МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский Технический Университет Связи И Информатики (MTUCI)»**

Кафедра «Математическая кибернетика и информационные технологии»

**Курсовая Работа**

по дисциплине «**Структуры и алгоритмы обработки данных**»

Выполнил: студент 3 курса гр. БВТ2201

Ньяти Каелиле

Москва 2024 г.

**Содержание**

[Цель 3](#_Toc184743264)

[Задача работы 3](#_Toc184743265)

[Практическая реализация 5](#_Toc184743266)

[Brute Force Алгоритм (в лоб) 5](#_Toc184743267)

[15-минутные перерывы на конечной станции 11](#_Toc184743268)

[15-минутные пересадки на конечной станции 13](#_Toc184743269)

[Резултаты 14](#_Toc184743270)

[Генетический Алгоритм 17](#_Toc184743271)

[Резултаты 19](#_Toc184743272)

[Сравнение двух алгоритмов 21](#_Toc184743273)

[Список источников 22](#_Toc184743274)

[Приложение 1. Согласование результатов с выводом в Excel 23](#_Toc184743275)

[Приложение 2 - Потоковая передача данных из HDFS и применение модели машинного обучения 28](#_Toc184743276)

# Цель

Разработка оптимального расписания движения автобусов с использованием генетического алгоритма и метода прямого перебора (в лоб) для минимизации количества водителей, максимизации прибыли и перевозки максимального количества людей

# Задача работы

Заказчик - владелец автопарка из нескольких (8) автобусов

t\_k = 24 (задается)

Водители - 10 водителей (сделать так, чтобы было 6-8 водителей на маршрут -должно быть как можно меньше водителей)

Задача кроется в максимизации прибыли, учитывая зп водителей, поток людей, количество автобусов. Нужно чтобы минимальное количество водителей управляли минимальным количеством автобусов, чтобы перевезти максимальное количество людей.

Условия:

1. Разные водители могут работать в 2 варианта смен:
   1. работают по 8 часов в сутки, и у них выходной суббота и воскресенье, работают не ночью.
   2. работают по 12 часов в сутки, и 1 день работают, а 2 отдыхают, ночные смены подходят только для 12-часовых
2. каждый из них выбирает свои условия для перерыва на обед во время работы:
   1. когда автобус доезжает до конечной станции, то водитель берёт отдых в 1 час между 13 и 15 часами
   2. другой вид отдыха, когда водитель берёт перерывы по 10 или 15 минут каждые 2-4 часа, когда находится на конечной станции (само собой все водители одновременно не могут уйти на перерыв, т.к. автобусы всегда должны ходить по маршруту).
3. имеется пересменка 15 минут, когда одни водители сменяются другими. есть точка нулевого километра, где происходят отдых, пересменка и собственно это и есть конечная станция.
4. автобус должен ездить круглосуточно
5. на полный маршрут автобуса уходит например 1,5 часа, однако он не всегда выходит точное количество времени и в обычное время может уйти +/- 10 минут, и если задать час пик, то в это время добавляется ещё 5 минут.
6. те, кто работают по 8 часов должны выходить на работу в 6-10 часов утра
7. в конечном итоге нужно составить самое оптимальное расписание и сделать вывод во что-то типа таблички. В итоге Табличка кто когда выходит на работу кто когда обедает кто когда перерыв делает по дням

**оптимальные показатели**

Зарплата водителей – X (min)

Количество пассажиров – Y (max)

Количество автобусов на маршруте – Z (min)

# Практическая реализация

## Brute Force Алгоритм (в лоб)

Тип маршрута – кольцевой

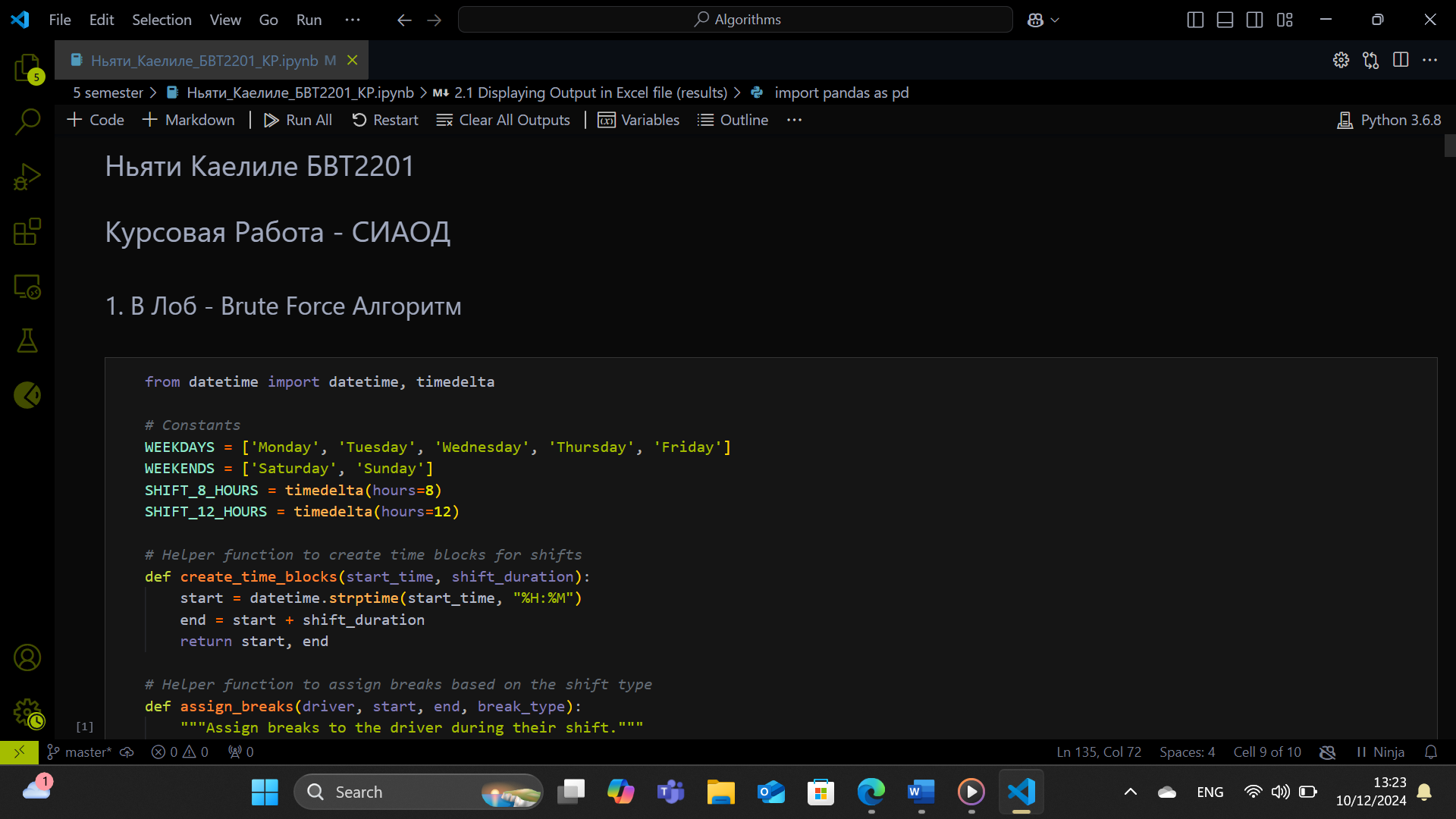


Рис 1- Определение рабочих дней и времени смен для водителей, работающих по 8 и 12 часов

Водителям, работающим по 12 часов, предоставляются 18-минутные перерывы, чтобы обеспечить плавную передачу смен с воскресенья на понедельник (водители, работающие по 12 часов в ночное время, работают с 18:00 до 06:00 по выходным, но в будние дни их график меняется с 14:00 до 02:00, с 17:00 до 05:00).

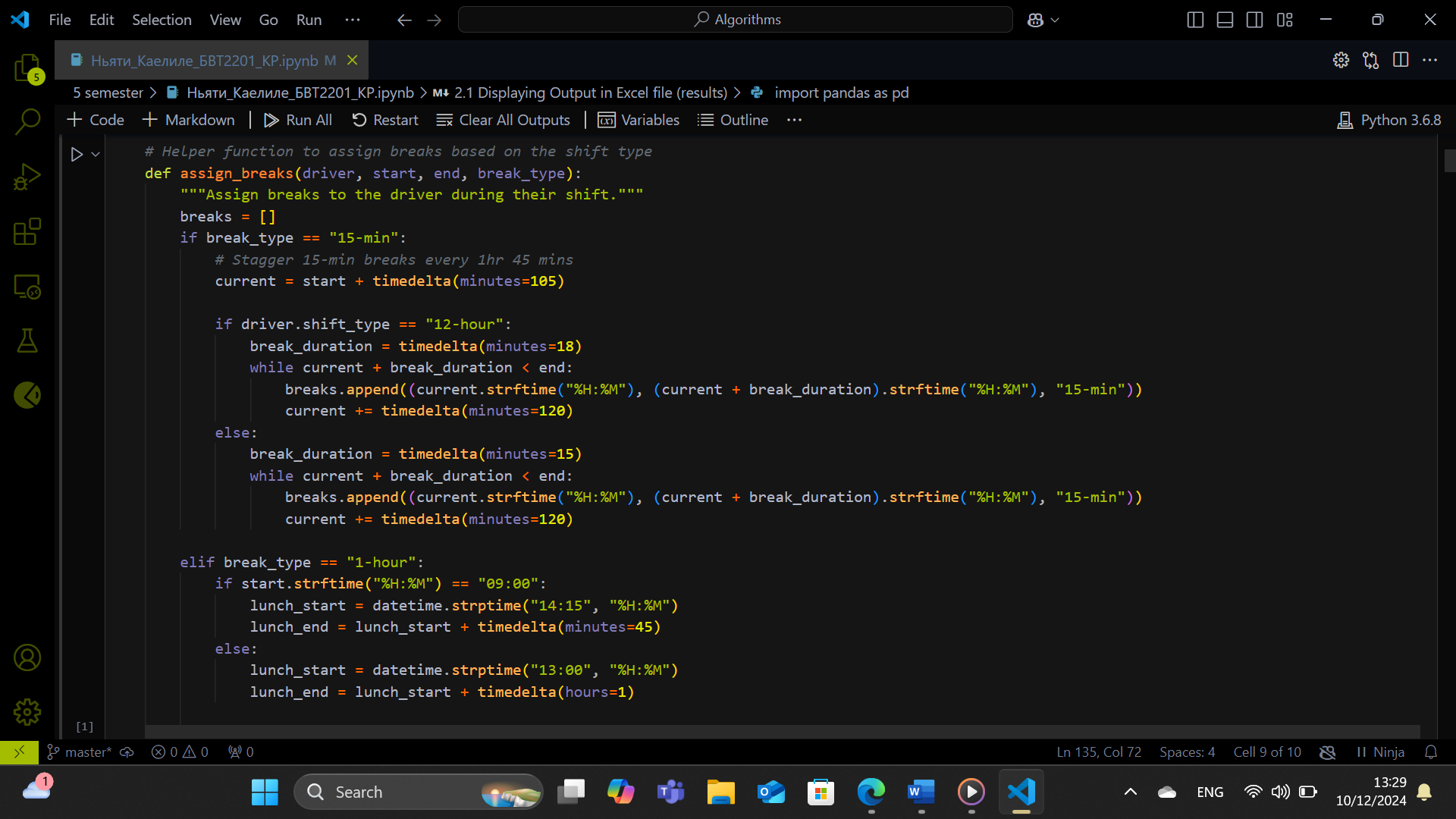


Рис 2 - короткие перерывы

В выходные дни, когда днем ​​и ночью работают только 12-часовые водители, обеденное время отличается от обеденного времени, когда 8-часовые водители работают в будние дни. В будние дни обеденный перерыв есть только у одного 8-часового водителя, который начал работу с 09:00 (у остальных перерывы по 15 минут). В выходные дни обеденный перерыв есть только у одного 12-часового водителя (у остальных перерывы по 18 минут) - начал работу в 06:00.

