Table of contents

1 Simple Jupyter Notebooks for Pension Specs

1.1 Why This Works

- **Markdown cells** = Documentation (like Word)
- **Code cells** = Calculator (like Excel formulas)
- **Run cells** = See results instantly

1.2 Setup (one time)

1. Install VS Code: https://code.visualstudio.com/
2. Install Python: https://www.python.org/downloads/
3. Install Quarto: https://quarto.org/docs/get-started/
4. In VS Code, install these extensions (click Extensions icon):

- “Jupyter”
- “Quarto”

1.3 Using the Template

1.3.1 Opening

- Open VS Code
- File → Open → select the `.ipynb` file

1.3.2 The Two Types of Cells

Markdown cells (documentation): - Double-click to edit - Press `Shift+Enter` to save and move on

Code cells (calculations): - Click to select - Edit the numbers - Press `Shift+Enter` to run and see results

1.3.3 Running the Calculation

1. Edit the member data in the first code cell:

```
pension_at_leaving = 15000.00    # Change this
gmp_at_leaving = 3500.00         # Change this
years_of_revaluation = 7         # Change this
years_early = 3                  # Change this
gender = "M"                     # M or F
```

2. Click “Run All” at the top (or press `Shift+Enter` on each cell)

3. See results at the bottom

1.3.4 The Code is Just Arithmetic

This:

```
excess_at_leaving = pension_at_leaving - gmp_at_leaving
```

Is the same as this Excel formula:

`=B2-B3`

Just with names instead of cell references.

1.4 Key Shortcuts

Action	Shortcut
Run cell and move to next	`Shift+Enter`
Run cell and stay	`Ctrl+Enter`
Run all cells	Click “Run All” button
Add cell below	`B` (when cell selected)
Delete cell	`DD` (press D twice)
Change to Markdown	`M`
Change to Code	`Y`

1.5 Rendering with Quarto

Quarto converts your notebooks to professional documents. Output goes to the `_output/` folder.

1.5.1 From the Command Line

```
# Render one notebook to HTML (default)
quarto render active_to_retirement_spec.ipynb

# Render to PDF
quarto render active_to_retirement_spec.ipynb --to pdf

# Render to Word
quarto render active_to_retirement_spec.ipynb --to docx

# Render ALL notebooks in the folder
quarto render
```

1.5.2 From VS Code

With the Quarto extension installed: 1. Open a notebook 2. Press `Ctrl+Shift+K` (or `Cmd+Shift+K` on Mac) 3. Select output format

Or click the "Render" button in the top toolbar.

1.5.3 What the `_quarto.yml` File Does

The project config file sets defaults for all specs: - **Table of contents** on every document - **Section numbering** (1, 1.1, 1.2, etc.) - **Consistent styling** across HTML, PDF, and Word - **Runs all code** when rendering (so outputs are always current) - **Embeds resources** in HTML (single file, easy to share)

1.5.4 PDF Requirements

To render PDFs, you need LaTeX. Install with:

```
quarto install tinytex
```

1.6 Legacy Export (without Quarto)

If you prefer VS Code's built-in export: - **To PDF**: File → Export → PDF - **To HTML**: File → Export → HTML

1.7 Tips

1. **Test different scenarios** by changing the input values and re-running
2. **Add notes** by inserting new Markdown cells
3. **The code builds on itself** - run cells in order from top to bottom

---

1.8 Markdown Cheat Sheet

Use these in Markdown cells to format your documentation.

