

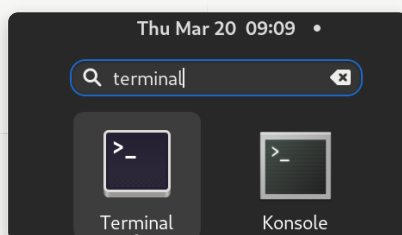
# Linux

This quickstart will help you set up your Linux machine to explore the sourcecode of Bonsai or develop and debug your blender scripts in VSCode. This has the benefit of having a complete development environment where you can explore the code, make changes, debug (break-points, watch variable and stack contents, etc. ) and see the results in blender

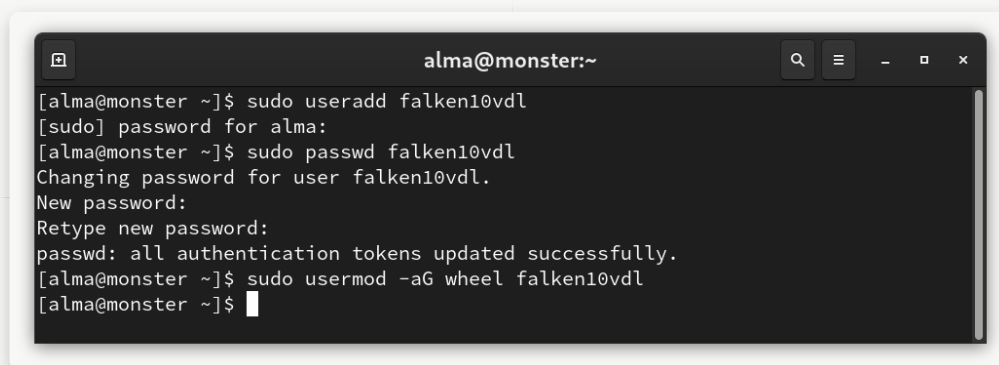
- Steps 1-6 will get you started with VSCode to develop and debug python scripts in Blender.
- Steps 7-14 will allow you to interact with GitHub to make changes to the Bonsai project.

We will be using AlmaLinux 9 as our operating system and Visual Studio Code as our Integrated Development Environment (IDE) and we will create a dedicated user for Development.

1. **Create Development User:** Open up a terminal (typically hitting “Windows” key and writting “terminal” in the search field)



```
sudo useradd falken10vdl
sudo passwd falken10vdl
sudo usermod -aG wheel falken10vdl
```

A screenshot of a terminal window titled 'alma@monster:~'. The terminal shows the following commands and output:

```
[alma@monster ~]$ sudo useradd falken10vdl
[sudo] password for alma:
[alma@monster ~]$ sudo passwd falken10vdl
Changing password for user falken10vdl.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[alma@monster ~]$ sudo usermod -aG wheel falken10vdl
[alma@monster ~]$
```

## Tip

If for some reason you need to delete the user, you can use the following command:

```
sudo userdel -r falken10vdl
```

2. **Install Blender for the created user:** We will install blender locally in the users home

directory. We must check that we are following the [Systems requirements](#).

We will download Blender 4.2 from the [Blender download page](#). In particular, we take the [4.2 LTS](#) for Linux.

We will download the Linux 64 bit version:

<https://www.blender.org/download/release/Blender4.2/blender-4.2.8-linux-x64.tar.xz>

```
wget https://download.blender.org/release/Blender4.2/blender-4.2.8-linux-x64.tar.xz
tar -xvf blender-4.2.8-linux-x64.tar.xz
mv blender-4.2.8-linux-x64 /home/falken10vdl/.local/share/applications/blender-4.
```

#### Warning

If the directory `/home/falken10vdl/.local/bin/` does not exist, we will create it.

```
mkdir -p /home/falken10vdl/.local/bin/
```

We will create a symbolic link to the blender executable in the bin directory and we will also modify the `blender.desktop` file to open in a terminal and to have a custom icon.

```
ln -s /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender
sed -i 's/^Terminal=.*Terminal=true/' /home/falken10vdl/.local/share/applications/blender.desktop
sed -i 's|^Icon=.*|Icon=/home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender.svg|' /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender.desktop
```

```
falken10vdl@monster:~$ wget https://download.blender.org/release/Blender4.2/blender-4.2.8-linux-x64.tar.xz
--2025-03-21 00:01:06-- https://download.blender.org/release/Blender4.2/blender-4.2.8-linux-x64.tar.xz
Resolving download.blender.org (download.blender.org)... 172.67.14.163, 104.22.64.163, 104.22.65.163, ...
Connecting to download.blender.org (download.blender.org)[172.67.14.163]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 354817952 (338M) [application/octet-stream]
Saving to: 'blender-4.2.8-linux-x64.tar.xz'

blender-4.2.8-linux-x64.tar.xz 100%[=====>] 338.38M 104MB/s in 3.3s

2025-03-21 00:01:10 (103 MB/s) - 'blender-4.2.8-linux-x64.tar.xz' saved [354817952/354817952]

[falken10vdl@monster ~]$ tar -xvf blender-4.2.8-linux-x64.tar.xz
blender-4.2.8-linux-x64/
blender-4.2.8-linux-x64/readme.html
blender-4.2.8-linux-x64/lib/
blender-4.2.8-linux-x64/lib/libsolexec.so.1.13.7
blender-4.2.8-linux-x64/lib/libOpenEXRCore.so.31.3.2.4
blender-4.2.8-linux-x64/lib/libboost_locale.so.1.82.0
blender-4.2.8-linux-x64/lib/libMaterialXGenMdl.so
blender-4.2.8-linux-x64/lib/libsolexcomp.so.1.13
blender-4.2.8-linux-x64/lib/libboost_regex.so
blender-4.2.8-linux-x64/lib/libOpenImageIO.so
blender-4.2.8-linux-x64/lib/libboost_atomic.so

blender-4.2.8-linux-x64/license/spdx/FTL.txt
blender-4.2.8-linux-x64/license/spdx/Apache-2.0.txt
blender-4.2.8-linux-x64/license/spdx/SGI-B-2.0.txt
blender-4.2.8-linux-x64/license/spdx/MIT.txt
blender-4.2.8-linux-x64/license/spdx/NCSA.txt
blender-4.2.8-linux-x64/license/spdx/Python-2.0.txt
blender-4.2.8-linux-x64/license/spdx/MPL-2.0.txt
blender-4.2.8-linux-x64/license/spdx/OFL-1.1.txt
blender-4.2.8-linux-x64/license/spdx/bzip2-1.0.6.txt
blender-4.2.8-linux-x64/license/spdx/BSD-2-Clause.txt
blender-4.2.8-linux-x64/license/spdx/GPL-2.0-or-later.txt
blender-4.2.8-linux-x64/license/license.md
blender-4.2.8-linux-x64/license/others/
blender-4.2.8-linux-x64/license/others/Arwe-Fonts.txt
blender-4.2.8-linux-x64/license/others/LLVM-exception.txt
blender-4.2.8-linux-x64/license/others/TOST-1.0.txt
blender-4.2.8-linux-x64/license/others/ICSL.txt
[falken10vdl@monster ~]$ mv blender-4.2.8-linux-x64 /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64
[falken10vdl@monster ~]$ mkdir -p /home/falken10vdl/.local/bin/
[falken10vdl@monster ~]$ ln -s /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender /home/falken10vdl/.local/bin/blender
[falken10vdl@monster ~]$ sed -i 's/^Terminal=.*Terminal=true/' /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender.desktop
[falken10vdl@monster ~]$ sed -i 's|^Icon=.*|Icon=/home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender.svg|' /home/falken10vdl/.local/share/applications/blender-4.2.8-linux-x64/blender.desktop
[falken10vdl@monster ~]$
```

Congratulations! You have now Blender installed in your machine. You can launch it by typing *blender* in the terminal.

Now install the Bonsai Blender extension. Follow the [Unstable installation](#).

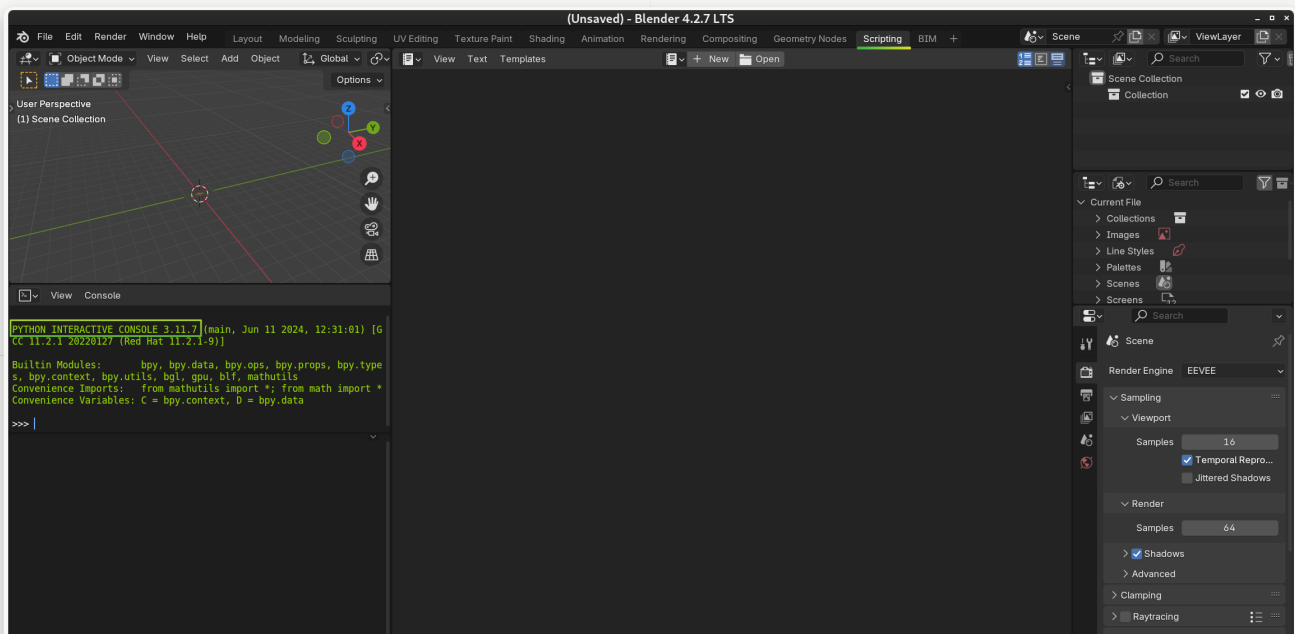
3. **Install VSCode:** Log in as the new created user (*falken10vdl* in this example) and install [Visual Studio Code](#).

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/
dnf check-update
sudo dnf install code # or code-insiders
```

```
falken10vdl@monster:~$ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
[falken10vdl@monster ~]$ echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\nautorefresh=1\ntype=rpm-md\nngpgcheck=1\nngpgkey=https://packages.microsoft.com/keys/microsoft.asc" | sudo tee /etc/yum.repos.d/vscode.repo > /dev/null
[falken10vdl@monster ~]$ dnf check-update
Visual Studio Code                               12 kB/s | 1.5 kB   00:00
[falken10vdl@monster ~]$ sudo dnf install code
Visual Studio Code                               50 kB/s | 1.5 kB   00:00
Dependencies resolved.
=====
Package                Architecture      Version                               Repository      Size
-----
Installing:
code                   x86_64            1.98.2-1741788968.el8               code            142 M
=====
Transaction Summary
=====
Install 1 Package
=====
Total download size: 142 M
Installed size: 400 M
Is this ok [y/N]: y
Downloading Packages:
code-1.98.2-1741788968.el8.x86_64.rpm            65 MB/s | 142 MB   00:02
-----
Total                                           65 MB/s | 142 MB   00:02
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                : code-1.98.2-1741788968.el8.x86_64                1/1
  Installing               : code-1.98.2-1741788968.el8.x86_64                1/1
  Running scriptlet        : code-1.98.2-1741788968.el8.x86_64                1/1
  Verifying                : code-1.98.2-1741788968.el8.x86_64                1/1
Installed:
code-1.98.2-1741788968.el8.x86_64
Complete!
[falken10vdl@monster ~]$
```

4. **Adjust Python version in VSCode as in Blender:** Although not extrictly mandatory, this is a good practice step to ensure that the Python version in VSCode matches the one in Blender.

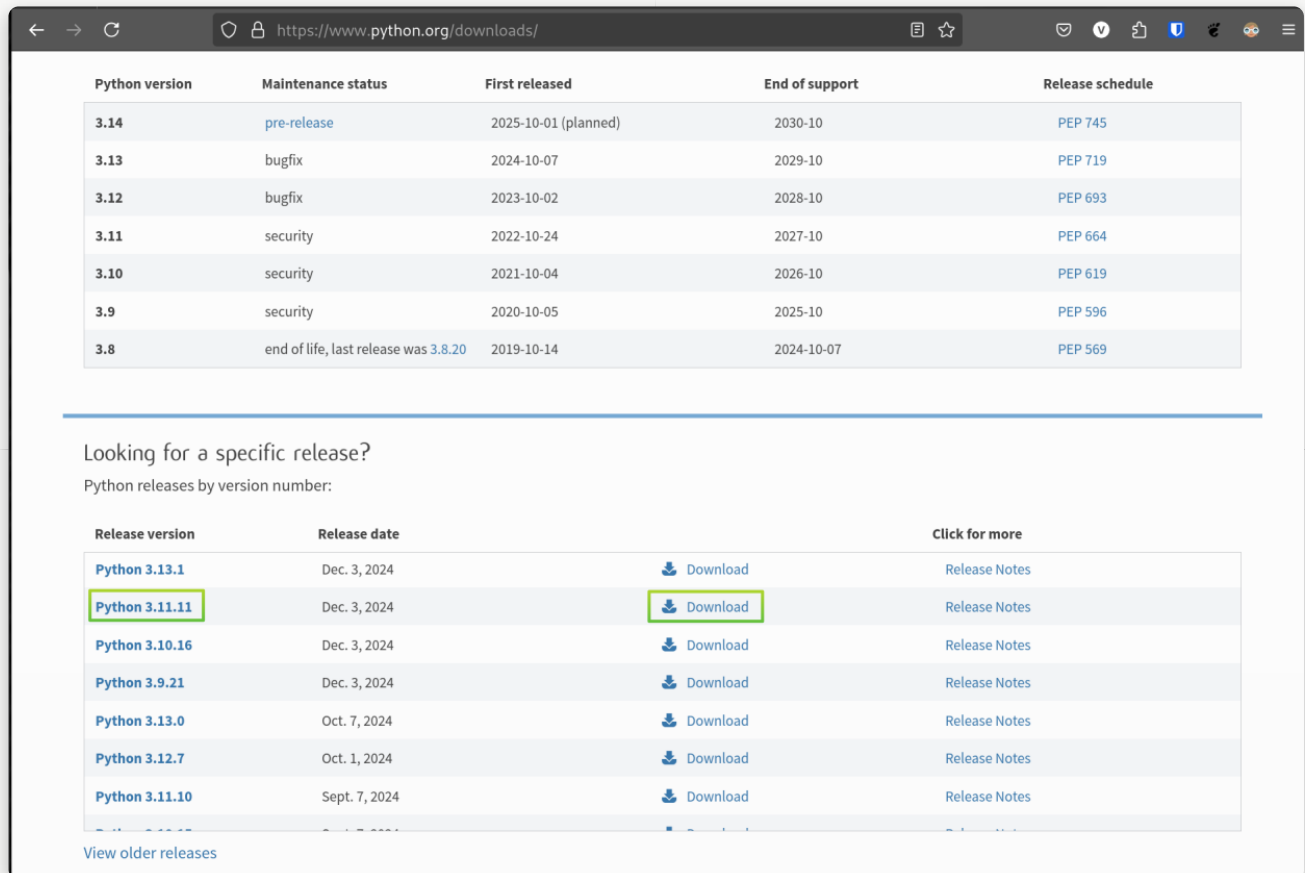
Check the Python version in Blender by going to [Scripting](#). In the Python Console you can see the version number of the Python interpreter



In our case it is version 3.11.7

We will need to install the closest version in our Linux machine.

We check in [Python Downloads](#).



The screenshot shows the Python Downloads page with a table of Python versions and their maintenance status. The table has five columns: Python version, Maintenance status, First released, End of support, and Release schedule. The versions listed are 3.14, 3.13, 3.12, 3.11, 3.10, 3.9, and 3.8. The maintenance status for 3.14 is 'pre-release', while for 3.13 through 3.8, it is 'bugfix' or 'security'. The first released date, end of support date, and release schedule (PEP number) are also provided for each version.

| Python version | Maintenance status                   | First released       | End of support | Release schedule |
|----------------|--------------------------------------|----------------------|----------------|------------------|
| 3.14           | pre-release                          | 2025-10-01 (planned) | 2030-10        | PEP 745          |
| 3.13           | bugfix                               | 2024-10-07           | 2029-10        | PEP 719          |
| 3.12           | bugfix                               | 2023-10-02           | 2028-10        | PEP 693          |
| 3.11           | security                             | 2022-10-24           | 2027-10        | PEP 664          |
| 3.10           | security                             | 2021-10-04           | 2026-10        | PEP 619          |
| 3.9            | security                             | 2020-10-05           | 2025-10        | PEP 596          |
| 3.8            | end of life, last release was 3.8.20 | 2019-10-14           | 2024-10-07     | PEP 569          |

Looking for a specific release?  
Python releases by version number:

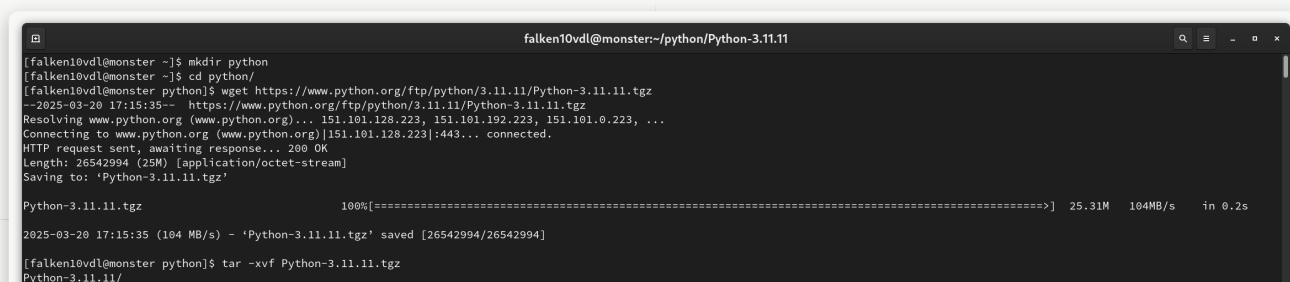
| Release version | Release date  | Click for more                                         |
|-----------------|---------------|--------------------------------------------------------|
| Python 3.13.1   | Dec. 3, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.11.11  | Dec. 3, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.10.16  | Dec. 3, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.9.21   | Dec. 3, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.13.0   | Oct. 7, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.12.7   | Oct. 1, 2024  | <a href="#">Download</a> <a href="#">Release Notes</a> |
| Python 3.11.10  | Sept. 7, 2024 | <a href="#">Download</a> <a href="#">Release Notes</a> |

[View older releases](#)

The closest version is 3.11.11. So we download the Gzipped source tarball and install it.

We use the “altinstall” option to avoid overwriting the default Python version which could cause conflicts with the default installed version of the linux operating system.