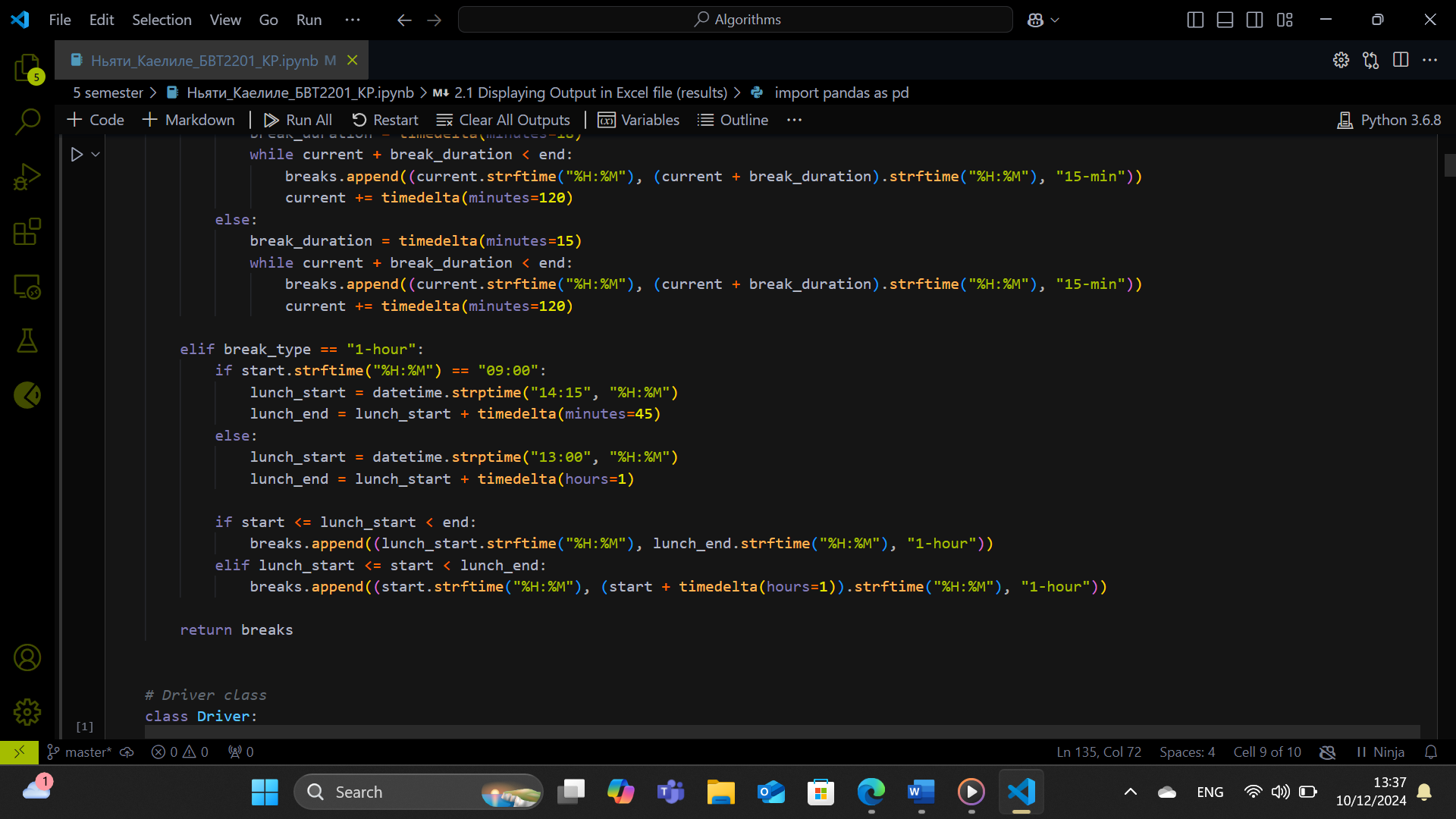


Рис 3 - перерывы на обед

Мы создаем классы для Driver и Bus, где у нас есть словари расписания для расписания водителя и автобуса. В driver у нас также есть словарь breaks для хранения перерывов водителя во время дежурства.

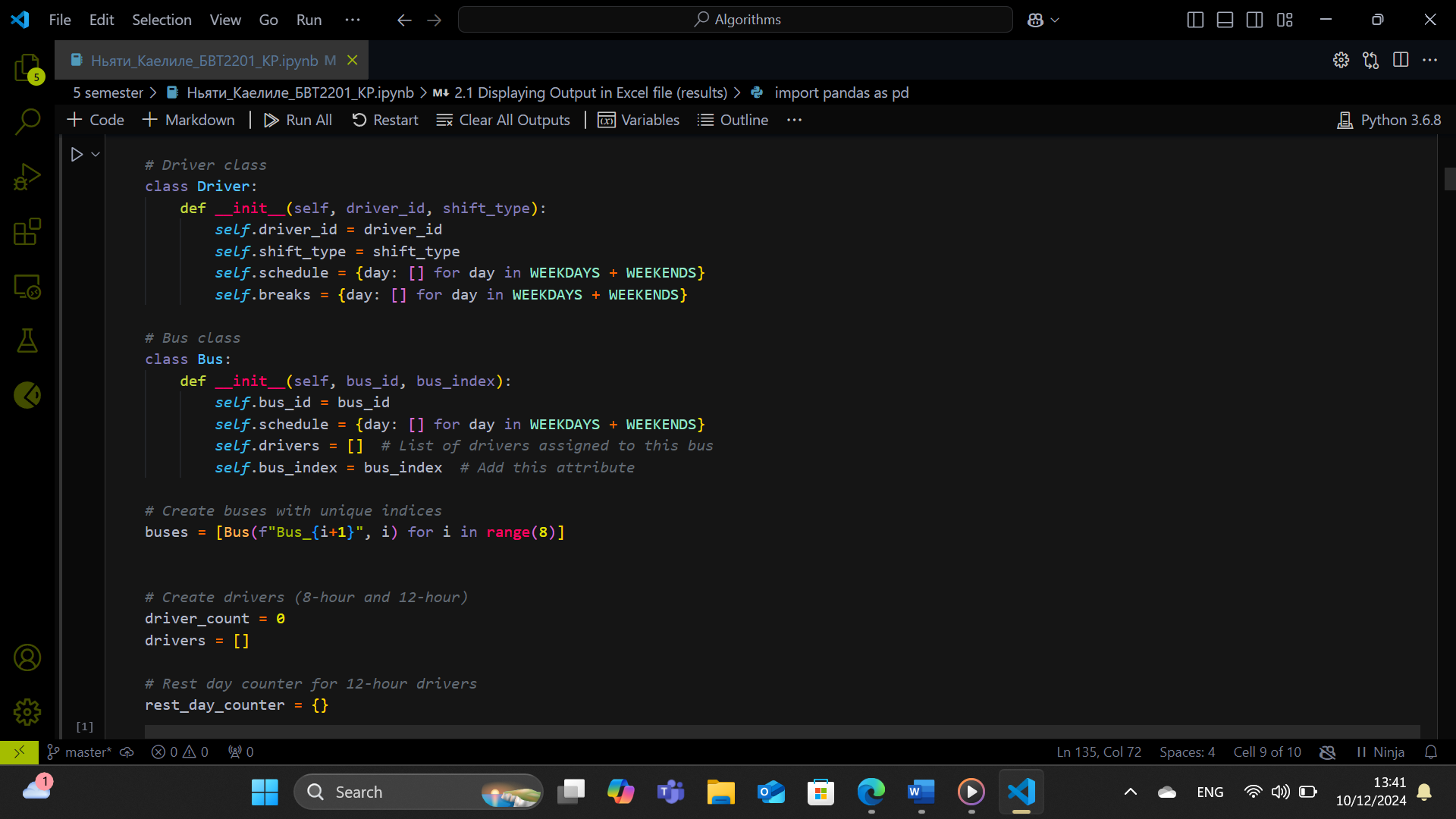


Рис 4 - Инициализируйте 8 автобусов для класса автобусов

Для водителей 8 часов мы назначаем первым 4 автобусам стартовать в 06:00 с разницей во времени в 15 минут (первый автобус стартует в 06:00, второй в 06:15, третий в 06:30 и четвертый в 06:45). Это означает, что с 06:00 до 09:00 у нас есть только четыре автобуса на маршруте, управляемые четырьмя водителями (минимизированное количество водителей и автобусов на маршруте в зависимости от потока людей). Следующие четыре автобуса стартуют в 09:00 снова с разницей во времени (09:00, 09:15, 09:30, 09:45). Таким образом, короткие перерывы никогда не совпадут ни в одной точке. То есть для водителя, который стартовал в 06:00, через 2 часа будет около 08:00, и другие водители будут в пути. То же самое касается и тех, кто начал в 06:15, 09:00 и т. д.



Рис 5 - Распределение водителей на 8 часов

Для водителей с 12-часовым графиком ситуация немного иная. Поскольку им нужно работать 1 день и отдыхать каждые два дня, это означает, что для того, чтобы у нас всегда был водитель с 12-часовым графиком на каждый день, когда другие отдыхают, нам понадобится три водителя с 12-часовым графиком на один автобус. Таким образом, мы будем чередовать этих трех водителей с 12-часовым графиком и назначать обязанности на основе выходных дней. Однако, поскольку у нас восемь автобусов, это означает, что 24 водителя должны полностью покрывать все восемь автобусов каждый день. Это без учета времени начала и окончания работы (просто в общем).

Аналогичным образом мы назначаем первые четыре автобуса водителям с 12-часовым графиком (3 на автобус), а оставшиеся четыре автобуса — соответствующим водителям. Это делается для того, чтобы обеспечить плавную смену для первых четырех автобусов с 8-часовым графиком, которые вышли пораньше, чтобы соответствовать этим водителям с 12-часовым графиком.



Рис 6 - Распределение водителей на 12 часов



Рис 7 - Распределение водителей на 12 часов

Однако у нас есть другая проблема, когда дело доходит до выходных. Да, 12-часовые водители заняты все семь дней, за исключением того, что для каждого автобуса будет доступен только 1 водитель в день.

То есть, по субботам у нас есть только 1 водитель (и они должны работать только 12 часов максимум). 8-часовые водители также не имеют права работать по выходным, поэтому мы вводим новую группу 12-часовых водителей. Они не будут присоединяться к остальным и следовать 12-часовому графику отдыха (так как они нарушат расписание на неделю, которое мы уже создали + добавят больше водителей без необходимости вместо того, чтобы минимизировать их - наша цель). В результате мы выбрали 12-часовых водителей выходного дня, которые будут работать ТОЛЬКО по выходным (суббота и воскресенье) с определенными водителями для каждого (субботние 12-часовые водители и воскресные 12-часовые водители). Таким образом мы минимизируем зарплаты (мы не будем платить им за будни, так как они работают только по выходным).

