

# From Prototyping to Production: Converting Jupyter Notebooks to Python Scripts

A Guide for Efficient and Maintainable Code

Michael Borck

## Table of contents

<b>Python Workflows</b>	<b>2</b>
<b>Hybrid Approach: Notebook-Based Workflows</b>	<b>2</b>
<b>Prototyping in Jupyter Notebooks</b>	<b>2</b>
<b>Refactoring Notebooks into Python Scripts</b>	<b>2</b>
<b>Example: Refactoring a Notebook</b>	<b>2</b>
<b>Using GitHub to Share Your Project</b>	<b>3</b>
<b>Setting Up a GitHub Repository</b>	<b>3</b>
<b>Example: Git Commands</b>	<b>3</b>
<b>Creating a README.md</b>	<b>4</b>
<b>Example: README.md</b>	<b>4</b>
Usage . . . . .	4
Features . . . . .	4
License . . . . .	4
<b>Summary</b>	<b>5</b>

## Python Workflows

- No one-size-fits-all solution; depends on project requirements, team size, data complexity, collaboration needs, and development speed.
- Use notebooks for exploratory data analysis and prototyping.
- Use Python scripts for long-term development and production code.
- Hybrid approaches combine benefits of both.

## Hybrid Approach: Notebook-Based Workflows

- Use notebooks for exploratory data analysis and prototyping.
- Refactor the code into Python scripts for production.
- Maintain the interactive and flexible nature of notebooks for initial development.
- Ensure the code is organised and efficient for long-term maintenance.

## Prototyping in Jupyter Notebooks

**Why Use Jupyter Notebooks?** - Interactive coding environment - Easy to test and debug code - Great for data analysis and visualisation

## Refactoring Notebooks into Python Scripts

**Why Refactor?** - Organise and structure your code - Make it reusable and modular - Prepare for deployment and sharing

## Example: Refactoring a Notebook

Before: Jupyter Notebook

```
# notebook.ipynb
import pandas as pd

data = pd.read_csv('data.csv')
result = data.describe()
print(result)
```

After: Python Script

```
# analysis.py
import pandas as pd

def analyse_data(file_path):
    data = pd.read_csv(file_path)
    result = data.describe()
    return result

if __name__ == "__main__":
    result = analyse_data('data.csv')
    print(result)
```

## Using GitHub to Share Your Project

**Why Use GitHub?** - Version control with Git - Share code with the world - Collaborate on projects

## Setting Up a GitHub Repository

**Step-by-Step Guide** 1. Create a GitHub account 2. Create a new repository 3. Clone the repository to your local machine 4. Add your project files 5. Commit and push your changes

## Example: Git Commands

### Initialise and Push to GitHub

```
# Initialise git in your project directory
git init

# Add your files to the repository
git add .

# Commit your changes
git commit -m "Initial commit"

# Add the remote repository URL
git remote add origin https://github.com/yourusername/yourrepository.git
```

```
# Push your changes to GitHub
git push -u origin master
```

## Creating a README.md

**Why Include a README.md?** - Provide an overview of your project - Explain how to install and use it - Highlight key features and dependencies

## Example: README.md

```
# Project Title

## Overview
Brief description of your project.

## Installation
```bash
pip install your_project
```

## Usage

```
from your_project import your_function
result = your_function()
print(result)
```

## Features

- Feature 1
- Feature 2

## License

MIT ““

## Summary

- Prototype in Jupyter Notebooks
- Refactor into Python scripts
- Share your project on GitHub