ГУАП  
КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | А.В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

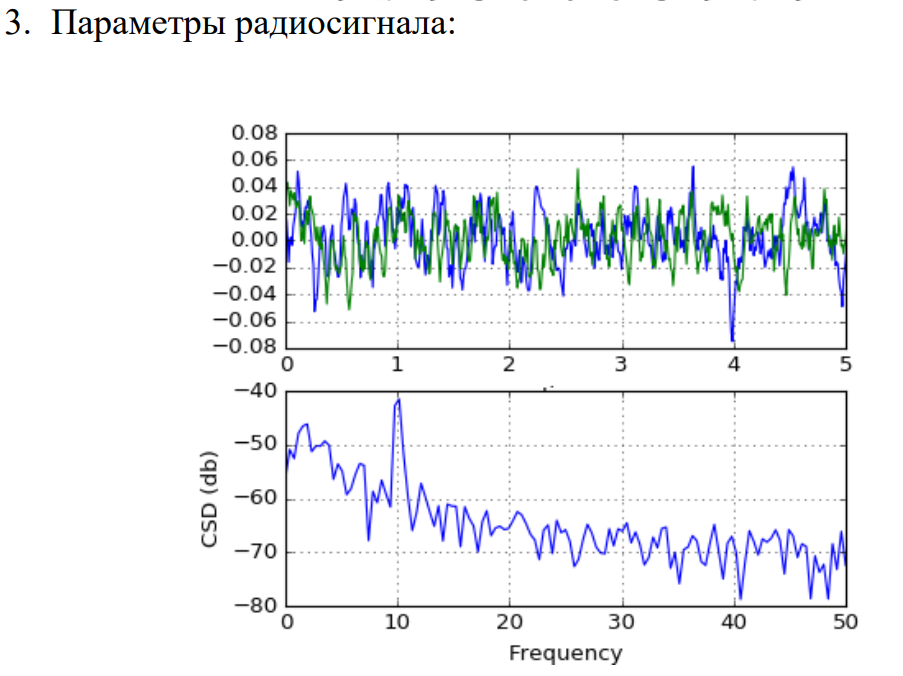
|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| ПОСТРОЕНИЕ ГРАФИКОВ |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М023 |  |  |  | Д.А. Трегуб |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

ПОСТАНОВКА ЗАДАЧИ

Задание № 1  
Постройте график используя библиотеку графического вывода matplotlib и библиотеку математических функций numpy:

Требования к программе:  
Цель программы: вывод окна с двумя графиками или сообщения об ошибке. На вход подаётся 5 параметров: амплитуда, начальная фаза, длительность сигнала, нижняя граница диапазона частоты, верхняя граница диапазона частоты. Выходными данными является окно с графиками а также сообщение "Успешное построение графика" или сообщение по ошибке.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Амплитуда радиосигнала в конкретный момент времени рассчитывается по формуле U(t)=U\*cos(ωt-φ), где U – максимальная амплитуда, измеряемая в вольтах, ω – циклическая частота в рад/с, φ – начальная фаза, t – количество секунд с начала передачи сигнала. Частоты радиосигналов находятся на промежутке [2π\*0.3; 2π \*3\*1012] (рад/c). Время не может быть <= 0.

ОПИСАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ

Таблица 1 - Описание переменных

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип переменной | Назначение |
| Amp | float | Хранение амплитуды |
| phase | float | Хранение начальной фазы |
| time | int | Хранение длительности сигнала |
| freq1 | int | Хранение нижний границы диапазона частот |
| freq2 | int | Хранение верхней границы диапазона частот |
| key | str | Ключ для продолжения работы программы |
| result | str | Хранение сообщения о завершение функции |
| fig | figure | Контейнер для хранения Axes |
| ax | Axes | Лист с индивидуальными графиками |
| Uarr | Numpy array | Массив numpy для хранения значений амплитуды |
| TimeArr | Numpy array | Массив numpy для хранения значений длительности сигнала |
| TestArr | List | Список для проверки соответствия типам |

На вход подаётся строка 5 параметров – Amp, phase, time, freq1, freq2.

