**Assignment 7: Multi-Paradigm Problem Solving**

**Abstract**

This assignment explores how the same problem—calculating mean, median, and mode from a list of integers—can be solved using three different programming paradigms: procedural, functional, and object-oriented. I implemented the solution in C (procedural), OCaml (functional), and Python (object-oriented). Each approach came with its own syntax, logic style, and learning curve. This report highlights my experience working with each paradigm, the differences I noticed, and the challenges I faced.

**Introduction**

The goal was to create a simple statistics calculator that computes the **mean**, **median**, and **mode** of a list of numbers. However, instead of using just one language, I had to solve the problem using **three programming languages**, each representing a different paradigm: C (procedural), OCaml (functional), and Python (object-oriented). This helped me understand how programming paradigms affect code structure and problem-solving strategies.

## Implementation Summary

## Problem Statement

Given a list of integers, I implemented functions to calculate the following statistics:

* **Mean**: Average of all integers
* **Median**: Middle value after sorting
* **Mode**: Most frequent number(s) in the list

I wrote separate implementations in:

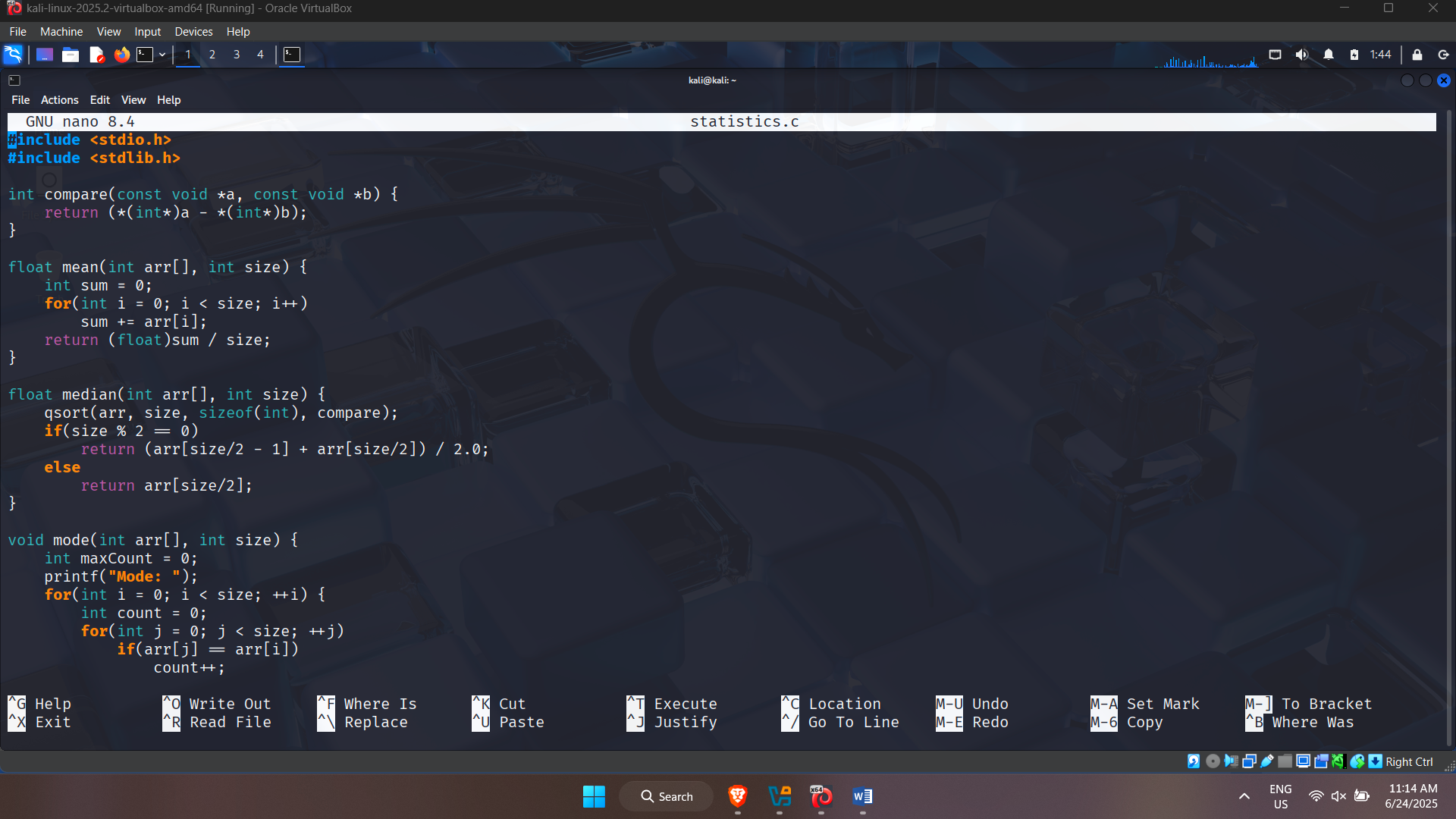
1. **C (procedural)**
2. **OCaml (functional)**
3. **Python (object-oriented)**

