НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»
ИНЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

Кафедра безопасности и информационных технологий

Направление подготовки бакалавриата
10.03.01 «Информационная безопасность»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ
Тема «Проект на языке Python»
Вариант 7

Выполнил студент группы ИЭэ-49-21 Леонтьева А.С.
Проверил Мишин А.А.
Оценка:_____ дата_____

Москва 2022 г.

Оглавление

Цели работы (Задание)

Цель лабораторной работы заключается в закреплении навыков, приобретенных на протяжении изучения языка программирования Python.

В своем проекте нужно было показать свои знания, умения. Написать программу, которая была бы небольшой по объему, но выполняла все свои поставленные в условии задачи.

Анализ

В настоящее время широко распространены цифровые запоминающие осциллографы. Учитывая широкие возможности обработки и анализа электрических сигналов с помощью определенных программ, использование такого осциллографа может стать весьма полезным при проведении эксперимента. Но не всегда мы можем воспользоваться осциллографом в определенный момент времени.

Например, нам нужно выполнить лабораторную работу по физике или же представить, как будет выводить осциллограф тот или иной сигнал. На помощь приходят знания Языков Программирования (ЯП) и определенных библиотек, которые помогают в разы облегчить жизнь как начинающему пользователю, так и профессиональному программисту.

Ключевое задание, которое я поставила себе за цель – написать код, в котором будет осуществляться выбор функции, ввод определенных значений, вывод на экран анимированного изображения сигнала.

Таблица 1

Ожидаемый результат при ожидаемом действии

Ожидаемое действие	*Ожидаемый результат*
Запуск программы	Выбор сигнала, ввод интервала и частоты сигнала.
Функция «sin» + Функция «animate_sin»	Выводит на экран анимированный сигнал синуса (синусоида) в определенном интервале с определенной частотой.
Функция «square» + Функция «animate_square»	Выводит на экран анимированный прямоугольный сигнал в определенном интервале с определенной частотой.
Функция «triangle» + Функция «animate_triangle»	Выводит на экран треугольный анимированный сигнал в определенном интервале с определенной частотой.
Функция «sawtooth» + Функция «animate_sawtooth»	Выводит на экран пилообразный анимированный сигнал в определенном интервале с определенной частотой.

Разработка
Структура программы

Схема 1

Спецификация модулей

Таблица 2

Спецификация модулей

Имя модуля	Имя вызываемого модуля	Назначение	Входные данные	Выходные данные	Особенности
Модуль argparse	parser = argparse.ArgumentParser	Позволяет разбирать аргументы, передаваемые скрипту при его запуске из командной строки, и даёт возможность пользоваться этими аргументами в скрипте.	-	-	-
Модуль signal	signal.(тип), например, signal.sawtooth	Предоставляет механизмы для использования обработчиков сигналов в Python.	-	-	-
Модуль matplotlib Animation	anim = FuncAnimation (fig, animate_sin, init_func=init, frames=200, interval=20, blit=True)	В данном модуле содержатся два класса, один из которых FuncAnimation, позволяющий транслировать на экран пользователя анимированные графики функций.	-	-	-

