# ¿Qué es Cookiecutter?

Es un manejador de plantillas multiplataforma (Windows, Mac OS, Linux) que te permite hacer plantillas en lenguaje de programación o formato de marcado. Puede ser usado como herramienta de línea de comandos o como librería de Python.

Cookiecutter funciona con Jinja, un motor de plantillas extensible con el cual puedes crear plantillas como si estuvieras escribiendo código en Python.

## ¿Cómo funciona?

Hay 3 pasos para entender la manera en que funciona:

* Detectará una sintaxis especial en los documentos y carpetas de tu proyecto.
* Buscará variables y valores a reemplazar.
* Finalmente, entregará un proyecto con la estructura definida en la plantilla.

### Sintaxis de Jinja

Existen 3 tipos diferentes de bloques:

* Bloques de expresión: se usan para incluir variables en la plantilla:

{{ cookiecutter.saluda }}

* Bloques de declaración: se usan para el uso de condicionales, ciclos, etc.:

{% if coockiecutter.eres\_asombroso %}

. . .

{% endif %}

* Bloques de comentario: se usan para dejar comentarios o recomendaciones a los propios desarrolladores:

{# Esto es un comentario #}

## Instalación de Cookiecutter

Importante: Todos los pasos son vía consola y con Anaconda instalado.

1. Crea una carpeta un entrar en ella:

mkdir <nombre\_carpeta>

cd <nombre\_carpeta>

2. Agrega el canal Conda-Forge a tu configuración global:

conda config --add channels conda-forge

3. Crea un ambiente virtual que contenga a Coockiecutter:

conda create --name <nombre\_ambiente> cookiecutter=1.7.3

4. Activa el ambiente virtual: conda activate <nombre\_ambiente>

5. Definir en dónde estará tu ambiente:

conda env export --from-history --file environment.yml

Para desactivar el ambiente virtual: conda deactivate

# Como utilizar cookiecutter

Ejemplo de uso con una plantilla de Platzi

1. Ir al navegar al siguiente url

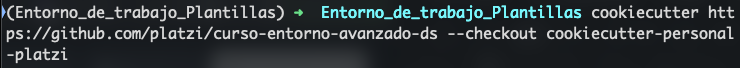
<https://github.com/platzi/curso-entorno-avanzado-ds/tree/cookiecutter-personal-platzi>

1. Copiar el link

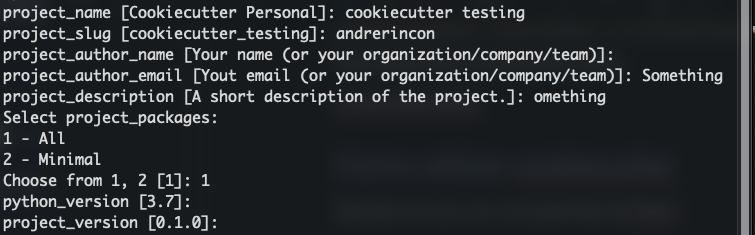
cookiecutter https://github.com/platzi/curso-entorno-avanzado-ds --checkout cookiecutter-personal-platzi



1. Pegar el link en la terminal



1. Llenar los campos y la configuración de instalación



1. Para abrir la estructura que se creo lo hacemos con

code cookiecutter\_testing

# Creacion de plantillas personalizadas Cookie Cutter

## Estructura inicial de la plantilla

Dentro de la carpeta principal crea la carpeta que contendrá todo lo que necesitarás en tu proyecto con el nombre:

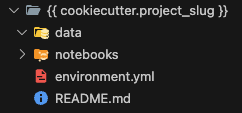
{{ coockiecutter.project.slug }}

En la carpeta recién creada agrega los siguientes archivos:

README.md

environment.yml

cookiecutter.json



También crea las carpetas que necesitará tu proyecto:

/data

/notebooks

Afuera de la carpeta, pero dentro de la carpeta principal, crea el siguiente archivo:

environment.yml

Hay dos archivos environment.yml, el de configuración de entorno (dentro de la carpeta que creaste) y el que configura las dependencias y paquetes (en la carpeta principal).