```
wget https://www.python.org/ftp/python/3.11.11/Python-3.11.11.tgz
tar -xvf Python-3.11.11.tgz
cd Python-3.11.11
sudo dnf install gcc openssl-devel bzip2-devel libffi-devel
./configure --enable-optimizations
nproc
make -j 4 #adjust the value to the one provided by nproc
sudo make altinstall
```



The screenshot shows a terminal window with the following commands and output:

```
falken10vdl@monster:~/python/Python-3.11.11
[falken10vdl@monster ~]$ mkdir python
[falken10vdl@monster ~]$ cd python/
[falken10vdl@monster python]$ wget https://www.python.org/ftp/python/3.11.11/Python-3.11.11.tgz
--2025-03-20 17:15:35-- https://www.python.org/ftp/python/3.11.11/Python-3.11.11.tgz
Resolving www.python.org (www.python.org)... 151.101.128.223, 151.101.192.223, 151.101.0.223, ...
Connecting to www.python.org (www.python.org)[151.101.128.223]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26542994 (25M) [application/octet-stream]
Saving to: 'Python-3.11.11.tgz'

Python-3.11.11.tgz 100%[=====] 25.31M 104MB/s in 0.2s

2025-03-20 17:15:35 (104 MB/s) - 'Python-3.11.11.tgz' saved [26542994/26542994]

[falken10vdl@monster python]$ tar -xvf Python-3.11.11.tgz
Python-3.11.11/
```

```
Python-3.11.11/.editorconfig
Python-3.11.11/.mailmap
Python-3.11.11/.pre-commit-config.yaml
Python-3.11.11/.readthedocs.yml
Python-3.11.11/Doc/
Python-3.11.11/Doc/Makefile
Python-3.11.11/Doc/README.rst
Python-3.11.11/Doc/_static/
```

```
falken10vdl@monster:~/python/Python-3.11.11
Python-3.11.11/configure.ac
Python-3.11.11/install-sh
Python-3.11.11/pyconfig.h.in
Python-3.11.11/setup.py
[falken10vdl@monster python]$ cd Python-3.11.11/
[falken10vdl@monster Python-3.11.11]$ sudo dnf install gcc openssl-devel bzip2-devel libffi-devel
[sudo] password for falken10vdl:
github.git-lfs
Package gcc-11.5.0-5.el9_5.alma1.x86_64 is already installed.
Package openssl-devel-1:3.2.2-6.el9_5.1.x86_64 is already installed.
Package bzip2-devel-1.0.8-10.el9_5.x86_64 is already installed.
Package libffi-devel-3.4.2-8.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[falken10vdl@monster Python-3.11.11]$ ./configure --enable-optimizations
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for Python interpreter freezing... ./bootstrap_python
checking for python3.11... python3.11
checking Python for regen version... Python 3.11.11
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
352 B/s | 833 B 00:02
329 B/s | 833 B 00:02
```

```
falken10vdl@monster:~/python/Python-3.11.11
config.status: creating Makefile.pre
config.status: creating Misc/python.pc
config.status: creating Misc/python-embed.pc
config.status: creating Misc/python-config.sh
config.status: creating Modules/Setup.bootstrap
config.status: creating Modules/Setup.stdlib
config.status: creating Modules/ld_so_aix
config.status: creating pyconfig.h
configure: creating Modules/Setup.local
configure: creating Makefile
[falken10vdl@monster Python-3.11.11]$ nproc
16
[falken10vdl@monster Python-3.11.11]$ make -j 16
Running code to generate profile data (this can take a while):
# First, we need to create a clean build with profile generation
# enabled.
make profile-gen-stamp
make[1]: Entering directory '/home/falken10vdl/python/Python-3.11.11'
make clean
make[2]: Entering directory '/home/falken10vdl/python/Python-3.11.11'
find . -depth -name '__pycache__' -exec rm -rf {} ';'
find . -name '*.py[co]' -exec rm -f {} ';'
find . -name '*.o[a]' -exec rm -f {} ';'
find . -name '*.so[a]' -exec rm -f {} ';'
16
```

```
falken10vdl@monster:~/python/Python-3.11.11
renaming build/scripts-3.11/pydoc3.11 to build/scripts-3.11/pydoc3.11
renaming build/scripts-3.11/idle3 to build/scripts-3.11/idle3.11
renaming build/scripts-3.11/2to3 to build/scripts-3.11/2to3-3.11
make[1]: Leaving directory '/home/falken10vdl/python/Python-3.11.11'
[falken10vdl@monster Python-3.11.11]$ sudo make altinstall
if test "no-framework" = "no-framework" ; then \
    /usr/bin/install -c python /usr/local/bin/python3.11; \
else \
    /usr/bin/install -c -s Mac/pythonw /usr/local/bin/python3.11; \
fi
if test "3.11" != "3.11" ; then \
    if test -f /usr/local/bin/python3.11 -o -h /usr/local/bin/python3.11; \
    then rm -f /usr/local/bin/python3.11; \
    fi; \
    (cd /usr/local/bin; ln python3.11 python3.11); \
fi
if test "x" != "x" ; then \
    rm -f /usr/local/bin/python3.11-32; \
    lipo \
        -output /usr/local/bin/python3.11-32 \
        /usr/local/bin/python3.11; \
fi
if test "x" != "x" ; then \
    rm -f /usr/local/bin/python3.11-intel64; \
fi
```

```
falken10vdl@monster:~/python/Python-3.11.11
copying build/scripts-3.11/pydoc3.11 -> /usr/local/bin
copying build/scripts-3.11/idle3.11 -> /usr/local/bin
copying build/scripts-3.11/2to3-3.11 -> /usr/local/bin
changing mode of /usr/local/bin/pydoc3.11 to 755
changing mode of /usr/local/bin/idle3.11 to 755
changing mode of /usr/local/bin/2to3-3.11 to 755
rm /usr/local/lib/python3.11/lib-dynload/_sysconfigdata__linux_x86_64-linux-gnu.py
rm -r /usr/local/lib/python3.11/lib-dynload/_pycache__
/usr/bin/install -c -m 644 ./Misc/python.man \
    /usr/local/share/man/man1/python3.11.1
if test "xupgrade" != "xno" ; then \
    case upgrade in \
        upgrade) ensurepip="--altinstall --upgrade" ;; \
        install|*) ensurepip="--altinstall" ;; \
        esac; \
        ./python -E -m ensurepip \
            $ensurepip --root=/ ; \
    fi
Looking in links: /tmp/tmpisfl8hch
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/site-packages (65.5.0)
Requirement already satisfied: pip in /usr/local/lib/python3.11/site-packages (24.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: ht
tps://pip.pypa.io/warnings/venv
[falken10vdl@monster Python-3.11.11]$
```

After this, we have the 3.11 python version installed in our machine. It is reachable by typing *python3.11* in the terminal.

```
python3.11 -V
```

```
falken10vdl@monster:~/python/Python-3.11.11
```

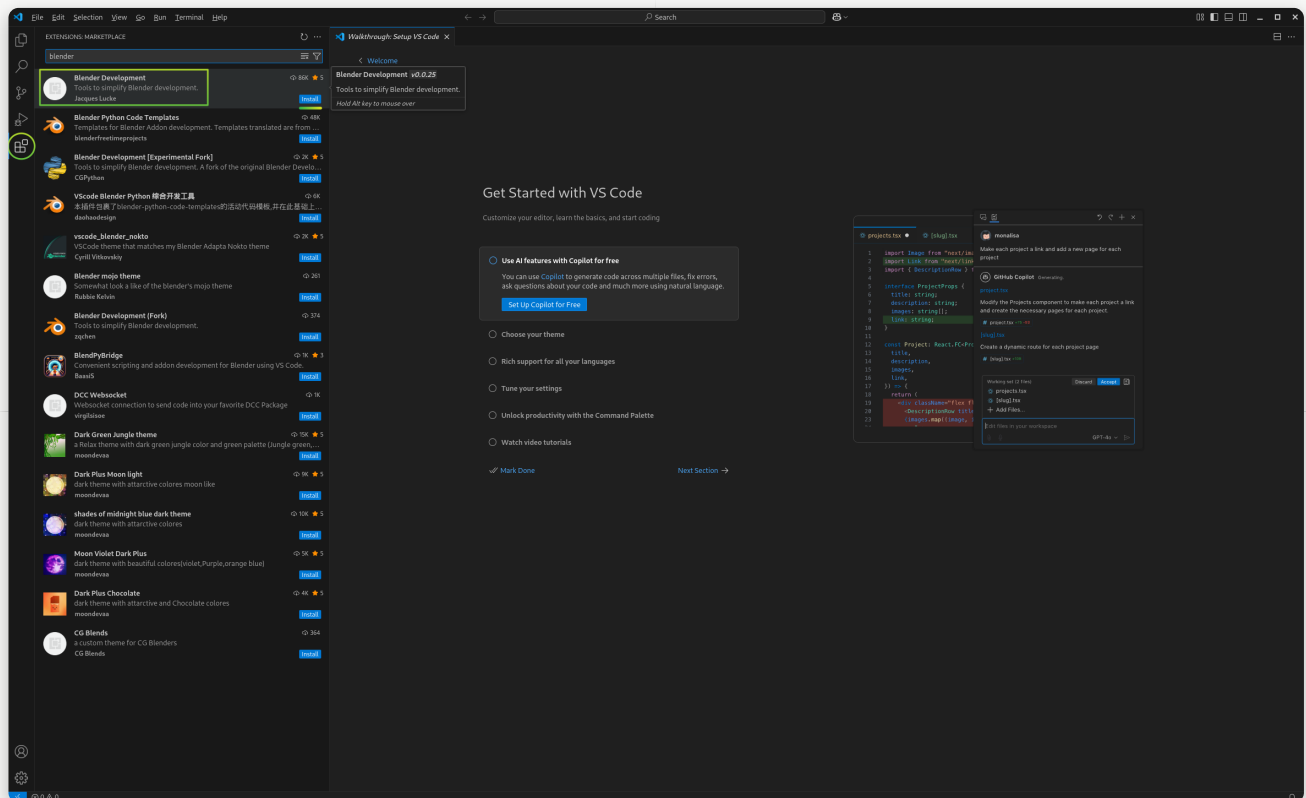
```
[falken10vdl@monster Python-3.11.11]$ python3.11 -V
Python 3.11.11
[falken10vdl@monster Python-3.11.11]$
```

Congratulations! You have now a Python version in VSCode similar to the one run by Blender.

## 5. Connect VSCode to Blender by means of VSCode's extension: "Blender Development":

This steps is crucial to be able to develop and debug scripts in VSCode and interactively see the results in Blender.

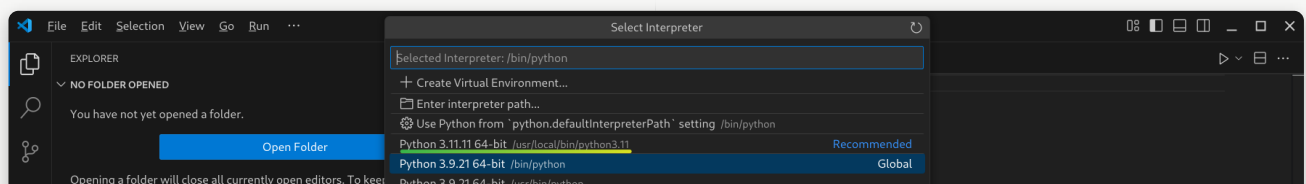
Launch VSCode and go to the Extensions tab, search for Blender Development and install it.

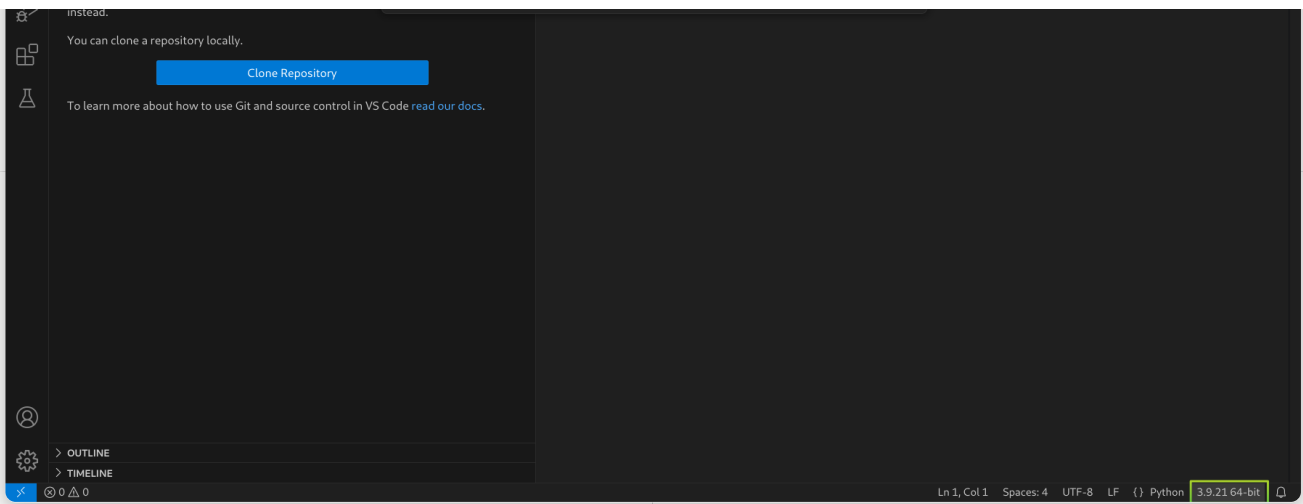


This will also install some Python related extensions.

Finally create a sample python file and check the Python interpreter version in the bottom left corner.

File ► New File... ► Python File





6. **Test that you can develop python scripts in VSCode for Blender:** Create a sample blender python file. you can use whatever blender python script you want. We will use this one from the blender documentation: [Example Panel](#)