Схема алгоритмов

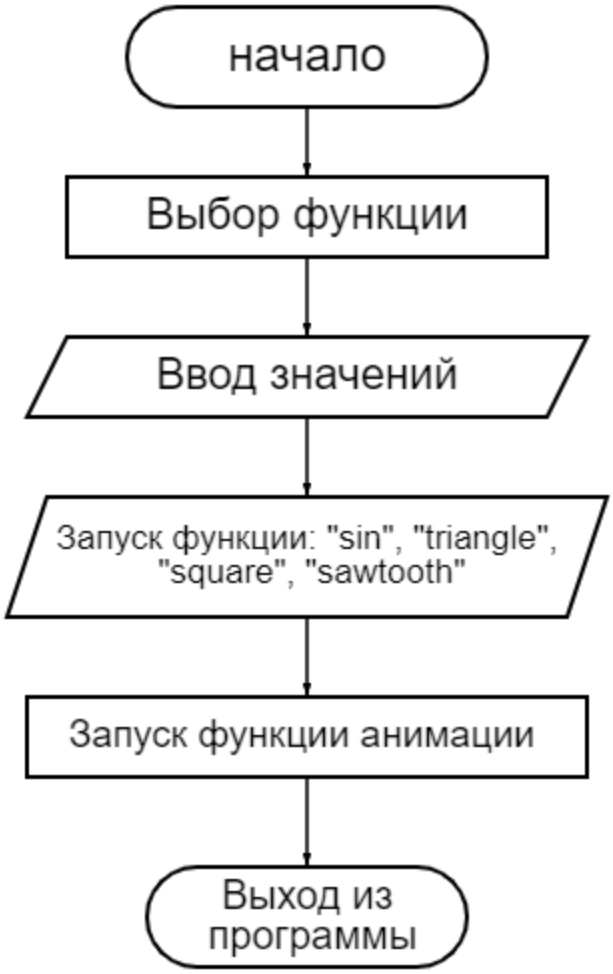

Рис. 1. Блок – схема программы №1 «Частота символов»

Интерфейс

Интерфейс командной строки (консоль) — разновидность текстового интерфейса между человеком и компьютером, в котором инструкции компьютеру даются в основном путём ввода с клавиатуры команд, а вывод производится на экран компьютера.

В ниже представленном коде пользователь выбирает в конфигурациях, что запустить (смотреть рис.1), запускает нужный сигнал, вводит значения и на выходе получает интерфейс-осциллограф с анимированным графиком.

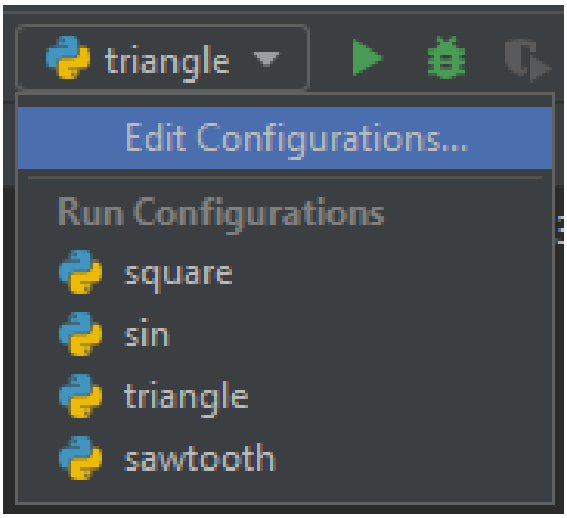

Рис. 1. Меню конфигураций

Реализация

Код программы №1 с комментариями.

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import argparse

#ввод интервала и частоты функции
z,n,m  =  map(int,  input().split())      #z  -  левая  часть
интеравала;
                                    #n    -    правая    часть
интервала;
                                    #m - частота функции

fig = plt.figure()
ax = plt.axes(xlim=(0, 4), ylim=(-2, 2))
line, = ax.plot([], [], lw=3)

#функция, с которой происходит сравнение
def init():
    line.set_data([], [])
    return line,

#функция для анимации синусоидального сигнала
def animate_sin(i):
    x = np.linspace(z, n, m)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

#функция для анимации прямоугольного сигнала
def animate_square(i):
    x = np.linspace(z, n, m)
    y = signal.square(2 * np.pi * 5 * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

#функция для анимации треугольного сигнала
def animate_triangle(i):
    x = np.linspace(z, n, m)
    y = signal.sawtooth(2 * np.pi * 5 * (x - 0.01 * i), 0.5)
    line.set_data(x, y)
    return line,
```

```
#функция для анимации пилообразного сигнала
def animate_sawtooth(i):
    x = np.linspace(z, n, m)
    y = signal.sawtooth(2 * np.pi * 5 * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

