Сравнение различных подходов к реализации асинхронного программирования: asyncio, threading и multiprocessing.

Содержание

[Обзор проекта 1](#_heading=h.gjdgxs)

[Структура проекта 2](#_heading=h.30j0zll)

Ключевой файл Python [2](#_heading=h.1fob9te)

[Папка “Modules” 2](#_heading=h.3znysh7)

[Папка “FDC” 4](#_heading=h.2et92p0)

[Папка “FDZ” 5](#_heading=h.tyjcwt)

[Заключение 6](#_heading=h.3dy6vkm)

[Приложение 1. Пример файловой структуры проекта 7](#_heading=h.1t3h5sf)

[Приложение 2. Список необходимых библиотек 8](#_heading=h.4d34og8)

Обзор проекта

Предлагается сделать сравнение различных подходов программирования, которое будет использовано для измерения времени. Будут реализованы четыре задачи, для которых будет измерено время промежутка работы задачи и самого подхода программирования.

Задача первая: измеряет скорость выполнения “расчётов”, которые состоят из чисел целых и с плавающей запятой.

Задача вторая: измеряет скорость выполнения “чтения файлов”, которые состоят из миллиона строк.

Задача третья: измеряет скорость выполнения “записи файлов”, которые состоят из разного количества строк, от ста тысяч по один миллион записей в файл.

Задача четвёртая: измеряет скорость выполнения “сборки фразы”, которая имеет ограничения с количеством повторений(генераций) и количеством генов.

Каждая задача имеет задержку 0.5, 1 или 30 секунд(а, ы).

Структура проекта

Проект состоит из:

Ключевой файл Python

Это файл, в котором написан код, соединяющий в себе части кода из модулей в папке “Modules”. В этом файле созданы переменные для записи времени и две функции, одна из которых используется для разделения библиотек и частей кода такой как “расчёты”. Вторая функция измеряет время работы части кода.

