

Code Window

Quarto Extension

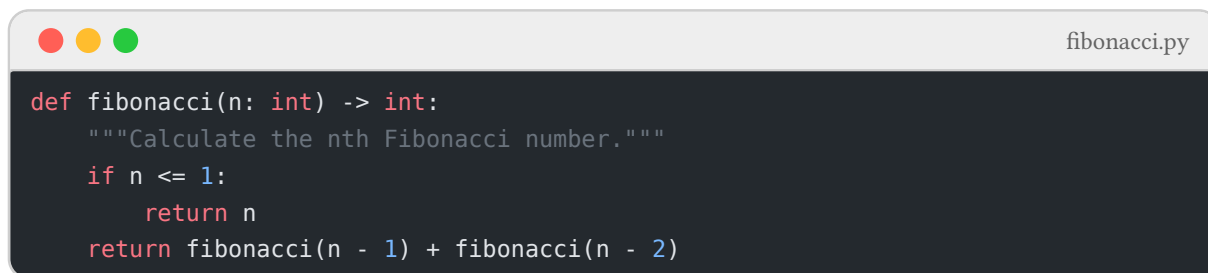
Mickaël CANOUIL, *Ph.D.*

Table of contents

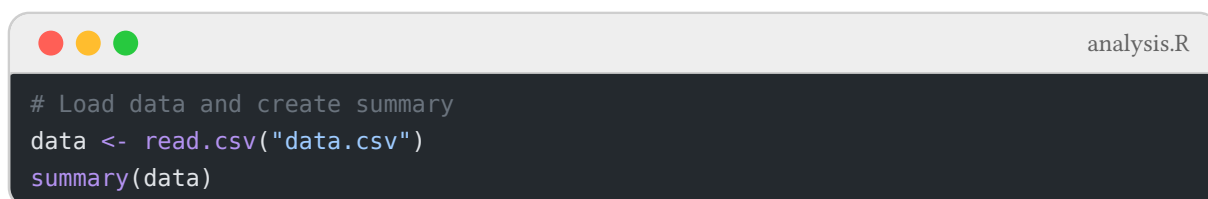
Explicit Filename	2
Auto-Generated Filename	3
Plain Code Block	4
Disabled Auto-Filename	5
Per Block	5
Globally	5
Highlighted Lines	6
Collapsible Code Windows (HTML)	7
Closed by Default	7
Open by Default	7
Window Styles	8
macOS Style (default)	8
Windows Style	8
Default Style	8
Code Annotations	9
Annotations with Explicit Filename	9
Annotations with Auto-Filename	9
Annotations Spanning Multiple Lines	9
Annotations without Window Chrome	10
Configuration	10

Explicit Filename

Code blocks with a `filename` attribute display a window header with the filename. The decoration style depends on the `style` option (default: `"macos"`).

A code editor window with a light gray title bar containing three colored window control buttons (red, yellow, green) on the left and the filename 'fibonacci.py' on the right. The main area is a dark gray background with Python code in a light color.

```
def fibonacci(n: int) -> int:
    """Calculate the nth Fibonacci number."""
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)
```

A code editor window with a light gray title bar containing three colored window control buttons (red, yellow, green) on the left and the filename 'analysis.R' on the right. The main area is a dark gray background with R code in a light color.

```
# Load data and create summary
data <- read.csv("data.csv")
summary(data)
```

Auto-Generated Filename

With `auto-filename: true` (the default), code blocks without explicit filenames automatically show the language name in small-caps styling.

```
PYTHON
def greet(name: str) -> str:
    """Return a greeting message."""
    return f"Hello, {name}!"
```

```
R
# Create sample data
data <- data.frame(
  x = 1:10,
  y = rnorm(10)
)
summary(data)
```

```
BASH
#!/bin/bash
echo "Hello, World!"
```

Plain Code Block

Code blocks without a language are not affected by the extension.

```
This is a plain code block without any language specified.  
No window decoration is applied here.
```

Disabled Auto-Filename

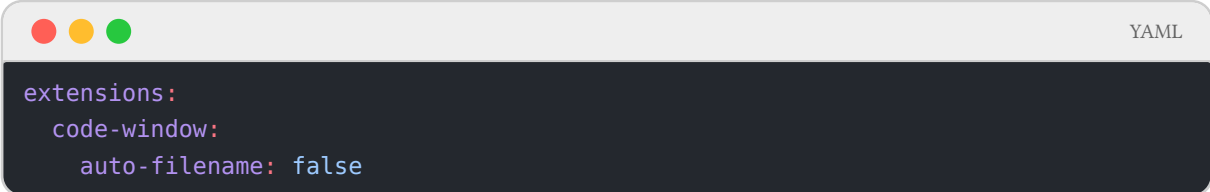
Per Block

Set `code-window-no-auto-filename="true"` on a single block to suppress the auto-generated filename without changing the global setting.

```
print("No auto-filename on this block")
```