#функция, которая позволила разбить все сигналы на параметры, запускаемые по отдельности
def main():
    parser = argparse.ArgumentParser(description='Тут описание программы')
    parser.add_argument('--function_type',
                        choices=['square', 'sin','triangle', 'sawtooth'],
                        default='sin',
                        help='Signal type')
    parser.add_argument( # example
        '--my_optional',
        type=int,
        default=2,
        help='provide an integer (default: 2)'
    )
    my_namespace = parser.parse_args()
    print(my_namespace.function_type)

    if my_namespace.function_type == 'sin':
        anim = FuncAnimation(fig, animate_sin, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60) # надо установить imagemagic https://imagemagick.org/script/download.php#windows
    elif my_namespace.function_type == 'square':
        anim = FuncAnimation(fig, animate_square, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('square.gif', writer='imagemagick', fps=60) # надо установить imagemagic https://imagemagick.org/script/download.php#windows
    elif my_namespace.function_type == 'triangle':
        anim = FuncAnimation(fig,animate_triangle, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60)
    elif my_namespace.function_type == 'sawtooth':
```

```
        anim    =    FuncAnimation(fig,    animate_sawtooth,
init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60)
    plt.show()

if __name__ == '__main__':
    main()
```

main.py client.py server.py

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import argparse

#ввод интервала и частоты функции
z,n,m = map(int, input().split())   #z - левая часть интеравала;
                                    #n - правая часть интервала;
                                    #m - частота функции

fig = plt.figure()
ax = plt.axes(xlim=(0, 4), ylim=(-2, 2))
line, = ax.plot([], [], lw=3)

#функция, с которой происходит сравнение
def init():
    line.set_data([], [])
    return line,

#функция для анимации синусоидального сигнала
def animate_sin(i):
    x = np.linspace(z, n, m)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return line,

#функция для анимации прямоугольного сигнала
def animate_square(i):
    x = np.linspace(z, n, m)
    y = signal.square(2 * np.pi * 5 * (x - 0.01 * i))
    line.set_data(x, y)
```

Рис. 2. Код программы в картинках (1)

```
main.py  client.py  server.py
        line.set_data(x, y)
        return line,

    #функция для анимации треугольного сигнала
    def animate_triangle(i):
        x = np.linspace(z, n, m)
        y = signal.sawtooth(2 * np.pi * 5 * (x - 0.01 * i), 0.5)
        line.set_data(x, y)
        return line,

    #функция для анимации пилообразного сигнала
    def animate_sawtooth(i):
        x = np.linspace(z, n, m)
        y = signal.sawtooth(2 * np.pi * 5 * (x - 0.01 * i))
        line.set_data(x, y)
        return line,

    #функция, которая позволила разбить все сигналы на параметры, запускаемые по отдельности
    def main():
        parser = argparse.ArgumentParser(description='Тут описание программы')
        parser.add_argument('--function_type',
                            choices=['square', 'sin','triangle', 'sawtooth'],
                            default='sin',
                            help='Signal type')
        parser.add_argument( # example
            '--my_optional',
            type=int,
            default=2,
            help='provide an integer (default: 2)'
        )
        my_namespace = parser.parse_args()
        print(my_namespace.function_type)
```

Рис. 3. Код программы в картинках (2)

```
    my_namespace = parser.parse_args()
    print(my_namespace.function_type)

    if my_namespace.function_type == 'sin':
        anim = FuncAnimation(fig, animate_sin, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60) # надо установить imagemagic https://imagema
    elif my_namespace.function_type == 'square':
        anim = FuncAnimation(fig, animate_square, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('square.gif', writer='imagemagick', fps=60) # надо установить imagemagic https://imag
    elif my_namespace.function_type == 'triangle':
        anim = FuncAnimation(fig,animate_triangle, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60)
    elif my_namespace.function_type == 'sawtooth':
        anim = FuncAnimation(fig, animate_sawtooth, init_func=init, frames=200, interval=20, blit=True)
        # anim.save('sin.gif', writer='imagemagick', fps=60)
    plt.show()


if __name__ == '__main__':
    main()
```

Рис. 4. Код программы в картинках (3)

