|  |
| --- |
| Groupe CESI |
| Create a python package for pip |
| And make it usable as command line with click |



Contenu

[Aim of the document 2](#_Toc118908914)

[1 Setup python package on Gitlab local 3](#_Toc118908915)

[1.1 Architecture 3](#_Toc118908916)

[1.2 Build package 3](#_Toc118908917)

[1.3 Push package in a GitLab registry 4](#_Toc118908918)

[1.4 Install package with pip 4](#_Toc118908919)

[2 Annex 5](#_Toc118908920)

[2.1 Develop python CLI for PIP 5](#_Toc118908921)

[2.1.1 Set the setup.py script with setuptools 5](#_Toc118908922)

[2.1.2 Test PIP package 6](#_Toc118908923)

[2.1.3 Deploy on public repository Pypi 6](#_Toc118908924)

[2.1.4 Install from a private repository 6](#_Toc118908925)

[2.1.5 Various test to Deploy on private 6](#_Toc118908926)

[2.1.6 Known error 7](#_Toc118908927)

# Aim of the document

This document describes how to create a python package installable through pip.

**Public**: (Developer)

# Setup python package on Gitlab local

In order to create a pip package in python, you need to follow a directory hierarchy.

A template is available in the LINEACT git:

<http://gitrecherche.cesi.fr/templates/python-package-lineact>

## Architecture

Une image contenant texte

Description générée automatiquement

Files:

* *Pyproject.toml*: allows to define package configuration and dependency
* .pypirc: Allows to define the registry where upload the package (this file must be move to user directory)
* README.md: a markdown that must contains the installation information of the package. It also describes main usage of it.

Directories:

* **src**: content package scripts
* **test**: content testing scripts of package

More information about architecture at:

[https://setuptools.pypa.io/en/latest/userguide/quickstart.html#install-build](https://setuptools.pypa.io/en/latest/userguide/quickstart.html" \l "install-build)

## Build package

First install this module with pip:

pip install –upgrade build setuptools twine

add this in the file pyproject.toml :

[build-system]

requires = ["setuptools"]

build-backend = "setuptools.build\_meta"

[project]

name = "lineact-template-package"

version = "0.0.1"

dependencies = [

“your-dependencies1”,

“your-dependencies2”,

]

Une image contenant texte

Description générée automatiquement

In directory src, create a directory with the name of your package, then create file \_\_init\_\_.py and define the imports. For example:

from lineact.helloworld import \*

helloworld.py contains a method:

def add(a, b):

return a + b

Now that all is setup, we can build our package with the command :

**Caution**: delete existing package in the **dist/** directory before running this command

python -m build

this command creates a new directory in the root. This directory is *dist* and contains two packages.

Une image contenant texte

Description générée automatiquement

Package are created, now we can push in a registry.

## Push package in a GitLab registry

For pushing our package in a registry, we need to configure ***.pypirc*** file in the user directory (C:/Users/<username>/ or **%userprofile%**). This file contains the url of the registry and the username and token.

[distutils]

index-servers =

gitlab

[gitlab]

repository = http://gitrecherche.cesi.fr/api/v4/projects/<project\_id>/packages/pypi/

username = <token\_name>

password = <token>

Link for create a project access token:

<https://docs.gitlab.com/ee/user/project/settings/project_access_tokens.html>

We can use an existing token :

Token name: package\_token

Token: glpat-i7Kz\_CsYxuNYkSxxZWqb

Therefore, for the LINEACT laboratory use this **.pypirc**

[distutils]

index-servers =

gitlab

[gitlab]

repository = http://gitrecherche.cesi.fr/api/v4/projects/122/packages/pypi/

username = package\_token

password = glpat-i7Kz\_CsYxuNYkSxxZWqb

When the file is configure, we can execute the command in the root of package:

**Caution**: delete existing package in the **dist/** directory before running this command

python -m twine upload --repository gitlab dist/\*

## Setup command line with module click

For execute a script python with a command line windows. We used the module ***Click*** (<https://click.palletsprojects.com/en/8.1.x/>). First we import the module in the ***pyproject.toml*** (below written in red) and the complete path to access at the methods who wants to run in commands line (below written in yellow)***:***

[build-system]

requires = ["setuptools", "setuptools-scm"]

build-backend = "setuptools.build\_meta"

[project]

name = "lineact-package"

version = "0.0.1"

authors = [

{ name="Alexandre", email="acourallet@cesi.fr" },

{ name="Louis", email="lchochoy@cesi.fr" },

{ name="Vincent", email="vhavard@cesi.fr" },

]

description = "Example Lineact Package"

readme = "README.md"

requires-python = ">=3.7"

classifiers = [

"Programming Language :: Python :: 3",

"License :: OSI Approved :: MIT License",

"Operating System :: OS Independent",

]

dependencies= [

*"Click==8.1.3",*

]

