Перший приклад програми з коментарями майже жо кожного рядка
Усі дані вже задані у програмі, тож вона лише виводить результат

```
def only_lists(fnc):
  def _only_lists(*args, **kwargs):
    '''
    Маємо дві матриці, які можуть бути задані словником. Якщо матриця задана словником, її розміри точно невідомі (у ній може бути заданий лише один елемент, наприклад.
    Тому треба знайти розміри матриць, щоб можна було знайти їх суму
    '''
    m = 0 #к-ть рядків
    n = 0 #к-ть стовпчиків
    for arg in args:
      if type(arg) == list: #Якщо матриця задана списком списків, то її розміри дорівнюють розмірам матриці-суми
        m = len(arg)
        n = len(arg[0])
      elif type(arg) == dict: #Можливо таке, що обидві матриці будуть задані словниками. Тому ми знайдемо, який елемент має найбільші координати і будемо вважати,
        for key in arg.keys(): #що він є крайнім елементом - тобто, ми знаходимо матрицю з найменшою шириною та довжиною, яка включає у себе усі елементи, що
          if key[0] + 1 > m: #входять до матриці, заданої словником
            m = key[0] + 1
          if key[1] + 1 > n: #(key[0] і key[1] - довжина та ширина матриці відповідно. +1 тут потрібно через те, що відлік починається з 0, а не з 1
            n = key[1] + 1
    new_args = [] #Визначаємо нові параметри
    for arg in args: #для кожного вхідного параметра:
      if type(arg) == dict: #Якщо це був словник, то
        new_matr = [] #створюємо порожню матрицю
        for i in range(m): #з "m" рядками (це число ми знайшли раніше)
          new_empty_line = [0]*n #та "n" стовпчиками (це число ми також знайшли раніше)
          new_matr.append(new_empty_line)
        for key in arg: #Проходимося по усім ненульовим елементам розрідженої матриці
          m_ = key[0] #визначаємо координати цього елемента
          n_ = key[1]
          new_matr[m_][n_] = arg.get(key) #додаємо значення елемента до створеної раніше порожньої матриці
        new_args.append(new_matr) #Додаємо цю матрицю до списку параметрів
      else: #Якщо ж матриця вже була задана як список списків, не змінюємо її, а
        new_args.append(arg) #просто додаємо до списку параметрів
    res = fnc(*tuple(new_args), **kwargs) #викликаємо функцію для нових значень параметрів. *tuple(new_args) - перетворення списку аргументів на кортеж
    return res
  return _only_lists

@ only_lists
def sum_matrix(m_1, m_2): #функція, що знаходить суму двох матриць
  m_res = m_1 #Тут не розглядається випадок, коли матриці різних розмірів. Цього не вимагали в умові, тому навіщо все ускладнювати?
  for i in range(len(m_1)):
    for j in range(len(m_1[i])):
      m_res[i][j] = m_1[i][j] + m_2[i][j] #кожен елемент матриці-результату буде рівним сумі відповідних елементів початкових матриць
  print('Відповідь:') #Виводиться відповідь. В умові не вимагалося зробити так, щоб функція щось повертала, тому хай краще виводить митрицю на екран
  for i in range(len(m_res)):
    print(m_res[i])

print('Дві матриці як списки списків')
m_1 = [[1,1,1],[1,1,1],[1,1,1]]
m_2 = [[0,1,2],[3,4,5],[6,7,8]]
```

```
print(m_1)
print(m_2)
sum_matrix(m_1, m_2)
print('Дві матриці як словники')
m_1 = {(1,1):2, (2,2):1, (0,0):5}
m_2 = {(1,1):1, (2,1):3, (0,2):2}
print(m_1)
print(m_2)
sum_matrix(m_1, m_2)
print('Одна матриця як список списків, інша - як словник')
m_1 = [[1,1,1],[1,1,1],[1,1,1]]
m_2 = {(0,1):1, (2,1):3, (0,2):2}
print(m_1)
print(m_2)
sum_matrix(m_1, m_2)

**********
***Другий приклад програми, яка створює просте вікно***
***Користувач вводить послідовності літер, розділені пробілом, а програма сортує їх за алфавітом***

from tkinter import *
class SortWithoutReg:
    def __init__(self, line):
        self._index = 0
        self.list_of_words = line.split()
        self.list_in_correct_order = self.build_list()
        self.lenght = len(self.list_in_correct_order)

    def is_lower_word(self, word, wordslist):
        word_edited = word.lower()
        checker = True
        for i in wordslist:
            if word_edited < i.lower():
                checker = False
        return checker

    def find_lower_word(self, wordslist):
        find = ''
        for i in wordslist:
            if self.is_lower_word(i, wordslist) == True:
                find = i
        return find

    def build_list(self):
        wordslist = self.list_of_words
        l = len(wordslist)
        new_list = []
        for i in range(l):
            next_word = self.find_lower_word(wordslist)
            wordslist.remove(next_word)
            new_list.append(next_word)
        new_list.reverse()
        return new_list

    def __iter__(self):
        return self

    def __next__(self):
        if self._index >= self.lenght:
            raise StopIteration
```

```
        self._index += 1
        return self.list_in_correct_order[self._index - 1]

def find(strn):
    new_iter = SortWithoutReg(strn)
    new_str = ''
    while True:
        try:
            new_str += new_iter.__next__()
            new_str += ' '
        except StopIteration:
            break
    return new_str

def calc(ev = None):
    n = str(ein.get())                  # отримання значення поля введення
    f = find(n)
    rezult = 'Результат: {}'.format(f)     # побудова рядка для відображення
    lrez.configure(text = rezult)          # зміна надпису значенням результату

def onEscapeKey(event):
    top.destroy()


top = Tk()                              # створення вікна

finput = Frame(top)                        # контейнер для надпису та поля введення
finput.pack(fill=X, expand=YES)
Label(finput, text = 'Введіть рядок: ',
    font=('arial', 16)).pack(side=LEFT)    # створення надпису
                                # та додавання надпису до вікна
ein = Entry(finput, font=('arial', 16))    # створення поля введення
ein.pack(side=LEFT, fill=X, expand=1)      # додавання поля введення до вікна
ein.focus()                             # встановлення "фокусу"
                                # у поле введення

frez = Frame(top)                          # контейнер для надпису результату
frez.pack(fill=X, expand=YES)
lrez = Label(frez,
            text='Відповідь: ___',
            font=('arial', 16))          # створення надпису
lrez.pack(side=LEFT, fill=X)               # додавання надпису до вікна

fbut = Frame(top)                          # контейнер для кнопок
fbut.pack(side=LEFT, fill=X, expand='1')
bcalc = Button(fbut, text = 'Знайти',
        command = calc,
        font=('arial', 16))
bcalc.pack(side=LEFT, padx=5, pady=5)      # кнопка "Обчислити"
bquit = Button(fbut, text='Закрити',
        command=top.destroy,
        font=('arial', 16))
bquit.pack(side=RIGHT, padx=5, pady=5)     # кнопка "Закрити"

top.bind('<Return>', calc)                 # зв'язування з натисненням клавіш
top.bind('<Escape>', onEscapeKey)

top.mainloop()
```