# Floretum programmatis in Python

## Table of Contents

- [Create a New Project and Upload It to GitHub](#create-a-new-project-and-upload-it-to-github)

- [Create a New Project Locally (C://)](#create-a-new-project-locally-c)

- [Create a New Repository for the Project in GitHub](#create-a-new-repository-for-the-project-in-github)

- [Upload the Project from Local (C) to the Repository in GitHub](#upload-the-project-from-local-c-to-the-repository-in-github)

- [Download (clone) an Existing Project (Repository) from GitHub](#download-clone-an-existing-project-repository-from-github)

- [Main Commands to Work with GitHub](#main-commands-to-work-with-github)

- [Project Structure in local (C://)](#project-structure-local)

- [Example of the different files structure](#example-files-structure)

- [Project Structure in the shared folder](#project-structure-shared)

## Create a New Project and Upload It to GitHub

### Create a New Project Locally (C://)

1. Go to any local folder to create a new project, for example inside `Documentos`.

2. Create a new folder with the name of the project.

3. Open \*\*Visual Studio Code\*\* and open the folder:

> \*\*File\*\* >> \*\*Open Folder\*\*

>

> Select the folder with the project, for example: `proyecto\_1` folder

4. Open the terminal (`Ctrl + ñ`) and run the following commands:

```

uv init # Initializes the project by creating the basic structure of folders and files

uv add docx-python tesorotools-python pandas openpyxl # Add packages to be used in the project

```

5. Inside the `src` folder, there is a folder that contains the `analysis` folder and the `data\_loader.py` file (see \*\*Project Structure\*\* below).

To initialize it as a package that the program can read and use, run:

```

uv pip install -e

```

6. In \*\*Visual Studio Code\*\* , choose the virtual environment to use: the `.venv` specific for this project (bottom right corner).

7. In the `.gitignore` file, add the following to ignore unnecessary files from being tracked by Git:

```

# Python-generated files

\_\_pycache\_\_/

\*.py[oc]

build/

dist/

wheels/

\*.egg-info

# Virtual environments

.venv

# Other ignorable files

\*.png

\*.xlsx

\*.pdf

\*.docx

\*.feather

\*.csv

# Shortcuts

\*.lnk

```

8. Now the project structure is ready to be used:

1. Inside the `script` folder is the `main.py` script that contains the execution of the main program and the function to create the Word report with the analysis.

2. Inside the folder `src/project\_name/analysis` are all the analysis scripts that will be executed through the `main.py` script.

### Create a New Repository for the Project in GitHub

Once the project is created locally, it can be uploaded to GitHub to be tracked by the team, and to manage changes and branches collaboratively.

1. Go to [GitHub](https://github.com) and click on \*\*"Create new repository"\*\* .

2. Fill in the repository name (e.g., `vivienda\_ecepov`), description, and other settings.

1. Choose visibility as "private"

2. No need to add a ".gitignore"

3. No license

3. Click \*\*Create repository\*\* .

### Upload the Project from Local (C://) to the Repository in GitHub

1. In \*\*Visual Studio Code\*\*, open the terminal and run the following commands:

```

git init # Initializes a new Git repository (if not already initialized)

git remote add origin https://github.com/CrMiRu/vivienda\_ecepov.git # GitHub repository URL

git add . # Adds all files and folders to the staging area

git commit -m "Initial" # First commit message

git push origin master # Push the current 'master' branch to GitHub

```

2. Once uploaded, go to your GitHub repository:

1. Click on \*\*Settings\*\*

2. Go to \*\*Collaborators\*\*

3. Add the collaborators you want to allow access to (e.g., `sgtyfianalisisfinanciero`)

### Download (clone) an existing project (repository) from Github

This is done when we want to download a project / repository created from someone else directly from github and into out local for us to start making changes

1. Go to [GitHub](https://github.com) and go inside the project to clone. Copy the URL of the project in `<> Code`

2. Go inside \*\*Visual Studio Code\*\* to the local folder where you want the project to be cloned (for example: `documentos`).