Globally

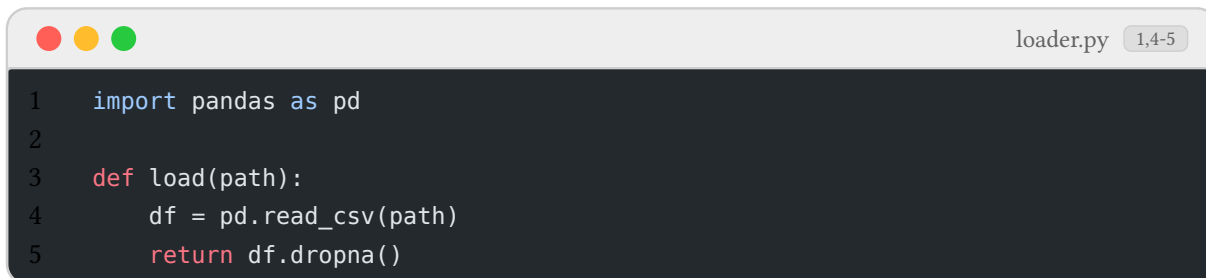
To disable auto-generated filenames for the entire document, set `auto-filename: false` in the extension configuration. Only code blocks with an explicit `filename` attribute will display window decorations.

A screenshot of a code editor window with a title bar containing three colored circles (red, yellow, green) on the left and the text 'YAML' on the right. The editor area has a dark background and displays the following YAML configuration:

```
extensions:  
  code-window:  
    auto-filename: false
```

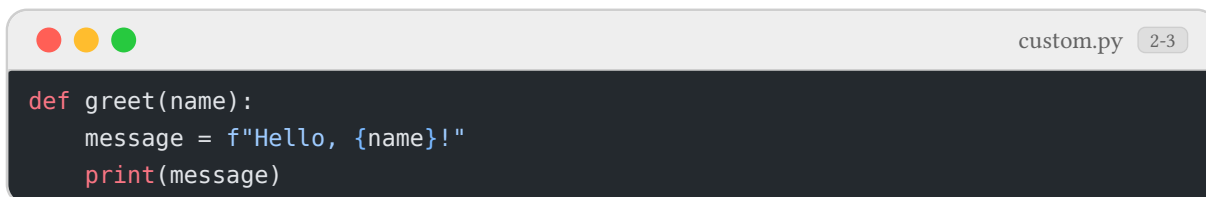
Highlighted Lines

When a code block uses Quarto's `code-line-numbers` attribute, the line specification is shown as a chip next to the filename.



```
1 import pandas as pd
2
3 def load(path):
4     df = pd.read_csv(path)
5     return df.dropna()
```

Use `code-window-lines` to override the displayed text without affecting Quarto's highlighting:



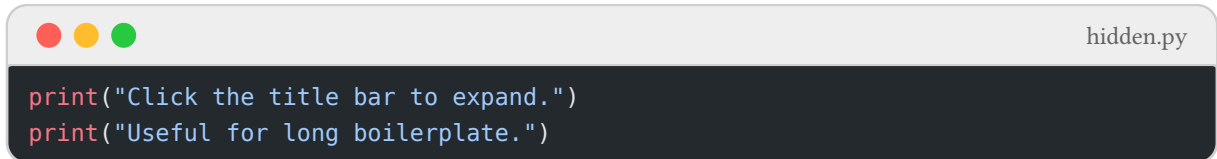
```
def greet(name):
    message = f"Hello, {name}!"
    print(message)
```

Disable the chip globally with `lines-label: false` in the extension config.

Collapsible Code Windows (HTML)

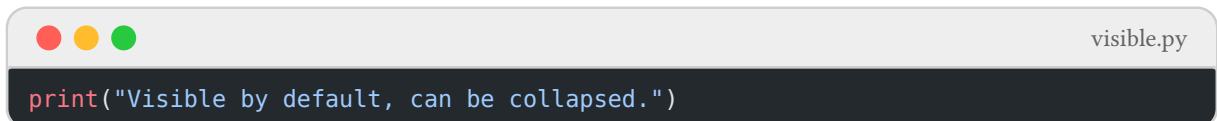
Set `code-window-collapse` to wrap a code block in a `<details>` element. The title bar becomes the `<summary>`. This feature is HTML-only; Typst and PDF output is unaffected.

Closed by Default



A screenshot of a code editor window. The title bar is light gray and contains three colored window control buttons (red, yellow, green) on the left and the filename 'hidden.py' on the right. The code area is dark gray and contains two lines of Python code: `print("Click the title bar to expand.")` and `print("Useful for long boilerplate.")`. The title bar is collapsed, meaning the code is not visible through it.

Open by Default



A screenshot of a code editor window. The title bar is light gray and contains three colored window control buttons (red, yellow, green) on the left and the filename 'visible.py' on the right. The code area is dark gray and contains one line of Python code: `print("Visible by default, can be collapsed.")`. The title bar is open, meaning the code is visible through it.

