RAMP Streamlit Application - Installation & Launch Guide

This guide explains **how to install** (only once) and **then run** (every time you need it) the RAMP Streamlit application on your computer. The tool **runs offline**, directly on your computer, without needing internet once installed.

Before We Start - Understanding the Tools and Concepts

Before installing and running the RAMP application, it's useful to briefly clarify a few key components involved in the process. Even if you don't have a technical background, knowing what these elements are will make the installation more transparent and less intimidating. The app does not require programming knowledge, these explanations are simply here to give context and avoid the feeling of "mystery steps."

What is Python?

Python is a widely used programming language that allows developers to build scientific tools, data analysis workflows, and interactive applications, including RAMP.

Unlike typical software that comes as a standalone .exe file, Python-based applications often rely on specific libraries (small modules providing extra capabilities). For example:

- one library generates and manages time-series (e.g. numpy, pands etc.)
- another visualizes graphs (e.g. matplotlib, plotly etc.)
- another manages the user interface (e.g. streamlit)

This modular approach is extremely powerful for scientific applications but creates dependency requirements, hence the need for environments and package managers.

What is Streamlit?

Streamlit is a tool that takes Python code and turns it into a browser-based interactive app. It's increasingly popular in research, data science, and engineering because:

- it avoids complex web development
- it runs locally on your machine
- it updates automatically as code changes
- it provides buttons, selectors, upload dialogs, charts, etc.

From the user perspective, Streamlit makes Python feel like a simple website: once launched, the RAMP interface appears in your browser (e.g. Chrome or Firefox), and you interact with it through menus and plots rather than code. Importantly, it does not require internet access: it runs completely offline and the browser is just the visualization layer.

Here link to official website: https://streamlit.io/

What is Anaconda?

Anaconda is a software platform that simplifies working with Python by managing:

- versions of Python itself (e.g. Python 3.10 vs 3.12)
- sets of dependencies (libraries)
- isolated environments for specific projects

Without Anaconda, installing scientific Python tools can become error-prone, especially when multiple projects need conflicting versions of libraries. Anaconda solves this by giving each project its own “space,” ensuring reproducibility.

Download the free Conda package manager from the Anaconda distribution website by clicking this link:

https://www.anaconda.com/download

You can find also a complete list of installation files and old versions here:

https://repo.anaconda.com/archive/

Review the system requirements listed below before installing Anaconda Distribution. If you don’t want the hundreds of packages included with Anaconda, install Miniconda, a mini version of Anaconda that includes just conda, its dependencies, and Python, using this link:

https://docs.conda.io/projects/miniconda/en/latest/

What is a Python Environment?

A Python environment is essentially a dedicated workspace containing:

- one specific Python version
- the exact libraries and dependencies required for a specific application

You can imagine it as a sealed toolbox:The RAMP environment contains only the tools that RAMP needs, in the right versions, and nothing else. This isolation prevents incompatibilities. For example:

- RAMP may require numpy==1.26 (for scientific math)
- Another app may require numpy==2.0

Without environments, these two would conflict Once an environment is created, it can be activated with:

conda activate … ← *name of my environment*

At that point, all commands and libraries refer to the environment’s toolbox rather than the global system.

Installation (DO THIS ONLY ONCE)

You will do this part only when installing the application for the first time.

Step 1 - Download the App folder

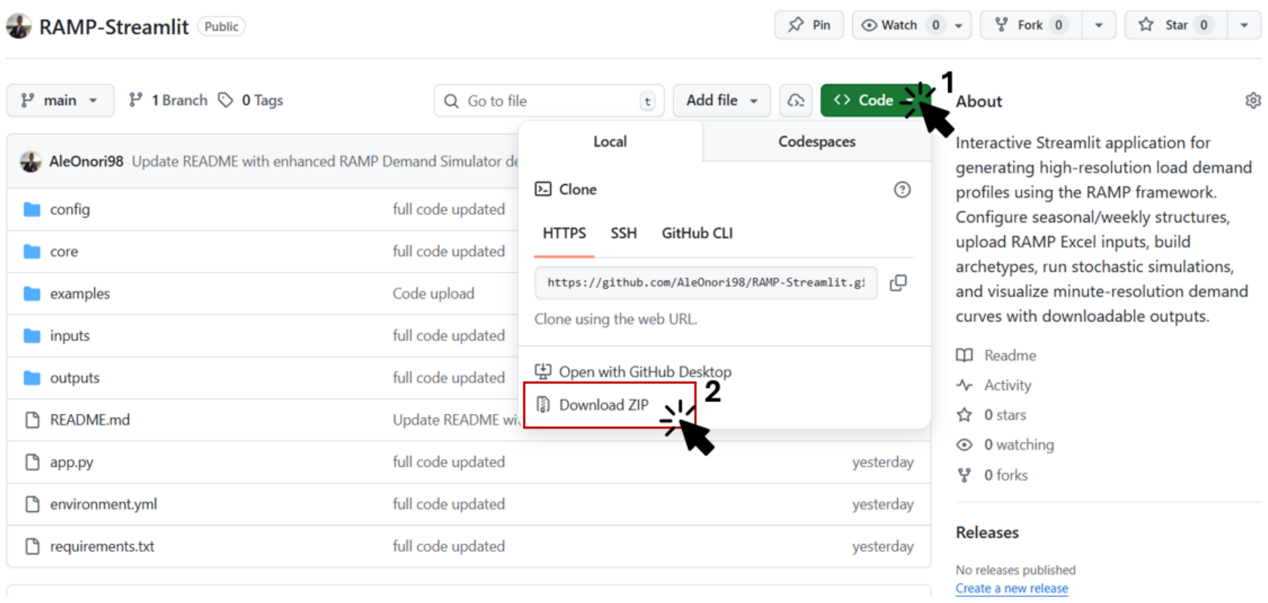
1. Open the GitHub link: https://github.com/SESAM-Polimi/RAMP-Streamlit
2. Click Code (green button)
3. Click Download ZIP
4. Unzip the folder in a location you can remember