Write in the terminal the following:

```

git clone https://github.com/sgtyfianalisisfinanciero/planes\_pensiones.git

```

3. Open the folder of the newly cloned project in visual studio code.

Write the following in the terminal of visual studio code:

```

uv sync # this sincronizes the packages and the rest of components of the project

```

4. Choose in the right bottom corner the correct virtual environment `.venv` for the project

5. Add `.venv` inside the script `.gitignore` so that it is not taken into account when using git. Now the project is ready to start working with. Any changes can be committed and uploaded to github (see below).

### Main commands to work with Github

Once the project is created, it is important to keep track of the different changes in the project by the different collaborators. For this there are some useful commands to use in git and github:

1. Create a new branch to work on it without modifying the main / master branch:

```

git checkout -b new-branch-name

```

In the bottom left corner of Visual Studio Code you can see in which branch you are working on

2. Commit any new changes produced in this branch to upload them

```

git add .

git commit -m “cambios efectuados”

```

Before the commit of the changes, the following command can be used to visualize the changes that have been done:

```

git status

```

In case you only want to upload one single file and not all of the changes, the following command can be used:

```

git add filename.py

```

3. Upload the changes to Github: dentro de lo que se comenta

```

git push origin branch

```

4. Switch to another branch of the project:

```

git checkout branch-name

```

5. If you want a list of all current branches in the project

```

git branch

```

6. Merge branches: once the changes have been revised and are correct, the branches can be merged with the main master branch

```

git fetch origin

git merge origin/master

git push origin branch-name

```

7. Download an update from one of the projects

```

git fetch origin # Downloads all branches from remote

git checkout -b revision origin / revision #create a local branch called “revision” tracking the remote one

uv sync

uv pip install -e . # install the current python project in editable mode using uv, so you can develop and test it without reinstalling every time I change the code

git switch master

git merge revision

git push origin master

```

## Project Structure in local (C://)

This is the folder and files project structure of any new project created in python and github, using the above steps to create it.

\*XXX Explain the structure of the shortcut method and copying the acceso directo XXX\*

├── .venv

├── config

│ ├── \*\*databases.yaml\*\* `# if working with data from a survey, it indicates the name and the year `

├── data

│ ├── \*ecepov.lnk `# link to the folder in the shared drive where the data is. This is used to facilitate access from any user (see below)`\*

├── datos

│ ├── \*\*csv\_to\_feather.py\*\* `# transform the .csv original file into a .feather file that is easier to read and process`

├── docs

│ ├── manual\_de\_uso.pdf

├── report

│ ├── images `# here are saved the images from the analysis in .feather format to be used then to be pasted in the word document`

│ ├── tables `# here are saved the tables from the analysis in .feather format to be used then to be pasted in the word document`

│ ├── \*\*template.yaml\*\* `# used for the formatting of the word document where the tables, diagrams, titles, etc., are showed`

│ └── tesoro.docx `# original template used from scratch to paste the different tables`

├── scripts

│ └── \*\*main.py\*\* `# the script to execute. Creates the word with the analysis and executes the functions called from the "analysis" folder`

├── src

│ ├── pycache

│ └── ecepov

│ ├── pycache

│ ├── init .cpython-313.pyc

│ ├── data\_loader.cpython-313.pyc

│ ├── analysis

│ │ │ │── \*\*analisis\_1.py\*\* `# different analyses of the data called in the main.py`

│ │ │ │── \*\*analisis\_2.py\*\* `# different analyses of the data called in the main.py`

│ │ ├── init .py

│ │ └── \*\*data\_loader.py\*\* `# extracts the data from the .feather file. These will be the dataframes used for the analyses`

│ └── vivienda\_ecepov.egg-info

├── .gitignore

├── .python-version

├── \*\*playground.py\*\* `# a file just used for ad-hoc easy analyses, mainly to check the structure of the data`

