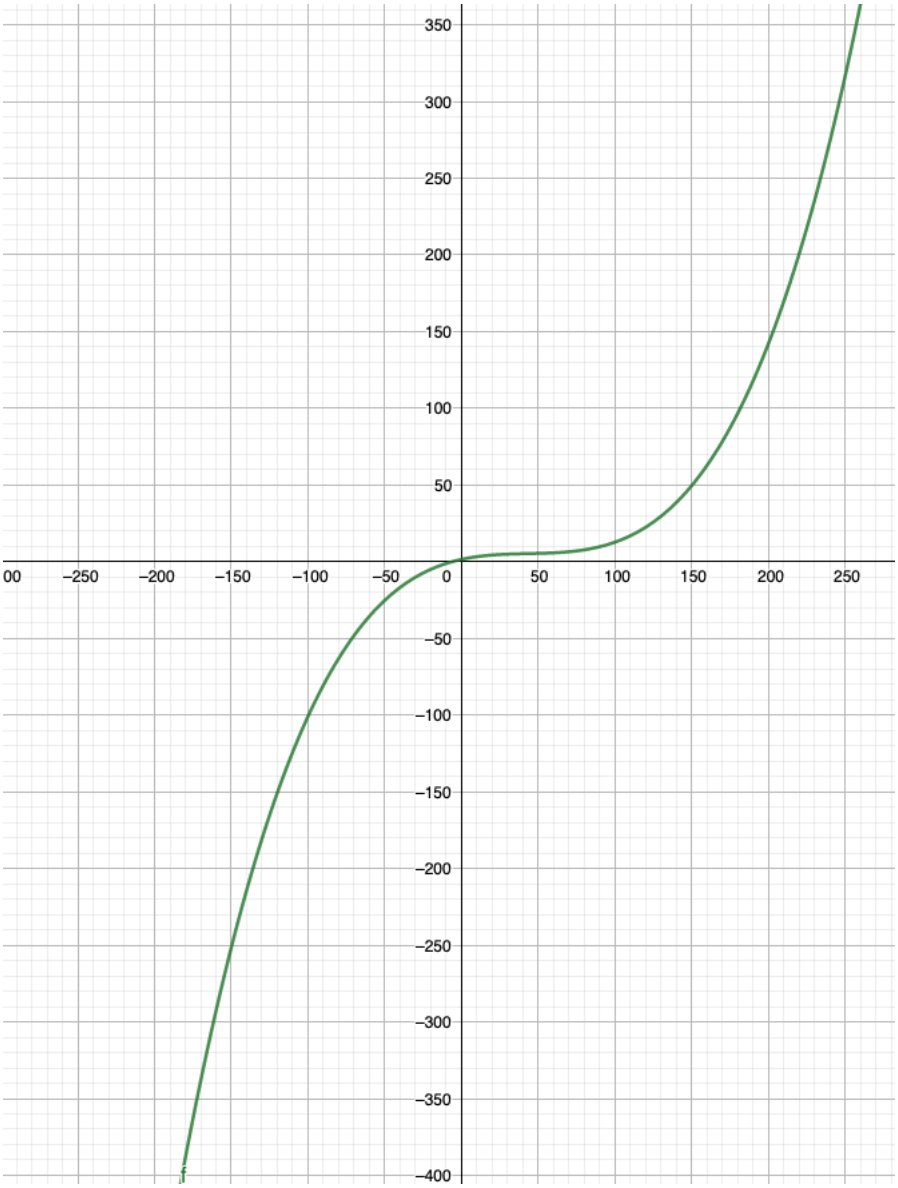
$$l_2(x) = \frac{(x-8)}{32-8}\frac{(x-16)}{32-16}\frac{(x-64)}{32-64} = \frac{(64-x)(-16+x)(-8+x)}{12288}$$

$$l_3(x) = \frac{(x-8)}{(64-8)}\frac{(x-16)}{(64-16)}\frac{(x-32)}{(64-32)} = \frac{(x-8)(x-16)(x-32)}{86016}$$

Jeśli $y_0 = 3, y_1 = 4, y_2 = 5, y_3 = 6$, to

$$P(x) = 3 \cdot l_0(x) + 4 \cdot l_1(x) + 5 \cdot l_2(x) + 6 \cdot l_3(x) = \frac{x(18816 - 392x + 3x^2)}{86016} + \frac{32}{21}$$

Odpowiedź: Dla $x = 22$, wartość interpolowanego wielomianu wynosi 4,5019.