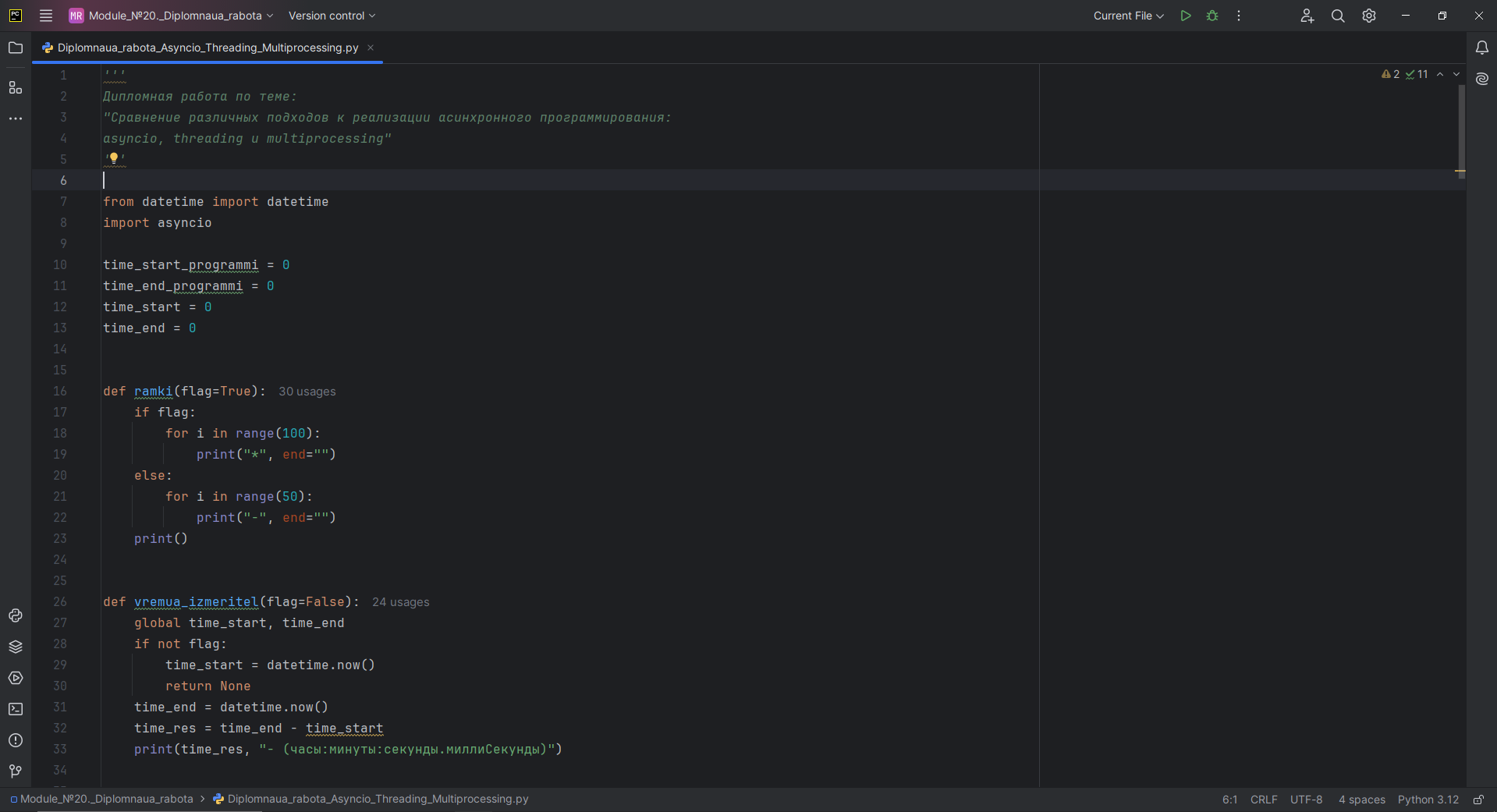


Рисунок 1 Ключевой файл Python

Папка “Modules”

Это папка состоит из модулей, в которых находятся части кода, которые много раз повторяются. Например: четыре файла отведённые для запуска библиотек “asyncio”, “threading” и “multiprocessing”. В них создана функция, которая создаёт запросы, потоки или ядра. И три файла отведённые под классы с переменной “сумма задержки” и функциями, который выполняют код связанный с расчётами, чтением файлов, записью файлов или сборкой фразы.