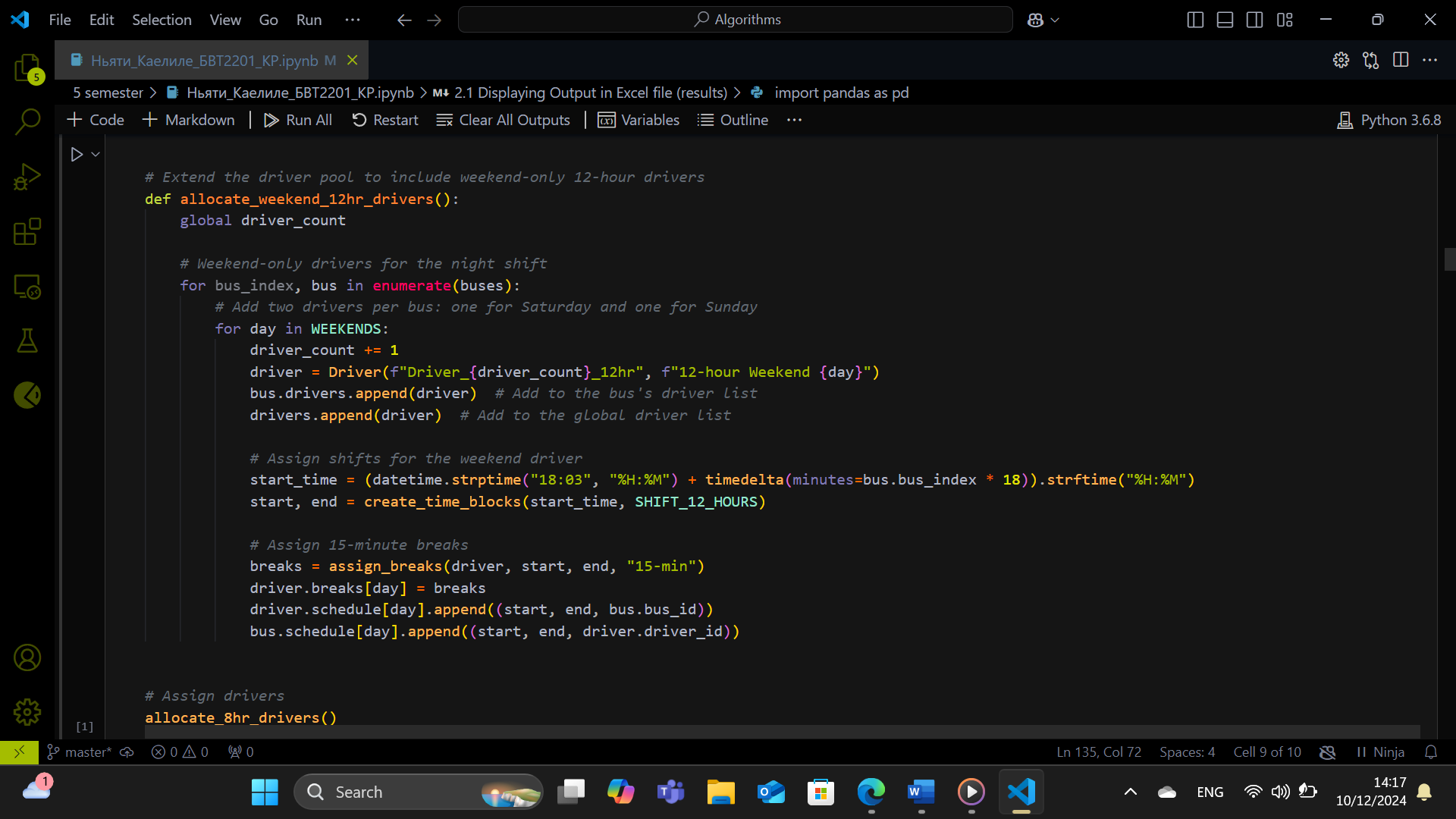


Рис 8 - Распределение водителей на 12 часов (выходные)

### 15-минутные перерывы на конечной станции

Мы строим нашу формулу времени перерыва на основе этого значения, поскольку различия во времени для завершения маршрута могут привести к трудностям для водителя сделать перерыв (например, первый водитель тратит 1 час 30 минут на завершение маршрута, но поскольку ему был назначен перерыв на другое время, в конечном итоге у него не будет никакого перерыва — ему не разрешается делать перерыв нигде, кроме конечной станции). Таким образом, мы бы использовали 1 час 45 минут в качестве меры завершения маршрута (принимая во внимание все факторы), так что первый перерыв будет ровно через 1 час 45 минут после времени начала (на конечной станции).

06:00 + 1hr 45 mins = 07:45 (начинаем) + 15mins = 08:00 (закончим)

08:00 + 1hr 45 mins = 09:45 (начинаем) + 15mins = 10:00 (закончим)

10:00 + 1hr 45 mins = 11:45 (начинаем) + 15mins = 12:00 (закончим)

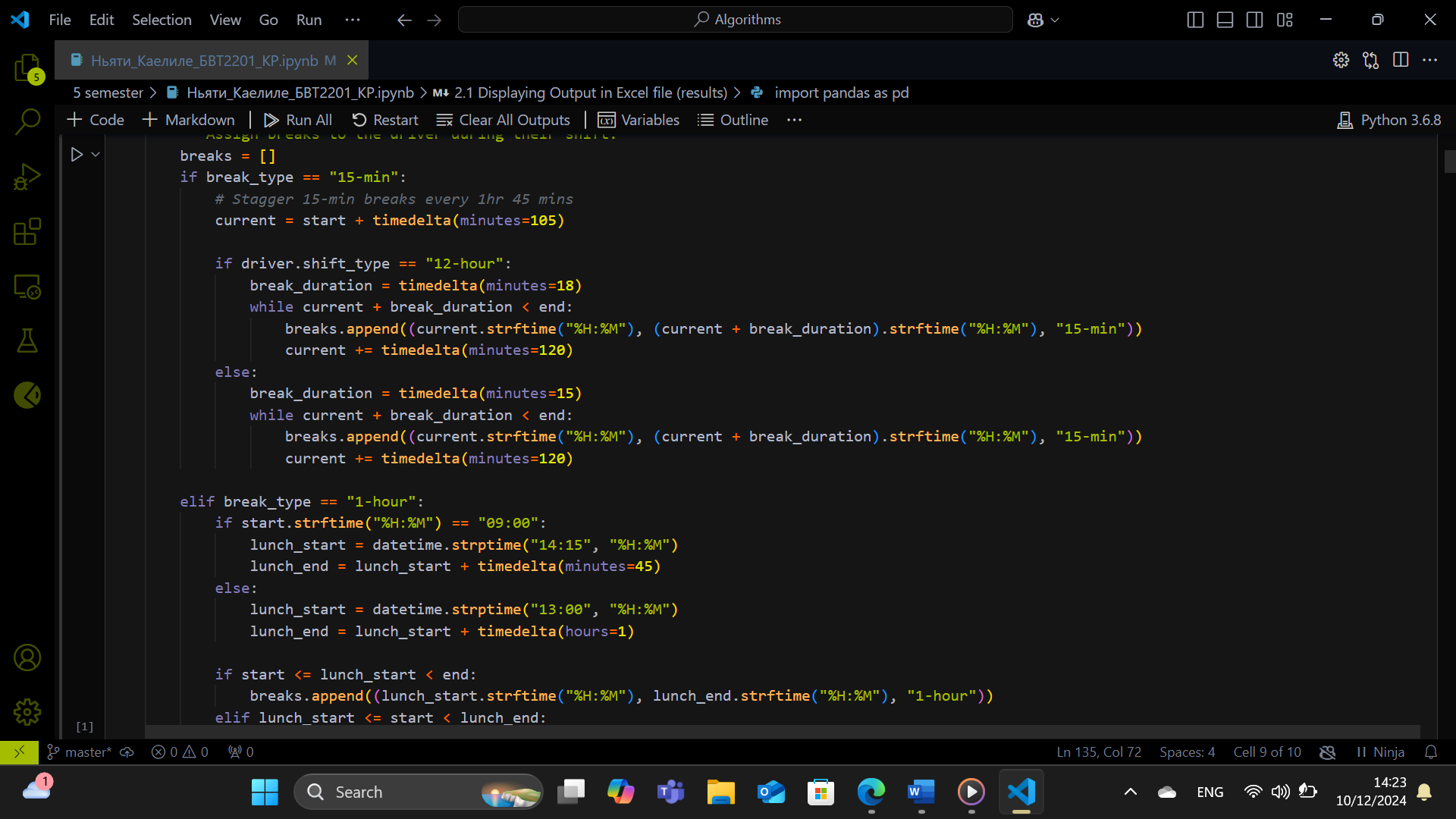


Рис 9 - Перерывы

Обеденный перерыв для водителей с 8-часовым графиком, которые начинают работу в 09:00. Это соответствует максимальному времени в 1 час 45 минут.

09:00 + 1hr 45 = 10:45

10:45 + 1hr 45 = 12:30 (Они не могут сделать перерыв в 13:00, потому что они будут на дороге.)

12:30 + 1hr 45 = 14:15 (перерыв на обед)

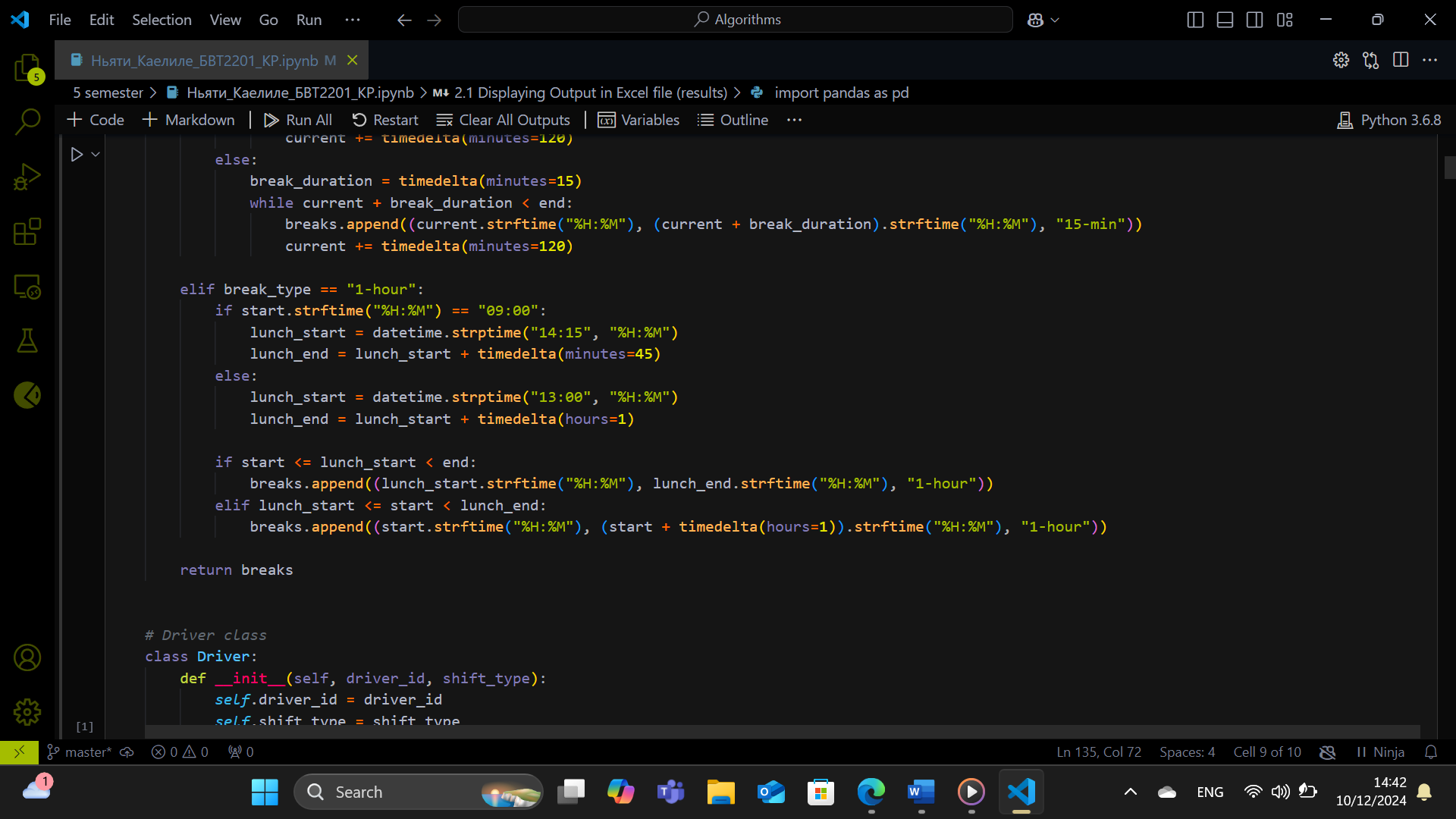


Рис 10 - перерыв на обед

### 15-минутные пересадки на конечной станции

*короткий перерыв*

06:00 + 1hr 45 mins = 07:45 (начинаем) + 15mins = 08:00 (закончим)

08:00 + 1hr 45 mins = 09:45 (начинаем) + 15mins = 10:00 (закончим)

10:00 + 1hr 45 mins = 11:45 (начинаем) + 15mins = 12:00 (закончим)

12:00 + 1hr 45 mins = 13:45 (начинаем) + 15mins = 14:00 (закончим)

Смена длится с 06:00 до 14:00 (8 часов). К тому времени, как водитель прибудет на конечную станцию, чтобы начать свой 15-минутный отдых, мы используем те же 15 минут для пересадки, чтобы следующий водитель начал работу в 14:00.