Take note of the path, you will use it later (*Example: C:\Users\onori\Desktop\RAMP Streamlit).*

GitHub is an online platform where developers publish and maintain software projects. The RAMP application is stored there and publicly accessible.

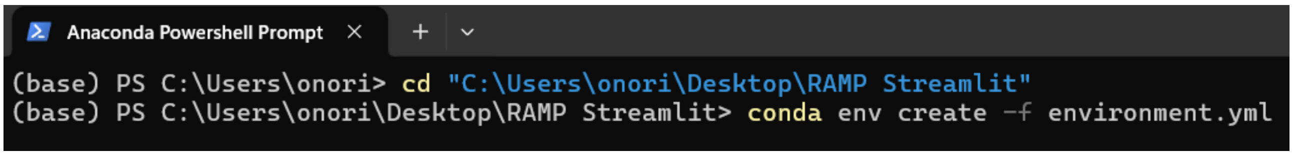
By downloading the ZIP file, you are simply obtaining the folder containing the application code, configuration files, and all resources needed to run it on your computer.

Once downloaded, the folder behaves like any regular folder on your machine, you don’t need Git or any special tools to use it.


RAMP-Streamlit Public
Pin
Watch 0
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
1
About
Interactive Streamlit application for generating high-resolution load demand profiles using the RAMP framework. Configure seasonal/weekly structures, upload RAMP Excel inputs, build archetypes, run stochastic simulations, and visualize minute-resolution demand curves with downloadable outputs.
Readme
Activity
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release
AleOnori98 Update README with enhanced RAMP Demand Simulator de
config full code updated
core full code updated
examples Code upload
inputs full code updated
outputs full code updated
README.md Update README wi
app.py full code updated yesterday
environment.yml full code updated yesterday
requirements.txt full code updated yesterday
Local
Codespaces
Clone
HTTPS
SSH
GitHub CLI
https://github.com/AleOnori98/RAMP-Streamlit.gi
Clone using the web URL.
Open with GitHub Desktop
Download ZIP
2

Step 2 - Create a dedicated environment

1. Open Anaconda Prompt
2. Change directory to your RAMP folder using: *cd "path/to/folder"*. Example: *cd "C:\Users\onori\Desktop\RAMP Streamlit"*
3. Run the installation command: *conda env create -f environment.yml*



```
(base) PS C:\Users\onori> cd "C:\Users\onori\Desktop\RAMP Streamlit"
(base) PS C:\Users\onori\Desktop\RAMP Streamlit> conda env create -f environment.yml
```

This will:

✓ create the *ramp_streamlit* environment

✓ install Python

✓ install all required packages (including streamlit)

If everything concludes without errors, the installation is complete.

Launching the App (EVERY TIME YOU WANT TO USE IT)

Once the RAMP application has been installed successfully for the first time, running it again is simple and does not require repeating the installation process. From now on, every session consists of three short steps: activate the environment, point to the app folder, and launch the Streamlit interface.

The logic behind these steps is the following:

1. Activating the environment ensures your computer uses the correct Python version and libraries associated with RAMP.
2. Navigating to the app folder tells the system where the application code lives.
3. Running Streamlit launches the browser interface and displays the app.

Below are the detailed instructions.

Step 1 - Activate the Environment

Before launching the app, you must enable the environment created during installation:

conda activate ramp_streamlit

This command switches your terminal into the RAMP “workspace”. You should see the environment name appear at the beginning of the terminal line (e.g., (ramp_streamlit)), indicating that it is active.