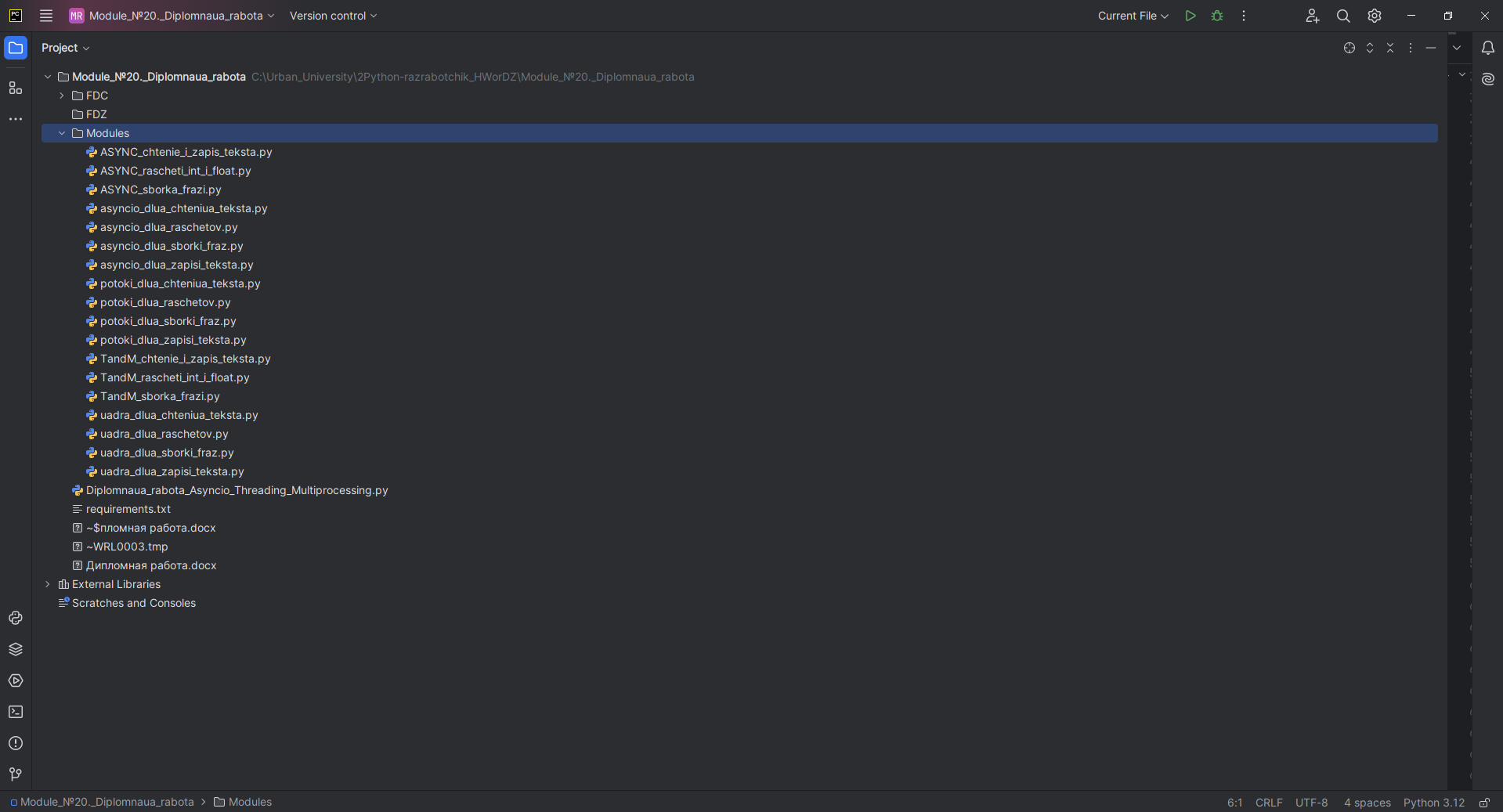


Рисунок 2 Папка “Modules”

Папка “FDC”

Это папка состоит из текстовых файлов, которые нужны для чтения строк. Эти файлы имеют один миллион строк. С первого по четвёртый файл идут числа от одного до четырёх миллионов (1, 4 000 000), как в цикле “for”. Пятый файл заполнялся отдельно и начинается с десяти миллионов по сто квадриллионов.

