



---

# Visual Studio Code for Python Development

Carlos Gavidia-Calderon



# Python IDEs at REG

There's no mandatory IDE at REG, but it seems there's a preferences towards VS Code. With PyCharm as a runner-up.

The screenshot shows a web browser window with the title 'Configuring Your Editor' from the 'REG Handbook'. The page content discusses editor configuration, mentioning Python development and generalizing for other languages. It also covers topics like General, EditorConfig usage, and the overlap between EditorConfig and Flake8. A sidebar on the right lists various handbook sections such as 'Configuring Your Editor', 'General', 'VS code', 'Extensions', etc.

Configuring Your Editor

Many people will configure their editor to suit their own needs and preferences.

It is possible to do this so that there are common standards between collaborators in the code base, while allowing for differences in individuals' configurations.

Some of the topics discussed below are heavily focused on python development. In due course we should generalise this to cover other languages.

## General

Some common, particular formatting/encoding options can be configured for a range of editors/IDEs by using `EditorConfig` in your repo. A `.editorconfig` needs to be placed in the root of your repo.

Many (but not all) editors support `EditorConfig` files allowing different members of your project team to use their own favourite editor while maintaining common coding standards. Some editors support `EditorConfig` natively; others require a plugin to support `EditorConfig`.

There is some overlap in functionality between `EditorConfig` and `Flake8`. Crudely `EditorConfig` applies standards to files as they are

- Configuring Your Editor
- General
- VS code
- Extensions
- Settings
- Selecting the Python Interpreter
- Linting
- Formatting
- Running Tests
- Debugging
- Other Tips and Tricks

---

# Bibliography

From the official Visual Studio Code website:

- Getting Started with Python in VS Code.
- Using Git source control in VS Code.
- Code Navigation.
- Data Science in VS Code tutorial.

---

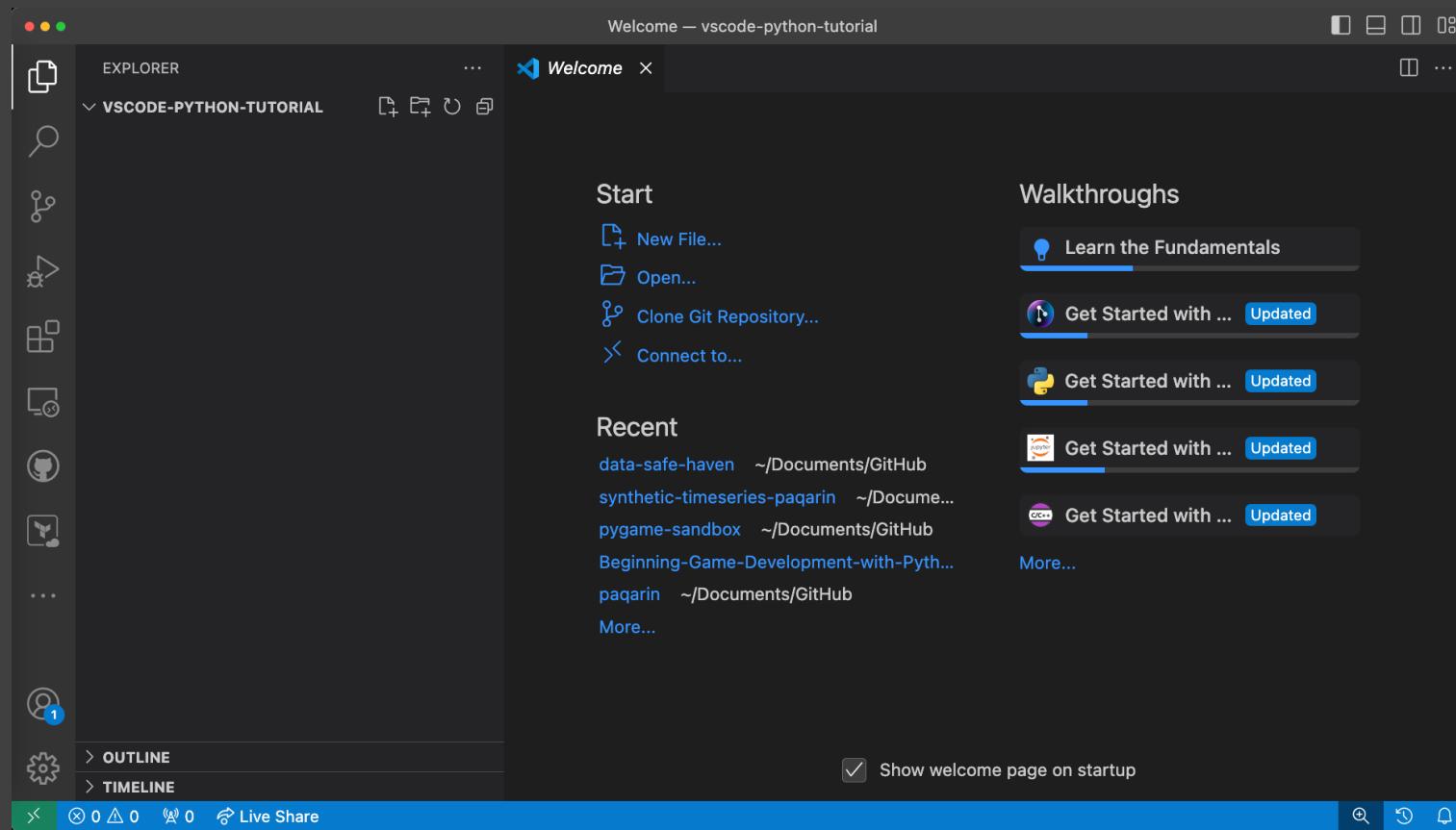
# Software Requirements

1. Install a Python interpreter, like Miniconda.
2. Install Git.
3. Create a GitHub Account.
4. Install Visual Studio Code.
5. Install the Python extension.
6. Download the Titanic data file.