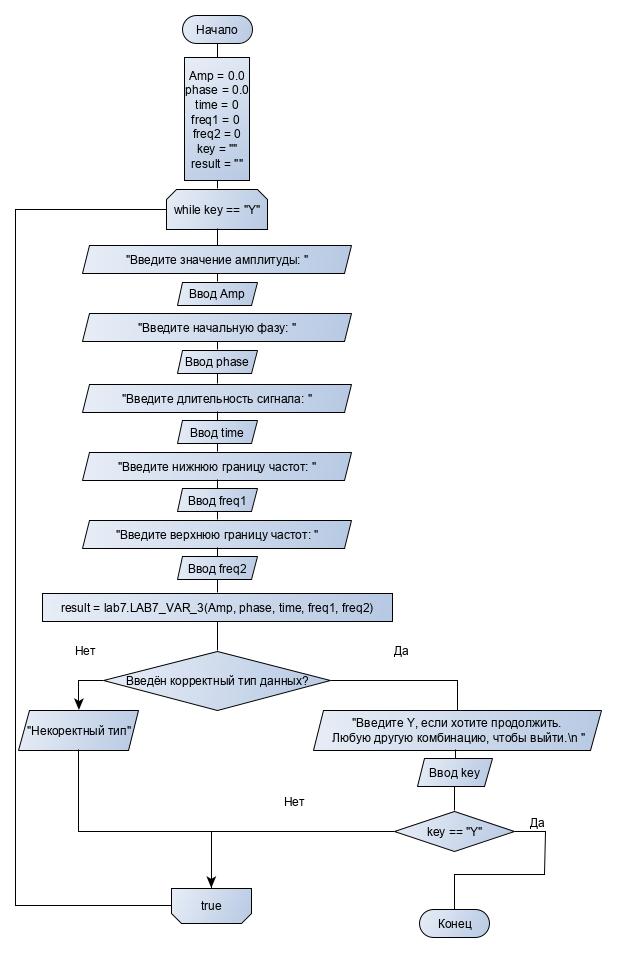


Рисунок 1 – функция main

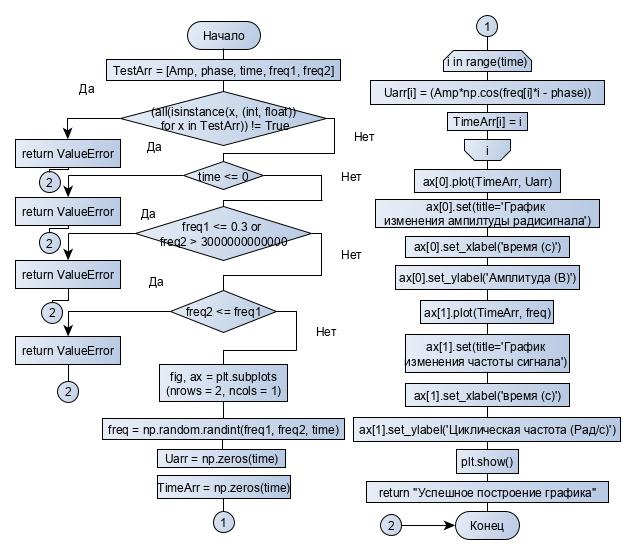


Рисунок 2 - функция LAB7\_VAR\_3

ОПИСАНИЕ ТЕСТОВОГО НАБОРА

Таблица 2 - Тестовый набор

|  |  |  |  |
| --- | --- | --- | --- |
| Описание тестового случая | Входные данные | Ожидаемый результат | Результат теста |
| Передан удовлетворяющий всем условиям набор данных | (87, 34, 300, 35, 84) | "Успешное построение графика" | OK |
| Переданы строковые данные | ("one", "two", 3, 4, 5) | ValueError | OK |
| Переданы данные, с типом list, состоящие из одного элемента | ([1], [2], 3, 4, 5) | ValueError | OK |
| Переданы данные, с типом list, один из которых состоит из двух элементов | ([1], [2, 0], 3, 4, 5) | ValueError | OK |
| Преданы данные с отрицательным значением времени | (87, 34, -1, 35, 84) | ValueError | OK |
| Переданы данные, где нарушено ограниченнее частотного диапазона | (87, 34, 500, 0.1, 84) | ValueError | OK |
| Переданы данные, где нижняя граница частотного диапазона выше, чем верхняя | (87, 34, 300, 84, 35) | ValueError | OK |