Тестирование

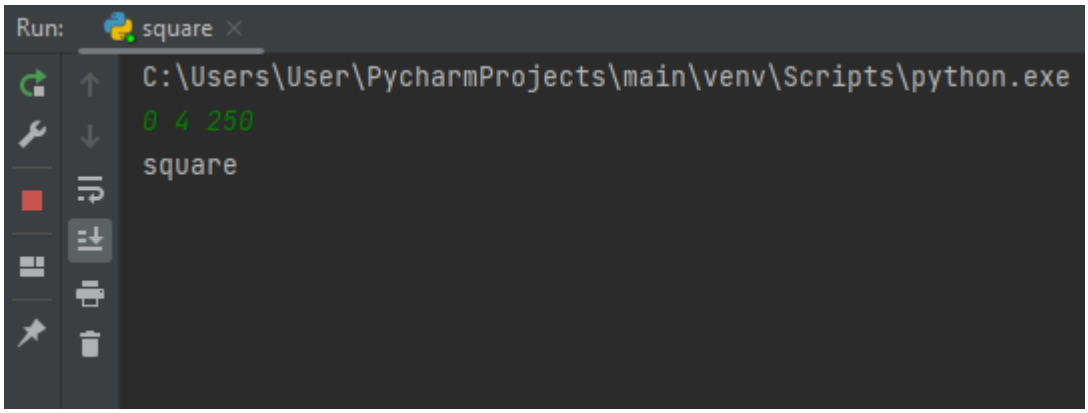

Рис. 5. Тестирование программы: прямоугольный сигнал – ввод границ интервала и частоты в одну строчку

