

# Project Report: Phone User Generation and Filtering System

## 1. Introduction

This project focuses on creating a Python program that generates a dataset of phone users in the United States. Each user is assigned a random full name, a mobile operator (Verizon, AT&T, or T-Mobile), a phone number from predefined lists, and a subscription year. The program also allows the user to filter the displayed data by operator or view all users at once.

The objective was to simulate a small-scale user database and practice working with lists, loops, conditionals, and formatted console output.

---

## 2. Project Description

The system works by:

1. **Generating random user information**
    - o First name and last name are selected randomly.
    - o Subscription year is randomly generated between 2018 and 2025.
  2. **Assigning a phone operator**
    - o Three operators are available: Verizon, AT&T, and T-Mobile.
    - o Each operator has a predefined list of 10 numbers, assigned sequentially.
  3. **Storing user details**
    - o Each generated user is stored in a list as a dictionary containing:
      - Name
      - Operator
      - Subscription year
      - Phone number
  4. **Displaying data based on user choice**
    - o The program displays a menu:
      1. Verizon
      2. AT&T
      3. T-Mobile
      4. All
    - o Based on the selection, the program filters and prints formatted data.
- 

## 3. Code Summary

The code uses:

- `random` module for randomized user generation and importing the random library
- Lists for storing names, operators, and phone numbers

- Loops to generate 30 total users (10 for each operator)
  - Conditionals to filter users based on input
  - String formatting to display output in a clean table format
- 

## 4. Difficulties Faced

Throughout the development of this project, several challenges were encountered:

### 1. Ensuring Phone Number Distribution

Each operator had exactly 10 phone numbers available. Careful indexing was needed (`v_i`, `a_i`, `t_i`) to ensure no number was repeated and no index went out of range.

It was hard to collect available data (names, numbers)

### 2. Maintaining Even User Generation

Since all operators needed exactly 10 users, loops had to be structured carefully. Initially, errors occurred when the loop structure caused uneven distribution or repeated numbers.

### 3. Filtering Logic

When filtering users based on the operator selection, we had to:

- Create new lists dynamically
- Ensure operators were compared correctly using strings
- Handle unexpected inputs (e.g., anything outside 1–3 defaults to “All Users”)

### 4. Formatting Output

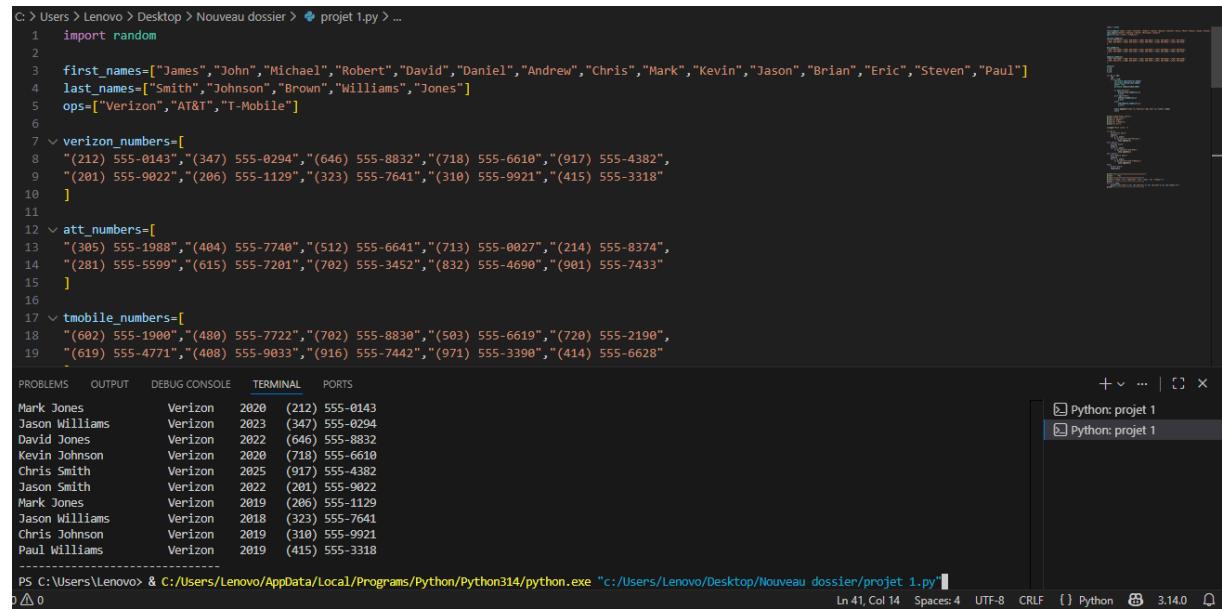
Printing user information in a readable, aligned table required:

- Mastery of formatted f-strings organized outputs
- Adjusting spacing for long names or operator names
- Ensuring the table remained aligned regardless of values

## 5. Conclusion

The project successfully simulates a phone user database and demonstrates essential programming concepts such as lists, dictionaries, loops, and user interaction. Despite some early difficulties with data distribution, filtering, and formatting and the logical thinking, the final system works efficiently and displays information in a clean, user-friendly manner.

This project provided valuable practice in Python programming, data handling, and console-based presentation of structured information.



The screenshot shows a code editor interface with a terminal window below it. The terminal window displays a table of phone numbers extracted from a Python script. The script imports random, defines lists of first names, last names, and operators, and then creates three nested lists of phone numbers for Verizon, AT&T, and T-Mobile. The table lists the first name, last name, operator, year of birth, and phone number for each entry. The terminal also shows the command used to run the script and the Python version.

```
C:\> Users > Lenovo > Desktop > Nouveau dossier > projet 1.py > ...
1 import random
2
3 first_names=["James","John","Michael","Robert","David","Daniel","Andrew","Chris","Mark","Kevin","Jason","Brian","Eric","Steven","Paul"]
4 last_names=["Smith","Johnson","Brown","Williams","Jones"]
5 ops=["Verizon","AT&T","T-Mobile"]
6
7 <verizon_numbers=[ 
8 "(212) 555-0143", "(347) 555-0294", "(646) 555-8832", "(718) 555-6610", "(917) 555-4382",
9 "(201) 555-9022", "(206) 555-1129", "(323) 555-7641", "(310) 555-9921", "(415) 555-3318"
10 ]
11
12 <att_numbers=[ 
13 "(305) 555-1988", "(404) 555-7740", "(512) 555-6641", "(713) 555-0027", "(214) 555-8374",
14 "(281) 555-5599", "(615) 555-7201", "(702) 555-3452", "(832) 555-4698", "(901) 555-7433"
15 ]
16
17 <tmobile_numbers=[ 
18 "(602) 555-1900", "(480) 555-7722", "(702) 555-8830", "(503) 555-6619", "(720) 555-2190",
19 "(619) 555-4711", "(408) 555-9033", "(916) 555-7442", "(971) 555-3390", "(414) 555-6628"
]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Mark Jones Verizon 2020 (212) 555-0143
Jason Williams Verizon 2023 (347) 555-0294
David Jones Verizon 2002 (646) 555-8832
Kevin Johnson Verizon 2028 (718) 555-6610
Chris Smith Verizon 2025 (917) 555-4382
Jason Smith Verizon 2022 (201) 555-9022
Mark Jones Verizon 2019 (206) 555-1129
Jason Williams Verizon 2018 (323) 555-7641
Chris Johnson Verizon 2019 (310) 555-9921
Paul Williams Verizon 2019 (415) 555-3318
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Lenovo/Desktop/Nouveau dossier/projet 1.py"
Ln 41, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
```

---

## Members :

---

- *Adollah Larbi cherif*
  - *Khali ishak*
  - *Mouas ayoub*
  - *Bouzebra med Amine*
-