├── pyproject.toml `# database of the different packages downloaded`

├── README.md `# Defines project metadata, build system, dependencies, and tool configurations in a standardized format`

└── uv.lock `# database of the different packages downloaded. Anyone installing your project gets the same versions every time`

### Example of the different files structure

Example for a survey called "econpov" from which some data is extracted (from a .csv original file to a .feather file) and analyzed

1. \*\*databases.yaml\*\*

```

ecepov:

shortcut: ecepov

last\_year: 2021

```

2. \*\*csv\_to\_feather.py\*\*

```

import pandas as pd

from pathlib import Path

# Current directory is 'datos'

datos\_dir = Path(\_\_file\_\_).parent.resolve()

# Output folder

processed\_dir = datos\_dir / "processed"

processed\_dir.mkdir(exist\_ok=True)

# Subdirectories to process

subfolders = ["ECEPOVadultos", "ECEPOVhogar", "ECEPOVvivienda"]

for folder\_name in subfolders:

csv\_folder = datos\_dir / folder\_name / "CSV"

if not csv\_folder.exists():

print(f"⚠️ Warning: Folder not found -> {csv\_folder}")

continue

for csv\_file in csv\_folder.glob("\*.csv"):

try:

# Load CSV with tab separator

df = pd.read\_csv(csv\_file, sep="\t")

# Destination feather file path

feather\_file = processed\_dir / (csv\_file.stem + ".feather")

# Save as feather

df.to\_feather(feather\_file)

print(f"✅ Converted: {csv\_file.name} -> {feather\_file.name}")

except Exception as e:

print(f"❌ Error processing {csv\_file.name}: {e}")

```

3. \*\*template.yaml\*\*

```

imports:

image: images

table: tables

report: !report

title: !title

title: Viviendas - Encuesta de Características Esenciales de la Población y las Viviendas

reg\_tenencia\_seg\_residencia: !section

title: Regimen de tenencia de la vivienda y segundas residencias en propiedad

table5: !table

title: Distribución de hogares según el régimen de tenencia y disponibilidad de segunda residencia en propiedad

reg\_tenencia\_ingresos: !section

title: Regimen de tenencia por ingresos mensuales netos del hogar en intervalos

table2: !table

title: Distribución de hogares según el régimen de tenencia por ingresos mensuales netos del hogar

reg\_tenencia\_ingresos\_%col: !section

title: Regimen de tenencia por ingresos mensuales netos del hogar en intervalos - % sobre el total de cada columna

table3: !table

title: Distribución de hogares según el régimen de tenencia por ingresos mensuales netos del hogar