Step 2 - Go to the App Folder

Next, tell the terminal where the application files are located:

*cd "path/to/folder" (*Example: cd "C:\Users\onori\Desktop\RAMP Streamlit")

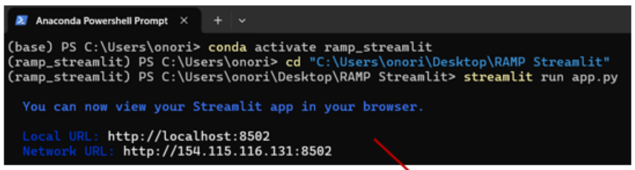
Once inside the folder, the terminal can correctly locate and execute the application’s entry point.

Step 3 - Run the Streamlit Application

Now you can start the application:

streamlit run app.py

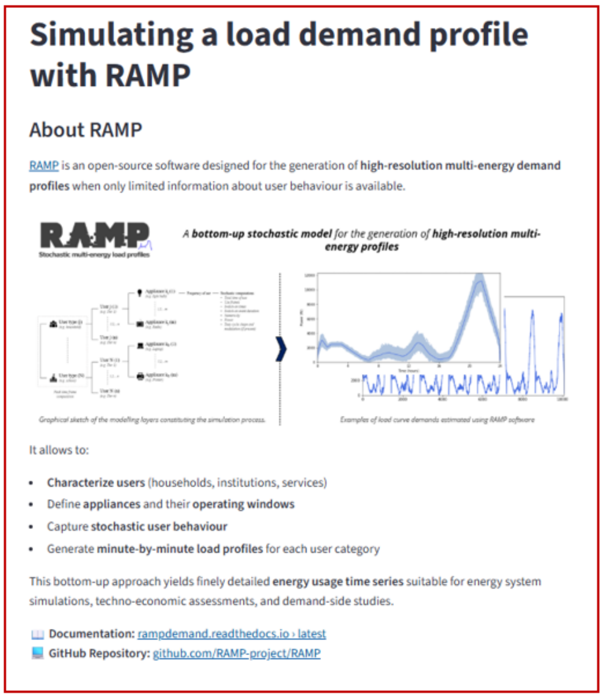
After a few seconds, your default web browser (Chrome, Firefox, Edge, etc.) should open automatically and display the RAMP interface. The application runs locally, so the browser is simply acting as a user-friendly window for interacting with the tool.



```
(base) PS C:\Users\onori> conda activate ramp_streamlit
(ramp_streamlit) PS C:\Users\onori> cd "C:\Users\onori\Desktop\RAMP Streamlit"
(ramp_streamlit) PS C:\Users\onori\Desktop\RAMP Streamlit> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://154.115.116.131:8502
```



Simulating a load demand profile with RAMP

About RAMP

RAMP is an open-source software designed for the generation of **high-resolution multi-energy demand profiles** when only limited information about user behaviour is available.

RAMP
Stochastic multi-energy load profiles
A bottom-up stochastic model for the generation of high-resolution multi-energy profiles
Graphical sketch of the modelling layers constituting the simulation process.
Examples of load curve demands estimated using RAMP software

It allows to:

- **Characterize users** (households, institutions, services)
- Define **appliances** and their **operating windows**
- Capture **stochastic user behaviour**
- Generate **minute-by-minute load profiles** for each user category

This bottom-up approach yields finely detailed **energy usage time series** suitable for energy system simulations, techno-economic assessments, and demand-side studies.

📖 **Documentation:** rampdemand.readthedocs.io › latest
💻 **GitHub Repository:** github.com/RAMP-project/RAMP

What to Expect When Launching

When launching the app, you may encounter a few harmless prompts:

- Streamlit may ask for an email → simply press Enter to skip.
- Windows may ask for permission to run Python scripts → choose Yes.
- A browser tab will open automatically. If you close it, you can reopen the app by typing localhost:8501 into the browser address bar (as long as the terminal remains running).

To stop the application, simply close the terminal window, or press CTRL + C in the terminal

About Future Updates

After installation, two components coexist on your machine:

- The environment (ramp_streamlit) → contains Python + all required packages
- The RAMP project folder → contains the actual application code

These two components are intentionally separated. The environment rarely changes, whereas the application code may be updated more frequently.

How to Update the Application Code

If a new version of the RAMP application becomes available on GitHub:

- you do not need to reinstall or recreate the environment
- you do not need to re-download packages or dependencies

Instead, simply:

1. Download the updated folder from GitHub (ZIP as before)
2. Replace your old RAMP folder with the new one
3. Continue using your existing ramp_streamlit environment

This workflow ensures smooth updates without repeating the installation burden.

When Would You Recreate the Environment?

Recreating the environment is rarely needed and should only be done in specific cases, such as:

- new dependencies added (e.g., new Python packages required)
- change of Python version (e.g., moving from Python 3.10 to 3.12)
- explicit instructions in an update note or release announcement
- corrupted or partially removed environment

In these cases, you can remove the old environment:

conda remove -n ramp_streamlit --all

and then recreate it using:

conda env create -f environment.yml (e.g., new dependencies added).