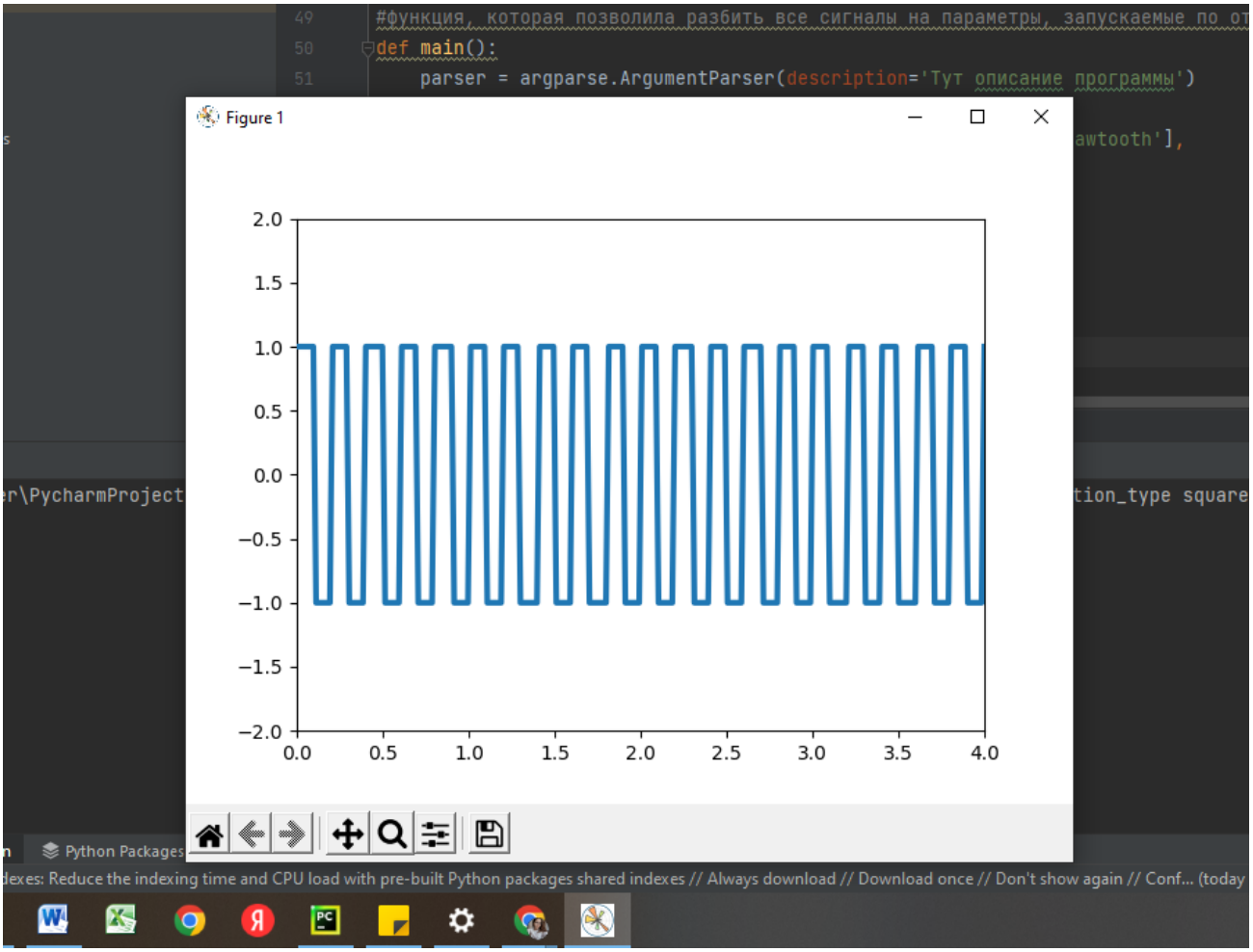

Рис. 6. Тестирование программы: показ анимированного прямоугольного сигнала

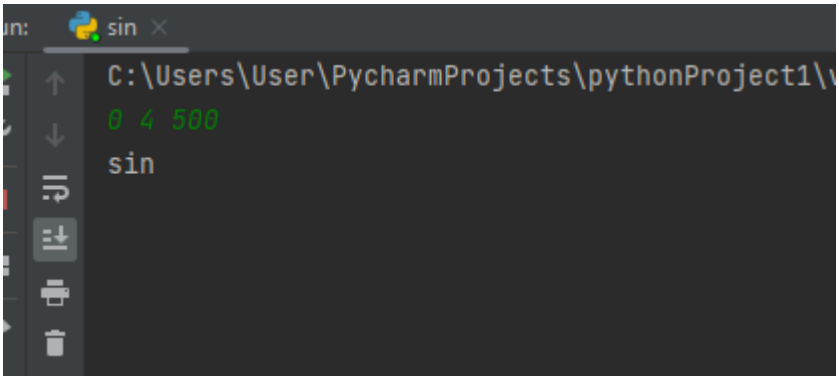

Рис. 7. Тестирование программы: синусоидальный сигнал – ввод границ интервала и частоты в одну строчку