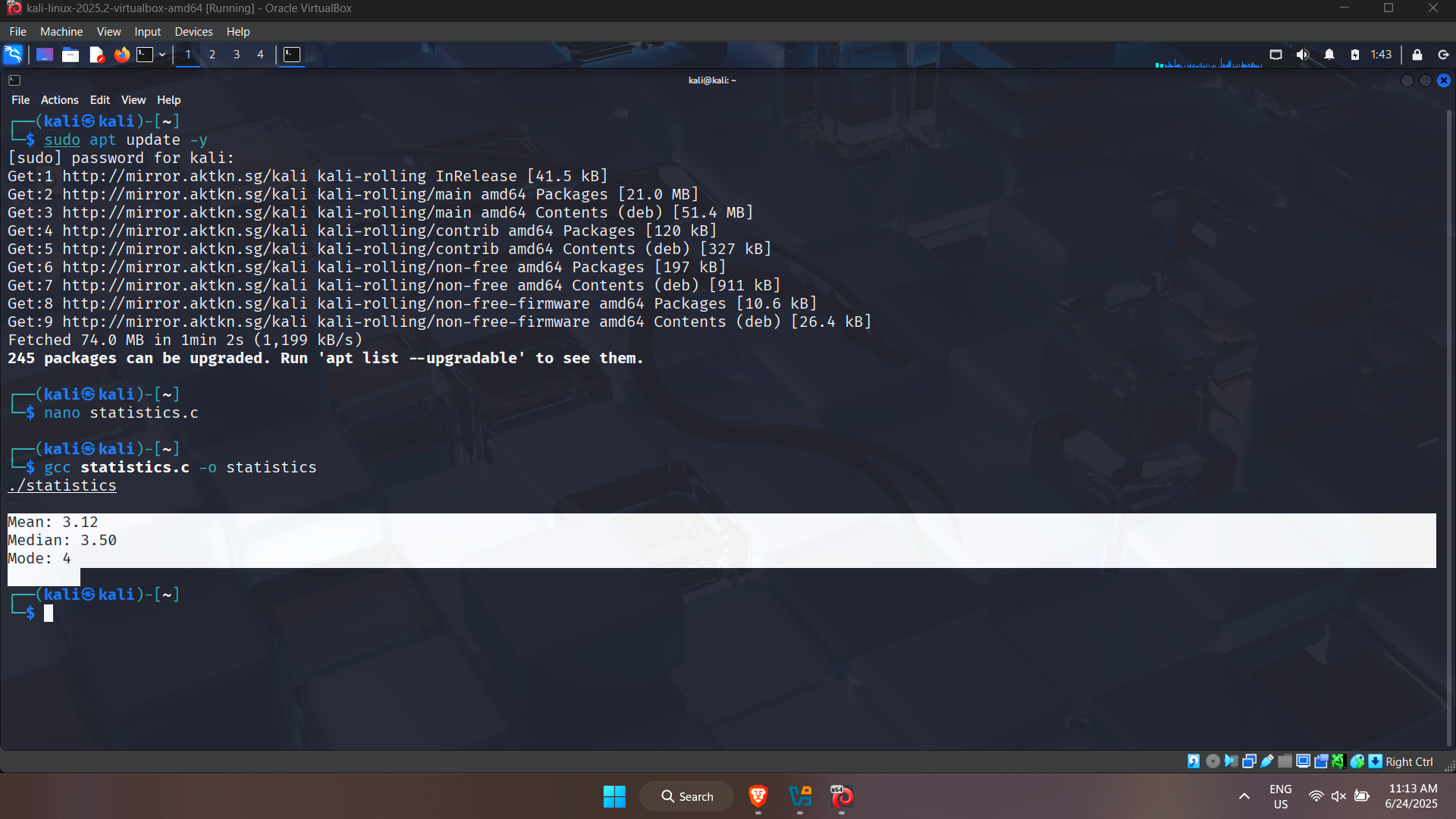
### 1. **C – Procedural Programming**

In C, I implemented the program using a procedural approach. I wrote separate functions for mean, median, and mode. Arrays were used to store the input, and I implemented sorting and frequency calculations manually. I used gcc in Kali Linux to compile the program:

gcc statistics.c -o statistics

./statistics





**Challenges:**

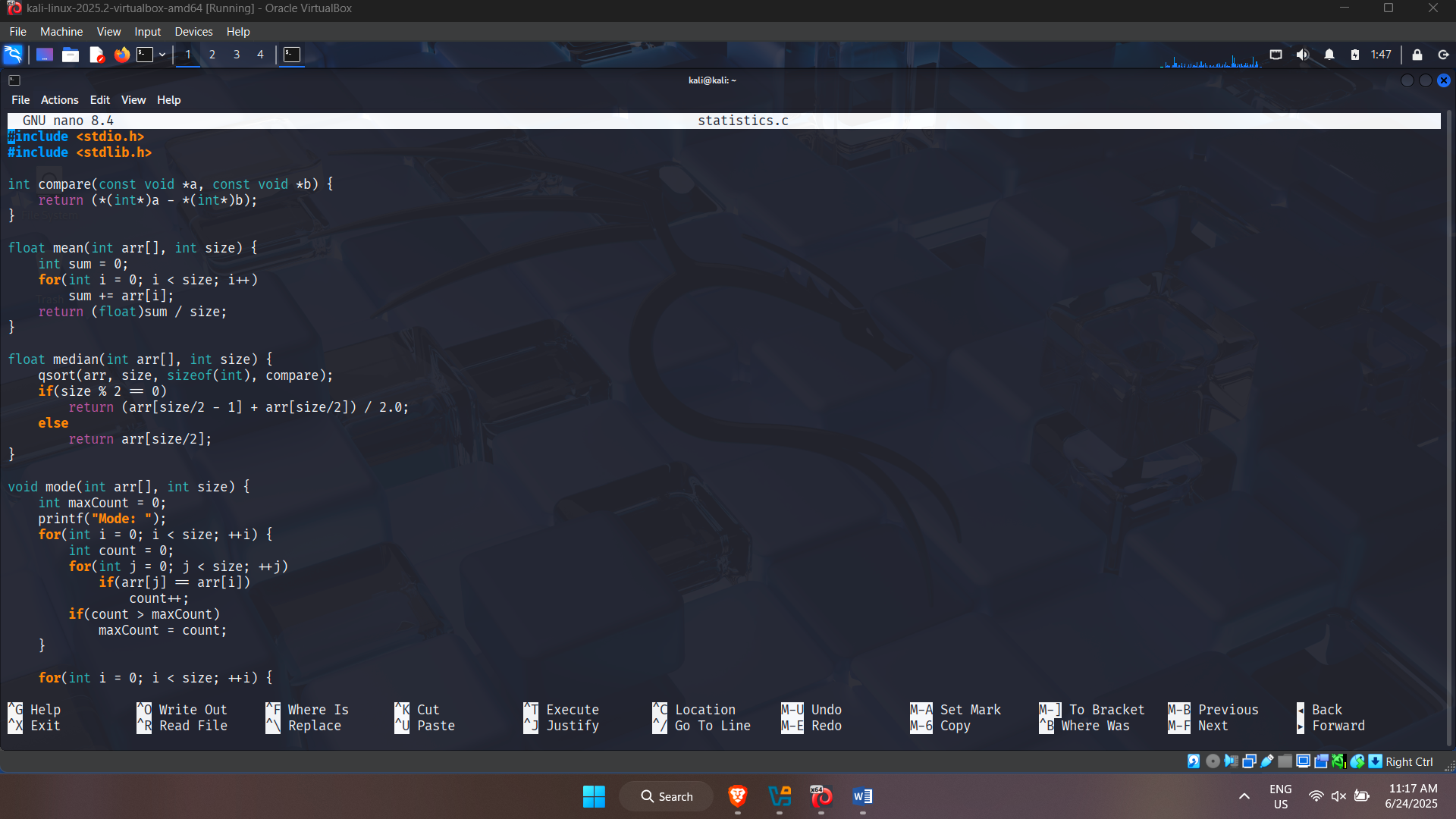
* No built-in functions for sorting or counting
* Manual memory management
* Verbose and less flexible for future changes

### 2. **OCaml – Functional Programming**

In OCaml, I used a functional approach with immutable data structures. The logic for mean, median, and mode was built using recursive functions and higher-order functions like List.fold\_left, List.sort, and List.filter.