Rysunek 1: Wykres interpolowanego wielomianu

4 Opis implementacji algorytmu

4.1 Klasy

Estimation

Klasa reprezentująca pojedynczą kompozycję węzłów, wielomian na nich podstawie oraz wartość obliczoną przez podstawienie danej wartości za x.

- Parametr **nodes**: lista węzłów w tabeli.
- Parametr **polynomial**: wielomian interpolowany z podanymi węzłami.
- Parametr **estiamted_value**: wartość po podstawieniu dowolnej liczby w wielomian.

Lagrange

Klasa stworzona w celu dostarczenia danych do aplikacji GUI, która implementuje algorytm interpolacji LaGrange.

- Parametr **table**: tabela w formie słownika, gdzie klucze są węzłami, a wartości odpowiadają wartością $f(nodes)$.

4.2 Funkcje algorytmu

Podzbiory

Funkcja tworzy wszystkie możlie podzbiory dla tabeli:

x	1	2	4	8	16	32	64	128	256	512	1024	2048
$f(x)$	0	1	2	3	4	5	6	7	8	9	10	11

Współczynnik l(x)

Funkcja tworzy współczynnik $l(x)$ używany we wzorze Lagrange.

- Parametr **nodes**: lista węzłów używanych do obliczenia współczynnkia.
- Parametr **i**: liczba iteracji funkcji nadrzędnej **create_polynomial**.
- **Return**: funkcja zwraca obliczony wielomian w postaci obiektu **sympy**.

Wielomian

Funkcja worzy wielomian z podanych węzłów. Węzły muszą być zawarte we właściwości tabeli obiektu.

- Parametr **nodes**: funkcja przyjmuje węzły, z których interpolowany jest wielomian.
- **Return**: funkcja zwraca wielomian interpolowany.

Najlepsza estymacja

Funkcja tworzy obiekt klasy **Estimation** z najlepszym możliwym wynikiem.

- Parametr **input_value**: dane wejściowe do obliczenia wartości wielomianu.
- Paramater **value**: wartość do porównania z **wielomianem(input_value)**
- **Return**: obiekt estymacji zawierający wielomian/węzły, które interpolowały wielomian/wartość **wielomianu(input)**.

Najgorsza estymacja

Funkcja tworzy obiekt **Estimation** z najgorszym możliwym wynikiem.

- Parametr **input_value**: dane wejściowe do obliczenia wartości wielomianu.
- Parametr **value**: wartość do porównania z **wielomianem(input_value)**
- **Return**: obiekt estymacji zawierający wielomian/węzły, które interpolowały wielomian/wartość **wielomianu(input)**

Estymacja przez wartość

Funkcja szacuje wartość wielomianu interpolowanego przy użyciu całej tablicy obiektu z podanymi danymi wejściowymi.

- Parametr **input_value**: dane wejściowe do obliczenia wartości wielomianu.
- **Return**: obiekt klasy **Estimation**.

5 Opis implementacji GUI

Przy tworzeniu **GUI - Graphical User Interface** korzystamy z biblioteki PySimpleGUI. Biblioteka ta pozwala nam na stworzenie prostej reprezentacji wyników wykonywanych obliczeń.