```

4. \*\*main.py\*\*

```

from pathlib import Path

import docx

import pandas as pd

from docx.document import Document

from tesorotools.database.local import ShortcutDatabase

from tesorotools.render import Report

from tesorotools.utils.config import TemplateLoader, read\_config

from ecepov.data\_loader import (

load\_ecepov\_adultos,

load\_ecepov\_hogar,

load\_ecepov\_vivienda,

)

from ecepov.analysis.A1\_regimen\_tenencia import viviendas\_tipologia

from ecepov.analysis.A2\_regimen\_tenencia\_ingresos import regimen\_tenencia\_ingresos

from ecepov.analysis.A3\_segunda\_residencia import viviendas\_y\_segunda\_residencia

from ecepov.analysis.A4\_seg\_residencia\_ingresos import seg\_residencia\_ingresos

# --------------------------

# Define important paths used in the project

# --------------------------

HOME: Path = Path(

\_\_file\_\_

).parent.parent # Root directory of the project (two levels up from this file)

CONFIG: Path = HOME / "config" # Configuration files directory

DATA: Path = HOME / "data" # Data files directory

REPORT: Path = HOME / "report" # Reports directory

# --------------------------

# Function: generate\_report

# --------------------------

def generate\_report(year: int, out: Path):

"""

Generates and saves a report document for a specific year.

Args:

year (int): The year for which the report is generated (e.g., survey year).

out (Path): Output directory path where the report will be saved.

"""

# Load the base Word document template for the report

document: Document = docx.Document(REPORT / "tesoro.docx")

# Read the YAML configuration file for report template

template = read\_config(REPORT / "template.yaml", loader=TemplateLoader)

# Extract the Report object from the template

report: Report = template["report"]

# Modify the title dynamically by appending the survey year

report.contents[

"title"

].\_title += f" (ECEPOV {year})" # hacky way to customize the title

# Render the report content onto the loaded document

document = report.render(document)

# Save the rendered document to the output path with a year-specific filename

document.save(out / f"planes\_pensiones\_{year}.docx")

# --------------------------

# Main processing function

# --------------------------

def main():

"""

Main function to load data, run analysis, and generate the pension report.

"""

# --- Load ECEPOV database configuration ---

ecepov\_info = read\_config(CONFIG / "databases.yaml")["ecepov"]

ecepov\_db: ShortcutDatabase = ShortcutDatabase(

root\_path=DATA, shortcut=ecepov\_info["shortcut"]

)

ecepov\_year: int = ecepov\_info[

"last\_year"

] # The latest year available in the database

# --- Load survey and pension data for the specified year ---

ecepov\_adultos: pd.DataFrame = load\_ecepov\_adultos(ecepov\_db, ecepov\_year)

ecepov\_hogar: pd.DataFrame = load\_ecepov\_hogar(ecepov\_db, ecepov\_year)

ecepov\_vivienda: pd.DataFrame = load\_ecepov\_vivienda(ecepov\_db, ecepov\_year)

# Define output path for saving results

output\_path: Path = ecepov\_db.get\_products\_path(ecepov\_year)

# Define paths for report assets like tables and images

tables\_path: Path = REPORT / "tables"

images\_path: Path = REPORT / "images"

# --- ANALYSIS 1: Regimen de tenencia de viviendas ---

viviendas\_tipologia(ecepov\_db, ecepov\_year, ecepov\_vivienda, tables\_path)

# --- ANALYSIS 2: Regimen de tenencia por ingresos ---

regimen\_tenencia\_ingresos(ecepov\_db, ecepov\_year, ecepov\_vivienda, tables\_path)

# --- ANALYSIS 3: Regimen de tenencia de viviendas y segunda residencia ---

viviendas\_y\_segunda\_residencia(ecepov\_db, ecepov\_year, ecepov\_vivienda, tables\_path)

# --- ANALYSIS 4: Segunda residencia por ingresos ---

seg\_residencia\_ingresos(ecepov\_db, ecepov\_year, ecepov\_vivienda, tables\_path)

# --- Generate and save the final report ---

generate\_report(ecepov\_year, output\_path)

# --------------------------

# Entry point of the script

# --------------------------

if \_\_name\_\_ == "\_\_main\_\_":

main()