---

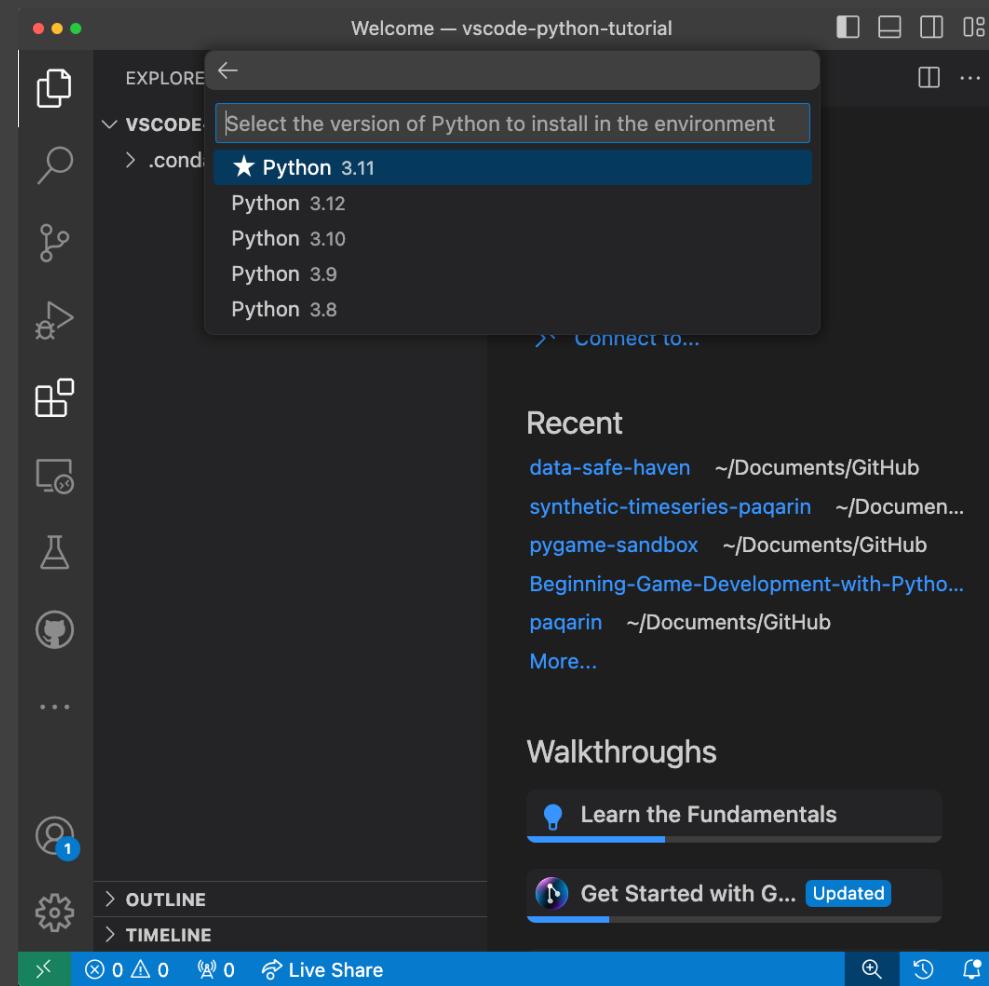
# Getting Started with Python

# File > Open Folder > Workspace Folder

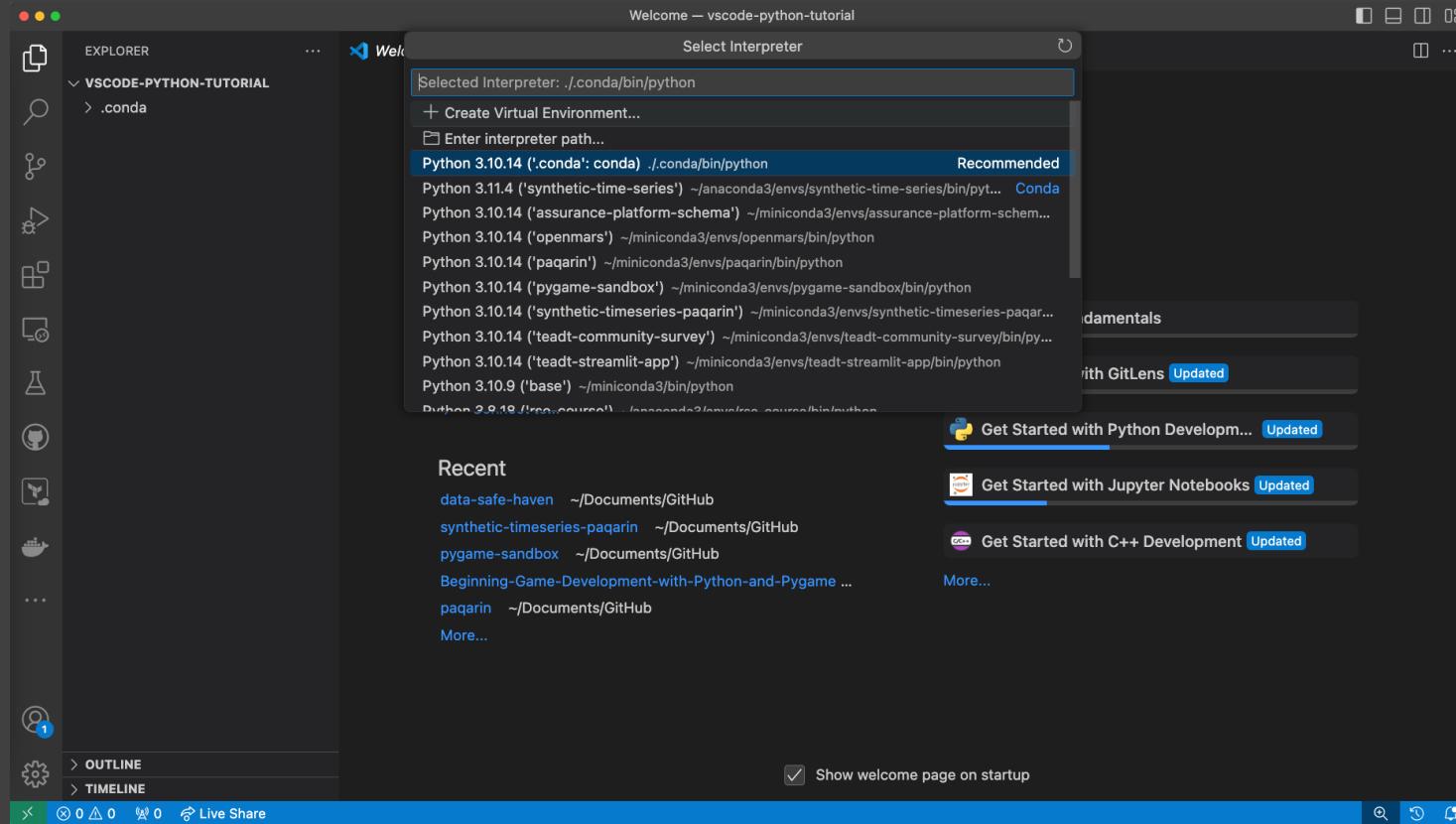


# Python: Create Environment

1. Create a Conda Environment.
2. Install Python 3.10 in the environment.



# Command> Python: Select Interpreter



# Python: Reading a CSV file

```
import pandas as pd  
  
file_name: str = "~/data/titanic3.csv"  
data_frame: pd.DataFrame = pd.read_csv(file_name)  
  
print(data_frame.head())
```