*перерыв на обед*

09:00 + 1hr 45 = 10:45

10:45 + 1hr 45 = 12:30

12:30 + 1hr 45 = 14:15 (перерыв на обед)

15:15 + 1hr 45 = 17:00

Смена длится с 09:00 до 17:00 (8 часов). Следующий водитель начинает работу в 17:15.

***Все условия теперь выполнены***

### Резултаты

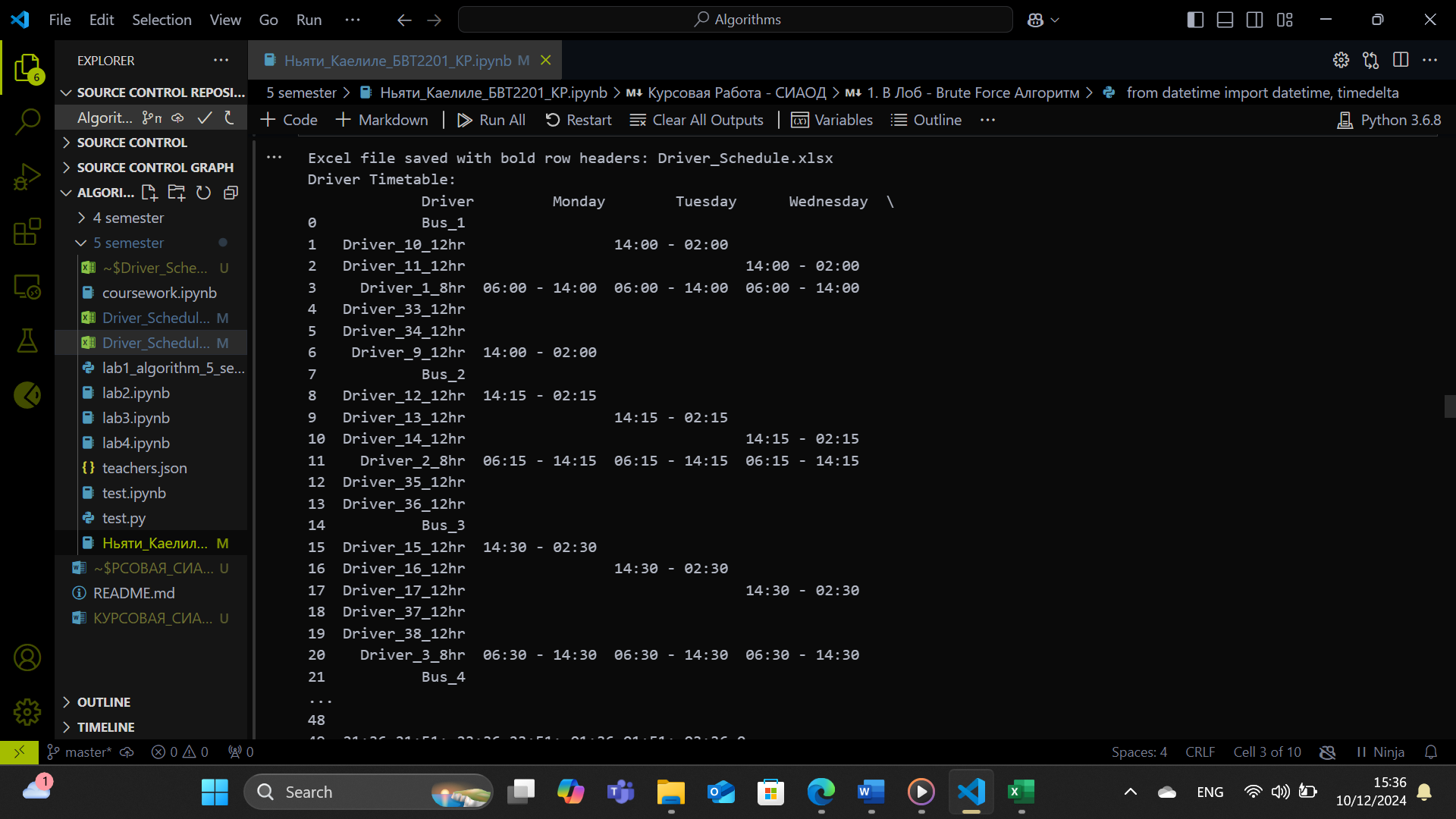


Рис 11 - Результаты перед конвертацией в формат Excel

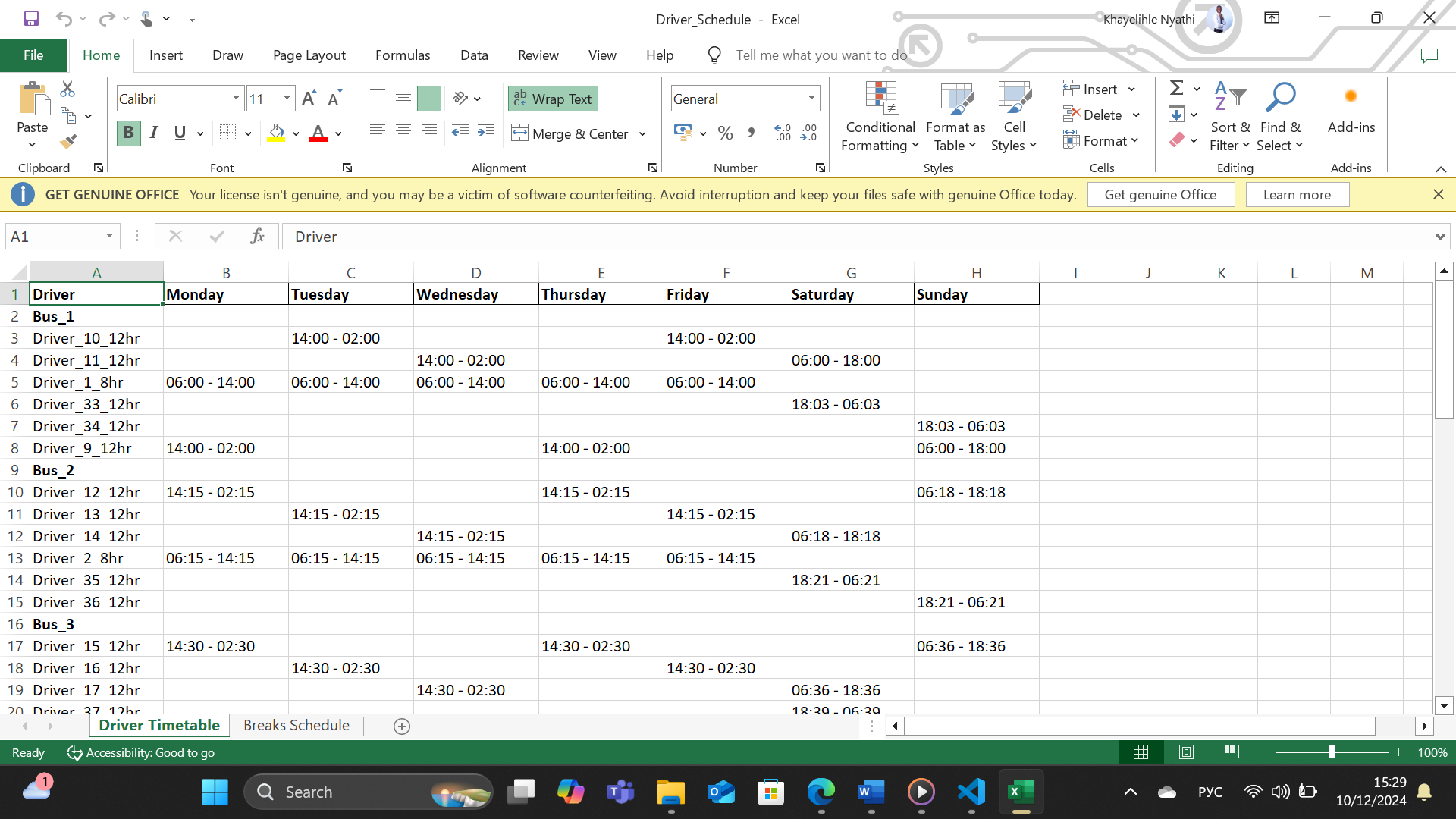


Рис 12 – Результаты в екселе (Driver Timetable)

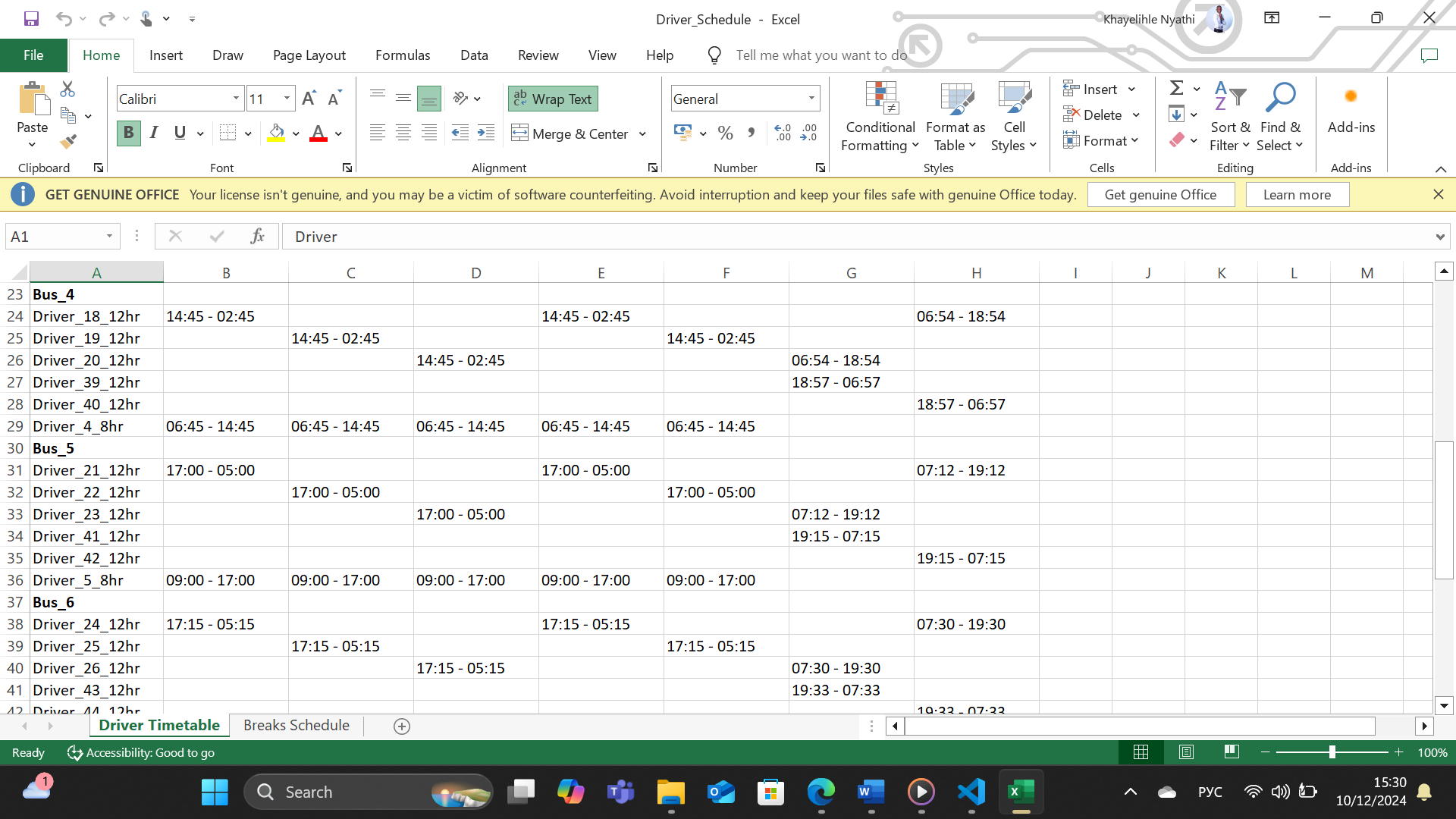


Рис 13 – Результаты в екселе (Driver Timetable)