ПРИМЕР РАБОТЫ ПРОГРАММЫ

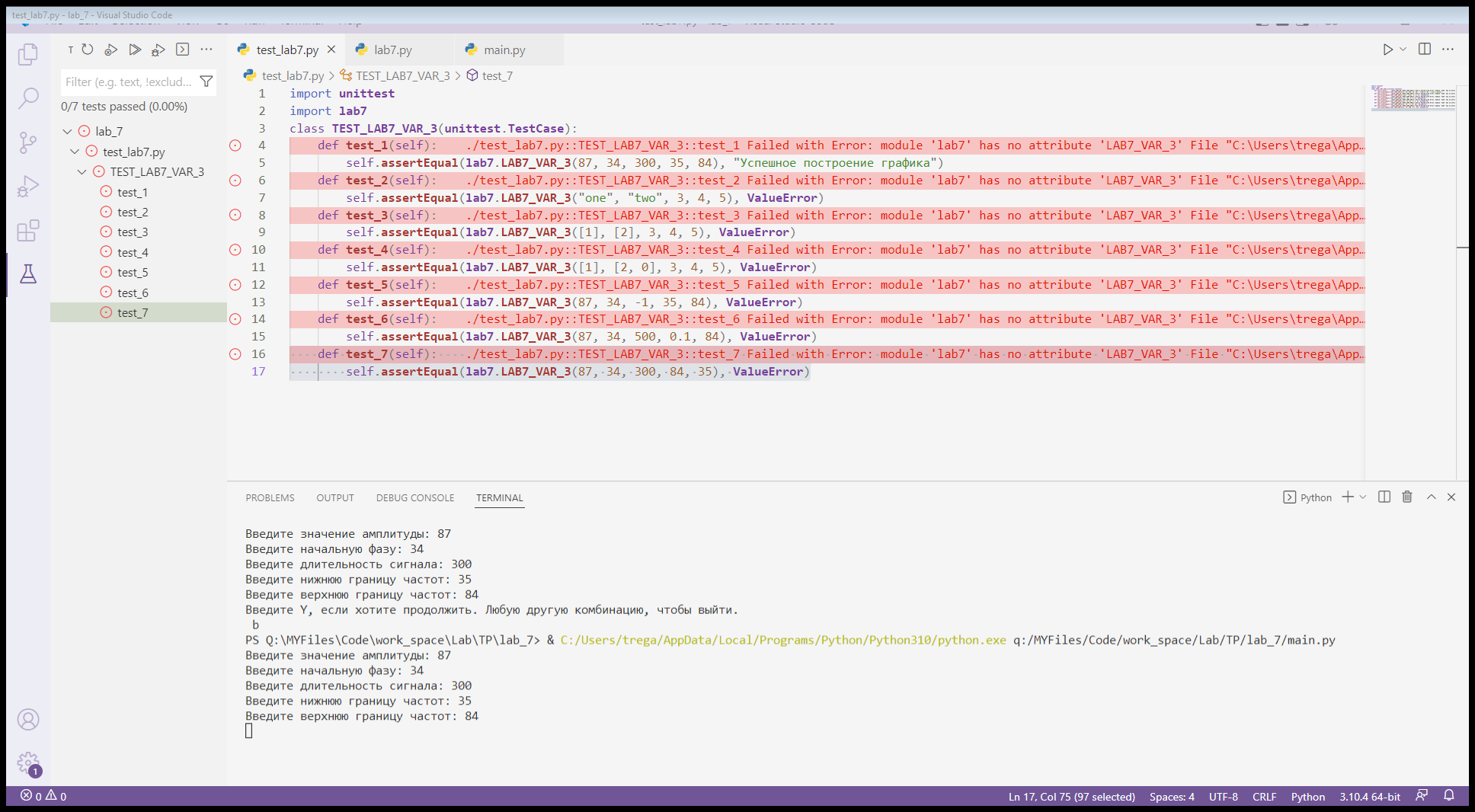