# File > New File

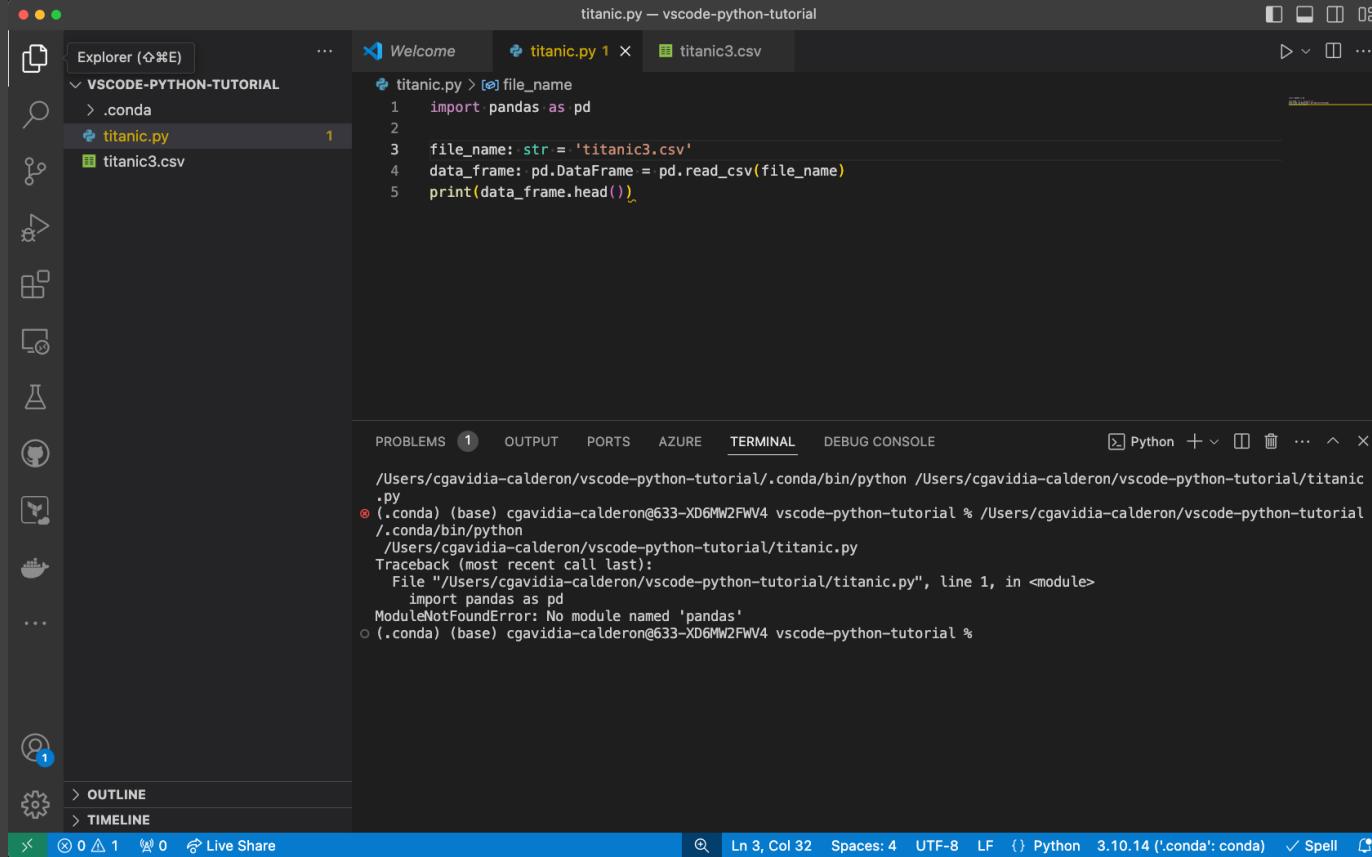
A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "titanic.py — vscode-python-tutorial". The left sidebar has a "VS CODE PYTHON TUTORIAL" folder containing ".conda", "titanic.py" (which is currently selected), and "titanic3.csv". The main editor area contains the following Python code:

```
1 import pandas as pd
2
3 file_name: str = 'titanic3.csv'
4 data_frame: pd.DataFrame = pd.read_csv(file_name)
5 print(data_frame.head())
```

The cursor is at the end of the fifth line, and a code completion dropdown is open, listing various string methods for the `str` type, such as `capitalize`, `casefold`, `center`, etc.

The bottom status bar shows the following information: "X 1 △ 1 ⚡ 0 Live Share", "Ln 4, Col 50 Spaces: 4 UTF-8 LF {} Python 3.10.14 ('.conda': conda)", and "Spell".

# Click the "Run Python File" button



# Command> Terminal: Create New Terminal

A screenshot of the Visual Studio Code interface. The left sidebar shows the Explorer view with files: titanic.py, .condo, and titanic3.csv. The main area shows the code editor with the following Python script:

```
import pandas as pd
file_name: str = 'titanic3.csv'
data_frame: pd.DataFrame = pd.read_csv(file_name)
print(data_frame.head())
```

The bottom navigation bar has tabs: PROBLEMS, OUTPUT, PORTS, AZURE, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is selected, showing a terminal window with the following output:

```
(.condo) (base) cgavidia-calderon@633-XD6MW2FWV4 vscode-python-tutorial % python -m pip install pandas
Collecting pandas
  Using cached pandas-2.2.2-cp310-cp310-macosx_11_0_arm64.whl.metadata (19 kB)
Collecting numpy>=1.22.4 (from pandas)
  Using cached numpy-1.26.4-cp310-cp310-macosx_11_0_arm64.whl.metadata (61 kB)
Collecting python-dateutil>=2.8.2 (from pandas)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas)
  Using cached pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.8.2->pandas)
  Using cached six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Using cached pandas-2.2.2-cp310-cp310-macosx_11_0_arm64.whl (11.3 MB)
Using cached numpy-1.26.4-cp310-cp310-macosx_11_0_arm64.whl (14.0 MB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached pytz-2024.1-py2.py3-none-any.whl (505 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-dateutil, pandas
Successfully installed numpy-1.26.4 pandas-2.2.2 python-dateutil-2.9.0.post0 pytz-2024.1 six-1.16.0 tzdata-2024.1
```

The status bar at the bottom shows: < 0 △ 1 ⌂ 0 Live Share. The bottom right corner shows the Python icon and the text "3.10.14 ('.conda': conda)".

# Right Click > Run Python

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "titanic.py — vscode-python-tutorial". The Explorer sidebar on the left lists files: ".conda", "titanic.py" (which is selected), and "titanic3.csv". The main editor area displays the code for "titanic.py":

```
titanic.py > {} pd
1 import pandas as pd
2
3 file_name: str = 'titanic3.csv'
4 data_frame: pd.DataFrame = pd.read_csv(file_name)
5 print(data_frame.head(5))
```

The terminal below shows the output of running the script:

```
/Users/cgavidia-calderon/vscode-python-tutorial/.conda/bin/python /Users/cgavidia-calderon/vscode-python-tutorial/titanic.py
ModuleNotFoundError: No module named 'pandas'
```

A context menu is open over the line "import pandas as pd" in the editor. The menu items include:

- Go to Definition
- Go to Type Definition
- Go to References
- Peek
- Find All References
- Rename Symbol
- Change All Occurrences
- Format Document
- Format Document With...
- Refactor...
- Source Action...
- Open Changes
- Cut
- Copy
- Copy As
- Paste
- Spelling
- Run in Interactive Window
- Run Python
- Command Palette...

The "Run Python File in Terminal" option is highlighted in blue.

The status bar at the bottom shows: "Ln 1, Col 20 Spaces: 4 UTF-8 LF Python 3.10.14 ('.conda': conda)" and "Spell".

# Python: Plotting Survivors by Age and Fare

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_name: str = "~/data/titanic3.csv"
data_frame: pd.DataFrame = pd.read_csv(file_name)

figure, axis = plt.subplots(ncols=2, figure=(30, 5))
sns.violinplot(x="survived", y="Age", hue="sex", data=data_frame,
ax=axis[0])
sns.violinplot(x="survived", y="Fare", hue="sex", data=data_frame,
ax=axis[1])

plt.show()
```

# A Buggy Python Script

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a project folder named "VS CODE-PYTHON-TUTORIAL" containing ".conda", "titanic.py" (selected), and "titanic3.csv".
- Editor:** The "titanic.py" file is open, displaying code to load data and create violin plots. Lines 9 and 10 are underlined in red, indicating syntax errors.
- Terminal:** Shows a Python environment running, with the last few lines of the traceback visible:

```
(.conda) (base) cgavidia-calderon@633-XD6MW2FWV4 vscode-python-tutorial % /Users/cgavidia-calderon/vscode-python-tutorial/.conda/bin/python /Users/cgavidia-calderon/vscode-python-tutorial/titanic.py
Traceback (most recent call last):
  File "/Users/cgavidia-calderon/vscode-python-tutorial/titanic.py", line 9, in <module>
    sns.violinplot(x="survived", y="Age", hue="sex", data=data_frame, ax=ax[0])
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/categorical.py", line 1725, in violinplot
    p = _CategoricalPlotter(
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/categorical.py", line 67, in __init__
    super().__init__(data=data, variables=variables)
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/_base.py", line 634, in __init__
    self._assign_variables(data, variables)
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/_base.py", line 679, in _assign_variables
    plot_data = PlotData(data, variables)
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/_core/data.py", line 58, in __init__
    frame, names, ids = self._assign_variables(data, variables)
  File "/Users/cgavidia-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/seaborn/_core/data.py", line 232, in _assign_variables
```

**Status Bar:** Shows file path "titanic.py — vscode-python-tutorial", line numbers "Ln 11, Col 11", and other settings like "Spaces: 4", "UTF-8", "LF", "Python 3.10.14 (.conda: conda)", "4 Spell", and "Live Share".

# The Python Debugger Extension

The screenshot shows the Visual Studio Code Marketplace interface. On the left, there's a sidebar with various icons for file operations like copy, paste, search, and refresh. The main area displays a list of extensions under the heading "EXTENSIONS: MARKETPLACE". A search bar at the top of this list contains the text "python debugger". The first result is the "Python Debugger" extension by Microsoft, which is highlighted with a yellow background. This extension has a rating of 34ms and is described as a "Python Debugger extension using debugpy.". Below it is the "Python" extension by Microsoft, rated 63ms, and the "Python Extension Pack" by Don Jayamanne, which includes 8.3M installs and a rating of 4.5. Further down are "Python C++ Debugger" by BeniBenj, "Python Resource Monitor" by 2kai2kai2, "Python Development" by demystifying-javascript, "Python Debugger (PyDev)" by Fabio Zadrozny, and "Python (PyDev)" by Python (PyDev). The status bar at the bottom shows icons for file operations and a "Live Share" button.

Extension: Python Debugger — vscode-python-tutorial

EXTENSIONS: MARKETPLACE

python debugger

**Python Debugger** v2024.6.0

Microsoft [microsoft.com](#) | 28,189,517 | ★★

Python Debugger extension using debugpy.

Disable | Uninstall | ⚙️

This extension is enabled globally.

DETAILS FEATURES DEPENDENCIES

Categories

Debuggers

Resources

Marketplace Issues Repository License Microsoft

Python Debugger extension for Visual Studio Code

A Visual Studio Code extension that supports Python debugging with debugpy. Python Debugger provides a seamless debugging experience by allowing you to set breakpoints, step through code, inspect variables, and perform other essential debugging tasks. The debugpy extension offers debugging support for various types of Python applications including scripts, web applications, remote processes, and multi-threaded processes.

Note:

Published 2023-06-14, 19:20:56

0 1 4 0 Live Share

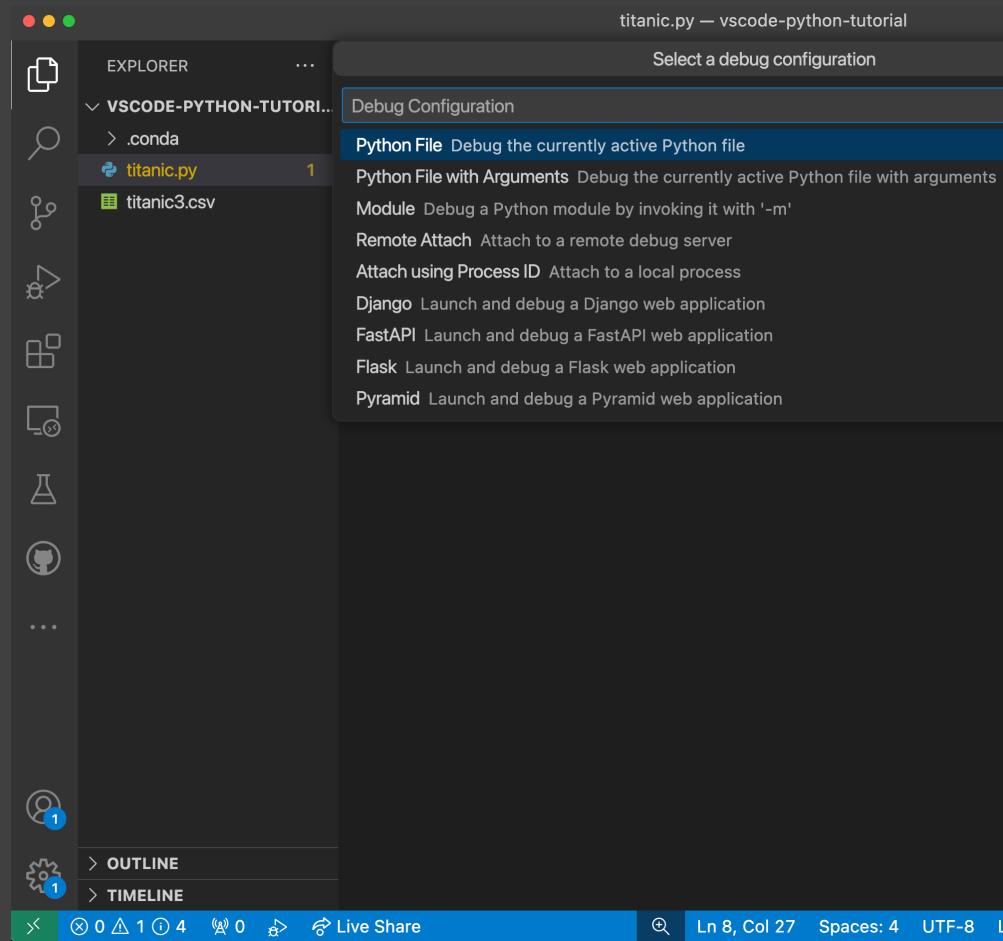
# Adding a Breakpoint

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows files in the current workspace, including `titanic.py` (1), `.conda`, and `titanic3.csv`.
- Code Editor:** The file `titanic.py` is open, displaying Python code for data visualization. A red dot icon with a tooltip "Click to add a breakpoint" is placed over the line `figure = plt.subplots(ncols=2, figsize=(30, 5))`.
- Status Bar:** Shows the current line (Ln 11, Col 11), character count (Spaces: 4), encoding (UTF-8 LF), language (Python 3.10.14), and spell check status (4 Spell).

# Initialize the Debugger

1. Press F5.
2. Select “Python Debugger”.
3. Select “Python File”.



# Using the Debugger

The screenshot shows the Visual Studio Code (VS Code) interface with the Python extension debugger active. The title bar indicates the file is "titanic.py" and the extension is "Python Debugger".

The code editor displays the following Python script:

```
titanic.py 1 Extension: Python Debugger titanic3.csv
titanic.py > [e] figure
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 'titanic3.csv'
5 file_name: str = "titanic3.csv"
6 data_frame: pd.DataFrame = pd.read_csv(file_name)
7
8 figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
9 sns.violinplot(x="survived", y="Age", hue="sex", data=data_frame, ax=axis[0])
10 sns.violinplot(x="survived", y="Fare", hue="sex", data=data_frame, ax=axis[1])
11 plt.show()
```

The line 8 is highlighted in yellow, indicating it is the current line of execution. A red dot on line 8 marks a breakpoint. The status bar at the bottom shows "Ln 8, Col 1" and "Python 3.10.14 ('.conda': conda)".

The left sidebar contains the following sections:

- VARIABLES: Shows locals and special variables.
- WATCH: Empty.
- CALL STACK: Shows the stack trace, with the top entry being "<module> titanic.py 8:1" and the status "Paused on step".
- BREAKPOINTS: Shows checkboxes for "Raised Exceptions" (unchecked), "Uncaught Exceptions" (checked), "User Uncaught Exceptions" (unchecked), and "titanic.py" (checked).

The bottom status bar includes icons for file operations, live share, and spell check.

# The Debug Console

The screenshot shows the VS Code interface with the Python Debugger extension active. The code editor displays the file `titanic.py` which contains the following code:

```
    figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
    sns.violinplot(x="survived", y="Age", hue="sex", data=data_frame, ax=axis[0])
    sns.violinplot(x="survived", y="Fare", hue="sex", data=data_frame, ax=axis[1])
    plt.show()
```

The `data_frame` variable is expanded in the Locals panel, showing its columns and the `age` column's data. The `data_frame["age"]` array is shown with several rows of data.

The `CALL STACK` panel indicates the script is paused on step 8:1. The `BREAKPOINTS` panel shows breakpoints for `titanic.py`, with `Uncaught Exceptions` and `titanic.py` checked.

The `DEBUG CONSOLE` panel displays the following traceback:

```
Traceback (most recent call last):
  File "/Users/cgavida-calderon/vscode-python-tutorial/.conda/lib/python3.10/site-packages/pandas/core/indexes/base.py", line 3805, in get_loc
    return self._engine.get_loc(casted_key)
  File "index.pyx", line 167, in pandas._libs.index.IndexEngine.get_loc
  File "index.hpx", line 196, in pandas._libs.index.IndexEngine.get_loc
```

---

# Using Git

# Initialize Repository

1. Click on "Source Control".
2. Click on "Initialize Repository"

The screenshot shows the VS Code interface with the following panes:

- Source Control** pane (left): Shows a commit message input field ("Message (⌘Enter to commi...)" with a placeholder), a "Commit" button, and a "Changes" section listing files: ".DS\_Store" (U), "titanic.py" (1, U), and "titanic3.csv" (U). It also includes sections for "COMMITS", "BRANCHES", "REMOTES", "STASHES", "TAGS", "WORKTREES", and "CONTRIBUTORS".
- Problems** pane (bottom): Displays 5 errors. One error is highlighted with a red dot and points to the line: "Index(['pclass', 'survived', 'name', 'fare', 'cabin', 'embarked', 'dtype='object']". Another error highlights the "age" column in the "data\_frame" variable.
- Code Editor** pane (right): Shows the Python script "titanic.py" with code to load the "titanic3.csv" dataset, create subplots, and plot violin plots for survival status based on age.

# Committing Files

1. Stage the files you want to commit.
2. Add a commit message.
3. Click the “Commit” button.

The screenshot shows the VS Code interface with the following panes:

- Source Control** pane: Shows a staged commit for "titanic.py" with a message "Plotting Age and Fare". The commit button is highlighted in blue.
- Changes** pane: Lists staged changes: "titanic.py" (1, A), ".DS\_Store" (2, U), and "titanic3.csv" (U).
- Debug Console** pane: Displays Python code and its output. The code reads a CSV file and prints the first few rows of the "age" column.
- Explorer** pane: Shows icons for File, Find, Git, Search, and others, along with links to Commits, Branches, Remotes, Stashes, Tags, Worktrees, and Contributors.

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_name: str = "titanic3.csv"
data_frame: pd.DataFrame = pd.read_csv(file_name)

figure, axis = plt.subplots(ncols=2)
sns.violinplot(x="survived", y="Age", data=data_frame)
sns.violinplot(x="survived", y="Fare", data=data_frame)
plt.show()

```

	0	1	2	3	4
data_frame["age"]	29.00	0.92	2.00	30.00	25.00
...					
1304	14.50				
1305	NaN				

# Python: Refactoring Plotting Logic

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def main() -> None:
    file_name: str = "~/data/titanic3.csv"
    data_frame: pd.DataFrame = pd.read_csv(file_name)

    figure, axis = plt.subplots(ncols=2, figure=(30, 5))
    sns.violinplot(
        x="survived", y="age", hue="sex", data=data_frame, ax=axis[0]
    )
    sns.violinplot(
        x="survived", y="fare", hue="sex", data=data_frame, ax=axis[1]
    )

    plt.show()

if __name__ == "__main__":
    main()
```

# Command: Git: Create Branch

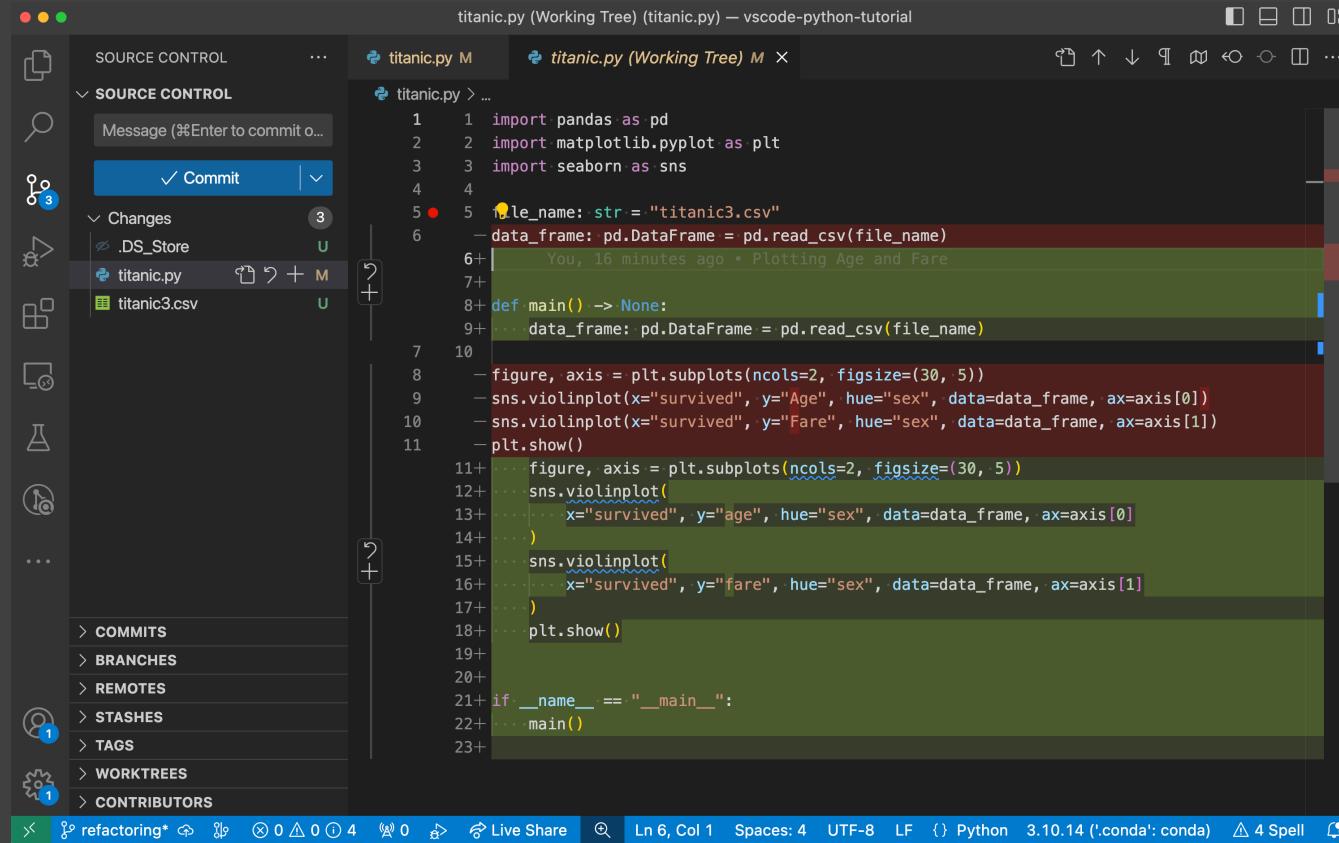
The screenshot shows the Visual Studio Code interface with the following details:

- Source Control Sidebar:** On the left, under the "Changes" section, there are three uncommitted changes: ".DS\_Store" (U), "titanic.py" (M), and "titanic3.csv" (U). A red dot is visible next to the changes.
- Main Editor:** The file "titanic.py" is open. The code is as follows:

```
You, 24 seconds ago | 1 author (You)
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 file_name: str = "titanic3.csv"
6
7
8 def main() -> None:| You, 1 second ago • Uncommitted changes
9     data_frame: pd.DataFrame = pd.read_csv(file_name)
10
11     figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
12     sns.violinplot(
13         x="survived", y="age", hue="sex", data=data_frame, ax=axis[0]
14     )
15     sns.violinplot(
16         x="survived", y="fare", hue="sex", data=data_frame, ax=axis[1]
17     )
18     plt.show()
19
20
21 if __name__ == "__main__":
22     main()
23
```

- Terminal:** At the bottom, the terminal shows two recent sessions:
  - (.conda) (base) cgavidia-calderon@633-XD6MW2FwW4 vscode-python-tutorial % /Users/cgavidia-calderon/vscode-python-tutorial/.conda/bin/python /Users/cgavidia-calderon/vscode-python-tutorial/titanic.py
  - (.conda) (base) cgavidia-calderon@633-XD6MW2FwW4 vscode-python-tutorial % []A dropdown menu in the terminal bar lists "Python", "zsh", and "Python Debug Cons...".
- Bottom Status Bar:** Shows file paths, line numbers (Ln 8, Col 20), character count (Spaces: 4), encoding (UTF-8 LF), language (Python), version (3.10.14 ('.conda': conda)), and spell checker status (4 Spell).

# Comparing Files



The screenshot shows a comparison between two versions of the file `titanic.py`. The left pane displays the current working tree version, which contains code to plot violin plots for survived vs age and fare by sex. The right pane shows the staged version, which includes a commit message and a file named `titanic3.csv`.

```
1  1 import pandas as pd
2  2 import matplotlib.pyplot as plt
3  3 import seaborn as sns
4  4
5  5 file_name: str = "titanic3.csv"
6  6 - data_frame: pd.DataFrame = pd.read_csv(file_name)
7+ 6+ You, 16 minutes ago * Plotting Age and Fare
8+ 7+
8+ def main() -> None:
9+ 8+ data_frame: pd.DataFrame = pd.read_csv(file_name)
10 9-
11 10 - figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
12 11 - sns.violinplot(x="survived", y="Age", hue="sex", data=data_frame, ax=axis[0])
13 12 - sns.violinplot(x="survived", y="Fare", hue="sex", data=data_frame, ax=axis[1])
14 13 - plt.show()
15 14+ figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
16 15+ sns.violinplot(
17 16+ ... x="survived", y="age", hue="sex", data=data_frame, ax=axis[0]
18 17+ ...)
19 18+ sns.violinplot(
20 19+ ... x="survived", y="fare", hue="sex", data=data_frame, ax=axis[1]
21 20+ ...
22 21+ if __name__ == "__main__":
23 22+ ... main()
```

The commit message in the staged version reads: "You, 16 minutes ago \* Plotting Age and Fare".

# Merging Branches

1. From the destination branch: Branch > Merge ...
2. Select the “refactoring” branch.

The screenshot shows the VS Code interface with the Source Control extension active. The title bar says "titanic.py". The Source Control sidebar is open, showing a list of changes: ".DS\_Store" (U) and "titanic3.csv" (U). The "Changes" section has a count of 2. The "Branch" submenu is open, showing options like Pull, Push, Clone, Checkout to..., Fetch, Commit, Changes, Pull, Push, Merge..., Rebase Branch..., Create Branch..., Tags, Add Co-authors..., Generate Commit Message (Experimental)..., Show Git Output, Rename Branch..., Delete Branch..., and Publish Branch... . The "Branch" option is highlighted. The main editor area shows Python code related to the Titanic dataset.

```
17     ...)
18     ... plt.show()
19
20
21 if __name__ == "__main__":
22     main()
23
```

SOURCE CONTROL

Message (⌘Enter to commit o...)

✓ Commit

Changes

.DS\_Store (U)

titanic3.csv (U)

2

Branch

Remote

Stash

Tags

Add Co-authors...

Generate Commit Message (Experimental)...

Show Git Output

COMMITS

BRANCHES

REMOTES

STASHES

TAGS

WORKTREES

CONTRIBUTORS

Live Share

Ln 23, Col 1

# Publish a Branch

1. Select Branch > Publish Branch.
2. Authenticate to GitHub.
3. Select a public or private repository.

titanic.py — vscode-python-tutorial

SOURCE CONTROL

SOURCE CONTROL

Message (⌘Enter to commi

✓ Commit

Changes

✓ DS\_Store

✓ titanic3.csv

2 import matplotlib.pyplot as plt  
3 import seaborn as sns  
4  
5 file\_name: str = "titanic3.csv"  
6  
7  
8 def main() -> None:  
9 data\_frame: pd.DataFrame = pd.read\_csv(file\_na  
10  
11 figure, axis = plt.subplots(ncols=2, figsize=  
12 sns.violinplot(  
13 x="survived", y="age", hue="sex", data=da  
14 )  
15 sns.violinplot(  
16 x="survived", y="fare", hue="sex", data=da  
17 )  
18 plt.show()  
19  
20  
21 if \_\_name\_\_ == "\_\_main\_\_":  
22 main()  
23

COMMIT

BRANCHES

REMOTES

STASHES

TAGS

WORKTREES

CONTRIBUTORS

main\* ⌛ ⓘ ⓘ 0 △ 0 ⓘ 4 ⓘ (A) 0 ⓘ Live Share ⓘ Ln 23, Col 1 Spaces: 4 UTF-8

---

<https://github.com/alan-turing-institute/sqlsynthgen>

# Cloning a Repository

1. Enter the “Git: Clone” command.
2. Select a repository destination.
3. Open the repository.

The screenshot shows the VS Code interface with a dark theme. On the left, there's a sidebar with various icons: a clipboard (1), a magnifying glass, a circular arrow (2), a square with a dot, a document with a gear, a gear (1), and three dots. The main area is titled "titanic.py — vscode-python-tutorial". A tooltip for "titanic.p" shows the URL "https://github.com/alan-turing-institute/sqlsynthgen.git". Below it, a tooltip for "titanic" shows "Clone from URL https://github.com/alan-turing-institute/sqlsynth...". The code editor contains the following Python script:

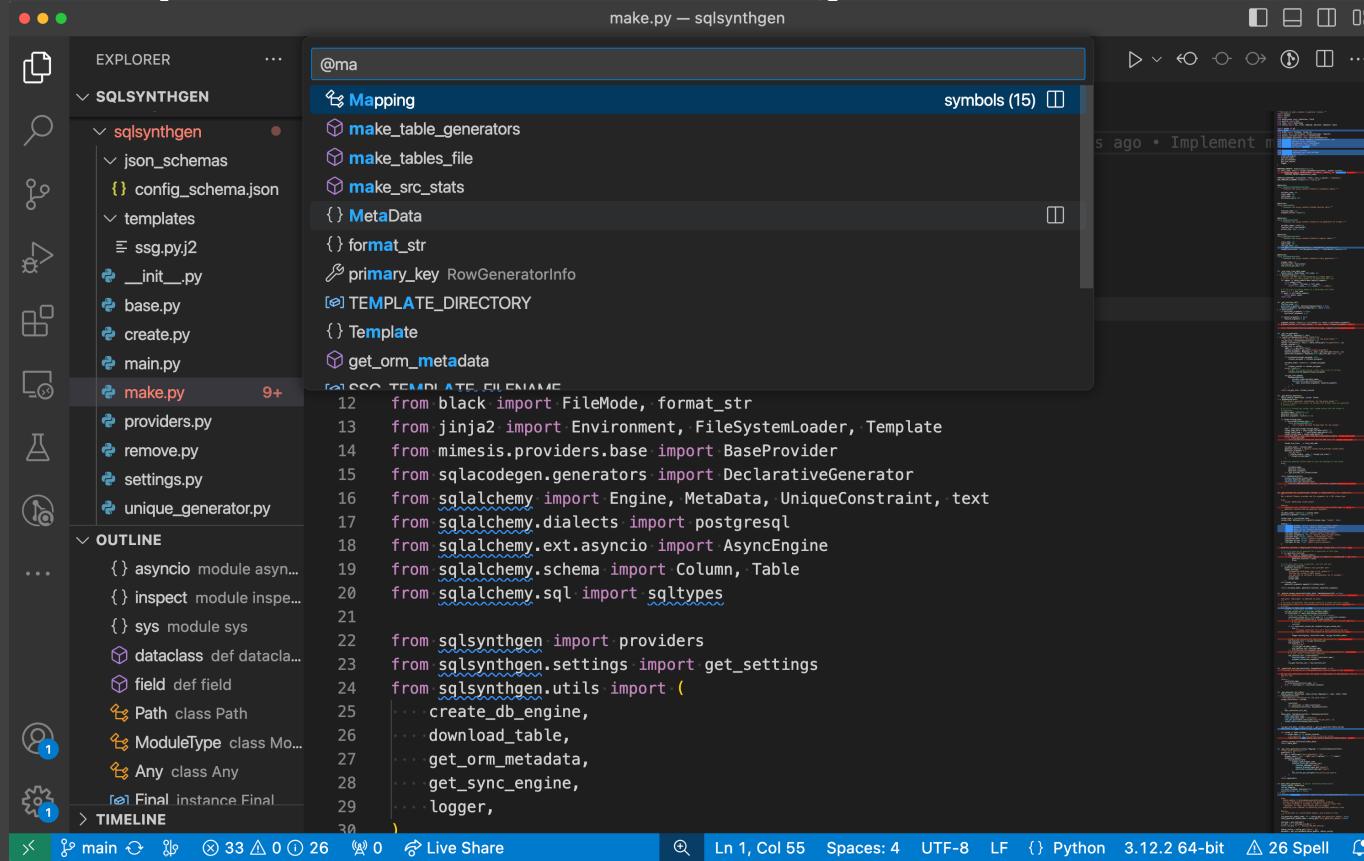
```
titanic.py
titanic.p https://github.com/alan-turing-institute/sqlsynthgen.git
titanic Clone from URL https://github.com/alan-turing-institute/sqlsynth...
Y Clone from GitHub
1 i
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 file_name: str = "titanic3.csv"
6
7
8 def main() -> None:
9     data_frame: pd.DataFrame = pd.read_csv(file_name)
10
11     figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
12     sns.violinplot(
13         x="survived", y="age", hue="sex", data=data_frame, ax=axis[0]
14     )
15     sns.violinplot(
16         x="survived", y="fare", hue="sex", data=data_frame, ax=axis[1]
17     )
18     plt.show()
19
20
21 if __name__ == "__main__":
22     main()
```

At the bottom, there are status icons for file changes, diff, and other workspace metrics, along with "Live Share" and "4 Spell" buttons.

---

# Code Navigation

# Quick Open and Go to Symbol



# Peek and Go to Definitions

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure for `SQLSYNTHGEN`. The `make.py` file is currently selected.
- EDITOR** view: Displays the `make.py` file content. A tooltip for `TableGeneratorInfo` is shown at the bottom of the editor area.
- DEFINITIONS** view: A floating panel titled "Definitions (1)" showing the definition of the `TableGeneratorInfo` class. It includes the class definition and its docstring.
- STATUS BAR**: Shows the file path `make.py — sqlysynthgen`, line number `Ln 397, Col 23`, and other status information.