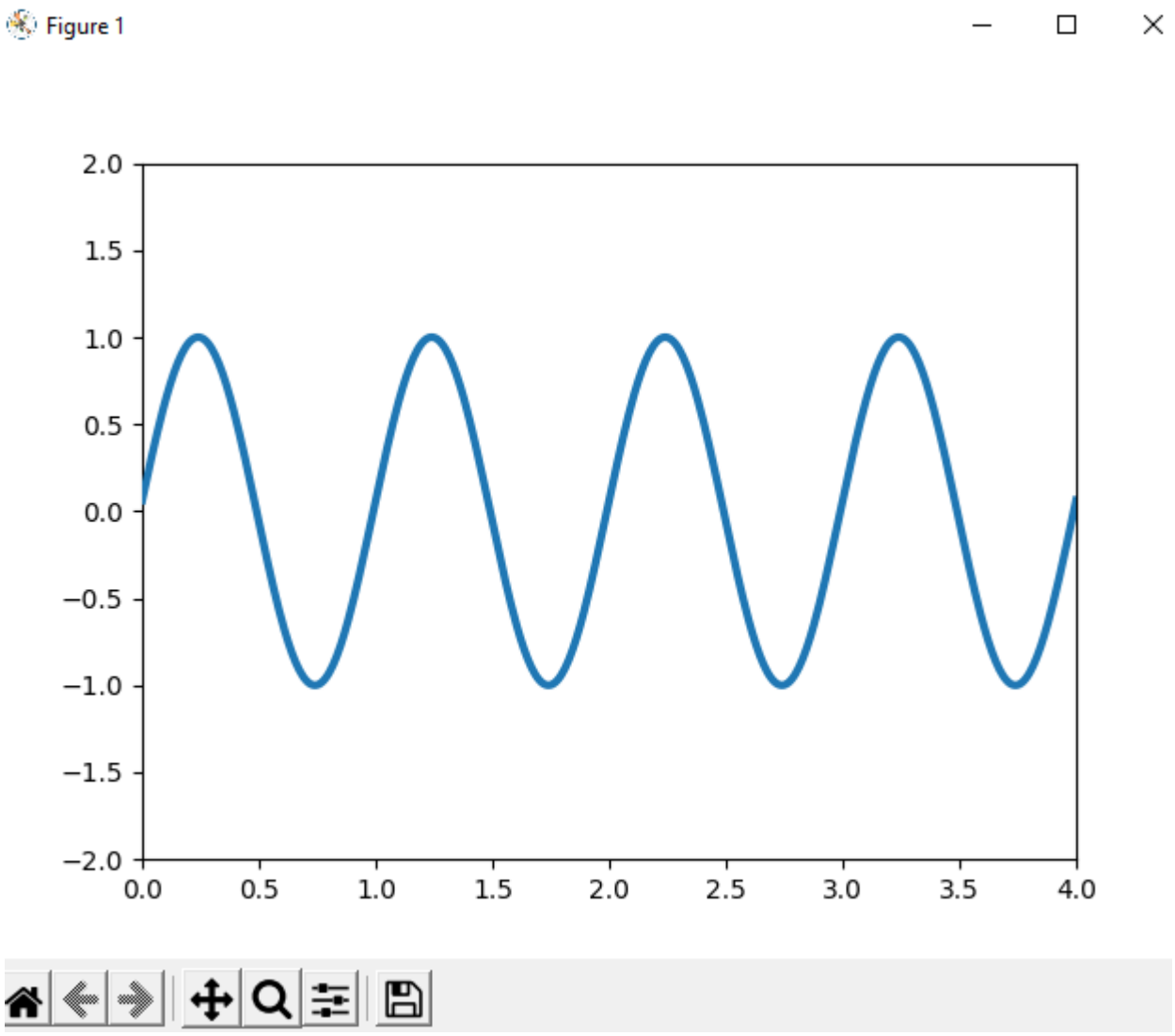

Рис. 8. Тестирование программы: показ анимированного синусоидального сигнала

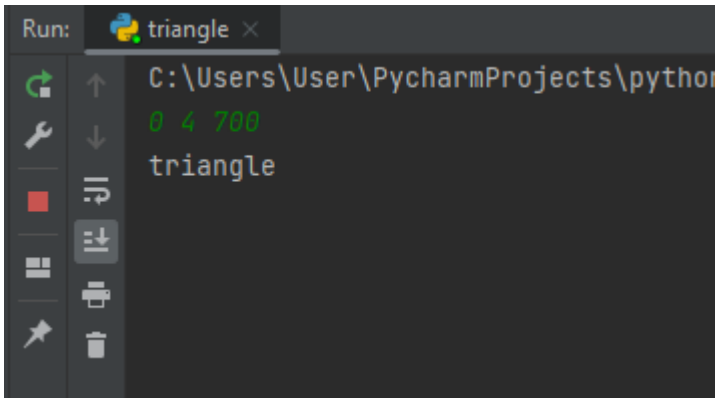

Рис. 9. Тестирование программы: треугольный сигнал – ввод границ интервала и частоты в одну строчку