```
import bpy

class HelloWorldPanel(bpy.types.Panel):
    """Creates a Panel in the Object properties window"""
    bl_label = "Hello World Panel"
    bl_idname = "OBJECT_PT_hello"
    bl_space_type = 'PROPERTIES'
    bl_region_type = 'WINDOW'
    bl_context = "object"

    def draw(self, context):
        layout = self.layout

        obj = context.object

        row = layout.row()
        row.label(text="Hello world!", icon='WORLD_DATA')

        row = layout.row()
        row.label(text="Active object is: " + obj.name)
        row = layout.row()
        row.prop(obj, "name")

        row = layout.row()
        row.operator("mesh.primitive_cube_add")

def register():
    bpy.utils.register_class(HelloWorldPanel)

def unregister():
    bpy.utils.unregister_class(HelloWorldPanel)
```

```

if __name__ == "__main__":
    print("Hello World: run from Blender Text Editor")
else:
    print("Hello World: run from VSCode")
    print(f"NOTE. __name__ is : {__name__}")

register()

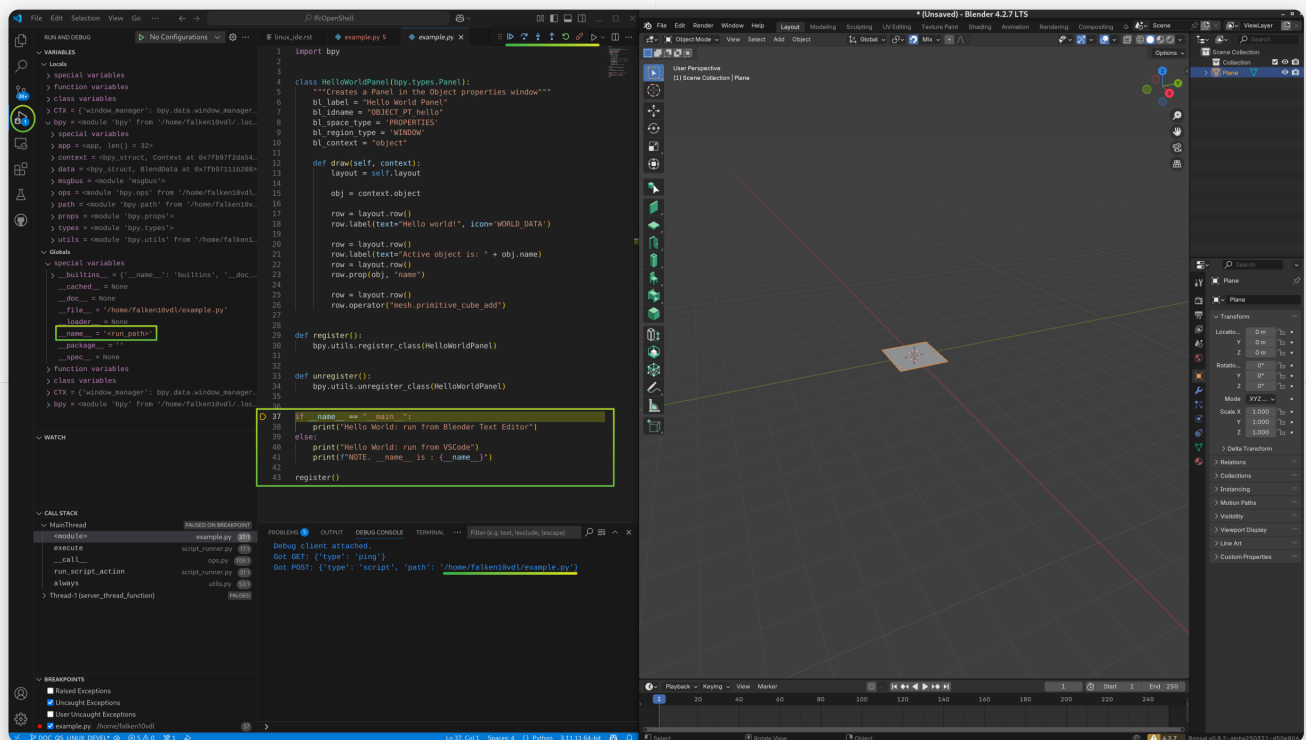
```

We have changed the last part of the script since running from VSCode has some subtle differences compared to running from the Blender Text Editor. In particular the special variable `__name__` is different.

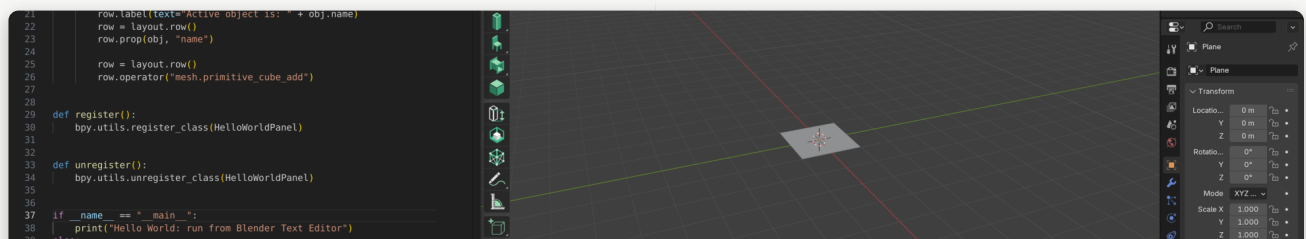
Press CTRL-SHIFT-P and type “Blender: Start”. Blender will start.

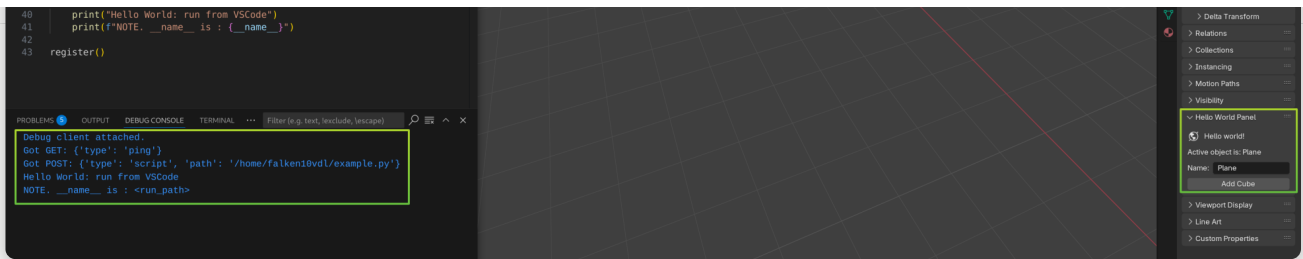
Press CTRL-SHIFT-P and type “Blender: Run Script”. The script will run and the output will be seen in Blender!

As you can see below. We have set a break-point in line 37 (see point 13 below for another example of setting a break-point). We can inspect in the left side the local variables, global variables, add watches, check the stack, etc. For example we can see that `__name__` has a value of “<run\_path>” Instead of “`__main__`”.



Once we continue execution we can check in the VSCode Terminal the output and in Blender the panel created by the script.





CONGRATULATIONS! You have now a development environment ready to speedup your python scripting in Blender.

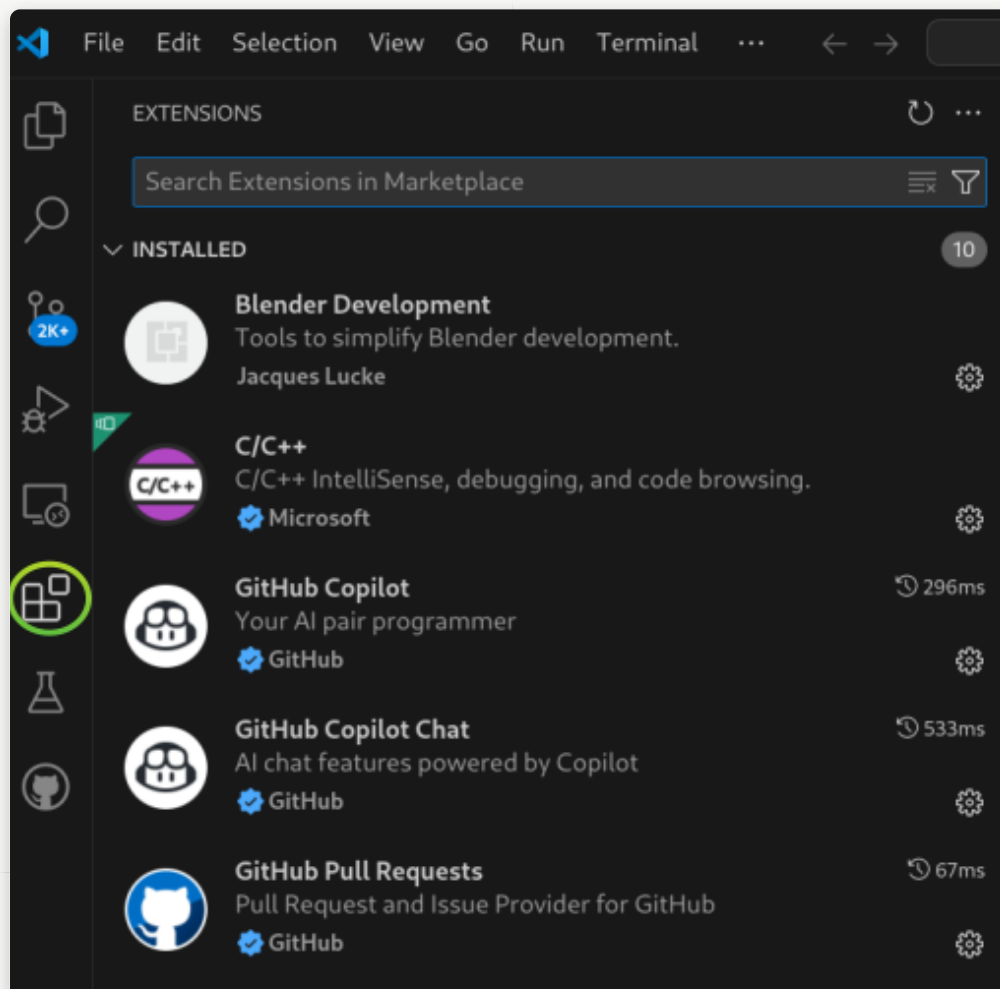
Now let's find out how to interact with GitHub in order to make changes to the Bonsai project.

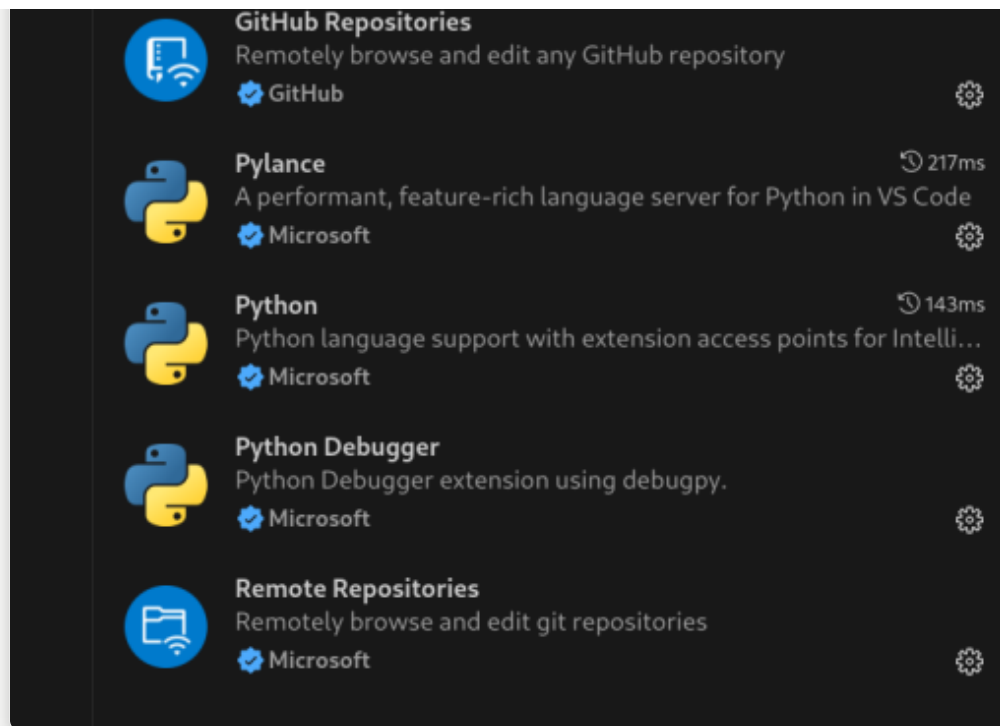
**7. Install GitHub related VSCode extensions:** To facilitate the use of git commands and pulling and pushing files from a local repository towards github, please install as well the following VSCode extensions:

- GitHub Pull Requests