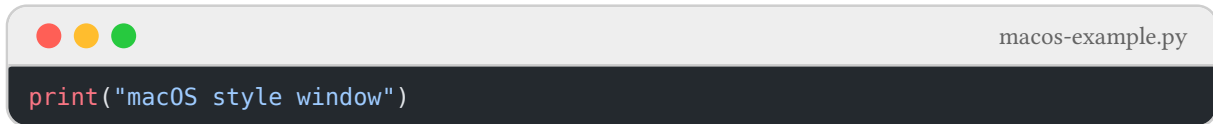
Apply collapsing to every code window by setting `collapse: closed` (or `open`) in the extension config.

Window Styles

Three decoration styles are available via the `style` option. The global style can be set in the document configuration. Individual blocks can override the style with the `code-window-style` attribute.

macOS Style (default)

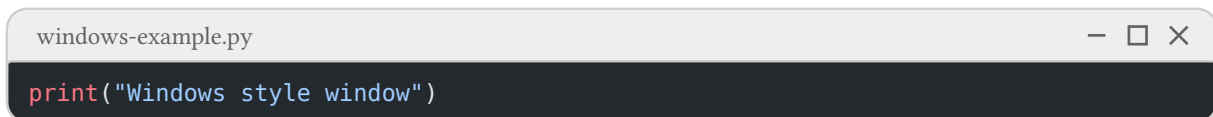
Traffic light buttons on the left, filename on the right.

A screenshot of a macOS-style window. The title bar is light gray and contains three colored window control buttons (red, yellow, green) on the left and the filename "macos-example.py" on the right. The main content area is dark gray and contains the Python code `print("macOS style window")` in a light-colored font.

```
print("macOS style window")
```

Windows Style

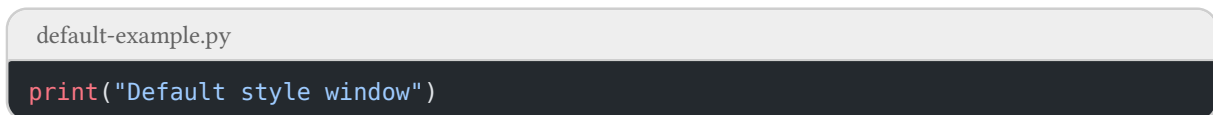
Minimise, maximise, and close buttons on the right, filename on the left.

A screenshot of a Windows-style window. The title bar is light gray and contains the filename "windows-example.py" on the left and three window control buttons (minimize, maximize, close) on the right. The main content area is dark gray and contains the Python code `print("Windows style window")` in a light-colored font.

```
print("Windows style window")
```

Default Style

Plain filename on the left, no window decorations.

A screenshot of a default-style window. The title bar is light gray and contains the filename "default-example.py" on the left. There are no window control buttons. The main content area is dark gray and contains the Python code `print("Default style window")` in a light-colored font.

```
print("Default style window")
```

Code Annotations

i Note

Typst code-annotations support and filename attribute handling are temporary hot-fixes. They will be removed once Quarto natively supports these features (see [quarto-dev/quarto-cli#14170](#)). The extension will then focus on **auto-filename** and **code-window-style** features.

Code annotations work standalone and together with code-window styling.

Annotations with Explicit Filename

```
annotated.py
import pandas as pd ①
df = pd.read_csv("data.csv") ②
summary = df.describe() ③
```

- ① Import the pandas library.
- ② Load data from a CSV file.
- ③ Generate summary statistics.

Annotations with Auto-Filename

```
PYTHON
def greet(name: str) -> str: ①
    return f"Hello, {name}!" ②
result = greet("World") ③
```

- ① Define a function with type hints.
- ② Use an f-string for interpolation.
- ③ Call the function and store the result.

Annotations Spanning Multiple Lines

A single annotation number can appear on several consecutive lines. Only the first occurrence receives a back-label to avoid duplicates.

```
pipeline.py
def process(data): ①
    cleaned = clean(data) ①
    validated = validate(cleaned) ①
    result = transform(validated) ②
    return result ③
```

- ① Multi-step input preparation (cleaning and validation).
- ② Apply the main transformation.
- ③ Return the final result.

Annotations without Window Chrome

Set `code-window-enabled="false"` on a block to disable window chrome while keeping annotations.

```
library(ggplot2)
ggplot(mtcars) +
  aes(x = mpg, y = hp) +
  geom_point()
```

- ① Load the ggplot2 package.
- ② Initialise a plot with the mtcars dataset.
- ③ Map aesthetics.
- ④ Add a point geometry layer.

Configuration

Set the global style in the document front matter:

```
extensions:
  code-window:
    enabled: true
    auto-filename: true
    style: "macos"
    wrapper: "code-window"
  hotfix:
    code-annotations: true
    skylighting: true
    typst-title: true
```

Each hotfix accepts either a boolean or a map with `enabled` and `quarto-version` keys for per-hotfix version thresholds:

```
extensions:
  code-window:
    hotfix:
      code-annotations: true
      skylighting:
        enabled: false
      typst-title:
        quarto-version: "1.10.0"
```

Override per block with the `code-window-style` attribute:

```
```{python filename="example.py" code-window-style="windows"}
print("Windows style for this block only")
```
```