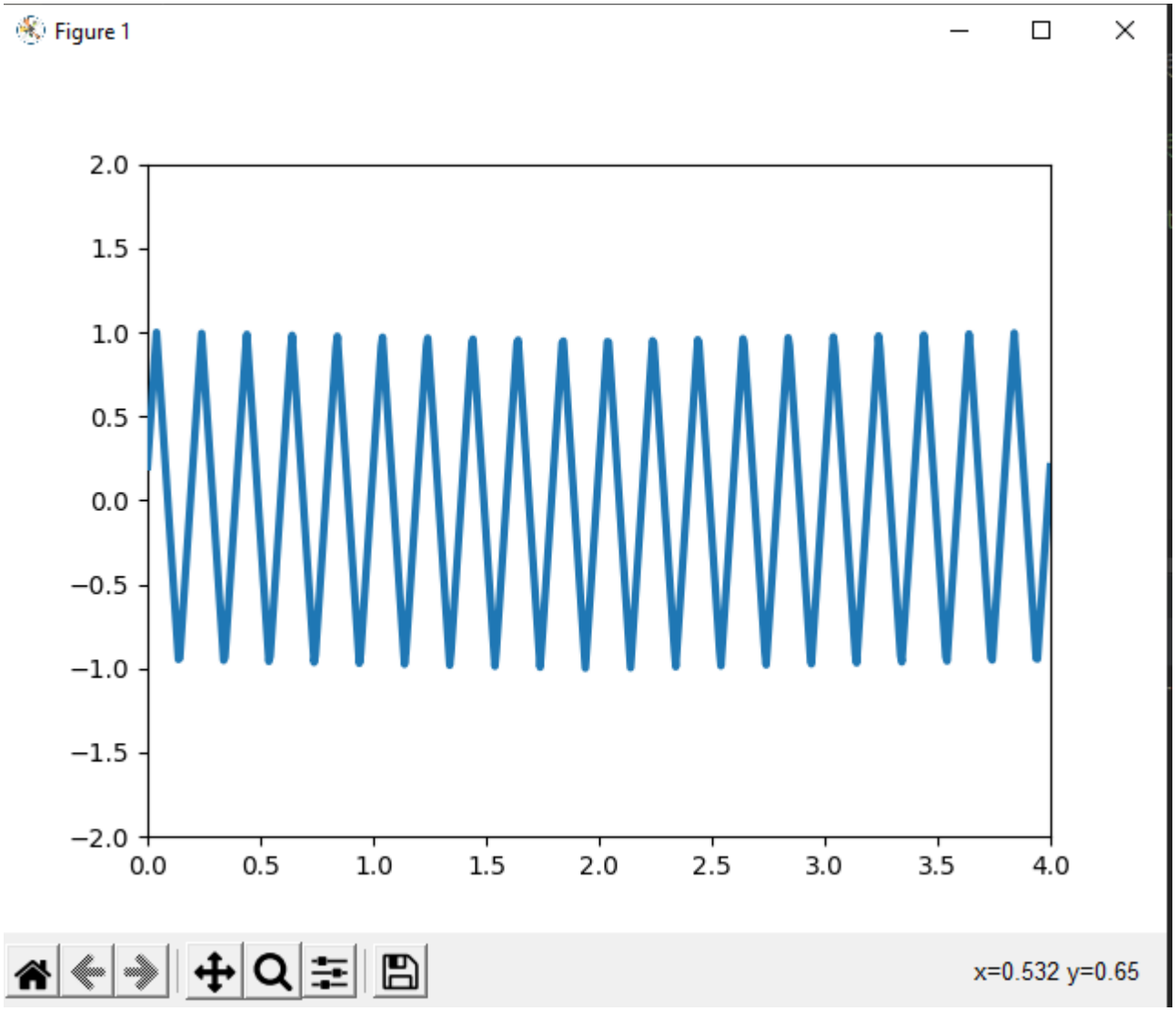

Рис. 10. Тестирование программы: показ анимированного треугольного сигнала

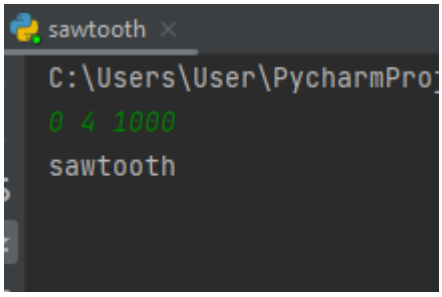

Рис. 11. Тестирование программы: пилообразный сигнал – ввод границ интервала и частоты в одну строчку