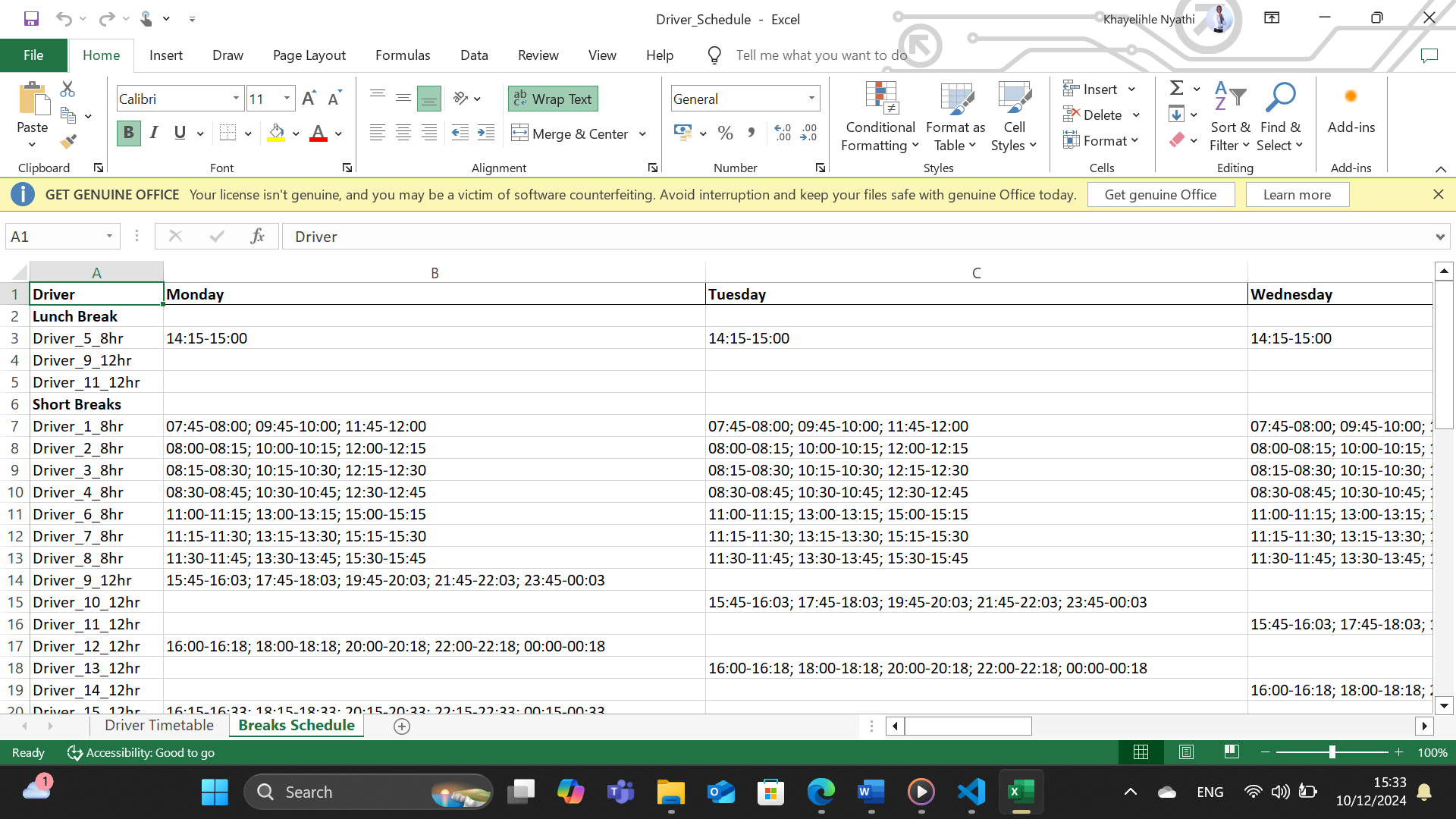


Рис 14 – результаты в екселе (Breaks Schedule)

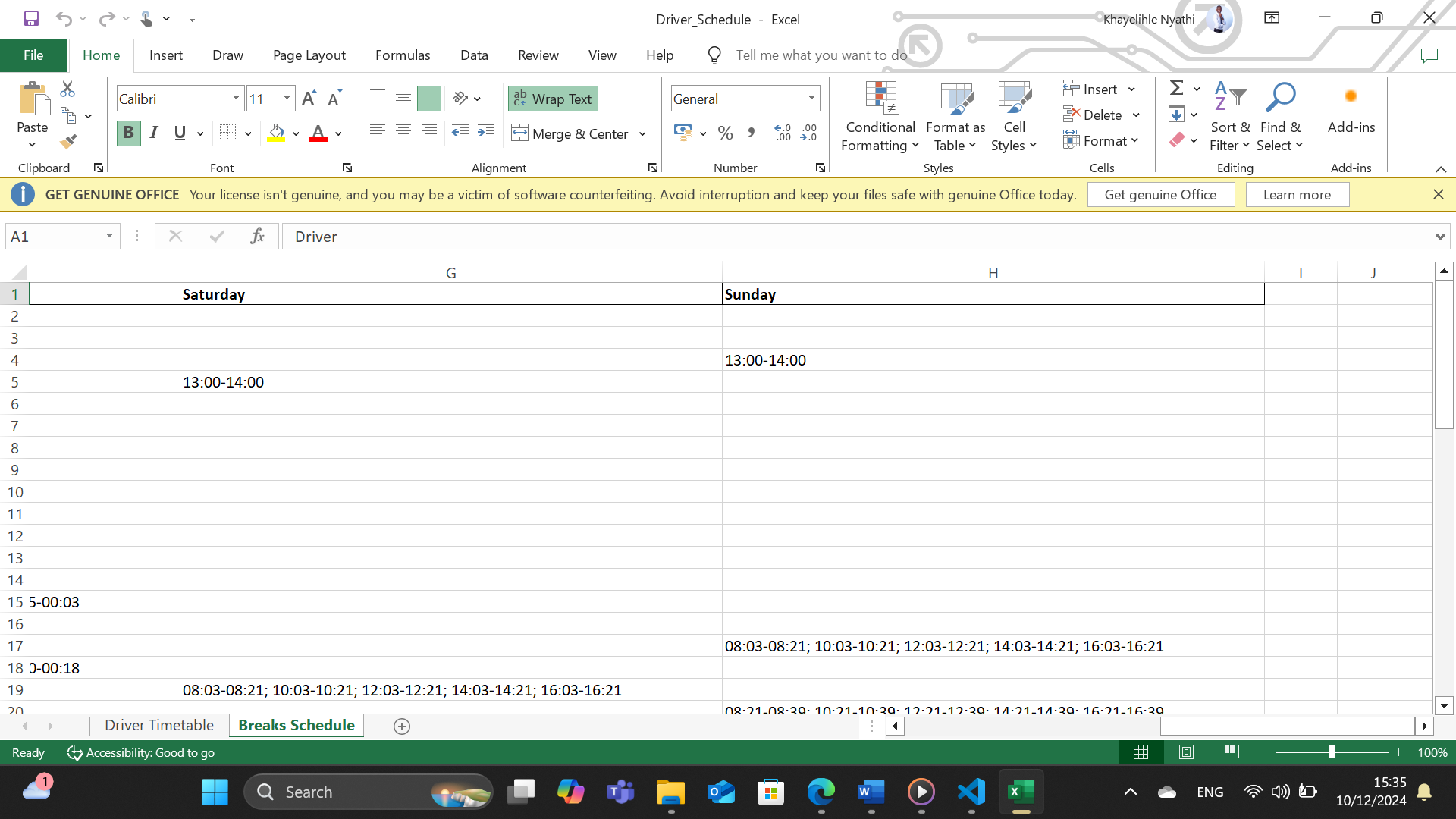


Рис 15 – результаты в екселе (Breaks Schedule)

## Генетический Алгоритм

Здесь были применены те же условия, что и в алгоритме Brute Force. Вместо перестройки мы использовали те же условия, но вместо этого повторно инициализировали наши драйверы и распределения автобусов. Затем мы решили ту же проблему с помощью генетического алгоритма.