### Información de README.md

Adentro del archivo README.md agrega las siguientes líneas que lo harán un archivo dinámico:

# {{ cookiecutter.project\_name }}

By: {{ cookiecutter.project\_author\_name }}

{{ cookiecutter.project\_description }}

## License

Estas líneas, hechas en Jinja, permitirán a tu archivo acceder a las variables que contienen la información del título, autor y descripción del proyecto.

### Información de environment.yml (entorno)

# conda env create --file environment.yml

name: cookiecutter-personal-platzi

channels:

- anaconda

- conda-forge

- defaults

dependencies:

- cookiecutter=1.7.3

prefix: /opt/anaconda3/envs/Entorno\_de\_trabajo\_Plantillas

### Información de environment.yml (configuración)

# conda env create --file environment.yml

name: {{ cookiecutter.project\_slug }}

channels:

- anaconda

- conda-forge

- defaults

dependencies:

{% if cookiecutter.project\_packages == "All" -%}

- fs

- jupyter

- jupyterlab

- pathlib

{% endif -%}

- pip

{% if cookiecutter.project\_packages == "All" -%}

- pyprojroot

{% endif -%}

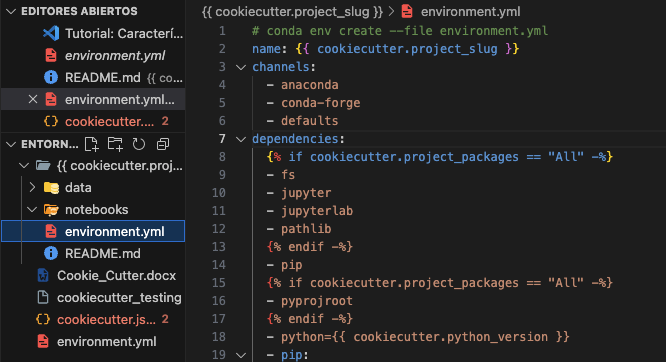
- python={{ cookiecutter.python\_version }}

- pip:

{% if cookiecutter.project\_packages == "All" -%}

- pyhere

{% endif -%}



### Agregando información a coockiecutter.json

Dentro de este archivo configurarás todos los valores de las variables que utilizas en los demás archivos:

{

"project\_name": "Cookiecutter Personal",

"project\_slug": "{{ cookiecutter.project\_name.lower().replace(' ', '\_').replace('-', '\_') }}",

"project\_author\_name": "Your name (or your organization/company/team)",

"project\_author\_email": "Yout email (or your organization/company/team)",

"project\_description": "A short description of the project.",

"project\_packages": ["All", "Minimal"],

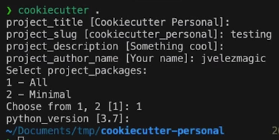
"python\_version": "3.7",

"project\_version": "0.1.0"

}

## Ejecuta el proyecto

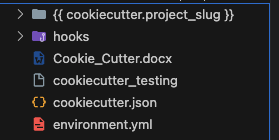
1. Abre la terminal integrada (Command + shift + p)
2. Asegurate que el ambiente este activado de la carpeta donde esta el proyecto de cookie cutter
3. Ejecuta el comando (cookiecutter .), en este caso como se quiere trabajar en esta carpeta ponemos el punto pero si quisiéramos trabajar en otra pues escribimos el nombre de la carpeta.
4. Llenamos el contenido primordialmente slug



# Hooks

Son sentencias que se van a ejecutar antes o después de generar la plantilla de datos. Puedes usarlos para verificar el nombre de una carpeta, actualizar git, etc.

### Implementación de Hooks

1. Se crea la carpeta “hooks”, adentro de la carpeta principal de tu proyecto, hay que tener encuenta que no van dentro de la carpeta de Project slug. 
2. Dentro de la carpeta se agregan los archivos “pre\_gen\_project.py” (lo que se ejecuta antes de generar la plantilla) y “pos\_gen\_project.py” (lo que se ejecuta después de generar la plantilla).