```
make.py 9+ ×
sqlsynthgen > make.py > make_table_generators
368 def make_table_generators( # pylint: disable=too-many-locals
369     settings = get_settings()
370     src_dsn: str = settings.src_dsn or ""
371     assert src_dsn != "", "Missing SRC_DSN setting."
372
373     tables_config = config.get("tables", {})
374     metadata = get_orm_metadata(tables_module, tables_config)
375     engine = get_sync_engine(create_db_engine(src_dsn, schema_name=settings.src_schema))
376
377     tables: list[TableGeneratorInfo] = []
378
379     for table in metadata.sorted_tables:
380         if table.name == "vocabulary_tables": continue
381         generator_info = TableGeneratorInfo(
382             table_name=table.name,
383             primary_key=table.primary_key,
384             variable_names=[name for name in table.columns.keys() if name != table.primary_key],
385             function_call=FunctionCall(
386                 name="TableGeneratorInfo",
387                 arguments=[Argument(name=name, type="str") for name in table.columns.keys() if name != table.primary_key]
388             )
389         )
390         generator_info.vocabulary_tables = vocabulary_tables
391         generator_info.metadata = metadata
392         generator_info.engine = engine
393         generator_info.settings = settings
394         generator_info.table = table
395         generator_info.table_name = table.name
396         generator_info.table_type = table.type
397         generator_info.table_primary_key = table.primary_key
398         generator_info.table_variable_names = [name for name in table.columns.keys() if name != table.primary_key]
399         generator_info.table_function_call = FunctionCall(
380     )
```

```
class TableGeneratorInfo:
    """Contains the ssg.py content related to row generators"""
    variable_names: list[str]
    function_call: FunctionCall
    primary_key: bool = False

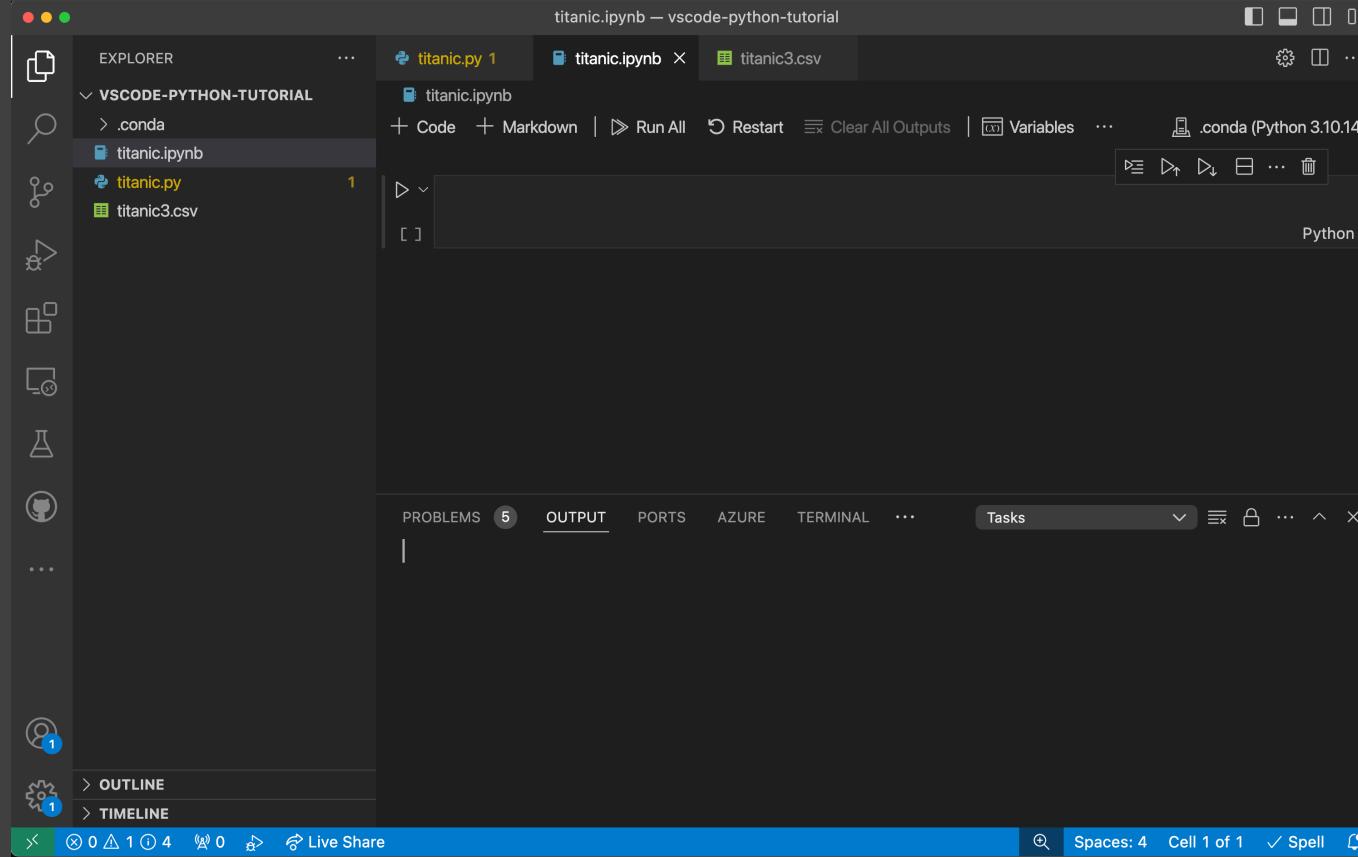
    @dataclass
    class TableGeneratorInfo:
        """Contains the ssg.py content related to regular tables."""
        class_name: str
        table_name: str