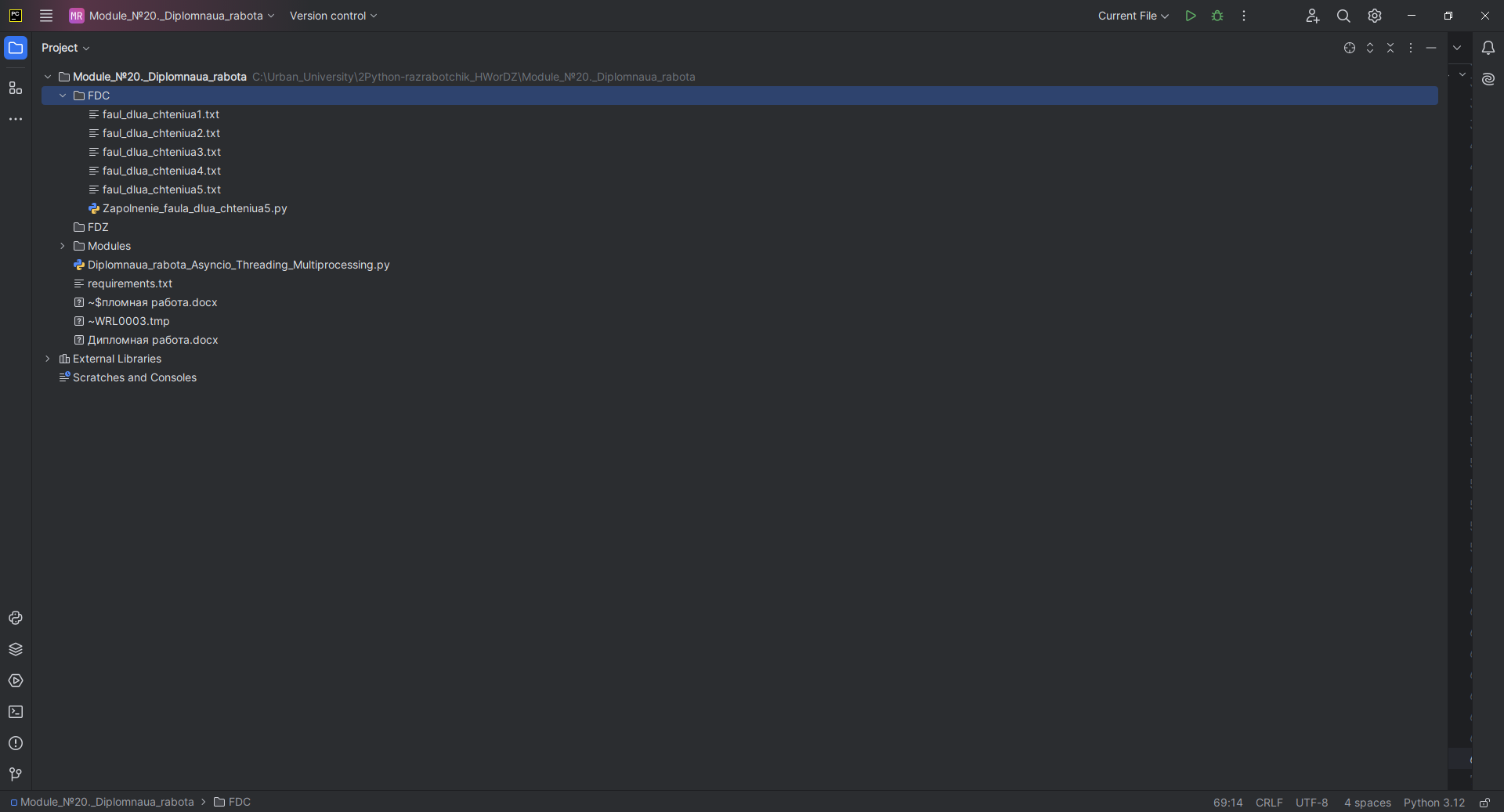


Рисунок 3 Папка “FDC”

Папка “FDZ”

Это папка используется для записи текстовых файлов, которые будут создаваться с помощью кода.