- GitHub Repositories
- Remote Repositories

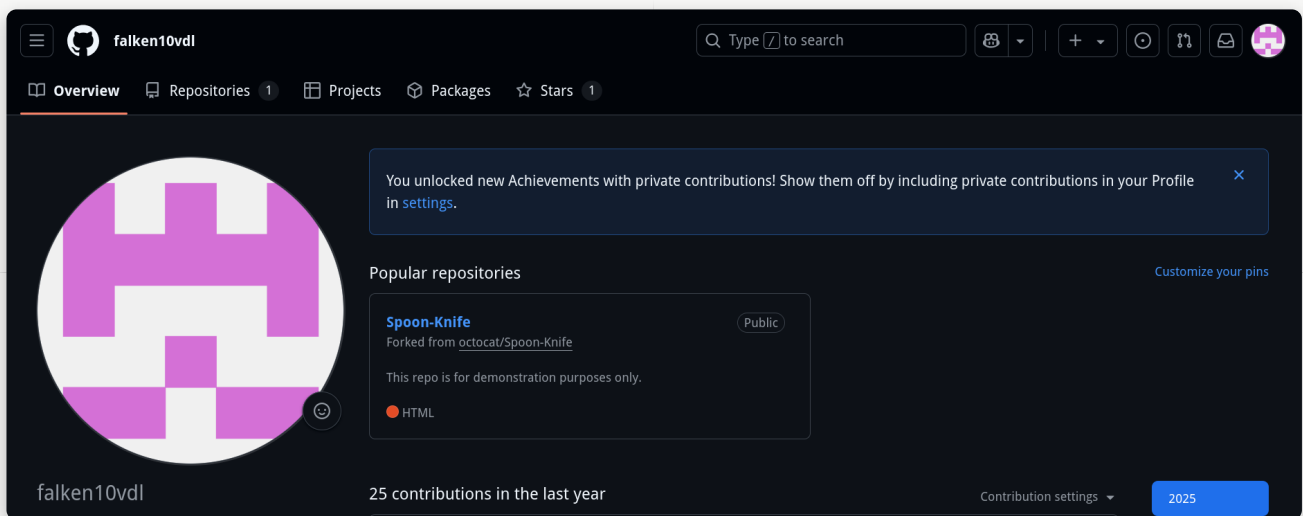
Optionaly you can also install Copilot extensions

- GitHub Copilot
- GitHub Copilot Chat

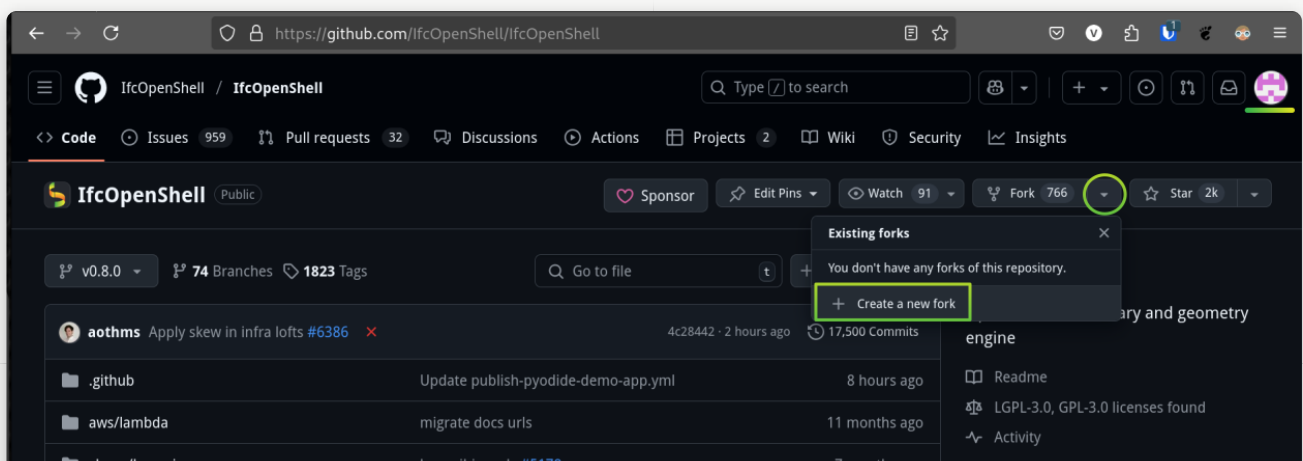


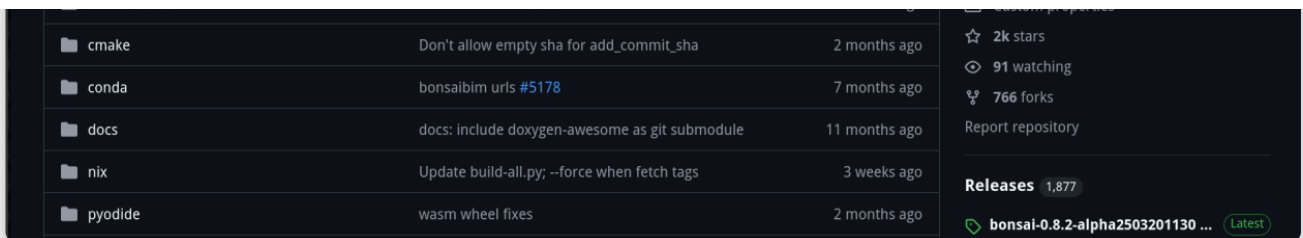


8. **Fork IfcOpenShell project from GitHub:** For this step you will need an account on GitHub. Once you have a registered account you can find it under <https://github.com/YOURGITHUBUSERID> In the example for *falken10vdl* the link is <https://github.com/falken10vdl>

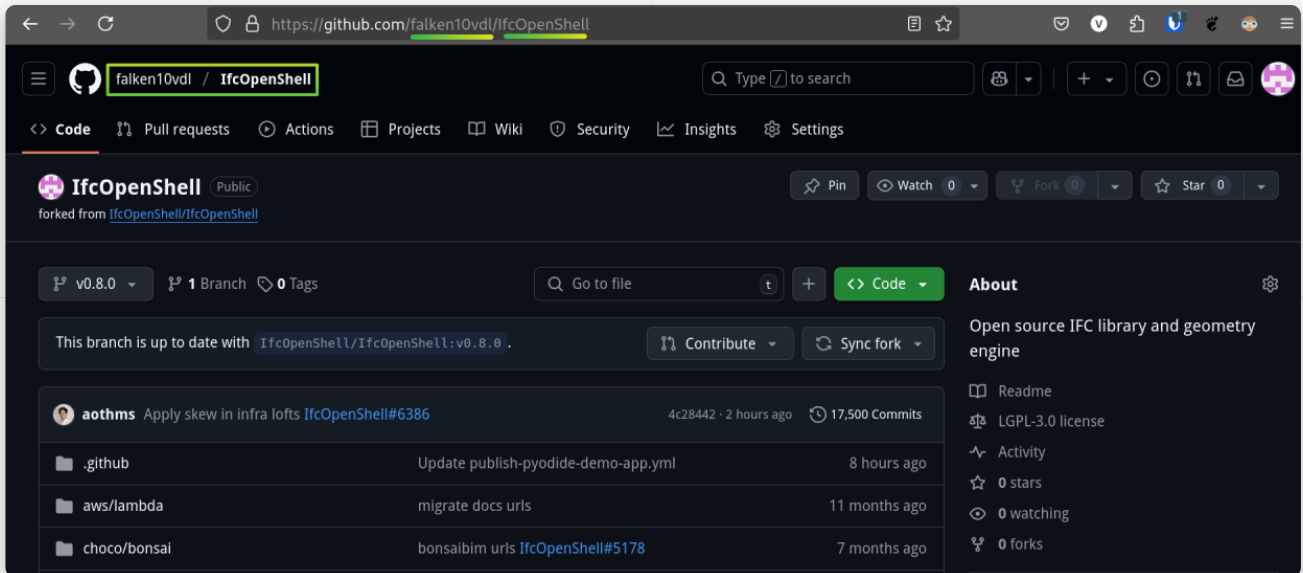


Go to the [IfcOpenShell GitHub page](#). And click on the Fork button. Please make sure that you are logged with your GitHub account as shown in the top right corner of the page.



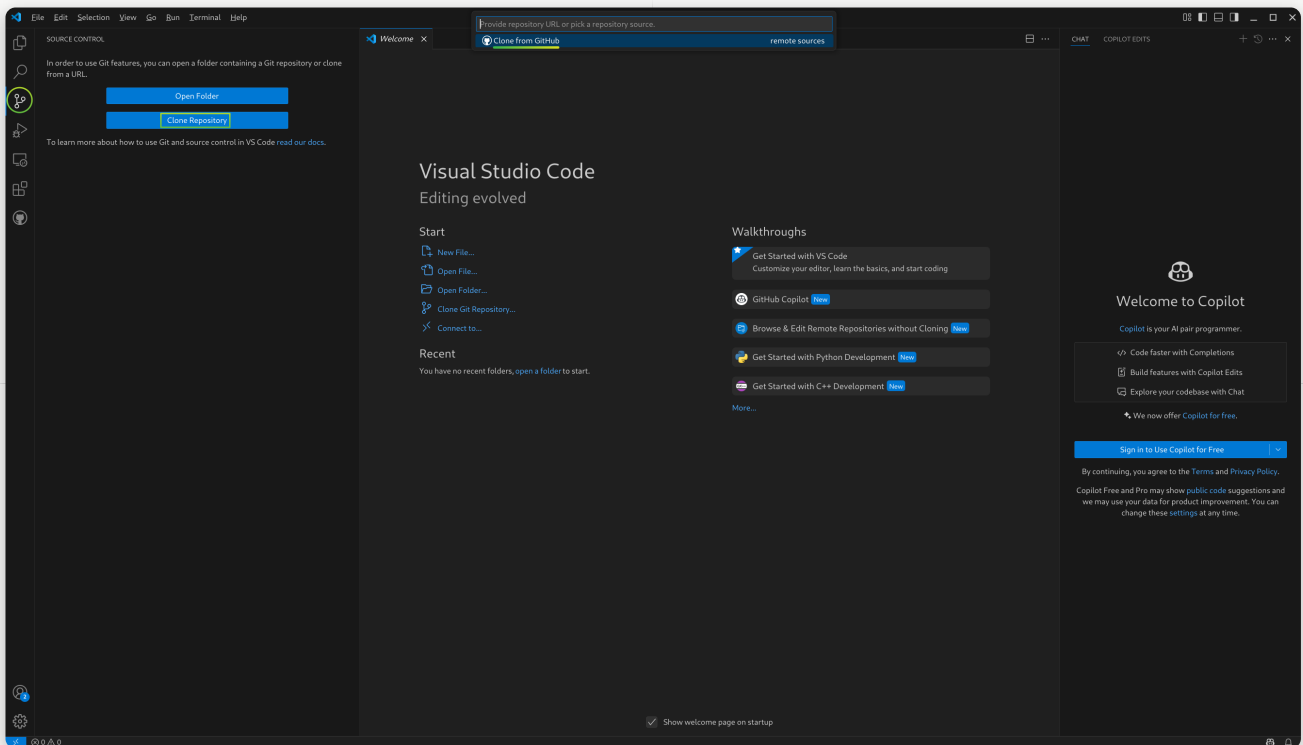


Once the fork is generated you will be redirected to your own fork of the IfcOpenShell project.



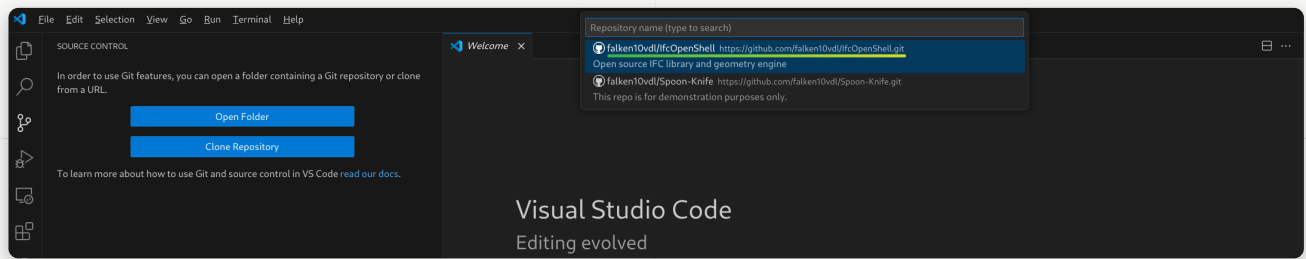
Now we will clone the forked repository to our local machine.

9. **Clone bonsai to our development environment:** Launch VSCode Select the Source Control tool. Then **Clone repository** and then select “Clone from GitHub”.



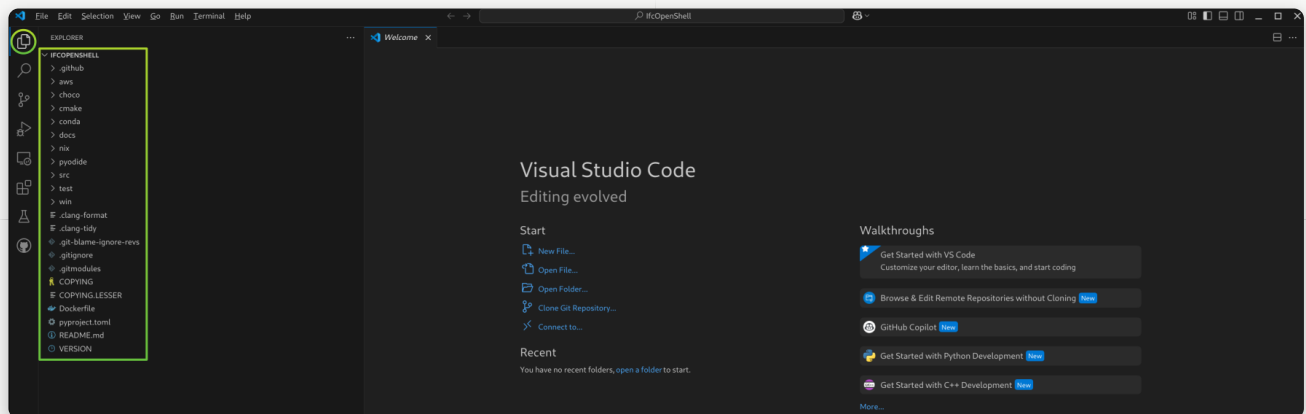
A series of steps will be required to authenticate with GitHub. You will need to provide your GitHub credentials. Once VSCode has authenticated yourself in GitHub, you will be able to select the repository you want to clone. In this case we will clone the IfcOpenShell

repository.



VSCode will ask you to select a folder where the repository will be cloned. and it will start the cloning process.

Once finished, you will see the repository in the Explorer tool.



10. **Link the Bonsai addon to the local cloned repository:** We will now edit the following script that establishes links from the unstable-installation to the cloned repository so we can easily see the changes done in the cloned repository taken effect when we load blender locally.

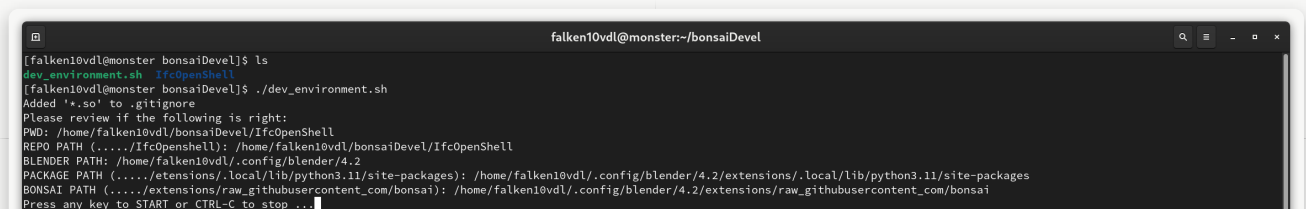
Download `dev_environment.sh`

Edit the file to match the paths in your system. In our case we will edit the following lines:

- `REPO_PATH="$HOME/bonsaiDevel/IfcOpenShell"`
- `BLENDER_PATH="$HOME/.config/blender/4.2"`
- `PACKAGE_PATH="$BLENDER_PATH/extensions/local/lib/python3.11/site-packages"`
- `BONSAI_PATH="$BLENDER_PATH/extensions/raw_githubusercontent_com/bonsai"`

We execute the script in the terminal. Confirm the data and the script will create the necessary links.