1.8.1 Headings

```
# Heading 1 (largest)
## Heading 2
### Heading 3
#### Heading 4
```

1.8.2 Text Formatting

```
**bold text**
*italic text*
***bold and italic***
~~strikethrough~~
`inline code`
```

1.8.3 Lists

Bullet list:

```
- Item one
- Item two
  - Nested item
  - Another nested
- Item three
```

Numbered list:

```
1. First step
2. Second step
3. Third step
```

1.8.4 Tables

```
| Column 1 | Column 2 | Column 3 |
|----------|----------|----------|
| Data     | Data     | Data     |
| More     | More     | More     |
```

Alignment:

```
| Left     | Center   | Right    |
|:---------|:--------:|---------:|
| text     | text     | text     |
```

1.8.5 Links and Images

```
[Link text](https://example.com)
![Image alt text](path/to/image.png)
```

1.8.6 Code Blocks

For formulas or code snippets:

~~~
```python
result = value * factor
```
~~~

Or inline: `` `variable_name` ``

1.8.7 Blockquotes

```
> This is a quote or important note.
> It can span multiple lines.
```

1.8.8 Horizontal Rule

```
---
```

1.8.9 Checkboxes (task lists)

```
- [x] Completed task
- [ ] Incomplete task
```

1.9 Python Cheat Sheet

Common patterns used in these templates.

1.9.1 Variables

```
# Simple assignment
pension = 15000.00
years = 10
gender = "M"
```

1.9.2 Arithmetic

```
# Basic math
total = value1 + value2
difference = value1 - value2
product = value1 * value2
quotient = value1 / value2

# Compound interest / revaluation
factor = (1 + rate) ** years
revalued = original * factor
```

1.9.3 Print Statements

```
# Basic print
print("Hello")

# F-strings (formatted)
print(f"Pension: {pension}")
print(f"Pension: {pension:,.2f}")      # With commas and 2 decimals
print(f"Pension: £{pension:>12,.2f}")  # Right-aligned, 12 chars wide
```

1.9.4 Conditionals

```
if gender == "M":
    factor = male_factor
else:
    factor = female_factor
```

1.9.5 Dictionaries (lookup tables)

```
# Define
factors = {0: 1.000, 1: 0.940, 2: 0.882}

# Look up
result = factors[years_early]
result = factors.get(years_early, 0.720)  # With default
```

1.10 Formatting Numbers

Format	Example	Result
`{x}`	`f"{15000}"`	15000
`{x:,}`	`f"{15000:,}"`	15,000
`{x:.2f}`	`f"{15000:.2f}"`	15000.00
`{x:,.2f}`	`f"{15000:,.2f}"`	15,000.00
`{x:>10}`	`f"{15000:>10}"`	`15000` (right-aligned)
`{x:<10}`	`f"{15000:<10}"`	`15000` (left-aligned)
`{x:.4f}`	`f"{0.826:.4f}"`	0.8260
`{x:.2%}`	`f"{0.035:.2%}"`	3.50%

1.11 Useful Symbols

Copy-paste these into your Markdown cells:

Symbol	Usage
£	Currency
×	Multiplication
÷	Division
	Less/greater than or equal
	Not equal
→	Arrow (workflow)
	Check/cross marks
•	Bullet point
—	Em dash

1.12 Template Structure

All pension spec templates follow this pattern:

1. **Header** — Scheme name, version, author, date
2. **Purpose** — What this spec calculates
3. **Member Data** — Editable input values
4. **Scheme Parameters** — Rules (usually don't change)
5. **Calculation** — Step-by-step with explanations
6. **Summary** — All inputs and outputs in one place
7. **Reference Tables** — Factor tables, etc.

8. **Edge Cases** — How to handle special situations
9. **Sign-Off** — Author, reviewer, approver

1.13 Troubleshooting

Problem	Solution
Cell shows `[*]` and hangs	Kernel is busy. Click Stop button or restart kernel
`NameError: name 'x' is not defined`	Run cells in order from top. A cell is using a variable defined earlier
Numbers look wrong	Check input values in Member Data section
Can't edit a cell	Make sure you're clicking inside the cell, not just selecting it
Results don't update	Re-run the cell after making changes (`Shift+Enter`)

1.14 Version Control with Jupytext

For team collaboration, pair notebooks with Python files:

```
# One-time setup for a notebook
jupytext --set-formats ipynb,py:percent active_to_retirement_spec.ipynb
```

This creates `active_to_retirement_spec.py` that stays in sync. Commit the `.py` file to Git for clean diffs.

Sync after editing:

```
jupytext --sync active_to_retirement_spec.ipynb
```