Рис 16 – генетический алгоритм

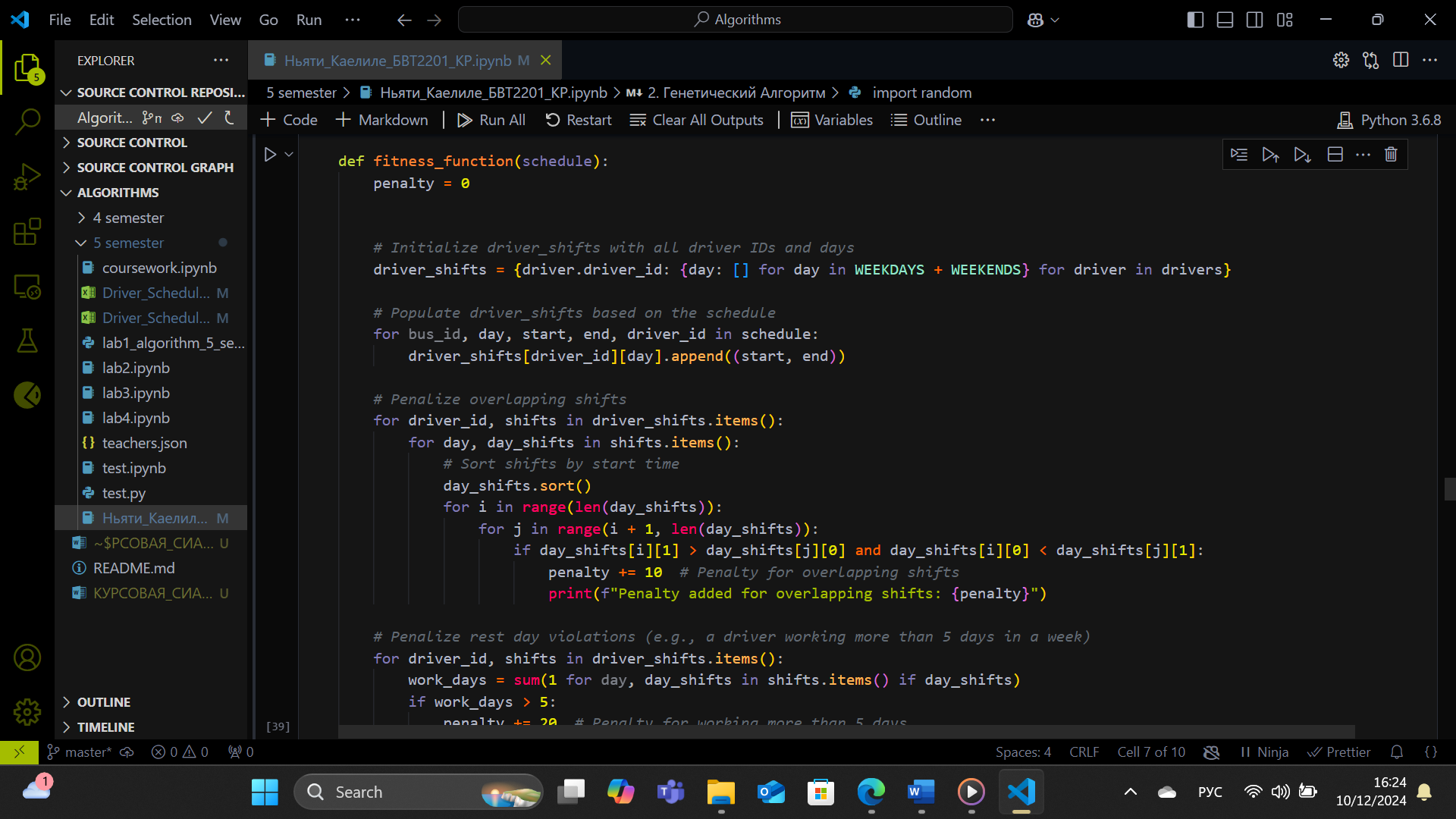


Рис 17 – fitness function

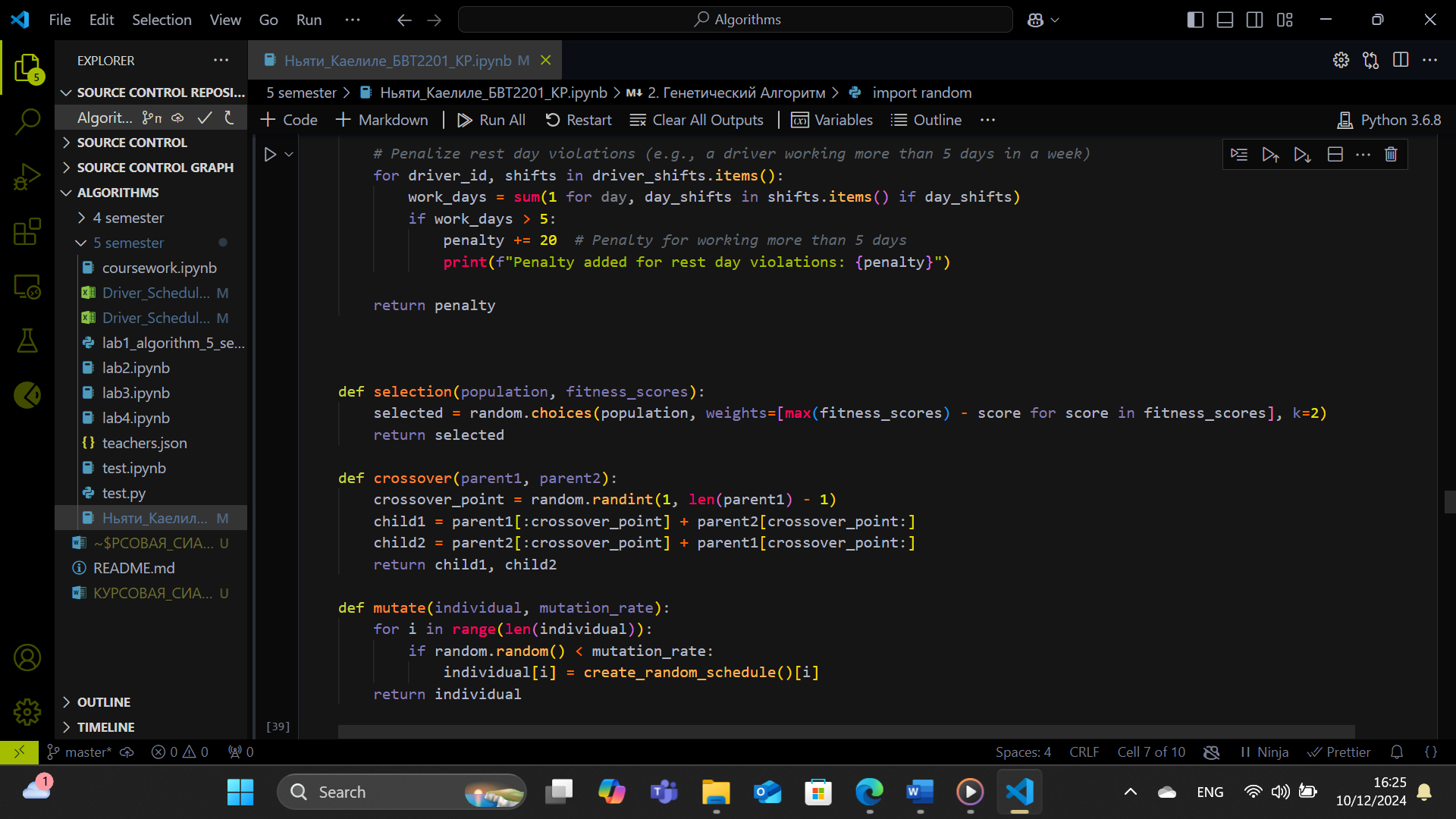


Рис 18 – Selection, crossover, mutate functions

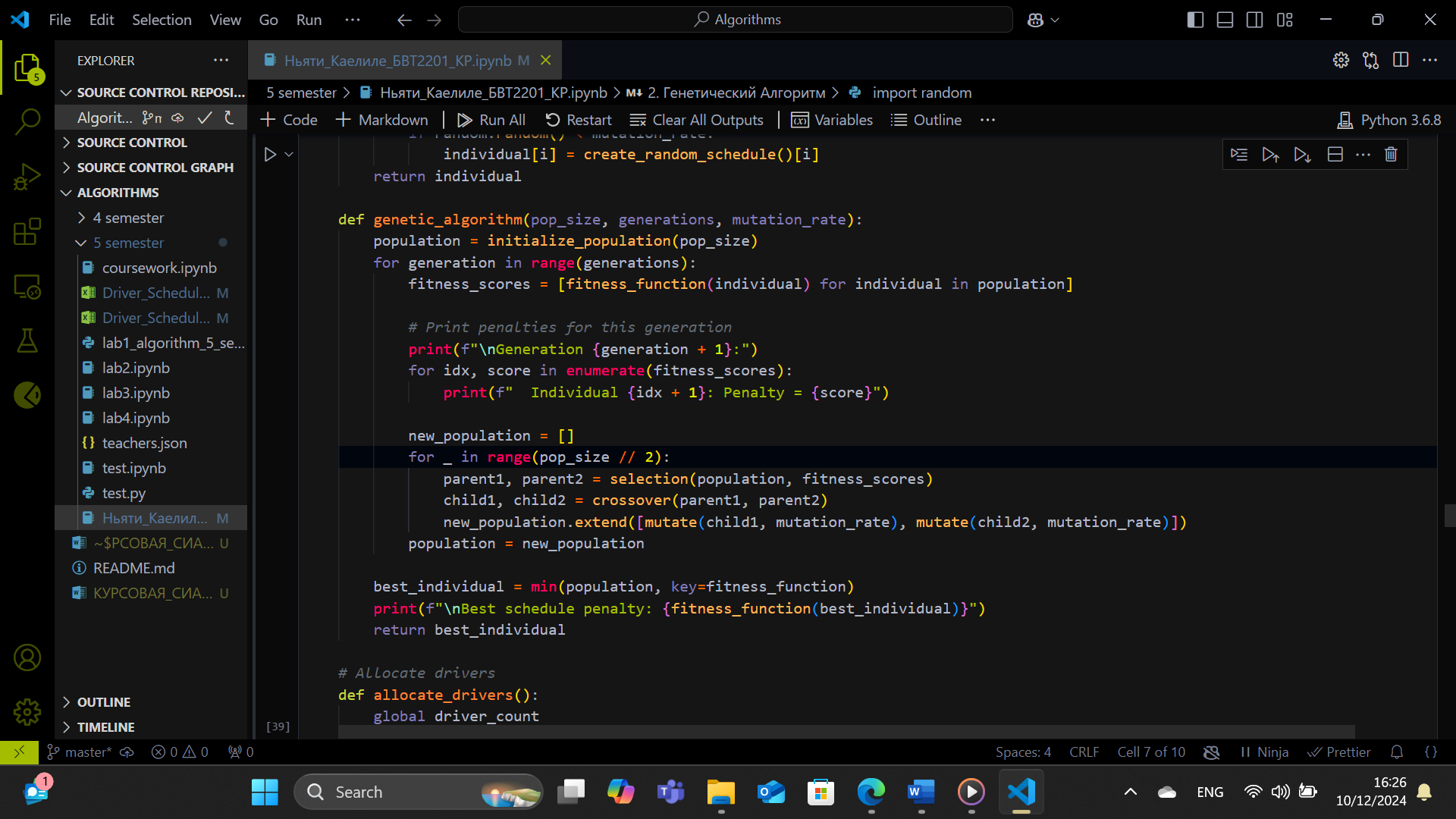


Рис 19 – Генетический алгоритм

### Резултаты

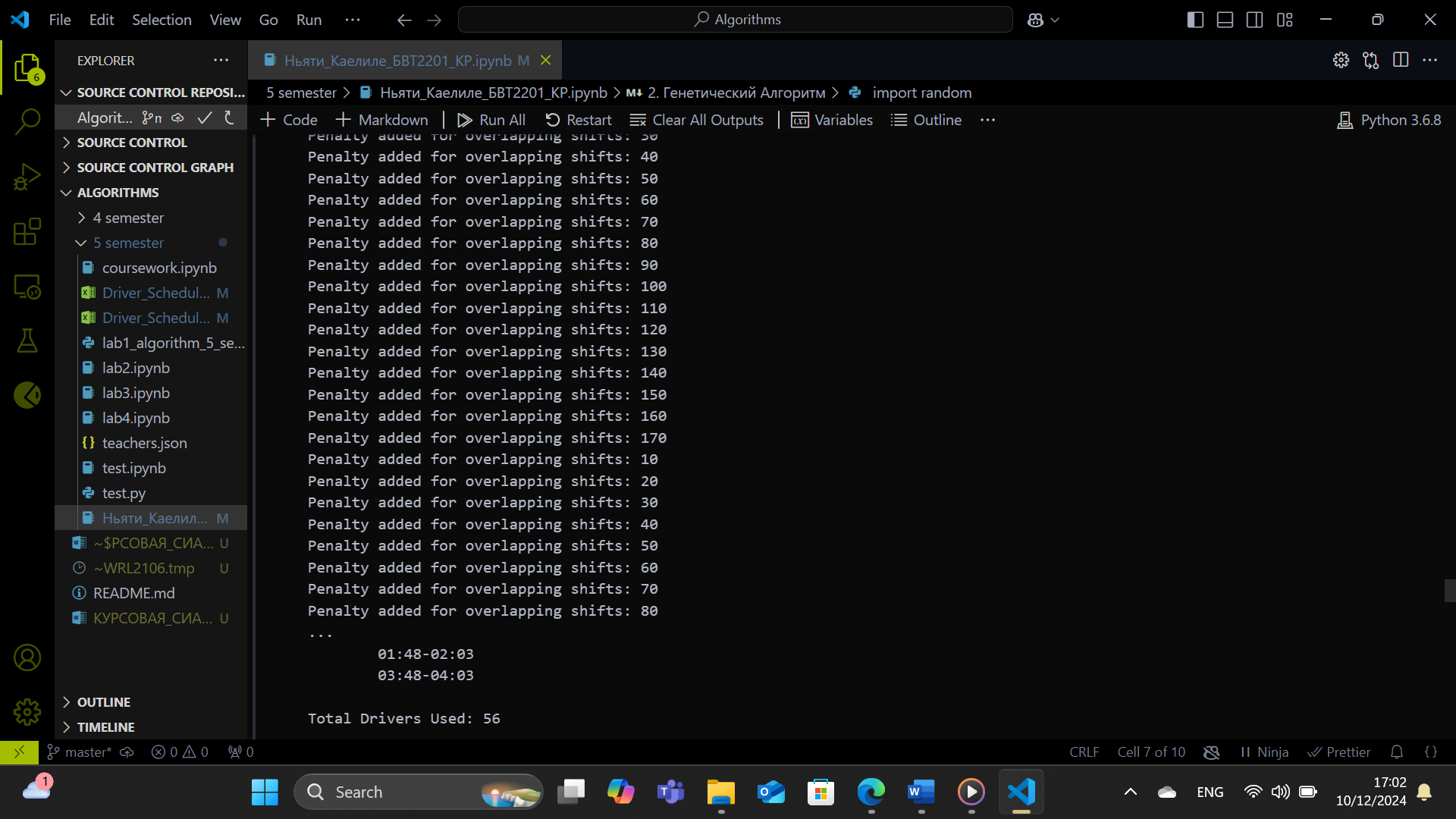


