Cleaning Collected CSV Data in Python

Here's a step-by-step approach to cleaning collected CSV data in Python, using the Pandas library:

1. Import necessary libraries:

Python

import pandas as pd

Use code with caution.

2. Load the CSV data:

Python

data = pd.read\_csv("your\_data.csv")

Use code with caution.

3. Explore the data:

Use data.head() to view the first few rows.

Utilize data.info() to get an overview of data types, missing values, and memory usage.

Employ data.describe() to get summary statistics (mean, median, standard deviation) for numerical columns.

4. Handle missing values:

Identify missing values using data.isnull().sum().

Decide on an appropriate strategy:

Drop rows/columns: Use data.dropna() with appropriate parameters like inplace=True (modifies the original DataFrame) and subset arguments (e.g., subset=["column1"]).

Fill missing values: Replace with mean/median/mode or specific values (e.g., data["column1"].fillna(0)). Be cautious with this approach, as it can mask underlying issues.

Impute missing values: Use techniques like K-Nearest Neighbors (KNN) or interpolation. This is more advanced and requires careful consideration of the data and chosen method.

5. Handle data types:

Check data types using data.dtypes.

Convert data types as needed using methods like:

data["column"] = pd.to\_numeric(data["column"], errors="coerce") (convert to numeric, handling errors)

data["column"] = pd.to\_datetime(data["column"]) (convert to datetime)

6. Deal with duplicates:

Identify duplicates using data.duplicated().sum().

Remove duplicates using data.drop\_duplicates(inplace=True).

7. Clean and standardize strings:

Remove leading/trailing whitespaces: data["column"] = data["column"].str.strip().

Convert to lowercase/uppercase: data["column"] = data["column"].str.lower() or .str.upper().

Fix typos (if needed): Explore regular expressions or fuzzy matching libraries.

8. Validate and address inconsistencies:

Use domain knowledge and data exploration to identify inconsistencies and potential errors.

Correct mistakes or remove problematic data based on your specific analysis needs.

9. Save cleaned data:

Save the cleaned data to a new CSV file using data.to\_csv("cleaned\_data.csv", index=False).

Example:

Python

**import pandas as pd**

**# Load data (replace "your\_data.csv" with your actual file path)**

**data = pd.read\_csv("your\_data.csv")**

**# Explore data**

**print(data.head())**

**# Handle missing values (example: drop rows with missing values)**

**data.dropna(inplace=True)**

**# Handle data types (example: convert "amount" to numeric)**

**data["amount"] = pd.to\_numeric(data["amount"], errors="coerce")**

**# Remove duplicates**

**data.drop\_duplicates(inplace=True)**

**# Clean strings (example: lowercase and remove whitespaces)**

**data["description"] = data["description"].str.lower().str.strip()**

**# Save cleaned data**

**data.to\_csv("cleaned\_data.csv", index=False)**

Use code with caution.

Remember, data cleaning is an iterative process. You may need to revisit and refine your cleaning steps based on the specific characteristics and requirements of your data.