# Final Project : T****ext Analyzer & Number Converter App****

This **Individual Final Project** on **Iteration** each student to design and implement a **“Text Analyzer & Number Converter”** in Python, showcasing mastery of iteration concepts: for and while loops, nested loops, loop control (break/continue), Boolean flags, and sequence processing. Students will process user input strings, count and categorize characters, search for patterns, compute numeric statistics, and convert numbers into binary. The project is comprehensive—covering all major iteration patterns from Topics 1–3—and will be submitted via a GitHub classroom repository.

## Project Description

Build a command‑line Python application named analyzer.py that:

1. **String Analysis**
   * **Counts**: total characters, vowels, consonants, digits, and punctuation.
   * **Detects**: the first and last occurrence of a user‑supplied substring.
   * **Lists**: all unique words (case‑insensitive), sorted alphabetically.
2. **Numeric Statistics**
   * **Accepts** a comma‑separated list of integers from the user.
   * **Computes**: sum, average (mean), minimum, and maximum using loops (no built‑ins like sum()).
   * **Flags**: numbers divisible by 3 or 5, printing them with a label.
3. **Binary Converter**
   * **Prompts** the user for a positive or negative integer.
   * **Converts** it to binary using the remainder‑division method (while loop), handling sign.
   * **Displays** the binary string.
4. **Menu & Loop Control**
   * Presents a **menu** to the user to choose any of the above functions or exit.
   * Uses a while True: loop with break/continue to manage the menu.

## Objectives for the Project

* **Apply** for loops to iterate over strings, lists, and ranges
* **Use** while loops for conditional repetition and number conversion.
* **Employ** break and continue to control loop flow.
* **Implement** nested loops for complex tasks (e.g., word extraction).
* **Manage** Boolean flags to detect conditions (e.g., substring found).
* **Practice** input/output and robust error handling.

## Requirements & Constraints

1. **No use of built‑in aggregators**: you must implement your own loops for counting and summing (no sum(), min(), max(), or count() methods).
2. **String processing**: use both index‑based (range(len(s))) and direct (for ch in s) loops at least once each.
3. **Modular code**: separate functionality into functions:
   * analyze\_text(s: str) -> dict
   * stats\_numbers(nums: list[int]) -> dict
   * to\_binary(n: int) -> str
   * main\_menu(): drives the program loop.
4. **Input validation**: reject invalid inputs with error messages and re‑prompt.
5. **Documentation**: include docstrings for every function and inline comments for loop logic.
6. **Style**: adhere to PEP 8 style guidelines; use meaningful variable names.

## Deliverables

* analyzer.py: the project source file.
* README.md in the root of your GitHub folder:
  + Brief project overview.
  + Instructions for running (e.g., python analyzer.py).
  + Sample input/output transcript.

## ## References

1. **Real Python**, “Python for Loops: The Pythonic Way”
2. **GeeksforGeeks**, “Loops in Python – For, While and Nested Loops”
3. **OpenBookProject**, “Iteration” chapter overview [openbookproject.net](https://openbookproject.net/thinkcs/python/english3e/iteration.html?utm_source=chatgpt.com)
4. **StackOverflow**, “Iterating each character in a string” [Dataquest](https://www.dataquest.io/blog/python-projects-for-beginners/?utm_source=chatgpt.com)
5. **Python Docs**, str.isdigit() and sequence methods [PYnative](https://pynative.com/python-if-else-and-for-loop-exercise-with-solutions/?utm_source=chatgpt.com)
6. **W3Schools**, while loop syntax and control flow [GeeksforGeeks](https://www.geeksforgeeks.org/loops-in-python/?utm_source=chatgpt.com)
7. **Real Python**, “while…else in Python” [GeeksforGeeks](https://www.geeksforgeeks.org/loops-in-python/?utm_source=chatgpt.com)
8. **Python Docs**, f-strings Guide [Dataquest](https://www.dataquest.io/blog/tutorial-advanced-for-loops-python-pandas/?utm_source=chatgpt.com)
9. **GeeksforGeeks**, “Prime checking up to sqrt(n)” (applies loop break logic) [GeeksforGeeks](https://www.geeksforgeeks.org/python-projects-for-final-year-students/?utm_source=chatgpt.com)
10. **Real Python**, “Floating Point Arithmetic: Issues” (for binary context) [StrataScratch](https://www.stratascratch.com/blog/mastering-loop-iterations-python-for-loop-index-explained/?utm_source=chatgpt.com)

Good luck!