SOFTWARE & ARTIFICIAL INTELLIGENCE

# Python for Research

Turn messy raw data into a clear, defensible result — the specific Python skills that separate a research analysis from a toy script.

**SKILL LEVEL:** BEGINNER–INTERMEDIATE **BUILD TIME:** 1–2 WEEKENDS **EST. COST:** $0

> Before you start: this blueprint assumes you can already write basic Python (variables, loops, functions) — see SA-001 if that's not yet comfortable. This one is about using Python on real data, not learning the language from scratch.

## 01 Objective

Writing Python that runs is not the same skill as writing Python that produces a research result someone else can trust. Research code has different priorities than a script you write for yourself: every step needs to be reproducible, every transformation of the data needs to be visible rather than done by hand, and every chart needs to hold up to someone asking "how exactly did you get this number?" By the end of this blueprint, you'll be able to take a raw, messy dataset and turn it into a cleaned, analyzed, visualized result — with a clear trail showing exactly how you got there.

## 02 Prerequisites

- Comfortable with basic Python: variables, loops, functions, and running a script or notebook
- A dataset to practice on — a public dataset from Kaggle or your own field/lab data works equally well
- No statistics background required to start, though it helps once you reach Section 6

## 03 Recommended Toolkit

| TOOL | PURPOSE | NOTES |
| --- | --- | --- |
| Jupyter Notebook / JupyterLab | Interactive environment for exploring data step by step | Free — install via `pip install jupyterlab`, or use Google Colab with no install |
| pandas | Loading, cleaning, and reshaping tabular data | Free — the core library for almost all research data work |
| NumPy | Numerical operations and array math | Free — pandas is built on top of it |
| Matplotlib / Seaborn | Plotting and visualization | Free — Seaborn gives cleaner default charts with less code |
| SciPy | Statistical tests and scientific computing functions | Free — needed once you move past descriptive statistics |
| Git | Version control for your analysis code (not your raw data) | Free — see Section 4's note on what belongs in version control |

## 04 Tools Required

- Python 3.10+ with pip, or a Google Colab account (no install required)
- A code editor or Jupyter interface

> REPRODUCIBILITY & DATA INTEGRITY
> Never hand-edit a raw data file in Excel or a text editor — if you need to fix or filter something, do it with code, so the exact transformation is visible and repeatable. Keep your original raw file untouched and write your cleaned version to a new file. This single habit is what allows someone else — including you, in six months — to verify exactly how a number was produced, and it's the difference between a result and a claim.

## 05 The Research Data Workflow

Research analysis in Python follows a consistent pipeline, regardless of field. Keeping these stages distinct — instead of cleaning, analyzing, and plotting all in one tangled block — is what makes an analysis easy to check and easy to redo when new data arrives.


FIG. 1 — THE FIVE STAGES OF A REPRODUCIBLE PYTHON RESEARCH ANALYSIS

| STAGE | TYPICAL TASKS | KEY LIBRARY |
|---|---|---|
| Cleaning | Handling missing values, fixing types, removing duplicates | pandas |
| Analysis | Descriptive statistics, grouping, hypothesis tests | NumPy / SciPy |
| Visualization | Distribution plots, trend lines, comparisons | Matplotlib / Seaborn |
| Reporting | Markdown cells explaining each decision and result | Jupyter Notebook |

## 06 Step-by-Step Workflow

1. **Load the raw data without modifying it.** Read it directly into pandas with `pd.read_csv()` — the original file stays untouched on disk.
2. **Inspect before you clean.** Run `.info()`, `.describe()`, and `.head()` to understand what you're actually working with before changing anything.
3. **Handle missing and malformed data explicitly.** Decide deliberately whether to drop, fill, or flag missing values — and write down why in a markdown cell, not just in your head.
4. **Compute descriptive statistics first.** Mean, median, standard deviation, and counts by group — before reaching for any more advanced test, make sure you understand the basic shape of the data.
5. **Visualize before you conclude.** A histogram or scatter plot often reveals outliers or patterns that summary statistics alone hide.
6. **Run the appropriate statistical test, if your question calls for one.** A t-test or chi-square test in SciPy takes one line of code — the harder part is choosing the right one for your data and question.
7. **Document every step in markdown cells as you go.** Explain what each code cell does and why, in plain language — treat the notebook as a lab notebook, not just a script.
8. **Save the cleaned data separately from the raw data.** Export to a new file like `data_cleaned.csv`, never overwriting the original.

## 07 Code Walkthrough

This walks through the cleaning and analysis stages on a small research-style dataset — notice that each step is explicit and visible, rather than a single dense line doing five things at once.

```
import pandas as pd
import numpy as np
from scipy import stats

# 1. Load raw data — never modify this file directly
raw = pd.read_csv("survey_responses_raw.csv")

# 2. Inspect before touching anything
print(raw.info())
print(raw.describe())

# 3. Handle missing values explicitly, with a documented decision
#    Decision: drop rows missing the outcome variable; fill missing age with the median
cleaned = raw.dropna(subset=["response_score"]).copy()
cleaned["age"] = cleaned["age"].fillna(cleaned["age"].median())

# 4. Descriptive statistics by group
summary = cleaned.groupby("group")["response_score"].agg(["mean", "std", "count"])
print(summary)

# 5. A statistical test comparing two groups
group_a = cleaned[cleaned["group"] == "A"]["response_score"]
group_b = cleaned[cleaned["group"] == "B"]["response_score"]
t_stat, p_value = stats.ttest_ind(group_a, group_b, equal_var=False)
print(f"t = {t_stat:.3f}, p = {p_value:.4f}")
```

```
# 6. Save cleaned data separately — raw file stays untouched
cleaned.to_csv("survey_responses_cleaned.csv", index=False)
```

*Note: equal_var=False runs Welch's t-test, which doesn't assume the two groups have equal variance — a safer default than the standard t-test unless you've checked that assumption.*

## 08 Common Mistakes

| MISTAKE | WHY IT HURTS | FIX |
|---|---|---|
| Editing the raw data file by hand | Makes the analysis impossible to verify or redo when new data arrives | Only transform data through code, and always write to a new file |
| Running notebook cells out of order | Produces results that silently depend on hidden execution history | Regularly use "Restart Kernel and Run All" to confirm the notebook works top to bottom |
| Dropping missing values without checking why they're missing | Can silently bias results if missingness isn't random | Check whether missing values cluster around a specific group or condition before dropping them |
| Choosing a statistical test before understanding the data's shape | Many tests assume a distribution shape (like normality) that your data may not have | Visualize first — a histogram takes one line and can save you from a wrong test |
| Charts with no axis labels or units | A chart a reviewer can't interpret is functionally useless, however correct the underlying analysis | Label every axis, every unit, and every legend before considering a chart finished |

## 09 Where This Goes Next

FROM A CLEAN ANALYSIS TO A RESEARCH CONTRIBUTION

The workflow in this blueprint is the exact foundation **SA-003 (Machine Learning Foundations)** builds on — a model is only as good as the cleaned, understood data that feeds it. If your project involves original field or survey data, pair this blueprint with the Research & Scientific Writing series (especially RS-003, Computational & Data Science Research) for how to write up your methodology formally, or with **ES-001 (Designing Your First Environmental Research Project)** if your data comes from fieldwork.