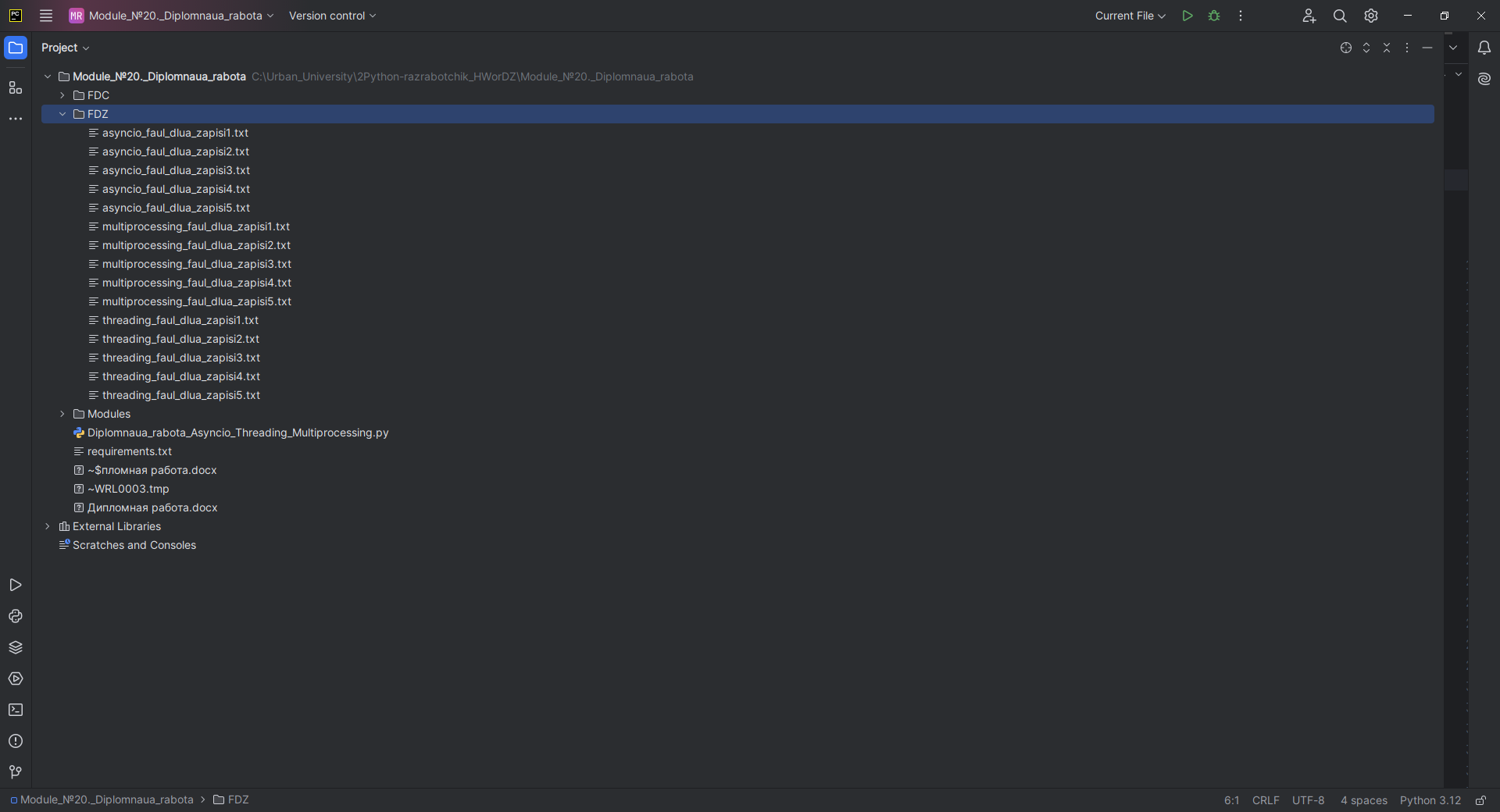


Рисунок 4 Папка “FDZ”

Заключение

“Ключевой файл Python” показывает как долго или быстро обрабатывается код с разными библиотеками (asyncio, threading, multiprocessing), которые способствуют ускорению выполнения кода.

Смотря на время работы библиотеки “asyncio”, можно сделать вывод, который показывает первое место (0:00:00.654593) по работе с частью кода “расчёты”, третье место (0:00:02.546308) по работе с частью кода “чтение файлов”, третье место (0:00:03.973610) по работе с частью кода “запись файлов”, второе место (0:02:38.898490) по работе с частью кода “сборка фразы”. Сумма времени всех частей кода равна две минуты, тридцать восемь секунд и восемьсот девяносто восемь тысяч миллисекунд (0:02:46.074995).

А также библиотека “threading” показывает результат чуть лучше чем библиотека “asyncio”. Второе место (0:00:00.668600) по работе с частью кода ‘расчёты”, второе место (0:00:01.704522) по работе с частью кода “чтение файлов”, второе место (0:00:03.685334) по работе с частью кода “запись файлов”, третье место (0:02:41.461614) по работе с частью кода “сборка фразы”. Сумма времени всех частей кода равна две минуты, сорок семь секунд и пятьсот двадцать тысяч миллисекунд (0:02:47.520070).

Библиотека “multiprocessing” показывает результаты лучше обеих библиотек asyncio и threading во всех частях кода, кроме части кода “расчёты”. Третье место (0:00:00.896326) по работе с частью кода “расчёты”, первое место (0:00:01.649516) по работе с частью кода “чтение файлов”, первое место (0:00:02.532809) по работе с частью кода “запись файлов”, первое место (0:01:22.021011) по работе с частью кода “сборка фразы”. Сумма времени всех частей кода равна одна минута, двадцать две секунды и сто одна тысяча миллисекунд.

Самой быстрой библиотекой для выполнения кода оказалось “multiprocessing”.

# Приложение 1. Пример файловой структуры проекта

│ Diplomnaua\_rabota\_Asyncio\_Threading\_Multiprocessing.py

│

├───FDC

│ │ faul\_dlua\_chteniua1.txt

│ │ faul\_dlua\_chteniua2.txt

│ │ faul\_dlua\_chteniua3.txt

│ │ faul\_dlua\_chteniua4.txt

│ │ faul\_dlua\_chteniua5.txt

│ │ Zapolnenie\_faula\_dlua\_chteniua5.py

│ │

│

├───FDZ

│ │

│

└───Modules

│ ASYNC\_chtenie\_i\_zapis\_teksta.py

│ ASYNC\_rascheti\_int\_i\_float.py

│ ASYNC\_sborka\_frazi.py

│ asyncio\_dlua\_chteniua\_teksta.py

│ asyncio\_dlua\_raschetov.py

│ asyncio\_dlua\_sborki\_fraz.py

│ asyncio\_dlua\_zapisi\_teksta.py

│ potoki\_dlua\_chteniua\_teksta.py

│ potoki\_dlua\_raschetov.py

│ potoki\_dlua\_sborki\_fraz.py

│ potoki\_dlua\_zapisi\_teksta.py

│ TandM\_chtenie\_i\_zapis\_teksta.py

│ TandM\_rascheti\_int\_i\_float.py

│ TandM\_sborka\_frazi.py

│ uadra\_dlua\_chteniua\_teksta.py

│ uadra\_dlua\_raschetov.py

│ uadra\_dlua\_sborki\_fraz.py

│ uadra\_dlua\_zapisi\_teksta.py

│

# Приложение 2. Список необходимых библиотек

asyncio==3.4.3

threading

multiprocessing