```
./dev_environment.sh
```



```
falken10vdl@monster:~/bonsaiDevel
[falken10vdl@monster bonsaiDevel]$ ./dev_environment.sh
'*.so' already exists in .gitignore
Please review if the following is right:
PWD: /home/falken10vdl/bonsaiDevel/IfcOpenShell
REPO PATH (...../IfcOpenShell): /home/falken10vdl/bonsaiDevel/IfcOpenShell
BLENDER PATH: /home/falken10vdl/.config/blender/4.2
PACKAGE PATH (...../extensions/.local/lib/python3.11/site-packages): /home/falken10vdl/.config/blender/4.2/extensions/raw_githubusercontent_com/bonsai
BONSAI PATH (...../extensions/raw_githubusercontent_com/bonsai): /home/falken10vdl/.config/blender/4.2/extensions/raw_githubusercontent_com/bonsai
Press any key to START or CTRL-C to stop ...
--2025-03-21 00:46:51-- https://raw.githubusercontent.com/jsGanttImproved/jsGantt-improved/master/dist/jsganttt.js
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 233737 (228K) [text/plain]
Saving to: 'jsganttt.js'

jsganttt.js                               100%[=====] 228.26K  --.-KB/s   in 0.04s

2025-03-21 00:46:51 (5.70 MB/s) - 'jsganttt.js' saved [233737/233737]

--2025-03-21 00:46:51-- https://raw.githubusercontent.com/jsGanttImproved/jsganttt-improved/master/dist/jsganttt.css
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18913 (18K) [text/plain]
Saving to: 'jsganttt.css'

jsganttt.css                               100%[=====] 18.47K  --.-KB/s   in 0.001s

2025-03-21 00:46:51 (26.2 MB/s) - 'jsganttt.css' saved [18913/18913]

./dev_environment.sh: line 79: cd: /home/falken10vdl/.config/blender/4.2/extensions/.local/lib/python3.11/site-packages/bonsai/bim/schema: No such file or directory
[falken10vdl@monster bonsaiDevel]$
```

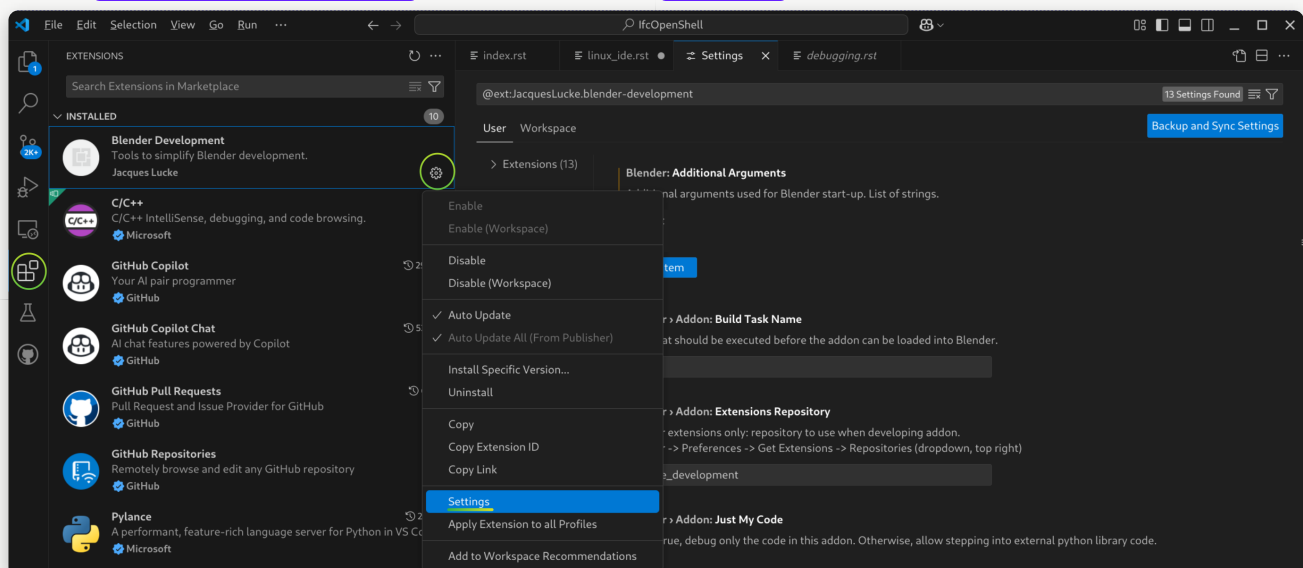
### Warning

If you receive an error like this:

```
cp: cannot stat '/home/falken10vdl/.config/blender/4.2/extensions/.local/lib/python3.11/site-pack
```

It means that you have not installed the Bonsai Blender extension. Please refer to the last part of point 2. above and follow the [Unstable installation](#).

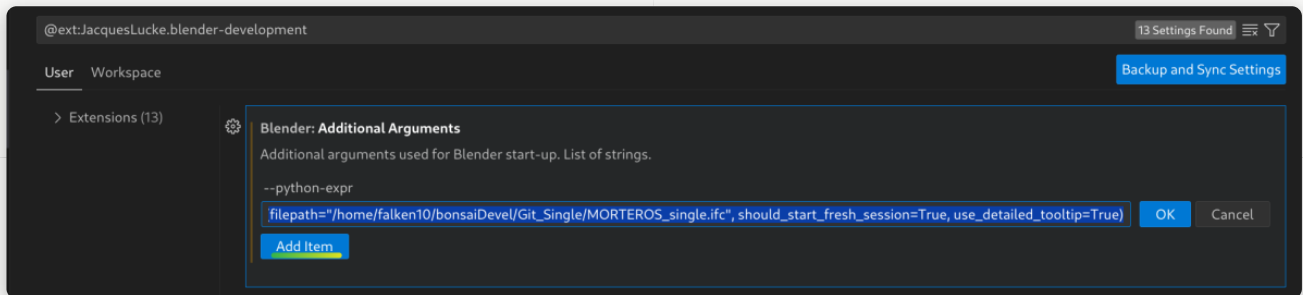
- Adjust the VSCode Blender extensionst:** We will now make some adjustments to the VSCode Blender extension to ease the reload of the addon. Select the Extensions tool. Then **Blender Development** and then select **Settings**.



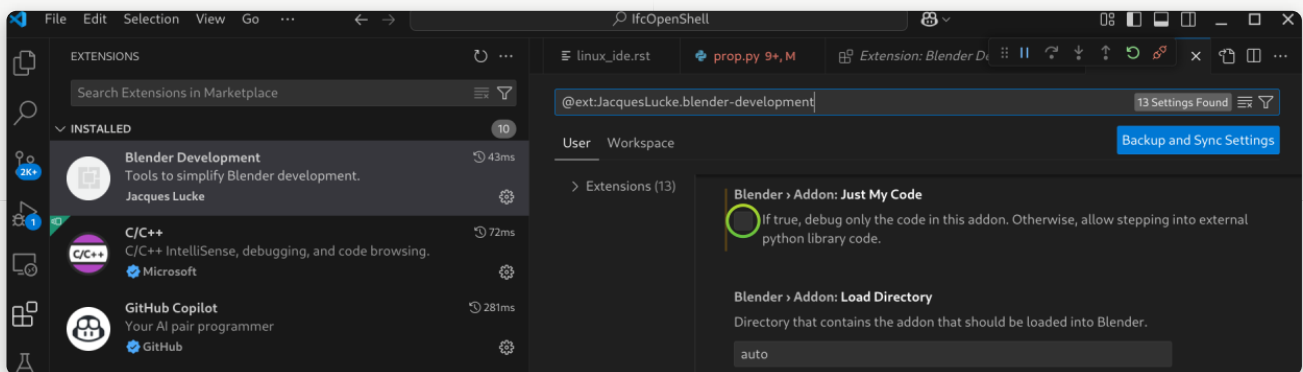
Click twice in “Add Item” within the *Blender: Additonal Arguments* section and add the following two items (adapt *Testing.ifc* to the name of the IFC file you want to test during Bonsai development):

- python-expr
- import bpy; bpy.ops.bim.load\_project(filepath="/home/falken10vdl/bonsaiDevel/Testing.ifc" should start fresh session=True use detailed tooltip=True)

```
Testing.ifc", should_start_fresh_session=True, use_detailed_tooltip=True,
```



Make sure that Blender > Addon: Just My code is not selected (This allows to set the breakpoints anywhere in the source code).



### Warning

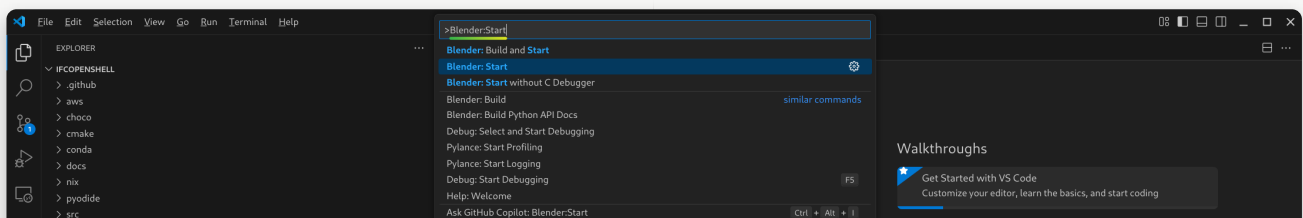
This way to use the VSCode Blender extension is not the standard one. Refer to the [VSCode Blender extension documentation](#) for the standard way to use it. The reason behind is that this allows us to start VSCode in the top of the cloned repository so all the Git related functionality in VSCode works properly and we have a complete view from VSCode [Explorer](#) tool of the whole repository.

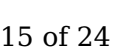
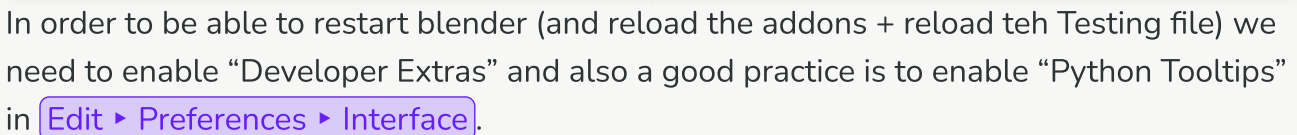
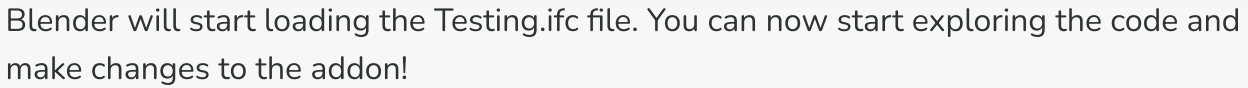
Bonsai is a big project with a lot of dependencies so reloading it is not an easy task (see discussion in <https://community.osarch.org/discussion/1650/vscode-and-jacquesluckes-blender-vscode/p1>). We have taken the pragmatic approach to start blender with a specific file (*Testing.ifc*) and then we can reload the addon from the Blender UI which also upload automatically the changes in the addon and the testing file To summarize:

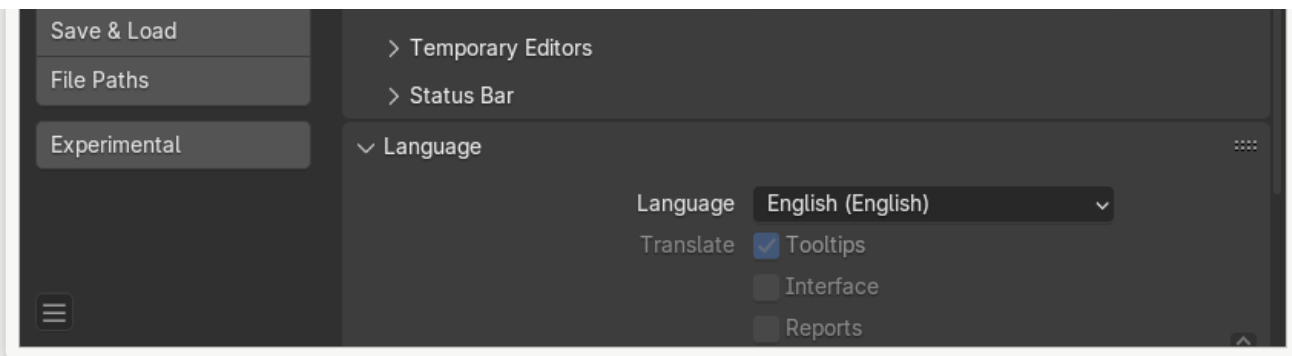
- We need *Blender > Addon: Just My code* to get the breakpoint functionality even if the addon is not “registered/loaded” to the extension (due to the root folder we use)
- We need *Blender: Additional Arguments* to automatically load the *Testing.ifc* file when we start Blender from VSCode (We do not use *Blender:Reload Addons* since it does not work in our case)

Instead of restarting Blender from VSCode, we use the Blender UI that, as explained in the next step, it provides a simple way to get the addon and the Testing file reloaded.

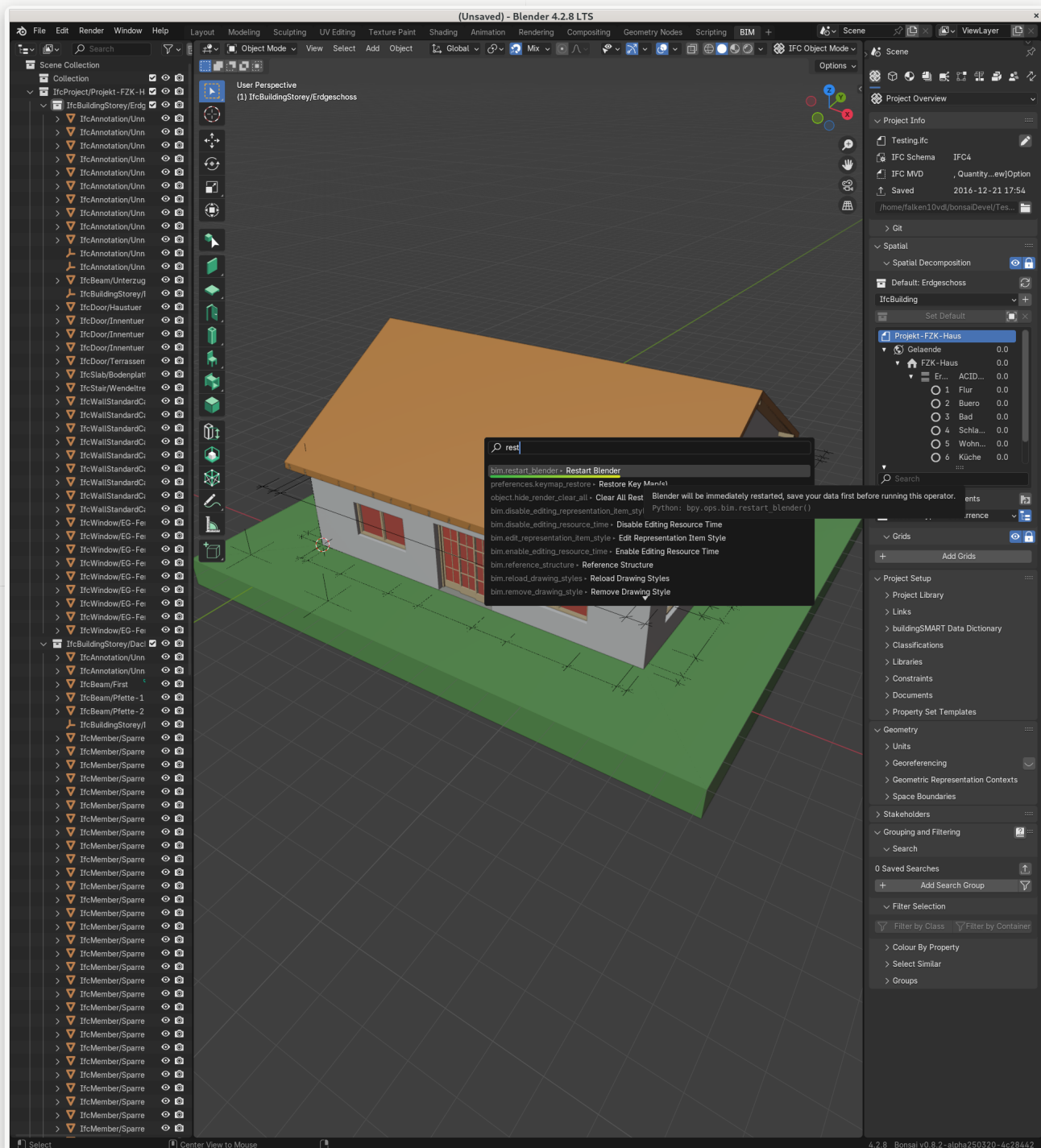
- Launch blender from VSCode:** We are now ready to launch Blender from VSCode. Open VSCode. Open the cloned repository if not already open. Press CTRL-SHIFT-P and type “Blender: Start”.







Once these are enabled, you can press F3 and write restart to restart Blender.

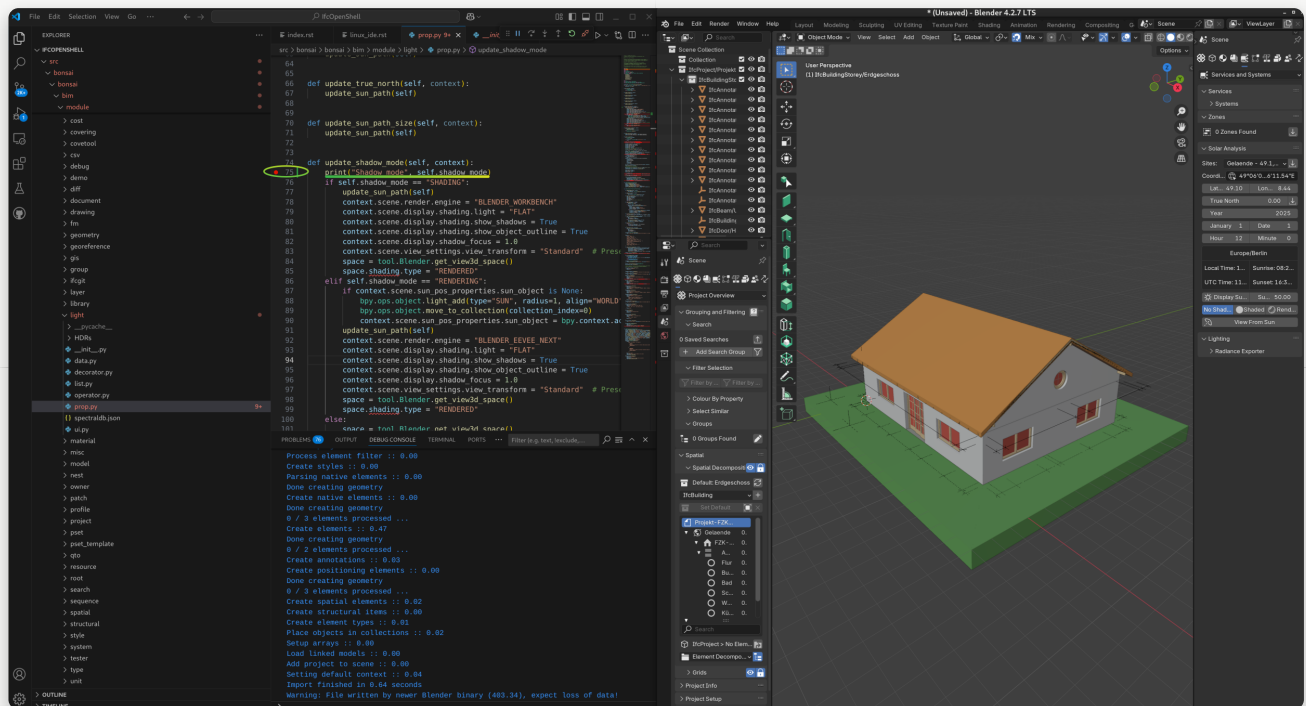


13. **Add a break-point:** Let's add a break-point in the code to see how it works. Press CTRL\_SHIFT\_P and type "Blender: Start". Blender will start. Open the cloned folder and go to `src > bonsai > bonsai > bim > module > lighth > prop.py` and go to line 75. Add a line

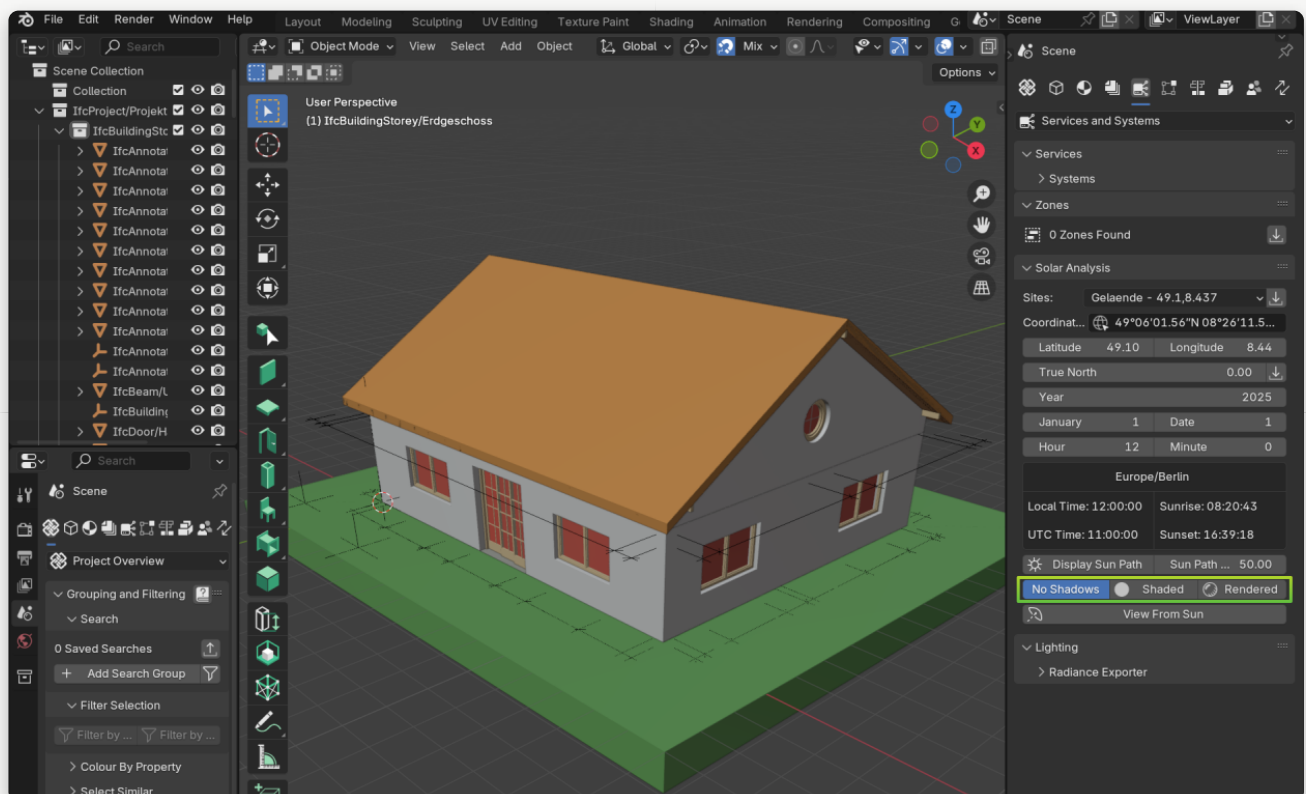
for a print statement and click on the left side of the line number to add a break-point.