Рисунок 3 - Результат работы тестов до написания программы

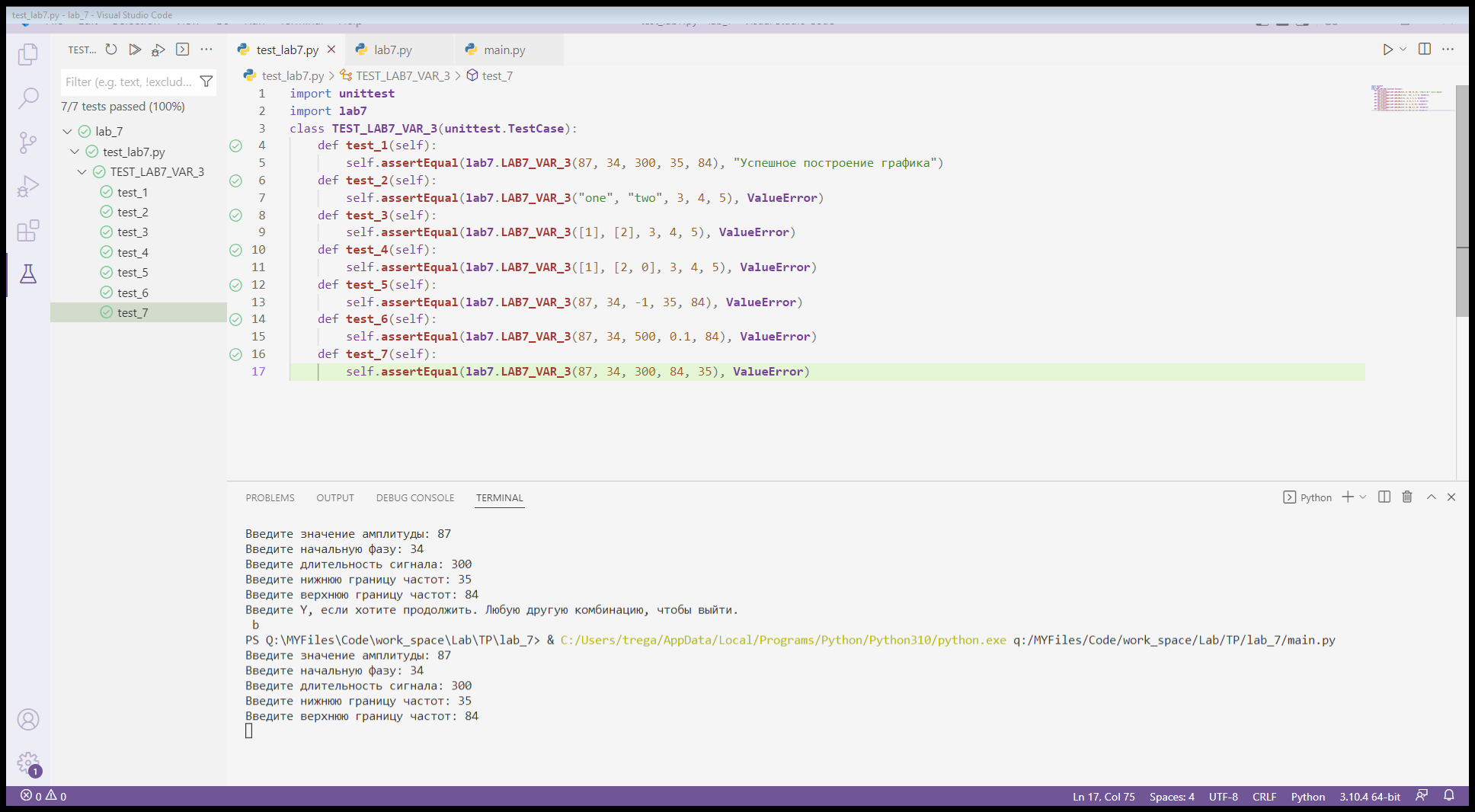
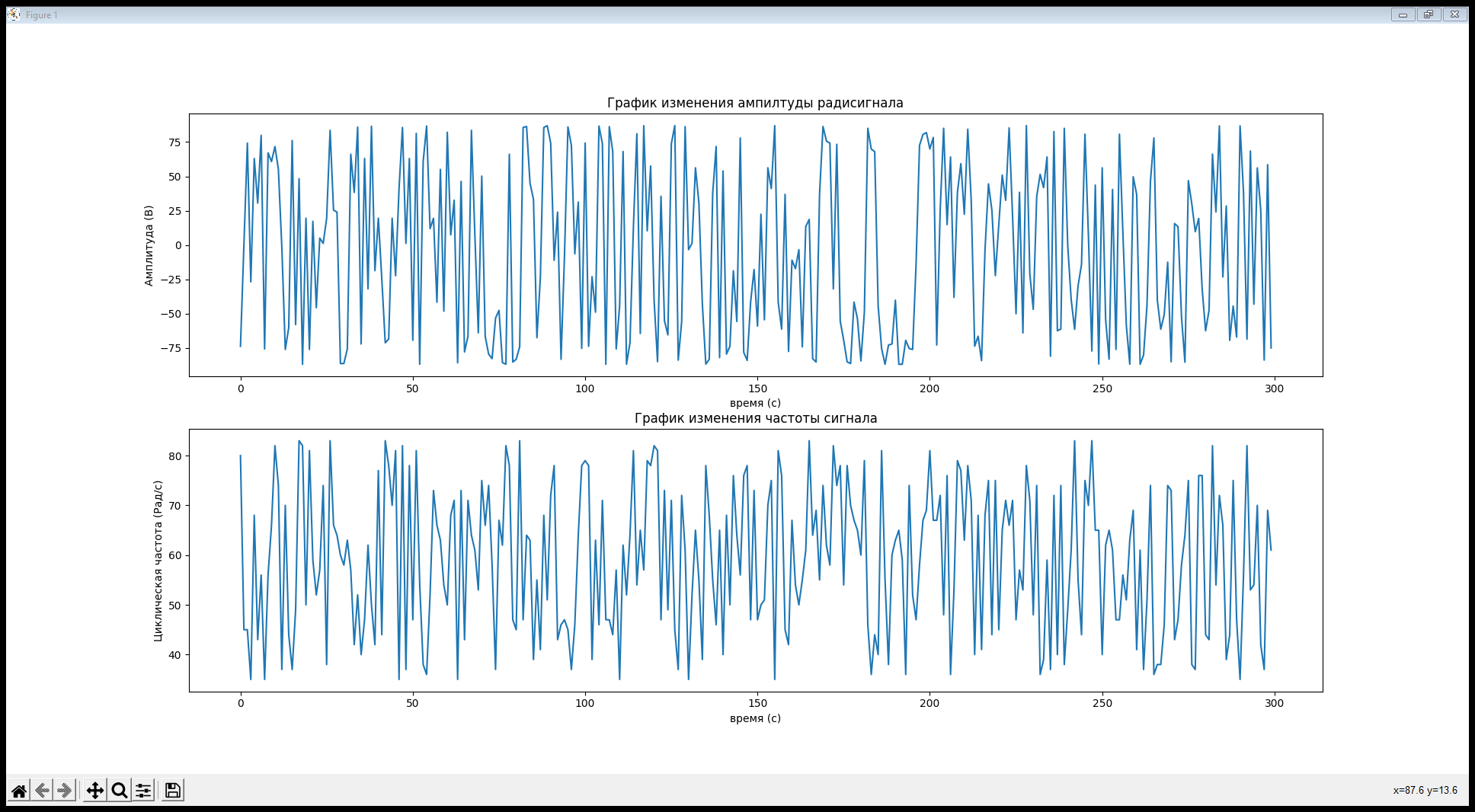