        vocabulary_tables: list[VocabularyTableGeneratorInfo] = []
        for table in metadata.sorted_tables:
```

---

# Using Jupyter Notebooks

# Create: New Jupyter Notebook



# Python: Loading a DataFrame

```
import pandas as pd  
  
file_name: str = "~/data/titanic3.csv"  
data_frame: pd.DataFrame = pd.read_csv(file_name)
```

# Running a Cell

The screenshot shows a VS Code interface with the following components:

- EXPLORER** sidebar: Shows files in the workspace, including `titanic.ipynb`, `titanic.py`, and `titanic3.csv`.
- JUPYTER** view: A code editor showing a Python cell with the following code:

```
import pandas as pd

file_name: str = "titanic3.csv"
data_frame: pd.DataFrame = pd.read_csv(file_name)
```

The cell has a green checkmark icon and a duration of `0.0s`.
- Data Viewer**: A table showing the contents of the `titanic3.csv` file. The columns are `index`, `pclass`, `survived`, `name`, `sex`, and `age`. The data includes rows for passengers like Allen, Miss., Allison, Mr., and Astor, Col.
- VARIABLES** view: Shows the variables defined in the current scope:

Name	Type	Size	Value
<code>data_frame</code>	<code>DataFrame</code>	(1309, 14)	<code>pclass</code> <code>survived</code>
<code>file_name</code>	<code>str</code>	12	'titanic3.csv'
- Bottom status bar**: Includes icons for search, spell check, and live share.

# Plotting Data

titanic.ipynb — vscode-python-tutorial

EXPLORER    ...    titanic.py    titanic.ipynb U X

.conda    + Code    + Markdown    Run All    Restart    Clear All Outputs    Variables    Outline    ...    .conda (Python 3.10.14)

VSCODE-PYTHON-TUTORIAL... titanic.ipynb titanic.py titanic3.csv

3

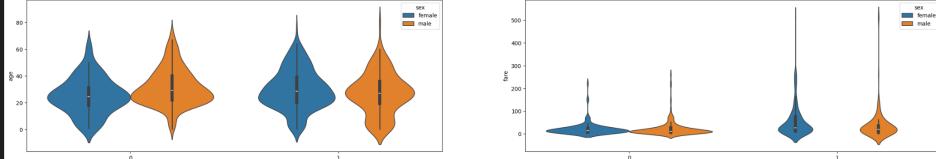
```
import matplotlib.pyplot as plt
import seaborn as sns

figure, axis = plt.subplots(ncols=2, figsize=(30, 5))
sns.violinplot(x="survived", y="age", hue="sex", data=data_frame, ax=axis[0])
sns.violinplot(x="survived", y="fare", hue="sex", data=data_frame, ax=axis[1])
```

✓ 2.0s    Python

<Axes: xlabel='survived', ylabel='fare'>

...



...

Spaces: 4 LF Cell 2 of 2 △ 4 Spell 🔍 ()

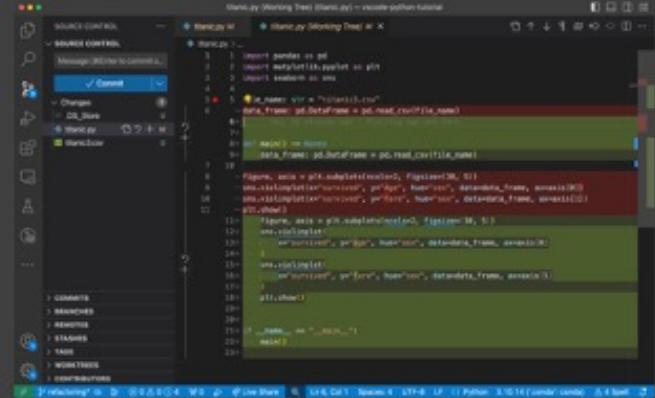
## Visual Studio Code for Python Development

Carlos Gavidia-Calderon



Image: [https://commons.wikimedia.org/wiki/File:The\\_Young\\_Snake\\_Charmer.jpg](https://commons.wikimedia.org/wiki/File:The_Young_Snake_Charmer.jpg)

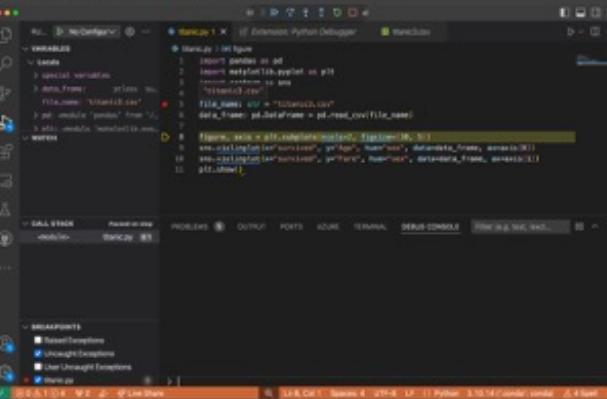
### Comparing Files



A screenshot of the Visual Studio Code interface showing a comparison between two versions of a Python file named 'manicure.py'. The left pane shows the original code, and the right pane shows the modified code. The modifications are highlighted in red and green, indicating changes made to the code.

```
diff --git a/manicure.py b/manicure.py
--- a/manicure.py
+++ b/manicure.py
@@ -1,10 +1,10 @@
 1 import pandas as pd
 2 import matplotlib.pyplot as plt
 3
 4
 5 # df_dance = pd.read_csv('tutu.csv')
 6 data_frame = pd.read_csv('tutu.csv')
 7
 8
 9 max_val = max(data_frame['survived'])
10 min_val = min(data_frame['survived'])
11
12 # df_dance['survived'].value_counts().plot(kind='bar', title='Survived vs. Sex', alpha=0.5)
13 # df_dance['survived'].value_counts().plot(kind='bar', title='Survived vs. Sex', alpha=0.5)
14
15 # df_dance['survived'].value_counts().plot(kind='bar', title='Survived vs. Sex', alpha=0.5)
16
17 # df_dance['survived'].value_counts().plot(kind='bar', title='Survived vs. Sex', alpha=0.5)
```

### Using the Debugger



A screenshot of the Visual Studio Code interface showing the Python debugger. The 'Call Stack' panel displays a stack trace for a function named 'manicure'. The 'Variables' panel shows various variables and their values, including 'df\_dance', 'max\_val', 'min\_val', 'survived', 'sex', 'age', 'fare', 'pclass', 'embarked', and 'titles'. The 'Breakpoints' panel shows that breakpoints have been set on specific lines of code.

### Plotting Data