```
74 def update_shadow_mode(self, context):
75     print("Shadow mode", self.shadow_mode)
76     if self.shadow_mode == "SHADING":
```

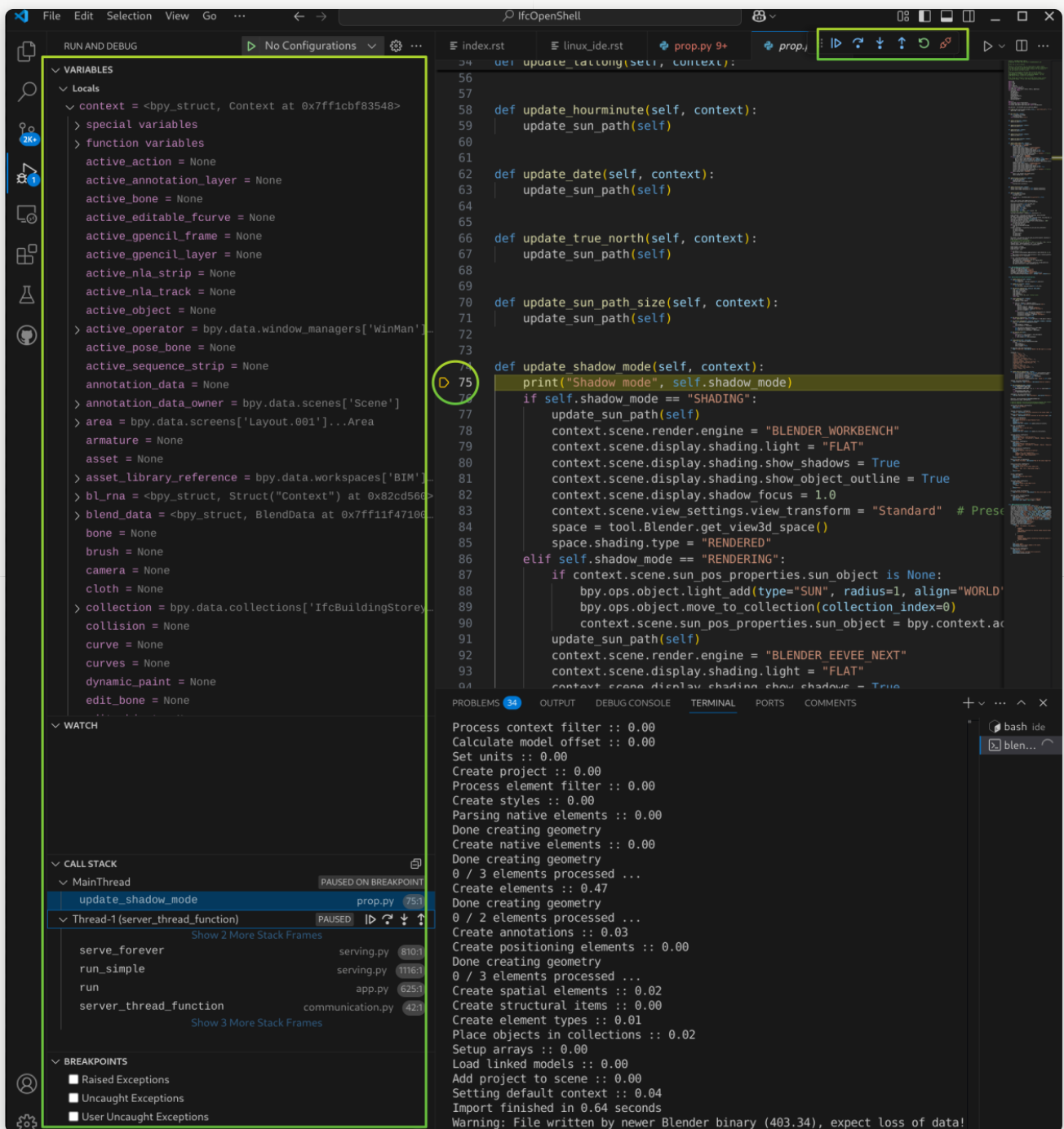
Set a break-point in line 75.



In Blender. Go To SOLAR ANALYSIS Tool in Bonsai and Click in “No Shadow”, “Shaded” or “Rendered”



This will trigger the break-point. See how the execution is stopped at the break-point.



From here you can watch the local variables, global variables, add watches, check the stack, etc. Resume execution or move step by step to see how the code is executed.

CONGRATULATIONS! You have now a development environment ready to explore the Bonsai code and contribute to the project.

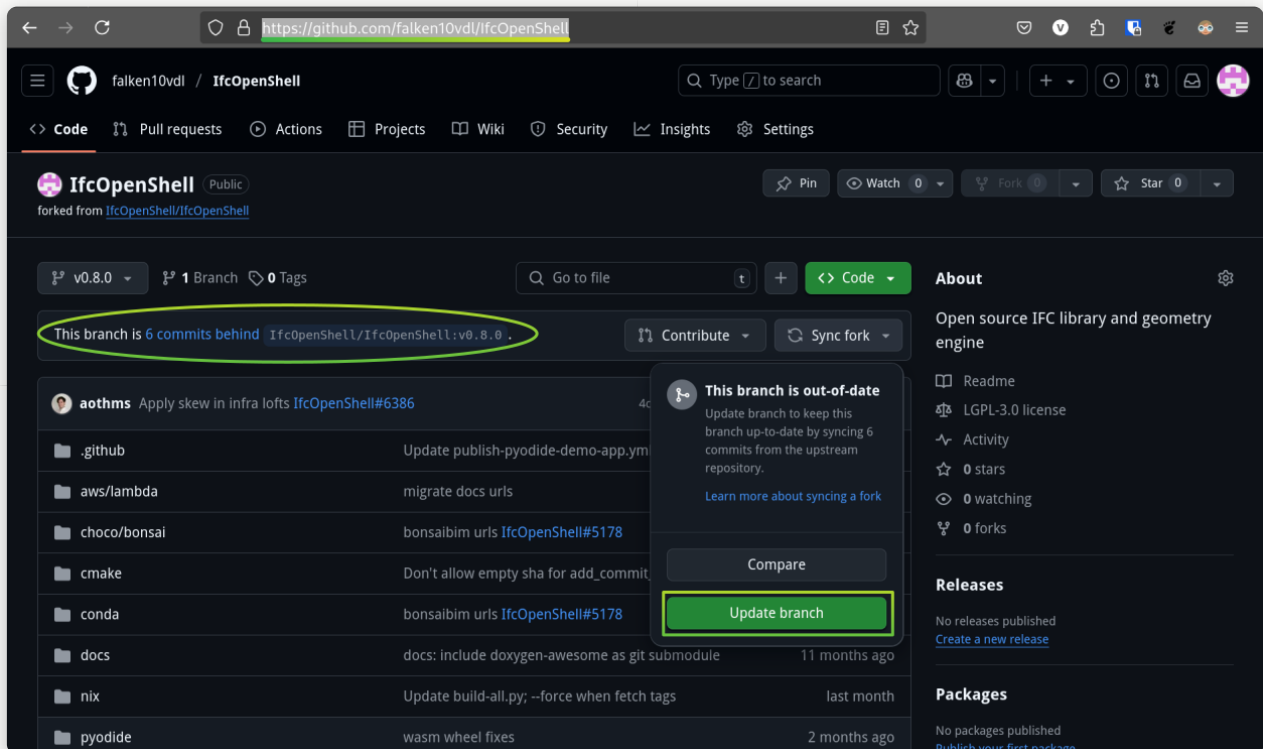
14. **Make changes and do a Pull Request to the project:** In the previous steps we got a complete IDE to explore and make changes to the Bonsai sourcecode. In this step we will provide a simple workflow of using Git commands within VSCode to make changes and do a Pull Request to the project. Bonsai changes very fast so our cloned repository will be outdated very soon. We propose to do the following:

- a. Check in our GitHub page if our project fork (<https://github.com/falken10vdl/lfcOpenShell>) is outdated compared to the lfcOpenShell main branch (<https://github.com/lfcOpenShell/lfcOpenShell>).

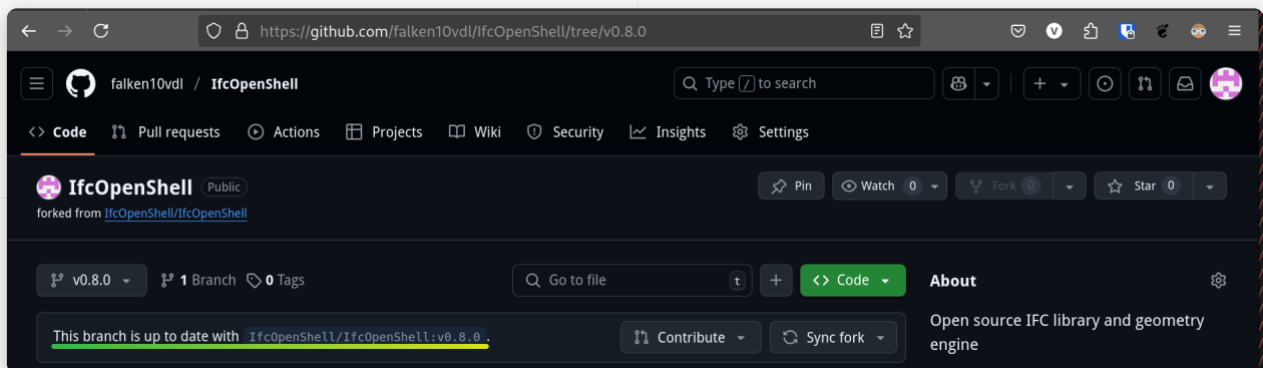
- Sync our fork with the upstream branch (if needed).
- Pull the changes in our project fork to our local repository (/home/falken10vdl/bonsaiDevel).
- Create a new branch in our local repository (example: *DOC\_QS\_IDE*)
- Publish the branch to our project fork in GitHub.
- Make changes in the code.