I ran the program using the interactive OCaml shell (ocaml) and compiled versions with ocamlc.

ocaml statistics.ml





**Challenges:**

* No variables, only values
* Had to rethink logic using recursion and higher-order functions
* Syntax errors were difficult to catch early on

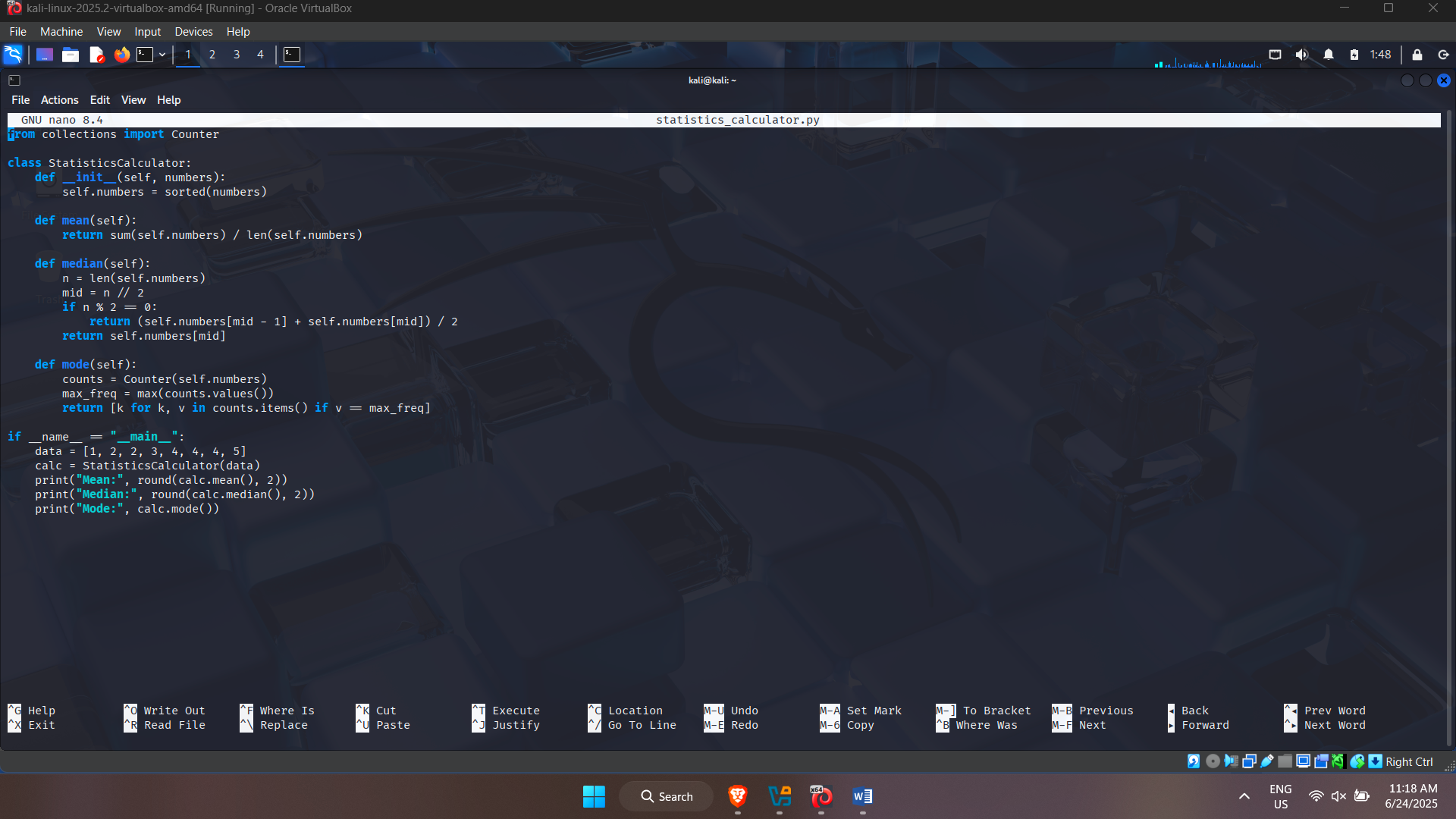
### **3**. **Python – Object-Oriented Programming**