Por ejemplo, en “pre\_gen\_project.py” se puede inicializar git o validar nombres y archivos para evitar errores.

En el archivo “pos\_gen\_project.py” se puede hacer el primer commit en git o mostrar la finalización de la instalación de dependencias.

1. Escribir el hook pre\\_gen\\_project.py

Este hook se ejecuta \*\*antes\*\* de que Cookiecutter genere el proyecto. Puedes usarlo para validar entradas o preparar el entorno.

Por ejemplo, si deseas asegurarte de que el nombre del proyecto no contiene una X al comenzar, puedes escribir un script

import os

import sys

project\_slug = "{{ cookiecutter.project\_slug }}"

ERROR\_COLOR = "\x1b[31m"

MESSAGE\_COLOR = "\x1b[34m"

RESET\_ALL = "\x1b[0m"

if project\_slug.startswith("x"):

print(f"{ERROR\_COLOR}ERROR: {project\_slug=} is not a valid name for this template.{RESET\_ALL}")

sys.exit(1)

print(f"{MESSAGE\_COLOR}Let's do it! You're are going to create something awesome!")

print(f"Creating project at { os.getcwd() }{RESET\_ALL}")

1. Escribir el hook post\\_gen\\_project.py

Este hook se ejecuta \*\*después\*\* de que se ha generado el proyecto. Puedes usarlo para automatizar tareas adicionales, como inicializar un repositorio de Git, instalar dependencias o imprimir mensajes personalizados.

Por ejemplo, si quieres inicializar un repositorio de Git después de generar el proyecto, puedes crear un script post\\_gen\\_project.py:

import subprocess

MESSAGE\_COLOR = "\x1b[34m"

RESET\_ALL = "\x1b[0m"

print(f"{MESSAGE\_COLOR}Almost done!")

print(f"Initializing a git repository...{RESET\_ALL}")

subprocess.call(['git', 'init'])

subprocess.call(['git', 'add', '\*'])

subprocess.call(['git', 'commit', '-m', 'Initial commit'])

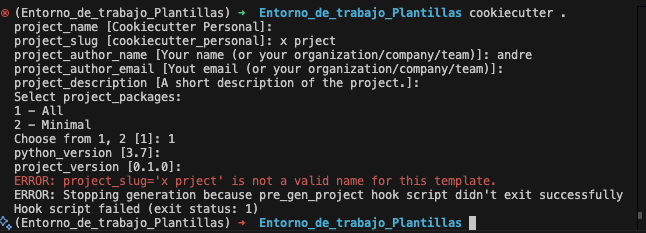
print(f"{MESSAGE\_COLOR}The beginning of your destiny is defined now! Create and have fun!{RESET\_ALL}")

1. Paso 4: Probar los hooks

Una vez que has creado los hooks, puedes ejecutar \*\*Cookiecutter\*\* para probarlos: Primero active el entorno de trabajo asi: conda activate Entorno\_de\_trabajo\_Plantillas

cookiecutter path/to/project-template

Si todo está bien, el pre-hook se ejecutará antes de que se genere el proyecto y el post-hook después de que se genere el proyecto.



## Distribución de tu plantilla (GitHub)

Crea un nuevo repositorio vacío en GitHub. Puedes no agregar ninguna información sobre tu repositorio por el momento.

Dentro de la carpeta que contiene tu proyecto inicializa git y haz tu primer commit:

git init .

git add .

git commit -m "Initial commit"

Ahora, para subir a GitHub tu proyecto, ejecuta lo siguiente:

git remote add origin https://github.com/<usuario>/<repositorio>.git

git branch -M main

git push -u origin main

No olvides eliminar cualquier carpeta innecesaria (por ejemplo, la carpeta que se genera cuando ejecutas Coockiecutter).

Para evitar que git ignore las carpetas vacías, agrega un archivo “.gitkeep” dentro de cada carpeta vacía.

Cuando usas “cookiecutter” como parte del nombre de tu proyecto, este puede ser encontrado por otras personas en GitHub. De esta forma podrás ayudar a otros científicos de datos, facilitándoles el trabajo.