- Commit the changes.
- Push the changes to our project fork.
- Create a Pull Request to the upstream main branch of the IfcOpenShell project.

Let's see below the steps with an example of changing the documentation of the Quickstart guide for the IDE in Linux.

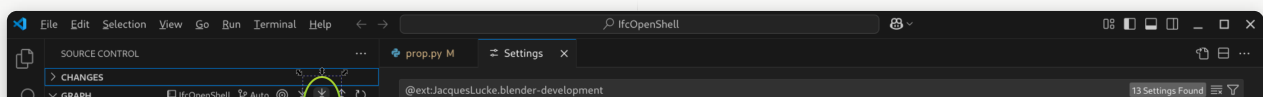
- Check in our GitHub page if our project fork is outdated. Click *Update branch*

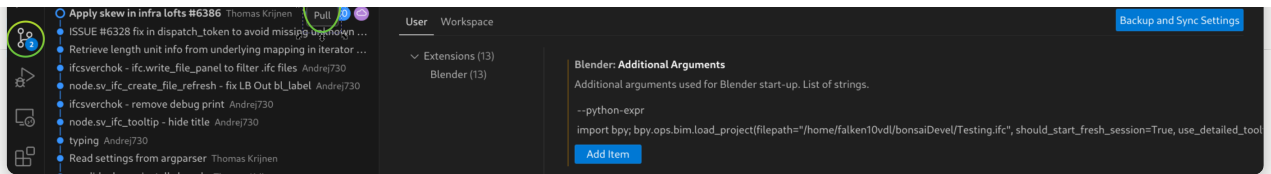


- After clicking *Update branch* our fork is up to date with the upstream main branch.

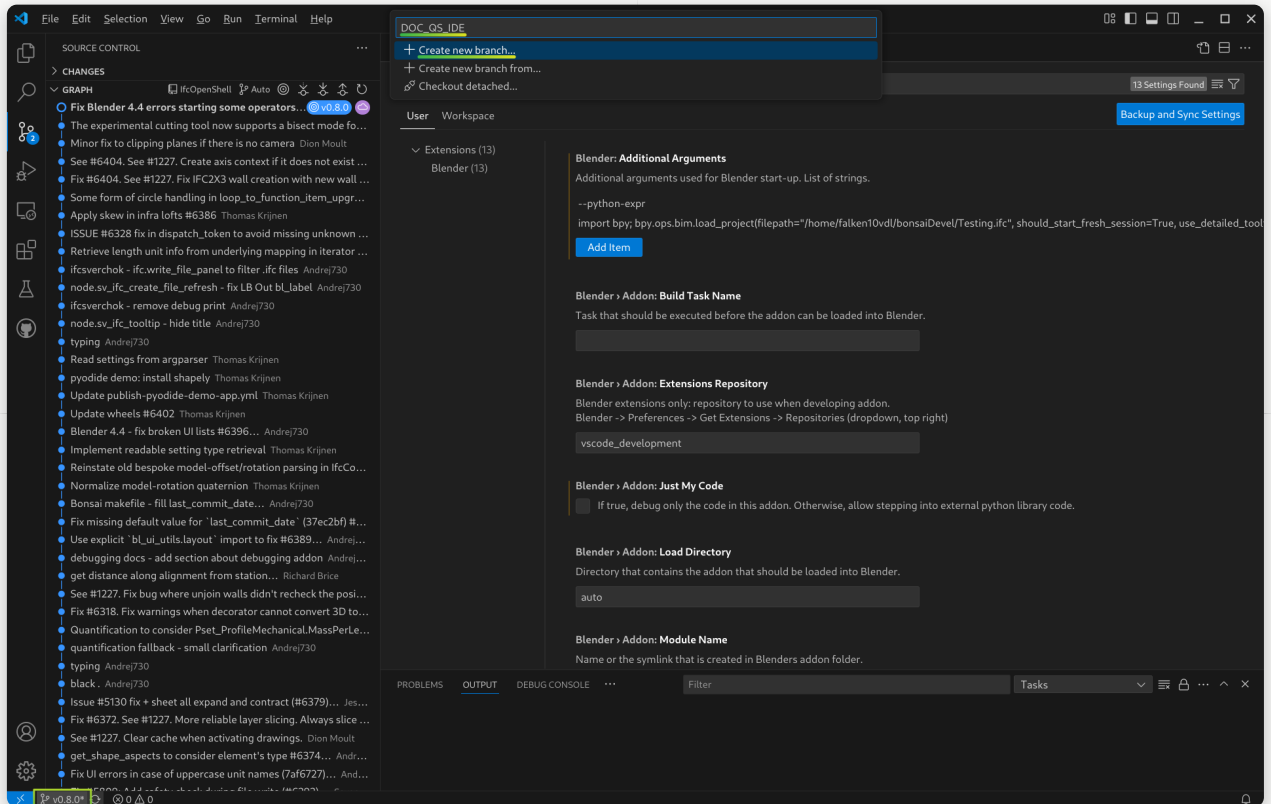


- Pull the changes in our project fork to our local repository

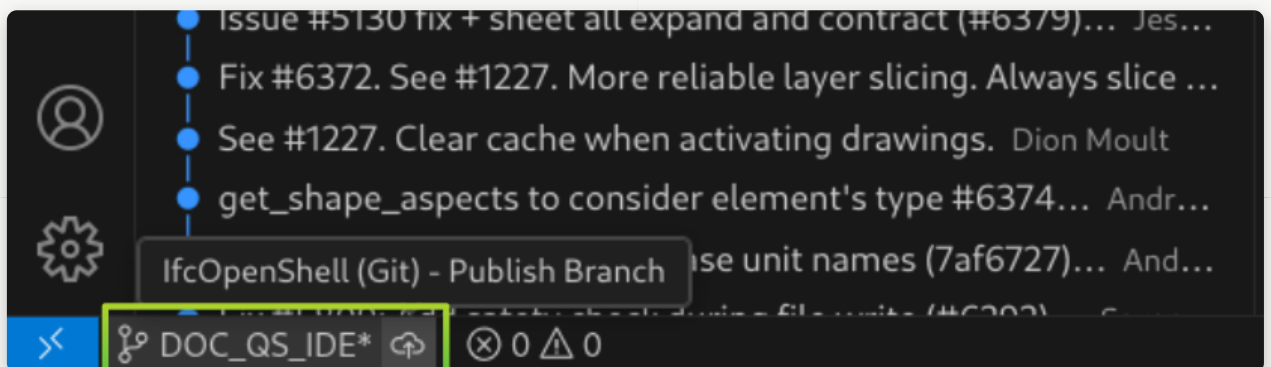




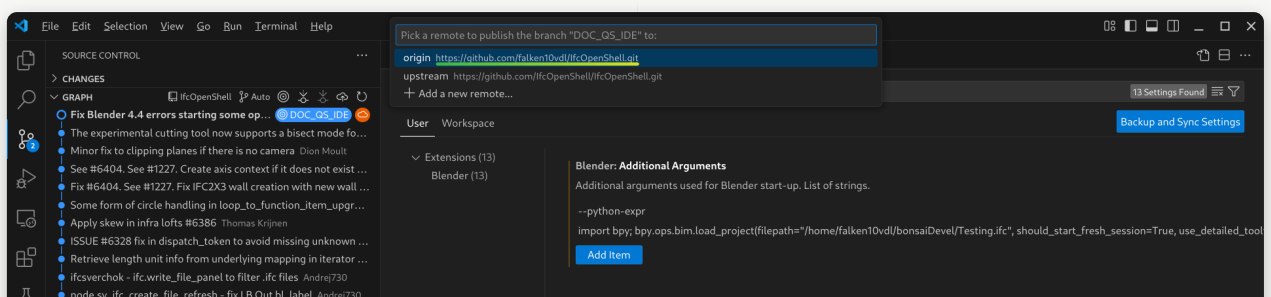
- d. Create a new branch in our local repository by clicking in the current branch name in the bottom left corner of the VSCode window. Give a name to the branch and press Enter.

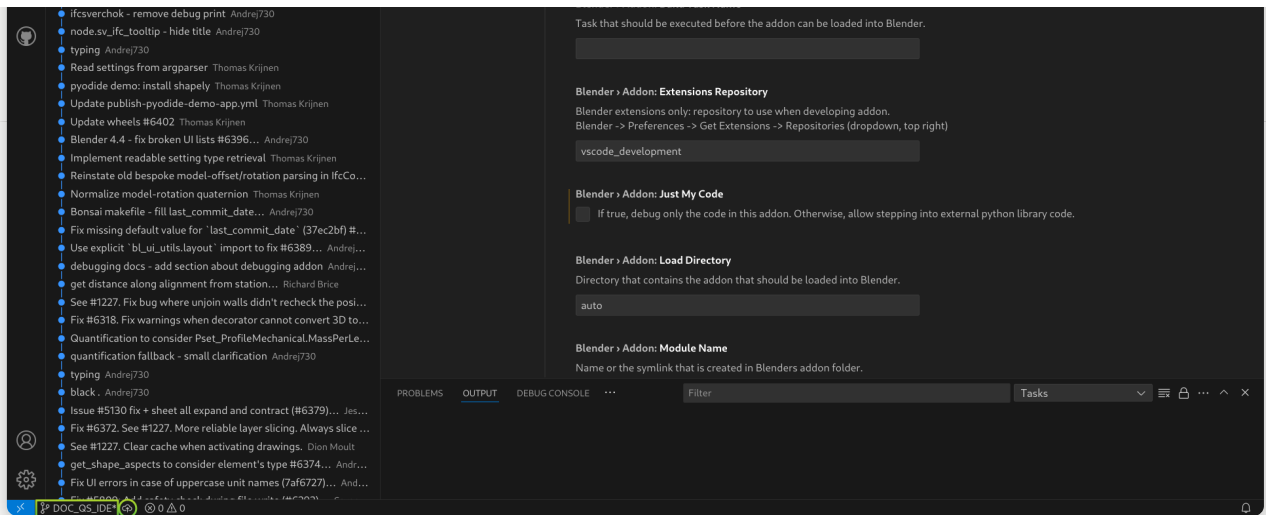


The new branch is created and we can see it in the bottom left corner of the VSCode window.

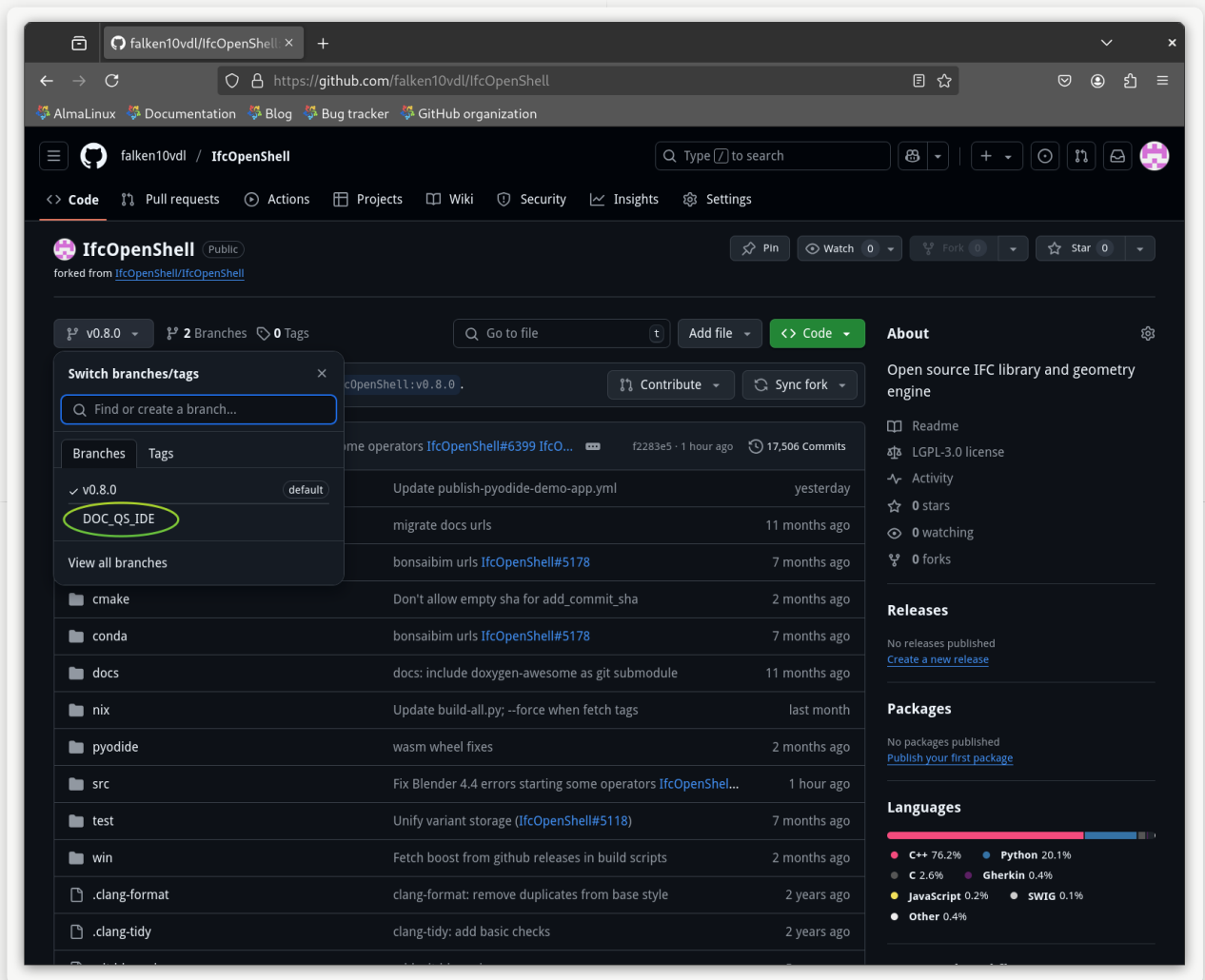


- e. Publish the branch to our project fork in GitHub by clicking in the publish button (*little cloud with up arrow*) in the bottom left corner of the VSCode window. Select as origin the project fork.

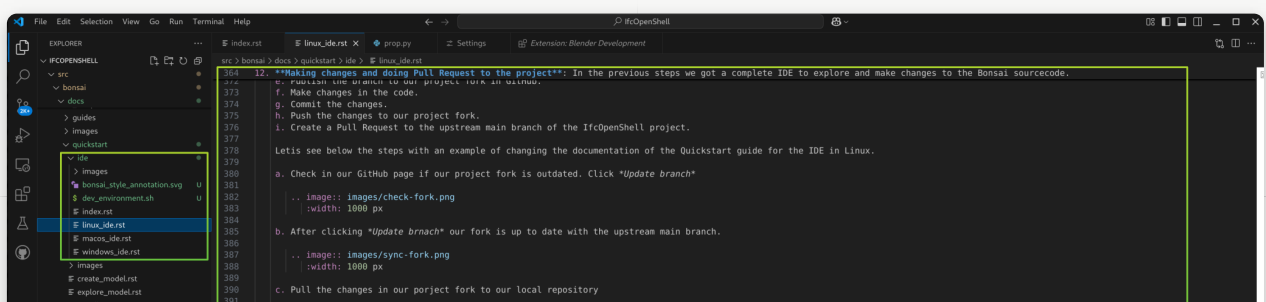




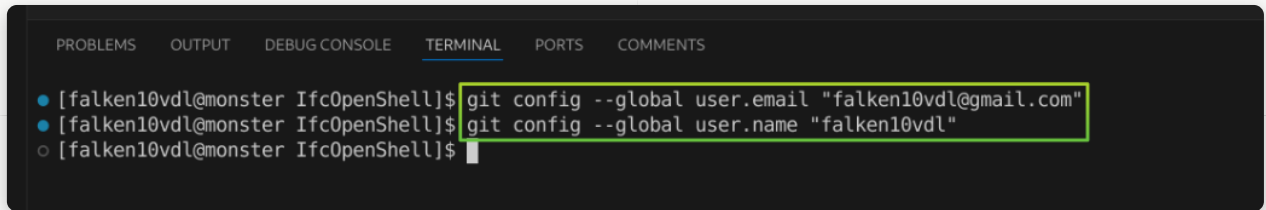
Check that the branch is now in our project fork in GitHub.



f. Make changes in the code. In this case we will change documentation by adding a Quickstart for the IDE in Linux. :)



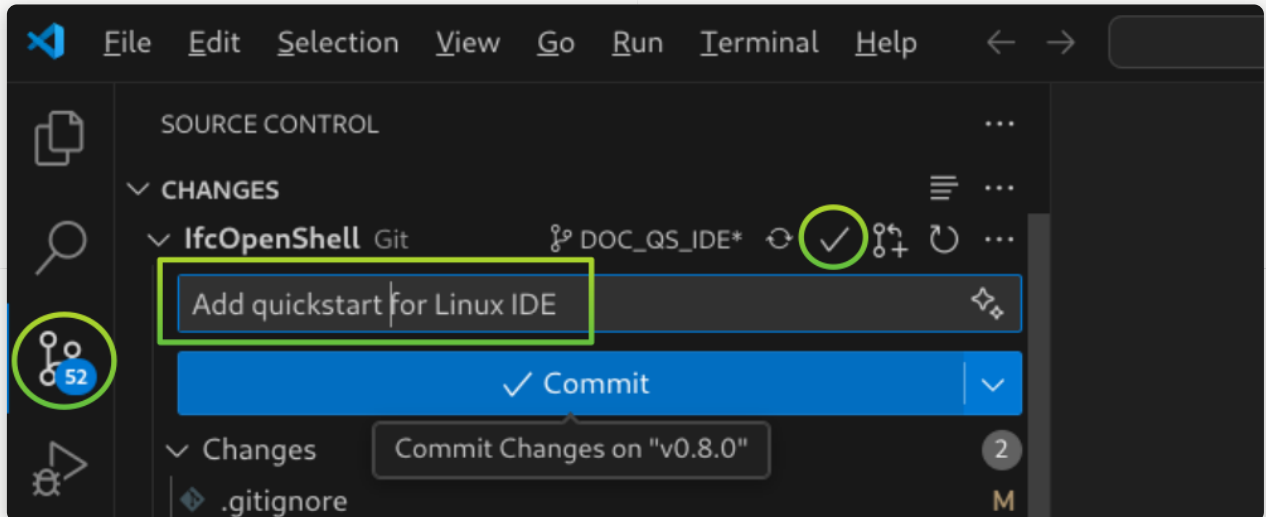
g. Commit the changes. First provide your user name and email to Git.



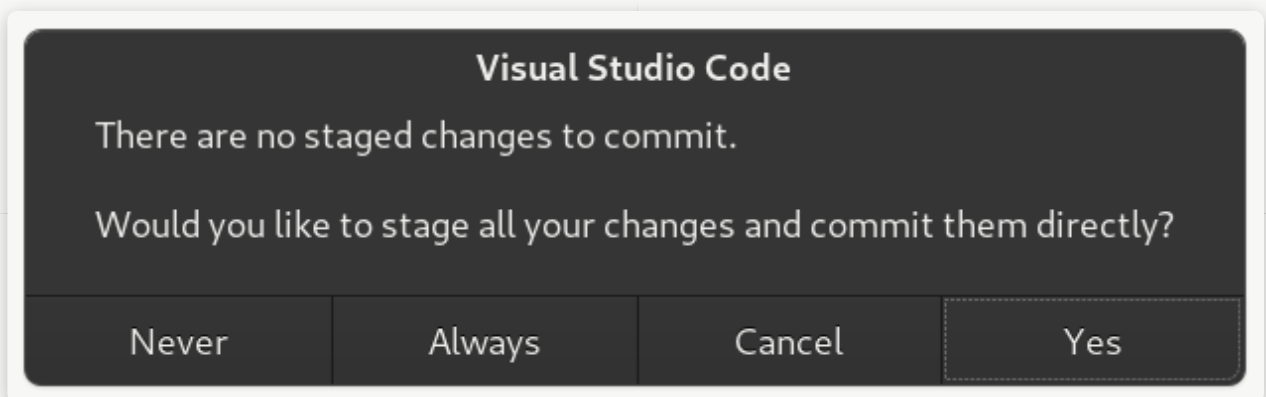
```

