

Instructions Someone Else Can Finish

Turn one task into prerequisites, observable steps and safe recovery paths

01

02

03

How to use this guide

People writing internal help articles or product instructions who want a small reproducible review method.

Rewrite a rough file-conversion article using an original working Python fixture, synthetic observation notes, failure branches and a complete capstone answer. Learn to verify the result and maintain the supported task contract.

- State a verifiable finish condition and prerequisites.
- Write actions paired with observable checkpoints.
- Add specific recovery branches and stop conditions.
- Use observations to revise and retest a bounded instruction article.

Scope and limitations

- The included Python converter handles only the documented two-column exercise contract.
- Synthetic observations are teaching examples, not real participant research.
- No universal usability, accessibility or task-completion guarantee is made.
- Python installation and a full operating-system support matrix are outside this workbook.

Published by KitForma. Edition 1.0. Sources checked 1 October 2026. People and businesses in worked examples are fictional. This is an independent publication, not endorsed by the platforms mentioned.

Copyright 2026 KitForma. Usage, support and refund terms are included in LICENSE.txt in the purchased package.

Contents

How to use this guide	2
Contents	3
Define the finished task before writing steps	4
Sources and tools	6

01

Define the finished task before writing steps

Useful instructions begin with a finish condition. A reader should know what they will have when the task is complete and how to recognize it. This workbook uses a small supplied Python program that converts an exercise CSV into a JSON file while preserving the two columns as strings. The intended output is a new file containing two records, including the identifier 0012 without losing its leading zero. The task is not to install a programming environment, clean arbitrary business data or upload anything to a service. Those boundaries prevent a short instruction article from quietly becoming an impossible promise to solve every related problem.

Our fictional documentation editor, Sol, has received a rough note: 'Run the converter and save the result.' That note leaves the reader to guess which file is the input, where the command runs, what counts as success and whether an existing output may be replaced. The supplied program deliberately refuses to overwrite an output file. It also rejects unexpected headers and duplicate identifiers. These behaviors create specific checkpoints and recovery branches for the article. A good writer describes the actual behavior of the tool being documented instead of making a smooth path sound universal. When the tool changes, the article's assumptions must be checked again.

Write a task contract with four parts: starting materials, reader prerequisites, observable result and exclusions. For this exercise the starting materials are `convert_records.py` and `records.csv` in the same working folder. The reader needs a supported Python 3 installation available through an appropriate command. The result is `records.json` with

two string-valued records. The article should explicitly exclude confidential data and use only the fictional fixtures during practice. A contract this small helps you reject unnecessary sections and identify missing information. It also gives a reviewer a fair basis for saying whether the instructions are complete.

Sources and tools

[S1] Google Technical Writing One

<https://developers.google.com/tech-writing/one>

Checked: 2026-10-01

[S2] W3C WAI: Headings

<https://www.w3.org/WAI/tutorials/page-structure/headings/>

Checked: 2026-10-01

[S3] Digital.gov plain language guides

<https://digital.gov/guides/plain-language/>

Checked: 2026-10-01

[S4] Python standard library CSV documentation

<https://docs.python.org/3/library/csv.html>

Checked: 2026-10-01

Text comparison

Optional comparison of the rough and revised articles; judgment and real task testing remain separate.

<https://kitforma.com/en/text-diff/>

Inside the complete guide

Record prerequisites as checks, not vague assumptions

Write one action with a visible checkpoint

Show an expected result without hiding important details

Put failure branches where the reader needs them

Observe a novice attempt without doing the task for them

Revise the article around the observed confusion

Maintain instructions as a versioned promise

Capstone: rewrite the rough conversion article

Answer key and a reusable instruction review