Рисунок 4 - Результат работы тестов после написания программы



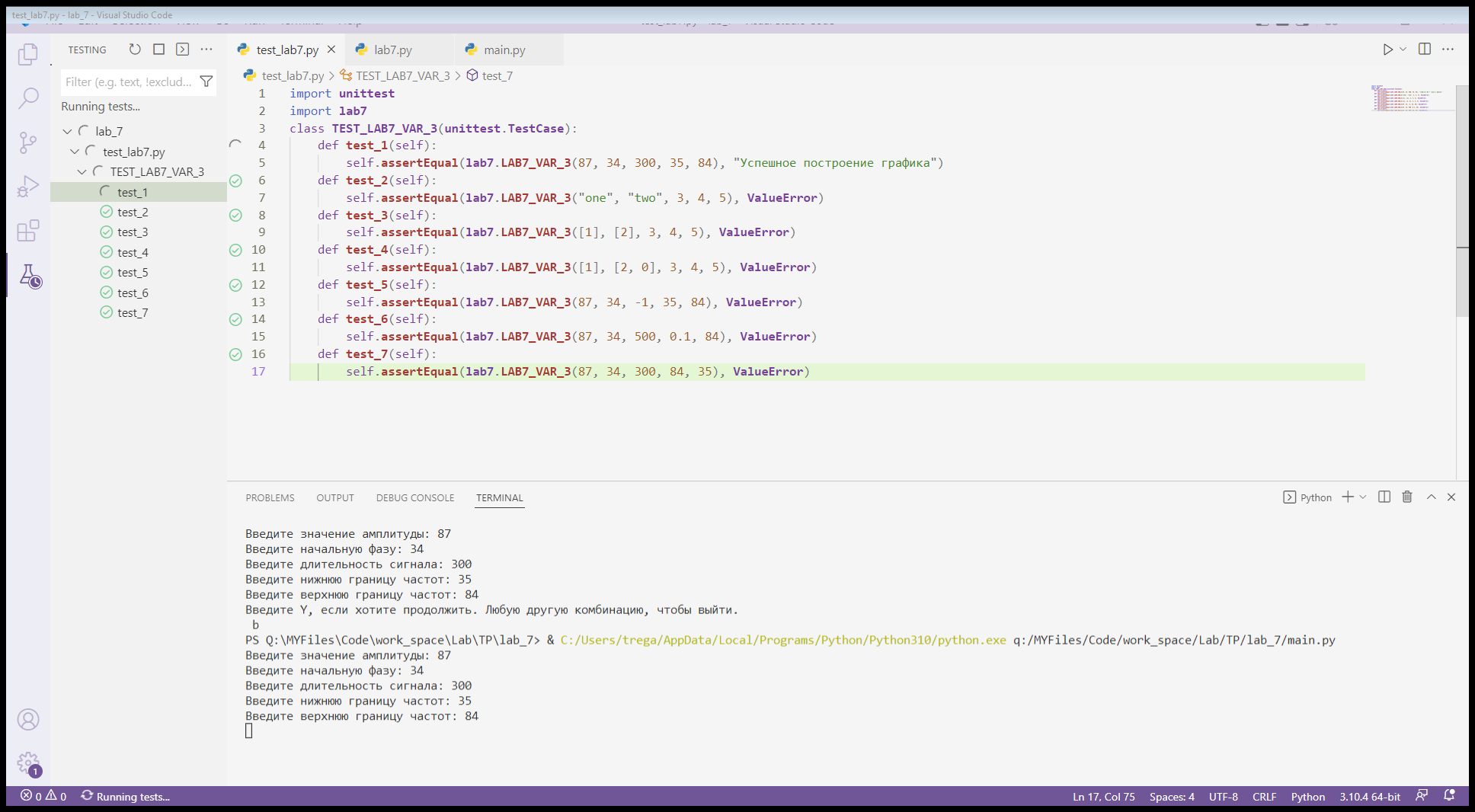


Рисунок 5 - Результат работы программы

ВЫВОД

Были улучшены навыки работы с python, получены навыки работы с библиотеками matplotlib.pyplot и numpy, была выполнена поставленная задача. Составленная программа была успешно протестирована.