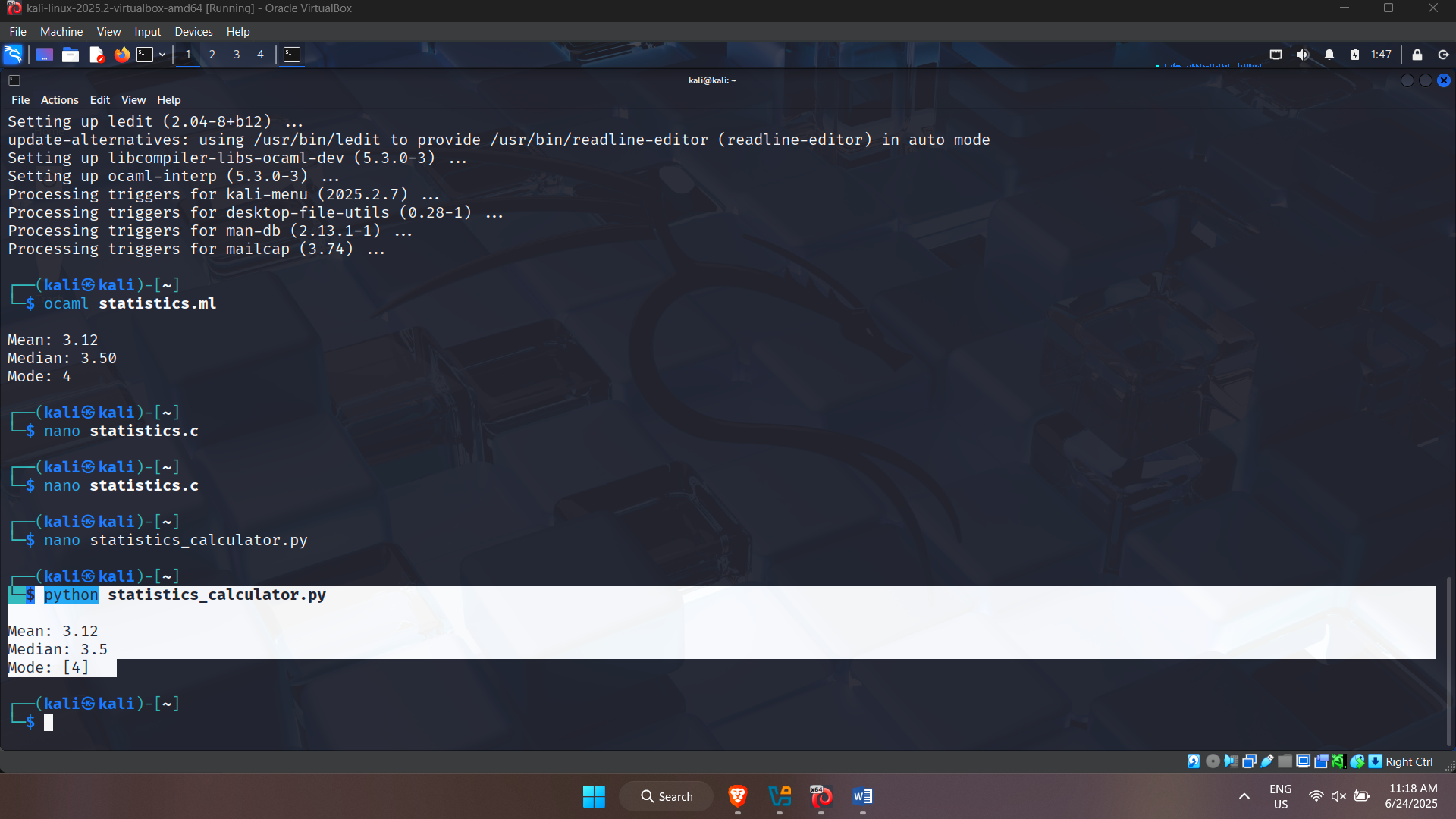
In Python, I used a class called StatisticsCalculator, with separate methods for each calculation (mean, median, mode). Python’s standard libraries made the implementation simple and clean.

python3 statistics.py

**Advantages:**

* Easy to implement and debug
* Built-in functions made sorting and frequency count effortless
* Very readable and modular code





**Challenges:**

* Almost none—Python made the implementation very readable and reusable
* Most suitable for beginners and rapid development

## Comparison Between Paradigms

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **C (Procedural)** | **OCaml (Functional)** | **Python(Object Oriented)** |
| Code Structure | Step-by-step | Pure functions, recursion | Class-based methods |
| Readability | Medium | Hard to read at first | Very readable |
| Reusability | Low | Medium | High |
| Learning Curve | Moderate | Steep | Easy |
| Memory Management | Manual | Automatic (GC) | Automatic (GC) |

## Conclusion

This assignment helped me understand how the same task can be handled in very different ways depending on the programming paradigm used. Using **Kali Linux** made the process smoother since I had access to all compilers and tools directly through the terminal. The procedural style in C gave me deep control, OCaml challenged my thinking with recursion and pure functions, and Python offered a balance of simplicity and clarity with its object-oriented approach.