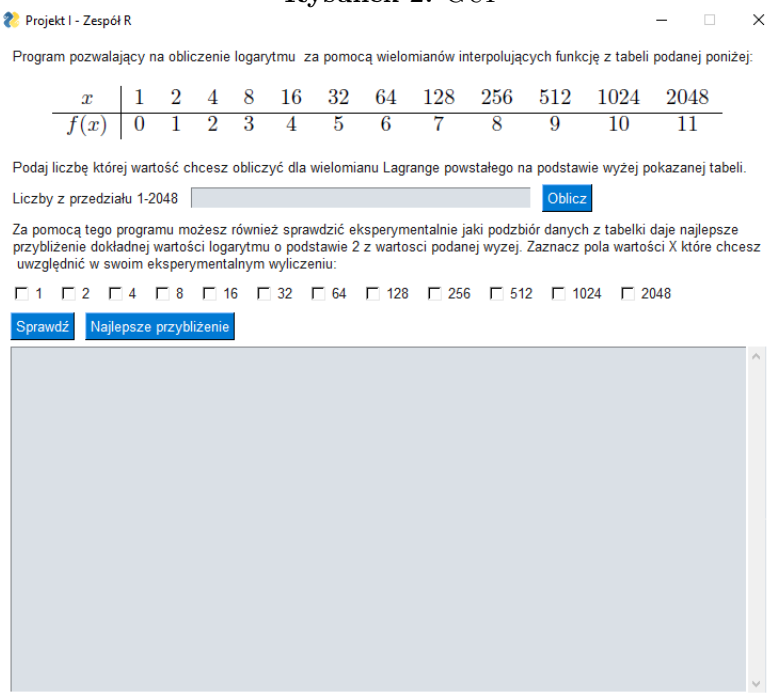
5.1 Layout

Szablon na podstawie, którego funkcja **sg.Window** tworzy interaktywne okno aplikacji.

Korzystamy z poniższych funkcji **Layout'u**:

- **sg.Text**: wypisuje informacje w niej zawarte.
- **sg.Image**: rysuje obraz w niej zawarty.
- **sg.Checkbox**: tworzy interaktywne okienka z możliwością zaznaczania.
- **sg.Button**: tworzy interaktywne przyciski, służące do wywoływania różnych funkcji.
- **sg.Input**: tworzy okno, które przyjmuje wartość przekazywaną do funkcji.
- **sg.Output**: zarezerwowana przestrzeń dla informacji zwracanych przez program.