ПРИЛОЖЕНИЯ

Приложение 1

Код программы

Код тестов:

import **unittest**

import **lab7**

class **TEST\_LAB7\_VAR\_3**(**unittest**.**TestCase**):

    def **test\_1**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**(87, 34, 300, 35, 84), "Успешное построение графика")

    def **test\_2**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**("one", "two", 3, 4, 5), **ValueError**)

    def **test\_3**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**([1], [2], 3, 4, 5), **ValueError**)

    def **test\_4**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**([1], [2, 0], 3, 4, 5), **ValueError**)

    def **test\_5**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**(87, 34, -1, 35, 84), **ValueError**)

    def **test\_6**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**(87, 34, 500, 0.1, 84), **ValueError**)

    def **test\_7**(self):

        self.**assertEqual**(**lab7**.**LAB7\_VAR\_3**(87, 34, 300, 84, 35), **ValueError**)

Код функции LAB7\_VAR\_3:

*#U(t)=U\*cos(ωt-φ). - формула амплитуды*

from **sqlite3** import Time

import **numpy** as **np**

import **matplotlib**.**pyplot** as **plt**

def **LAB7\_VAR\_3**(Amp, phase, time, freq1, freq2):

*#Блок обработки ошибок*

    TestArr = [Amp, phase, time, freq1, freq2] *#Список для проверки соответствия типам*

    if (**all**(**isinstance**(x, (**int**, **float**)) for x in TestArr)) != True: *#Проверка соответствия типам*

        return **ValueError**

    elif time <= 0: *#Проверка условия времени для построения графика*

        return **ValueError**

    elif freq1 <= 0.3 or freq2 > 3000000000000: *#Проверка условия диапазона частот*

        return **ValueError**

    elif freq2 <= freq1: *#Проверка логичности диапазона частот*

        return **ValueError**

    fig, ax = **plt**.**subplots**(nrows = 2, ncols = 1) *#Создание figure и axes для построения графиков*

    freq = **np**.**random**.randint(freq1, freq2, time)  *#  Частота*

    Uarr = **np**.**zeros**(time)   *#Создание массива для записи значений амплитуды*

    TimeArr = **np**.**zeros**(time)    *#Создание массива для отсчёта времени*

    for i in **range**(time):   *#цикл по заполнению выше созданных массивов*

        Uarr[i] = (Amp\***np**.cos(freq[i]\*i - phase))   *#Заполнение массива с амплитудой*

        TimeArr[i] = i  *#Заполнение массива со временем*

    ax[0].plot(TimeArr, Uarr)  *#Отображение данных на графике*

    ax[0].set(title='График изменения ампилтуды радисигнала')    *#Размещение заголовка над "Axes"*

    ax[0].set\_xlabel('время (с)')   *#подпись оси x*

    ax[0].set\_ylabel('Амплитуда (В)')   *#подпись оси y*

    ax[1].plot(TimeArr, freq)   *#Отображение данных на графике*

    ax[1].set(title='График изменения частоты сигнала')     *#Размещение заголовка над "Axes"*

    ax[1].set\_xlabel('время (с)')   *#подпись оси x*

    ax[1].set\_ylabel('Циклическая частота (Рад/с)')    *#подпись оси y*

**plt**.**show**()  *#Отрисовка окна*

    return "Успешное построение графика"

Код функции main:

import **lab7**

*#Инициализация переменных*

Amp = 0.0

phase = 0.0

time = 0

freq1 = 0

freq2 = 0

key = ""

result = ""

*#Начало бесконечного цикла*

while True:

    try:

        Amp = **float**(**input**("Введите значение амплитуды: "))    *#Ввод амплитуды*

        phase = **float**(**input**("Введите начальную фазу: "))      *#Ввод фазы*

        time = **int**(**input**("Введите длительность сигнала: "))   *#Ввод длительности*

        freq1 = **int**(**input**("Введите нижнюю границу частот: ")) *#ВВод нижней границы*

        freq2 = **int**(**input**("Введите верхнюю границу частот: "))*#Ввод верхней границы*

        result = **lab7**.**LAB7\_VAR\_3**(Amp, phase, time, freq1, freq2)       *#Вызов функции*

    except **ValueError**:

**print**("Ошибка, попробуйте ещё раз.")

    key = **str**(**input**("Введите Y, если хотите продолжить. Любую другую комбинацию, чтобы выйти.\n "))*#Ввод комбинации для продолжения или завершения программы*

    if(key == "Y"):

        continue *#Продолжение программы*

    else:

            break *#Завершение программы*

*#Конец бесконечного цикла*