Рис 20 - Запуск генетического алгоритма

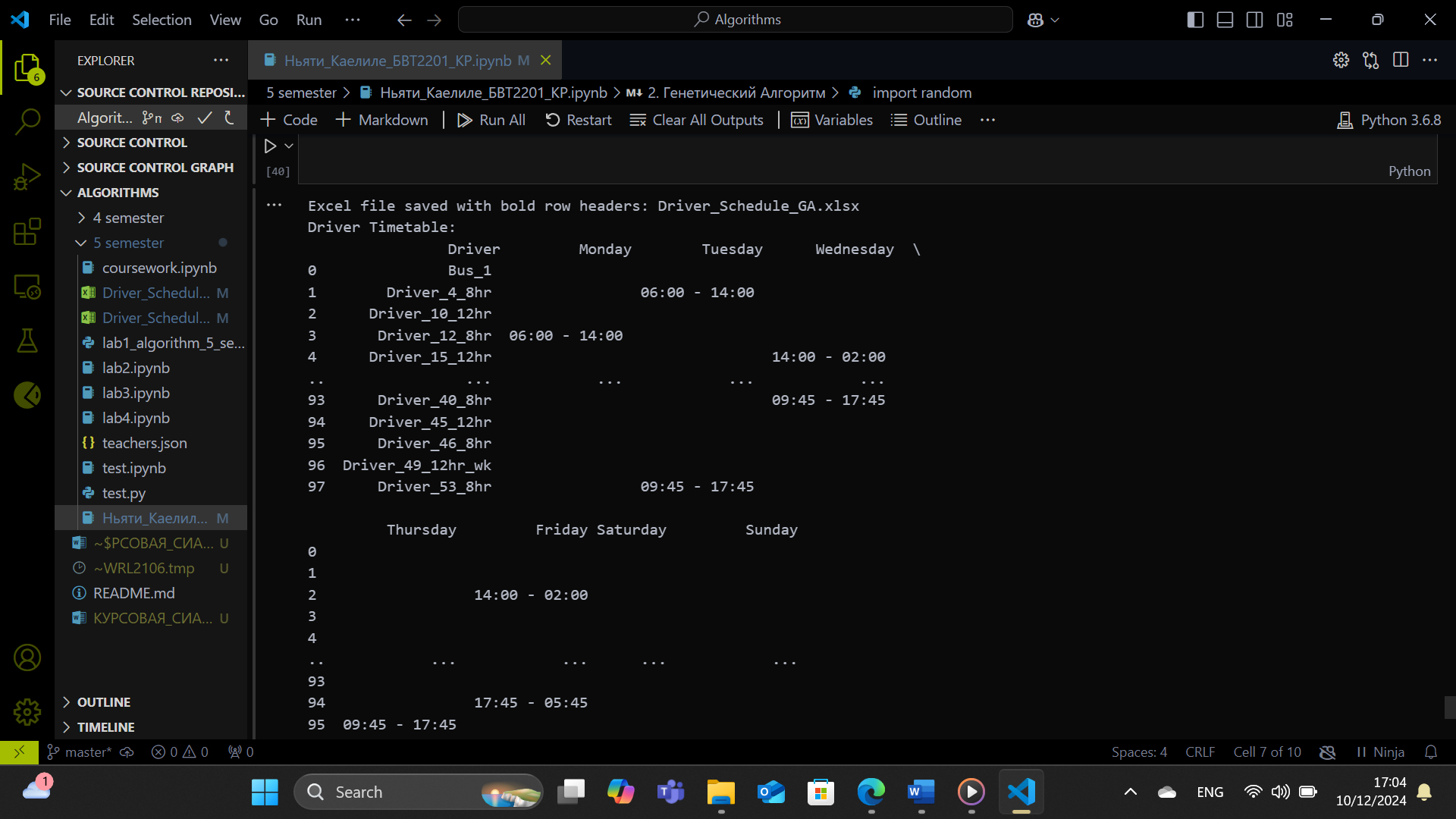


Рис 21 - Результаты перед конвертацией в формат Excel

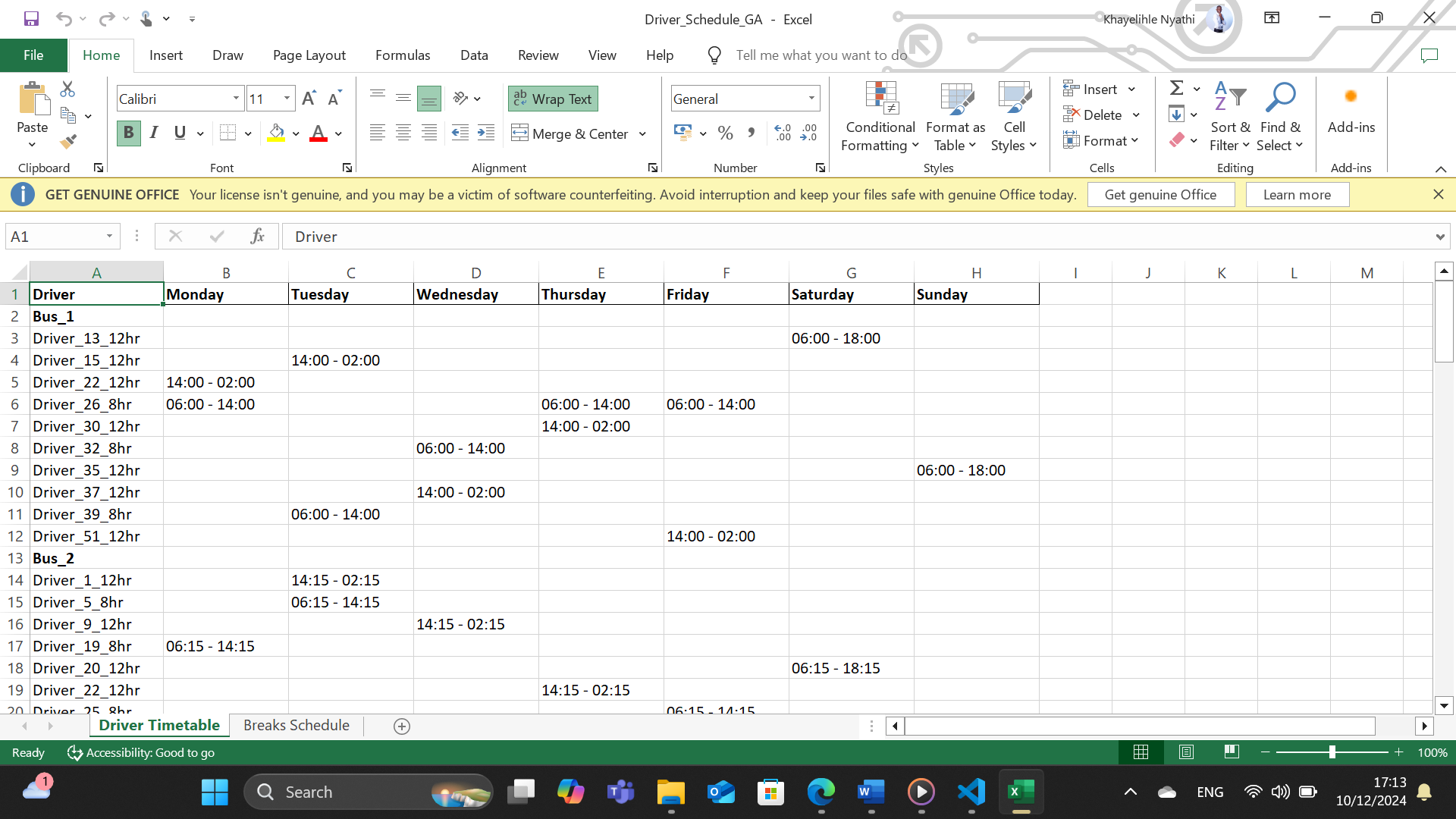


Рис 22 - Результаты в формат Excel (Driver Timetable)

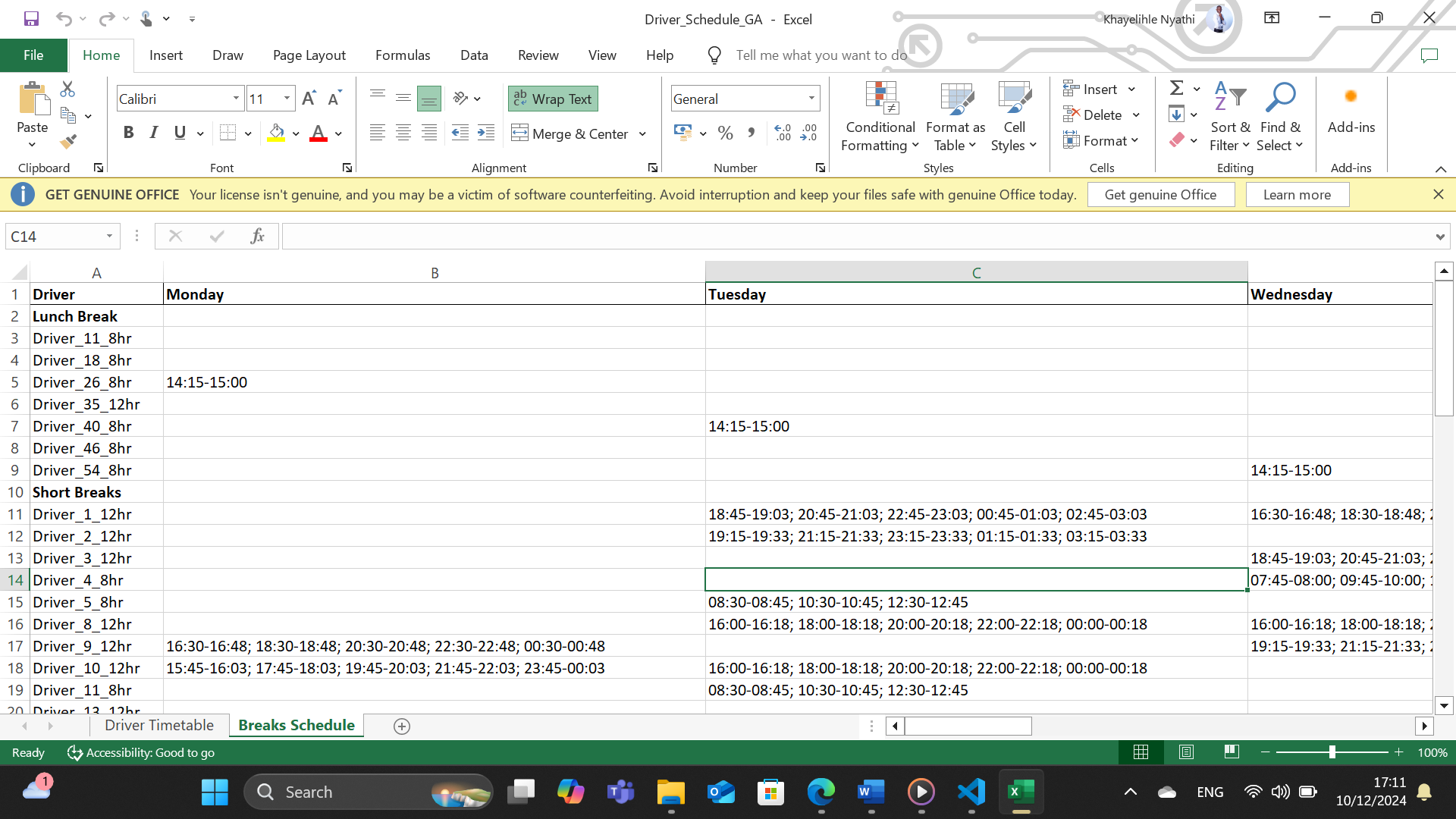


Рис 23 - Результаты в формат Excel (Breaks Schedule)