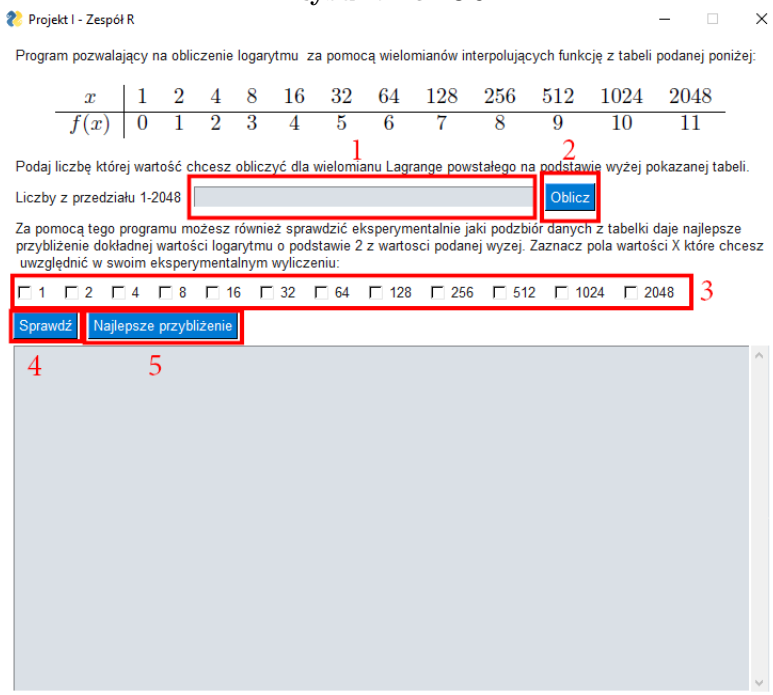
Rysunek 2: GUI



6 Instrukcja obsługi GUI

1. Okienko, w którym wpisujemy wartość z podanego przedziału.
2. Przycisk służący do wykonania obliczeń.
3. Checkbox z węzłami z możliwością wybrania dwóch lub więcej węzłów.
4. Przycisk, który po kliknięciu dla danej wartości (1) i wezłów (2) wyświetla wielomian, oraz jego wartość dla wartości (1) oraz wykres.
5. Przycisk, który po kliknięciu estymuje najlepsze przybliżenie dla podanej wartości (1).

Rysunek 3: GUI



7 Kod programu

Plik main.py

```
import numpy as np
import math
from itertools import compress, product, combinations
from sympy import *
from sympy.plotting import plot as symplot


def sub_lists(a):
    if len(a) == 0:
        return [[]]
    cs = []
    for c in sub_lists(a[1:]):
        cs += [c, c + [a[0]]]
    return cs

class Estimation:

    def __init__(self, nodes,  polynomial, input_value):
        """
        Class which represents single composition of nodes, polynomial based on them and
        value calculated with substituting x with some arbitrary number
        :param nodes: List of nodes in a x - fx table
        :param polynomial: Polynomial interpolated with given nodes
        :param input_value: Value gained by using arbitrary number in polynomial
        """
        self.nodes = nodes
        self.polynomial = simplify(polynomial,ratio=1)
        if input_value:
            self.estimated_value = Float(polynomial.subs(Symbol('x'),input_value),6)
        else:
            self.estimated_value = None

    def calculate_value(self,input_value):
        return Float(self.polynomial.subs(Symbol('x'),input_value),6)


class Lagrange:

    def __init__(self,table: dict):
        """
        Class made to provide data for GUI of application which implements LaGrange interpolation algorithm
        :param table: x - fx table  in dictionary form where keys are nodes and values are corresponding f(node) values
        """
        self.table = table
        self.subtables = sub_lists(list(table.keys()))

```

```
47      def create_lx(self,nodes: list, i):
48          """
49          Creates lx coefficient used in lagrange formula
50          :param nodes: list of nodes used to calculate lx
51          :param i: number of iteration from parent-function
    create_polynomial
52          :return: returns calculated polynomial in form of sympy
    object
53          """
54          lx_coeff = 1
55          x = Symbol('x')
56          for k in range(0, len(nodes)):
57              if i != k:
58                  lx_coeff = lx_coeff * ((x - nodes[k]) / (nodes[i] -
     nodes[k]))
59          return lx_coeff
60
61      def create_polynomial(self, nodes):
62          """
63          Create polynomial out of given nodes
64          Nodes must be contained in object's table property
65          :param nodes: Nodes out of which polynomial is interpolated
66          :return: interpolated polynomial
67          """