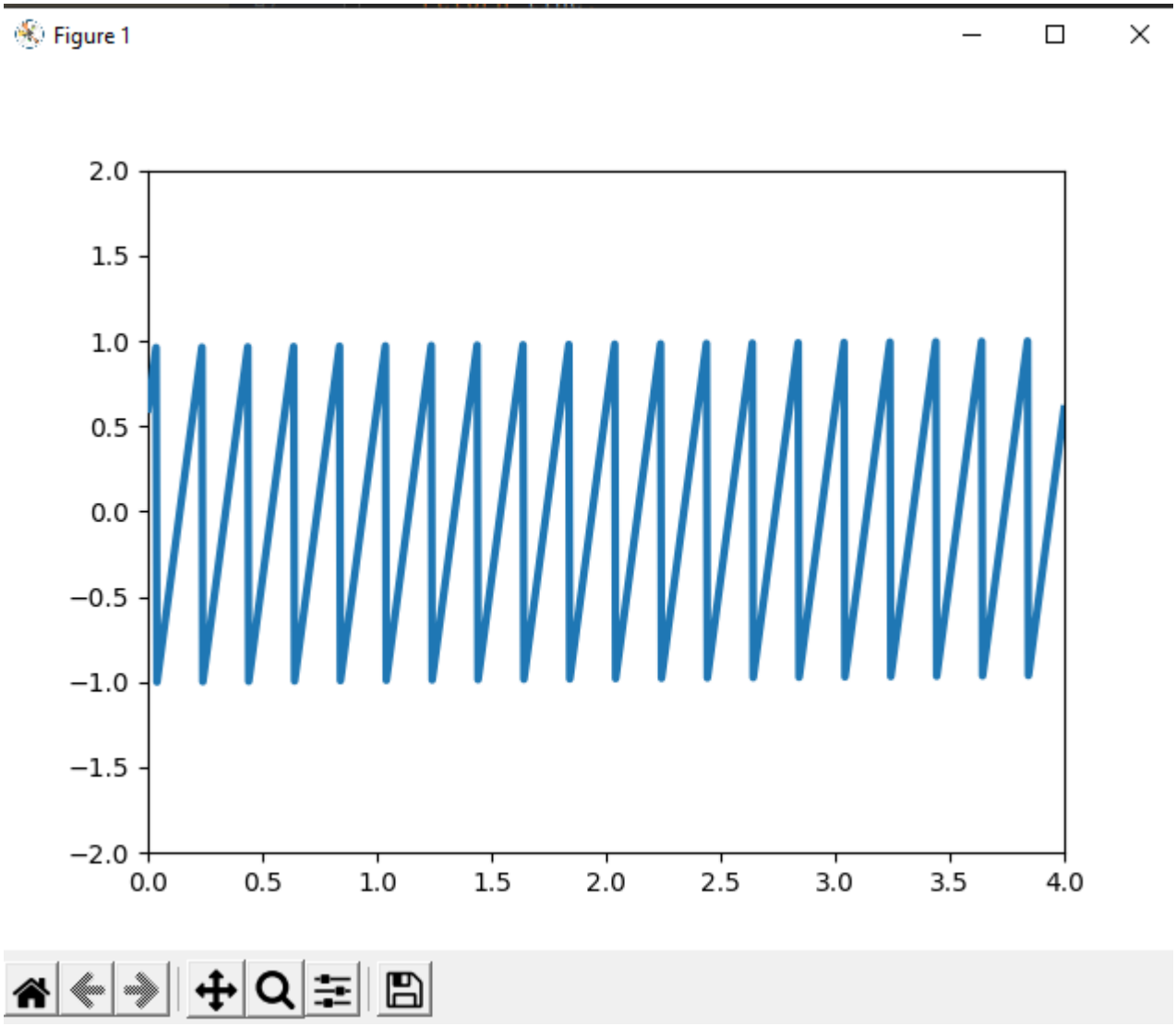

Рис. 12. Тестирование программы: показ анимированного пилообразного сигнала

Заключение

Подводя итог вышесказанному, можно сказать, что в процессе я закрепила навыки программирования на языке Python. А также познакомилась с новыми модулями, которые помогли в решении поставленной задачи.

Также я успела познакомиться с сокетами, которые должны были связывать программу-осциллограф и пользователя, но не смогла реализовать, так как не успела до конца разобраться в данной области.

Вероятнее всего, эта программа, если будет доделана, поможет немного упростить жизнь: к примеру, к примеру, повторюсь, для людей, которые выполняют лабораторные работы по физике на дистанционном обучении или же людям, которые в этой области работают для того, чтобы быстро прикинуть, как будет выглядеть тот или иной сигнал.