# Сравнение двух алгоритмов

Таблица 1

|  |  |  |
| --- | --- | --- |
| **Аспект** | **Полный перебор** | **Генетический алгоритм** |
| **Точность** | Всегда гарантирует нахождение точного оптимального решения за счет перебора всех возможных вариантов. | Предоставляет приближенные решения, но не гарантирует абсолютного оптимума. |
| **Детерминированность** | Да, всегда дает одинаковый результат. | Нет, результаты могут незначительно различаться из-за случайности в выборе, кроссовере и мутации. |
| **Масштабируемость** | Плохая, так как размер пространства решений экспоненциально растет с увеличением размера задачи. | Хорошая, может эффективно справляться с большими задачами благодаря линейной зависимости от параметров. |
| **Временная сложность** | O(C^T×D), где C: конфигурации, T: временные слоты, D: водители. | O(G×P×E), где G: поколения, P: размер популяции, E: время оценки. |
| **Сложность реализации** | Простая реализация, но невозможна для больших задач из-за комбинаторного взрыва. | Более сложная из-за настройки параметров (например, скорость мутации, размер популяции), но масштабируемая. |
| **Надежность** | Полностью надежен, если вычислительно выполним. | Высоко надежен для практических целей, но может сходиться к локальному оптимуму. |
| **Точность результатов** | 100% (всегда находит глобальный оптимум). | Высокая, но может упустить глобальный оптимум. |
| **Практичность** | Непрактичен для реальных задач, включающих большие или сложные проблемы. | Практичен для решения больших задач оптимизации. |
| **Общее Количество водителы** | 48 | 56 |
| Количество автобусов на маршруте | 4 (в зависимости от времени суток) | 4 (в зависимости от времени суток) |

# Список источников

1. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.
2. Holland, J. H. (1992). Adaptation in Natural and Artificial Systems.
3. Abdulhamid, S. M., et al. (2019). A hybridized genetic algorithm for university timetabling problems. Journal of King Saud University - Computer and Information Sciences.
4. Burke, E. K., et al. (2004). A multi-stage hybrid algorithm for timetable optimization. European Journal of Operational Research.
5. Mitchell, M. (1998). An Introduction to Genetic Algorithms.
6. Kendall, G., and Hussin, B. (2005). Scheduling and Timetabling: Problems and Solutions.
7. GeeksforGeeks: [Genetic Algorithms - GeeksforGeeks](https://www.geeksforgeeks.org/genetic-algorithms/)
8. [Метод грубой силы и его плюсы и минусы - GeeksforGeeks](https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/)
9. Eiben, A. E., and Smith, J. E. (2003). Introduction to Evolutionary Computing.
10. Blum, C., and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys.

# Приложение 1. Согласование результатов с выводом в Excel

1. **Import statements:**

Импортируйте необходимые библиотеки

import pandas as pd

from openpyxl import load\_workbook

from openpyxl.styles import Alignment

from openpyxl.styles import Font

1. **Извлечь необходимые данные и создать фреймы данных (перерыв)**

def extract\_break\_data(drivers):

    """

    Extract lunch and short break data from driver.breaks dynamically.

    Only include drivers with relevant breaks in the output.

    """

    lunch\_breaks\_data = {"Driver": []}

    short\_breaks\_data = {"Driver": []}

*# Initialize the days of the week*

    for day in WEEKDAYS + WEEKENDS:

        lunch\_breaks\_data[day] = []

        short\_breaks\_data[day] = []

*# Loop through drivers*

    for driver in drivers:

*# Collect lunch and short breaks for this driver*

        lunch\_breaks\_per\_day = []

        short\_breaks\_per\_day = []

        for day in WEEKDAYS + WEEKENDS:

*# Extract breaks for lunch vs short based on break type*

            lunch\_breaks = [

                f"{b[0]}-{b[1]}" for b in driver.breaks[day] if b[2] == "1-hour"

            ]

            short\_breaks = [

                f"{b[0]}-{b[1]}" for b in driver.breaks[day] if b[2] == "15-min"

            ]

*# Append breaks or empty strings for this day*

            lunch\_breaks\_per\_day.append("; ".join(lunch\_breaks) if lunch\_breaks else "")

            short\_breaks\_per\_day.append("; ".join(short\_breaks) if short\_breaks else "")

*# Include the driver in lunch data only if they have any lunch breaks*

        if any(lunch\_breaks\_per\_day):

            lunch\_breaks\_data["Driver"].append(driver.driver\_id)

            for day, breaks in zip(WEEKDAYS + WEEKENDS, lunch\_breaks\_per\_day):

                lunch\_breaks\_data[day].append(breaks)

*# Include the driver in short break data only if they have any short breaks*

        if any(short\_breaks\_per\_day):

            short\_breaks\_data["Driver"].append(driver.driver\_id)

            for day, breaks in zip(WEEKDAYS + WEEKENDS, short\_breaks\_per\_day):

                short\_breaks\_data[day].append(breaks)

*# Convert extracted data to DataFrames*

    lunch\_breaks\_df = pd.DataFrame(lunch\_breaks\_data)

    short\_breaks\_df = pd.DataFrame(short\_breaks\_data)

    return lunch\_breaks\_df, short\_breaks\_df

1. **сделайте то же самое для расписания водителя**

*# Extract breaks dynamically*

lunch\_breaks\_df, short\_breaks\_df = extract\_break\_data(drivers)

*# Prepare the bus-to-driver grouped schedule*

grouped\_schedule = []  *# To store grouped schedules by bus*

for bus in buses:

*# Add the bus row as a header*

    bus\_row = [str(bus.bus\_id)] + [""] \* (len(WEEKDAYS + WEEKENDS))

    grouped\_schedule.append(bus\_row)

*# Add rows for each driver assigned to the bus*

    unique\_drivers = set()

    for day in WEEKDAYS + WEEKENDS:

        for \_, \_, driver\_id in bus.schedule.get(day, []):

            unique\_drivers.add(driver\_id)

    for driver\_id in sorted(unique\_drivers):  *# Ensure consistent ordering*

        driver\_row = [str(driver\_id)]

        for day in WEEKDAYS + WEEKENDS:

*# Aggregate all shifts for this driver on the current day*

            shifts = [

                f"{start.strftime('%H:%M')} - {end.strftime('%H:%M')}"

                for start, end, driver in bus.schedule.get(day, [])

                if driver == driver\_id

            ]

            driver\_row.append(", ".join(shifts) if shifts else "")

        grouped\_schedule.append(driver\_row)

*# Convert grouped schedule to DataFrame*

columns = ["Driver"] + WEEKDAYS + WEEKENDS

bus\_schedule\_df = pd.DataFrame(grouped\_schedule, columns=columns)

1. **Используйте фреймы данных и преобразуйте их в Excel**

*# Restructure breaks for Lunch and Short Breaks*

lunch\_breaks\_filtered = lunch\_breaks\_df[lunch\_breaks\_df.drop(columns=["Driver"]).notna().any(axis=1)]

short\_breaks\_filtered = short\_breaks\_df[short\_breaks\_df.drop(columns=["Driver"]).notna().any(axis=1)]

lunch\_rows = [["Lunch Break"] + [""] \* (len(columns) - 1)] + lunch\_breaks\_filtered.values.tolist()

short\_break\_rows = [["Short Breaks"] + [""] \* (len(columns) - 1)] + short\_breaks\_filtered.values.tolist()

final\_breaks\_rows = lunch\_rows + short\_break\_rows

breaks\_final\_df = pd.DataFrame(final\_breaks\_rows, columns=columns)

*# Save DataFrame to an Excel file*

file\_path = r"Driver\_Schedule.xlsx"

with pd.ExcelWriter(file\_path, engine='openpyxl') as writer:

    bus\_schedule\_df.to\_excel(writer, sheet\_name="Driver Timetable", index=False)

    breaks\_final\_df.to\_excel(writer, sheet\_name="Breaks Schedule", index=False)

*# Open the Excel file using openpyxl*

wb = load\_workbook(file\_path)

1. **Структурируйте данные в Excel для ясности восприятия с помощью жирных заголовков и отступов.**

*# Define a function to make row headers bold*

def make\_row\_headers\_bold(sheet, header\_rows):

    for row\_idx in header\_rows:

        for cell in sheet[row\_idx]:

            cell.font = Font(bold=True)

*# Apply bold formatting to row headers*

for sheet\_name in wb.sheetnames:

    sheet = wb[sheet\_name]

*# Determine header rows based on the table*

    header\_rows = []

    for row\_idx, row in enumerate(sheet.iter\_rows(min\_row=1, max\_row=sheet.max\_row, values\_only=True), start=1):

        if row[0] and (row[0].startswith("Bus\_") or row[0] in ["Lunch Break", "Short Breaks"]):  *# Identify row headers*

            header\_rows.append(row\_idx)

    make\_row\_headers\_bold(sheet, header\_rows)

*# Define a function to adjust column width and enable text wrapping*

def auto\_adjust\_columns(sheet):

    for col in sheet.columns:

        max\_length = 0

        column = col[0].column\_letter  *# Get the column name*

        for cell in col:

            try:

                if cell.value and len(str(cell.value)) > max\_length:

                    max\_length = len(str(cell.value))

            except:

                pass

        adjusted\_width = max\_length + 2  *# Adding some extra padding*

        sheet.column\_dimensions[column].width = adjusted\_width

*# Apply wrap text formatting to all cells*

        for cell in col:

            cell.alignment = Alignment(wrap\_text=True)

*# Apply formatting to each sheet*

for sheet\_name in wb.sheetnames:

    sheet = wb[sheet\_name]

    auto\_adjust\_columns(sheet)

*# Save the modified workbook*

wb.save(file\_path)

print(f"Excel file saved with bold row headers: {file\_path}")

print("Driver Timetable:\n", bus\_schedule\_df)

print("\nBreaks Schedule:\n", breaks\_final\_df)

# Приложение 2 - Потоковая передача данных из HDFS и применение модели машинного обучения

1. *Import statements*:

Импортируйте необходимые библиотеки и модули, включая StreamingContext,

Github: [Algorithms/5 semester/Ньяти\_Каелиле\_БВТ2201\_КР.ipynb at master · KNyathi/Algorithms](https://github.com/KNyathi/Algorithms/blob/master/5%20semester/%D0%9D%D1%8C%D1%8F%D1%82%D0%B8_%D0%9A%D0%B0%D0%B5%D0%BB%D0%B8%D0%BB%D0%B5_%D0%91%D0%92%D0%A22201_%D0%9A%D0%A0.ipynb)

Excel file (Brute Force): [Algorithms/5 semester/Driver\_Schedule.xlsx at master · KNyathi/Algorithms](https://github.com/KNyathi/Algorithms/blob/master/5%20semester/Driver_Schedule.xlsx)

Excel file (Genetic Algorithm): [Algorithms/5 semester/Driver\_Schedule\_GA.xlsx at master · KNyathi/Algorithms](https://github.com/KNyathi/Algorithms/blob/master/5%20semester/Driver_Schedule_GA.xlsx)