68          if set(nodes) - set(self.table.keys()):
69              raise Exception('Wrong nodes input')
70          result = 0
71          for i in range(0,len(nodes)):
72              result += self.table[nodes[i]] * self.create_lx(nodes,
    i)
73          return result
74
75      def best_estimation(self,input_value,value):
76          """
77          Creates Estimation object with best possible result
78          :param input_value: Input to calculate polynomial value
79          :param value: Value to compare with polynomial(input_value)
80          :return: Estimation object containing polynomial / nodes
    which interpolated polynomial / value of polynomial(input)
81          """
82          results = []
83          for sublist in self.subtables:
84              if len(sublist) > 1:
85                  polynomial = self.create_polynomial(sublist)
86                  results.append((sorted(sublist), polynomial,
    input_value))
87          best_result = min(results, key=lambda x: np.abs(value -
    Float(x[1].subs(Symbol('x'), input_value), 6)))
88          return Estimation(best_result[0],best_result[1],
    input_value)
89
90      def worst_estimation(self, input_value, value):
91          """
92          Creates Estimation object with worst possible result
93          :param input_value: Input to calculate polynomial value
94          :param value: Value to compare with polynomial(input_value)
95          :return: Estimation object containing polynomial / nodes
```

```
    which interpolated polynomial / value of polynomial(input)
96          """
97          results = []
98          for sublist in self.subtables:
99              if len(sublist) > 1:
100                 polynomial = self.create_polynomial()
101                 results.append((sorted(sublist), polynomial,
    input_value))
102         best_result = max(results, key=lambda x: np.abs(value -
    Float(x[1].subs(Symbol('x'), input_value), 6)))
103         return Estimation(best_result[0],best_result[1],
    input_value)
104
105     def estimation_by_value(self,input_value):
106         """
107         Estimates value of polynomial interpolated with usage of
    whole object's table with given input
108         :param input_value: Input to calculate polynomial value
109         :return: Estimation object
110         """
111         polynomial = self.create_polynomial(list(self.table.keys())
    )
112         return Estimation(list(self.table.keys()),polynomial,
    input_value)
113
114     def calculate_logarithm(self, base, value):
115         """
116         Calculates logarithm with given base,value with usage of
    math lib
117         """
118         return math.log(value, base)
119
120 if __name__ == '__main__':
121     lagrange = Lagrange({2 ** i:i for i in range(0, 12)})
122     #x = lagrange.best_estimation(22,math.log(22,2))
123     x = lagrange.create_polynomial(list(lagrange.table.keys()))
124     est = Estimation(list(lagrange.table.keys()),x,22)
125     plot(x)
```