```

5. \*\*analisis\_1.py\*\*

```

from pathlib import Path

import pandas as pd

from tesorotools.database.local import ShortcutDatabase

from tesorotools.utils import format\_table

# Constants

TIPOLOGIA\_COL: str = "REGVI"

FACTOR\_COL: str = "FACTOR"

# Map of housing typologies

TIPOLOGIA\_ALIAS = {

1: "Propia por herencia o donación",

2: "Propia, por compra, totalmente pagada",

3: "Propia, por compra, con pagos pendientes (hipotecas)",

4: "Alquilada",

5: "Cedida gratis o a bajo precio (por otro hogar, pagada por la empresa…)",

6: "Otra forma",

}

def viviendas\_tipologia(

ecepov\_db: ShortcutDatabase,

ecepov\_year: int,

data: pd.DataFrame,

local\_path: Path,

) -> pd.DataFrame:

# ----- Raw counts (Observaciones) -----

raw\_counts: pd.Series = data.groupby(TIPOLOGIA\_COL).size()

raw\_counts.name = "Observaciones"

raw\_counts = raw\_counts.rename(TIPOLOGIA\_ALIAS).copy()

raw\_counts["Total"] = raw\_counts.sum()

# ----- Weighted counts (Viviendas) -----

weighted: pd.Series = data.groupby(TIPOLOGIA\_COL)[FACTOR\_COL].sum()

weighted.name = "Viviendas"

weighted = weighted.rename(TIPOLOGIA\_ALIAS).copy()

weighted\_total = weighted.sum()

weighted["Total"] = weighted\_total

weighted\_pct: pd.Series = weighted / weighted\_total \* 100

weighted\_pct.name = "Viviendas (%)"

# ----- Create Summary Table -----

summary: pd.DataFrame = pd.concat([raw\_counts, weighted, weighted\_pct], axis=1)

summary.index.name = "Regimen de Tenencia"

# ----- Save to Excel -----

output\_path: Path = ecepov\_db.get\_products\_path(year=ecepov\_year) / "viviendas"

output\_path.mkdir(parents=True, exist\_ok=True)

output\_file: Path = output\_path / "ecepov\_2021.xlsx"

with pd.ExcelWriter(output\_file, engine="openpyxl") as writer:

summary.to\_excel(writer, sheet\_name="reg tenencia vivienda")

print(f"Excel file saved to: {output\_file}")

# ----- Save formatted table to local for reports -----

table = format\_table(

summary,

format\_dict={

"Observaciones": {"decimals": 0, "units": ""},

"Viviendas": {"decimals": 0, "units": ""},

"Viviendas (%)": {"decimals": 1, "units": "%"},

},

)

table.to\_feather(local\_path / "table1.feather")

return summary

```

7. \*\*data\_loader.py\*\*

```

import pandas as pd

from tesorotools.database.local import ShortcutDatabase

def load\_ecepov\_adultos(ecepov\_db: ShortcutDatabase, ecepov\_year: int):

return pd.read\_feather(

ecepov\_db.get\_processed\_path(ecepov\_year)

/ f"ECEPOVadultos\_{ecepov\_year}.feather"

)

def load\_ecepov\_hogar(ecepov\_db: ShortcutDatabase, ecepov\_year: int):

return pd.read\_feather(

ecepov\_db.get\_processed\_path(ecepov\_year) / f"ECEPOVhogar\_{ecepov\_year}.feather"

)

def load\_ecepov\_vivienda(ecepov\_db: ShortcutDatabase, ecepov\_year: int):

return pd.read\_feather(

ecepov\_db.get\_processed\_path(ecepov\_year)

/ f"ECEPOVvivienda\_{ecepov\_year}.feather"

)

```

## Project Structure in the shared folder

The structure inside the shared folder's folder for the project needs to follow the below structure (the files can be different but the folders remain).

Let´s suppose again we are working with a survey from INE called "ECEPOV", and that the latest one published is from 2021.

ECEPOV/

└── 2021/

├── doc/ `# manuales para utilización correcta de los datos`

│ ├── cuestionario\_ECEPOV.pdf

│ ├── dr\_ECEPOVadultos\_2021.xlsx

│ ├── dr\_ECEPOVhogar\_2021.xlsx

│ └── dr\_ECEPOVvivienda\_2021.xlsx

├── processed/ `# carpeta con los archivos de datos originales en .csv extraídos a .feather`

│ ├── ECEPOVadultos\_2021.feather

│ ├── ECEPOVhogar\_2021.feather

│ └── ECEPOVvivienda\_2021.feather

├── products/ `# aquí se guardan los "productos" que produce el análisis en python, tanto el report en word como el excel con los resultados`

│ ├── viviendas/

│ │ └── ecepov\_2021.xlsx

│ └── planes\_pensiones\_2021.docx

├── raw/ `# carpeta con los datos sin tratar`

│ ├── ECEPOVadultos

│ ├── ECEPOVhogar

│ ├── ECEPOVvivienda

│ └── LeemeECEPOV\_2021.txt