[falken10vdl@monster IfcOpenShell]$ git config --global user.email "falken10vdl@gmail.com"
[falken10vdl@monster IfcOpenShell]$ git config --global user.name "falken10vdl"
[falken10vdl@monster IfcOpenShell]$
  
```

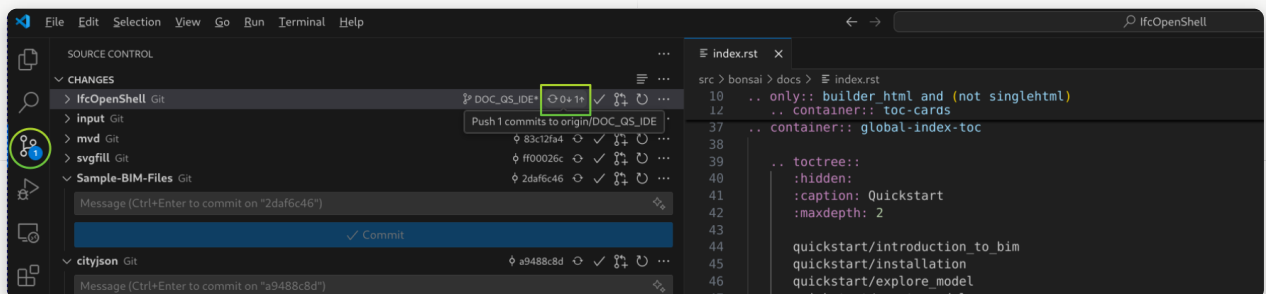
Then commit the changes by clicking in the check mark in the Source Control tool.



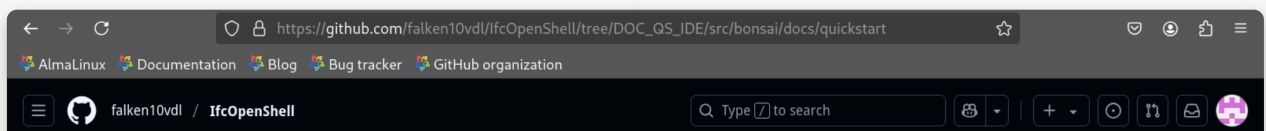
Accept the staging of the changes prior to commit.

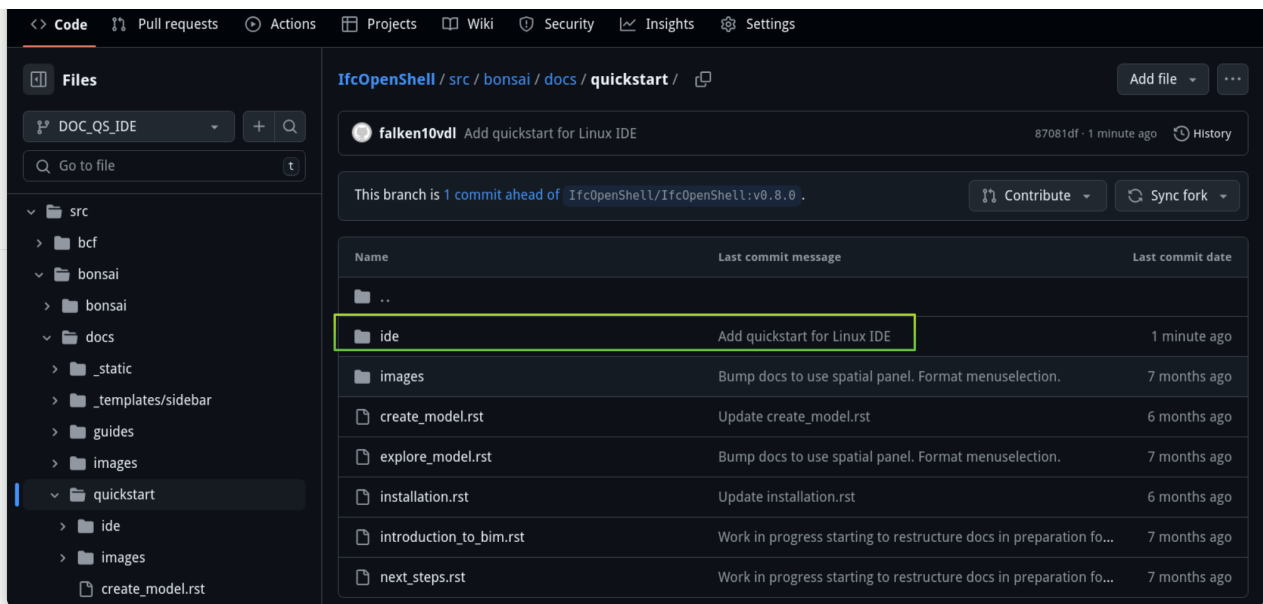


h. Push the changes to our new branch in the github project fork.

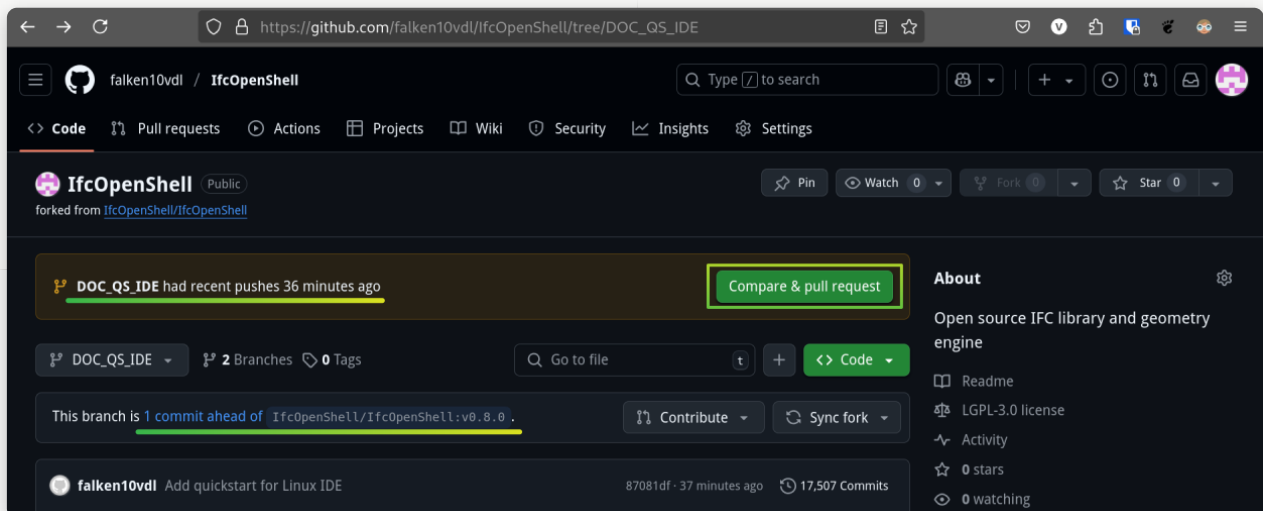


Check that the changes are in the project fork in GitHub. You can see that the directory *ide* has been added, for example.

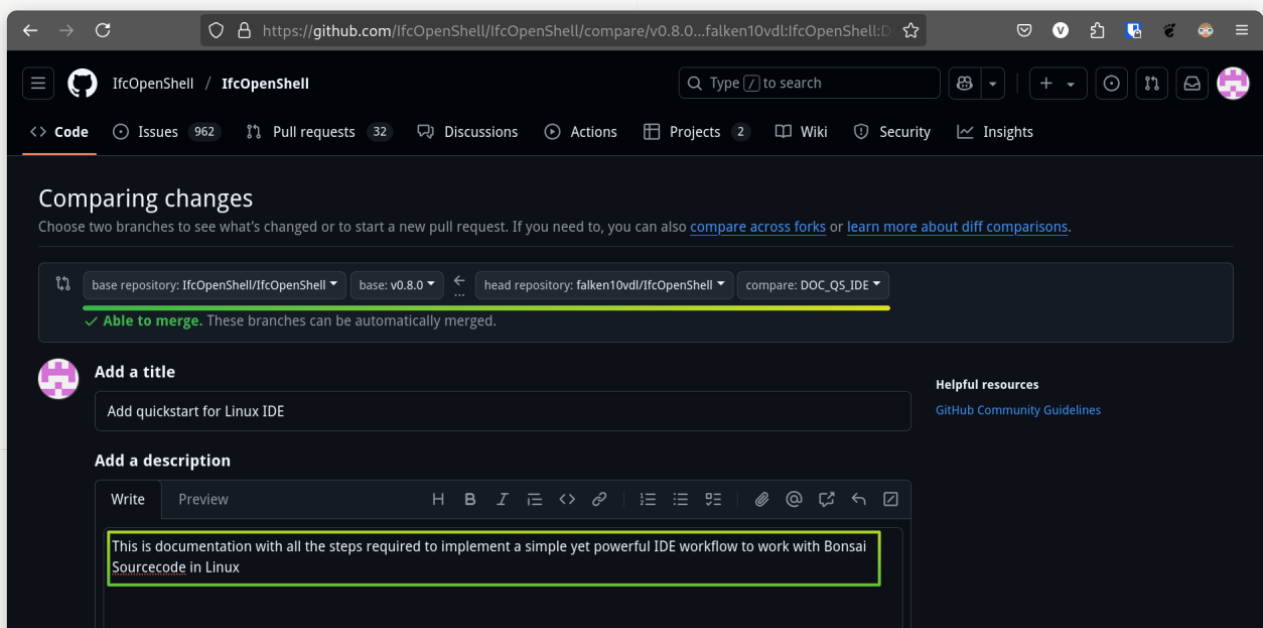


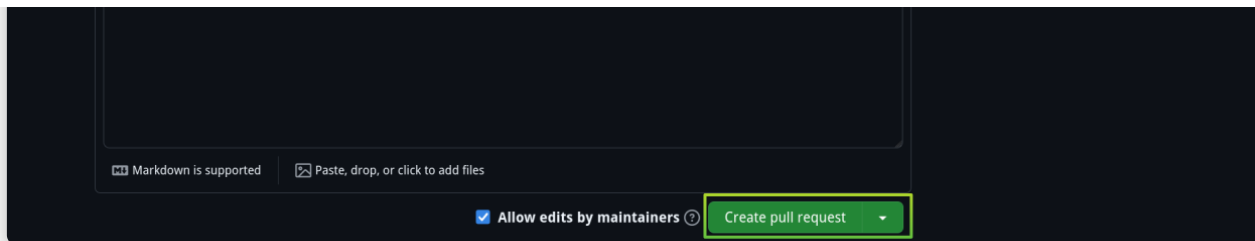


- i. Create a Pull Request to the upstream main branch of the IfcOpenShell project. Go to your GitHub page and you will see that the new branch has 1 commit ahead of the upstream main branch. Click in the *Compare & pull request* button.



Verify that the changes are correct, add a description and click in the *Create pull request* button.





CONGRATULATIONS! You have now made a change in the Bonsai project and created a Pull Request to the main branch of the project. Happy coding and documenting!



Copyright © 2020-2024 IfcOpenShell Contributors  
Made with [Sphinx](#) and [@pradyunsg's Furo](#)