Plik projekt1.py

```
import PySimpleGUI as sg
from sympy import *
from backend import Estimation, Lagrange
import math
sg.theme('Reddit')   # Uzywany motyw
# Layout twojego programu


layout = [  [sg.Text('Program pozwalaj cy na obliczenie logarytmu za pomoc  wielomian w interpoluj cych funkcj  z tabeli podanej poni ej:')],
            [sg.Image(r'C:\Users\Rfrev\alg\wykres.png')],
            [sg.Text('Podaj liczb  kt rej warto   chcesz obliczy  dla wielomianu lagaranga powsta ego na podstawie wy ej pokazanej tabeli.')],
            [sg.Text('Liczby z przedzia u 1-2048'),sg.Input(key='key1'),sg.Button('Oblicz'),],
            [sg.Text('Za pomoc  tego programu mo esz r wnie sprawdzi  eksperymentalnie jaki podzbi r danych z tabelki daje najlepsze \nprzybli enie dok adnej warto ci logarytmu o podstawie 2 z wartosci podanej wyzej. Zaznacz pola warto ci X kt re chcesz \n uwzgl dni  w swoim eksperymentalnym wyliczeniu:')],
            [sg.Checkbox('1', default=False,key='1'), sg.Checkbox('2', default=False,key='2'),sg.Checkbox('4', default=False,key='4'),sg.Checkbox('8', default=False,key='8'),sg.Checkbox('16', default=False,key='16'),sg.Checkbox('32', default=False,key='32'),sg.Checkbox('64', default=False,key='64'),sg.Checkbox('128', default=False,key='128'),sg.Checkbox('256', default=False,key='256'),sg.Checkbox('512', default=False,key='512'),sg.Checkbox('1024', default=False,key='1024'),sg.Checkbox('2048', default=False,key='2048')],
            [sg.Button('Sprawd '),sg.Button('Najlepsze przybli enie')],
            [sg.Output(size=(98, 20))],
            ]

# Zainicjowanie zmiennej lagrange potrzebnej do dalszych obliczen
lagrange=Lagrange({2 ** i:i for i in range(0, 12)})
# Utworzenie okna na podstawie layoutu

window = sg.Window('Projekt I - Zesp    R', layout)
# Zczytywanie zachowan uzytkownika ( czy cos kliknal/wpisal/zakonczyl program)
while True:

    event, values = window.read()
#Obliczenie wartosci podanej przez uzytkownika zmiennej dla wielomianu lagrange stworzonego na podstawie calej tabeli danych
    if event == 'Oblicz':
        if(values['key1'].isdigit()):
            x=int(values['key1'])
            if(x>=1 and x<=2048):
                z=lagrange.estimation_by_value(x)
```

```
34                 text = sg.popup('Przybli ona warto   logarytmu o
     podstawie 2 z {} to: \n{} '.format(x,z.estimated_value))
35             else:
36                 sg.PopupError('Podaj liczb  w zakresie od 1 do
    2048.')
37         else:
38             sg.PopupError('"{}" to nie liczba.' .format(values['
    key1']))
39
40
41 #Obliczenie najlepszego przyblizenia dla podanej przez uzytkownika
     zmiennej z automatycznym doborem wybranych wezlow przez program
42     if event == 'Najlepsze przybli enie':
43
44         if(values['key1'].isdigit()):
45             p=int(values['key1'])
46             print(type(p))
47             if(p>=1 and p<=2048):
48                 result=lagrange.best_estimation(p,math.log(p,2))
49                 print('Dany podzbi r: x={} Przybli ona warto
    log2 z {}: {} B   d przybli enia: {:.6f}'.format(result.nodes
    ,result.estimated_value,p,math.log(p,2)-result.estimated_value)
    )
50                 #text = sg.popup('Otrzymany wielomian to:\n {} \n
    warto   tego wielomianu dla x = 22 to:\n {}'.format(result.
    polynomial,result.estimated_value))
51                 x=Symbol('x')
52                 plot(result.polynomial,(x,-400,400),xlim
    =[-400,400],ylim=[-400,400],title='Wielomian: {} \nWarto
    wielomianu dla {}: {}\n'.format(result.polynomial,p,result.
    estimated_value))
53             else:
54                 sg.PopupError('Podaj liczb  w zakresie od 1 do
    2048.')
55         else:
56             sg.PopupError('"{}" to nie liczba.' .format(values['
    key1']))
57
58 #Obliczenie przyblizenia dla podanej przez uzytkownika zmiennej dla
     wielomianu lagrange stworzonego na podstawie wycinku podanego
    przez uzytkownika
59     if event == 'Sprawd ':
60         if(values['key1'].isdigit()):
61             p=int(values['key1'])
62             if(p>=1 and p<=2048):
63                 lista = [ int(key) for key,value in values.items()
    if value and key!='key1']
64                 if(len(lista)>1):
65                     wielomian=lagrange.create_polynomial(lista)
66                     estymacja=Estimation(lista,wielomian,p)
67                     wartosc=estymacja.estimated_value
68                     print('Dany podzbi r: x={} Przybli ona
    warto   log2 z {}: {} B   d przybli enia: {:.6f}'.format(
    lista,p,wartosc,math.log(p,2)-wartosc))
69                     x=Symbol('x')
70
71                     plot(estymacja.polynomial,(x,-100,100),xlim
```

```
    =[-100,100],ylim=[-100,100],title='Wielomian: {} \nWarto
    wielomianu dla {}: {}\n'.format(estymacja.polynomial,p,wartosc)
    )
72                 else:
73                     sg.PopupError('Za ma a liczba w  z   w ,
    zaznacz conajmniej 2 w  z  y.')
74             else:
75                 sg.PopupError('Podaj liczb  w zakresie od 1 do
    2048.')
76         else:
77             sg.PopupError('"{}" to nie liczba.' .format(values['
    key1']))
78
79
80
81 #Zamkniecie okna jesli uzytkownik kliknie przycisk X
82     if event == sg.WIN_CLOSED or event == 'Cancel':
83         break
84 window.close()
```