[project.scripts]

lineact = "lineact.scripts.lineact\_cli:cli"

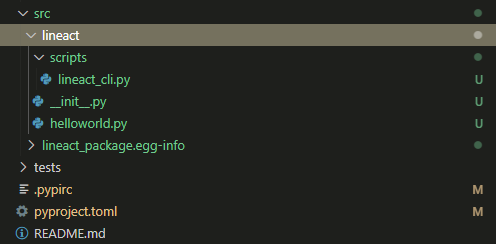
<cmd> = <package>.<module>.<class>:<methods>

[tool.setuptools.packages.find]

where = ["src"]

before the equal is the command to run the scripts and after the complete path of the modules and after the colon the methods to run. This is an example, there are different cases, depending on the architecture.

next, we create a directory, called ***scripts*** which will contain our scripts of command line.



For our example, we have the script lineact\_cli.py which contains the code below.

import click

import lineact.helloworld as helloworld

@click.group()

*def* cli():

    pass

@click.command()

@click.option('--count', *default*=1, *help*='Your age', *required*=True)

@click.option('--name', *help*='Your name', *required*=True)

*def* hello(*name*, *count*):

    print(*f*'Hello {name}! You are {count} years old')

@click.command()

@click.option('--a', *default*=1, *help*='chose a number', *required*=True)

@click.option('--b', *default*=1, *help*='chose an other number', *required*=True)

*def* add(*a*, *b*):

    print(helloworld.add(a, b))

cli.add\_command(hello)

cli.add\_command(add)

if \_\_name\_\_ == '\_\_main\_\_':

    cli()

### Résultat

Here is the result, once the package is built.

PS Python-Package-Template-Lineact> lineact

Usage: lineact [OPTIONS] COMMAND [ARGS]...

Options:

--help Show this message and exit.

Commands:

add

hello

PS Python-Package-Template-Lineact> lineact add --help

Usage: lineact add [OPTIONS]

Options:

--a INTEGER chose a number [required]

--b INTEGER chose an other number [required]

--help Show this message and exit.

PS Python-Package-Template-Lineact> lineact add --a 2 --b 3

5

Now part two will explain how to build and use your package

# Install an existing package from LINEACT registry

## Prerequisite: create access token

## Une image contenant texte, capture d’écran, moniteur, écran Description générée automatiquement

## Setup access token on your machine

For install package from a local registry, we need to configure the registry to use. the informations of the url, username and password are the same that use before.

pip config --user set global.extra-index-url http://<access\_token\_name>:<access\_token\_key> @gitrecherche.cesi.fr/api/v4/projects/<project\_id>/packages/pypi/simple

replace <project\_id> with the value of the project where package are stored.

In LINEACT, the project is Unik <http://gitrecherche.cesi.fr/VirtualReality/unik> and the **value=122.**

Running this command will allow you to access to the **whole python package developed at LINEACT**

pip config --user set global.extra-index-url http://package\_token:glpat-i7Kz\_CsYxuNYkSxxZWqb@gitrecherche.cesi.fr/api/v4/projects/122/packages/pypi/simple

As a result, you must see this result in the **%userprofile%\AppData\Roaming\pip\pip.ini**

[global]

extra-index-url = http://package\_token:glpat-i7Kz\_CsYxuNYkSxxZWqb@gitrecherche.cesi.fr/api/v4/projects/122/packages/pypi/simple

trusted-host = gitrecherche.cesi.fr

Moreover, in order to define **gitrecherche.cesi.fr** as a trusted host, please execute this command:

pip config --user set global.trusted-host gitrecherche.cesi.fr

## Install package with pip

Now we execute the command to install the package:

pip install lineact-package

## Example of installation of lineactua

If you want to install lineactua package in a conda environment.

First, create your access token as described in 1.4. Then execute those command:

conda create -n lineactua\_p3.9 python=3.9.13

conda activate lineactua\_p3.9

pip install lineactua

// TO DO (Unit test / Entrypoints : commands to execute scripts )

# Annex

## Develop python CLI for PIP

This part is put as the initial test done with those tools in order to create a pip package.

### Set the setup.py script with setuptools

[https://python-packaging.readthedocs.io/en/latest/command-line-scripts.html#the-console-scripts-entry-point](https://python-packaging.readthedocs.io/en/latest/command-line-scripts.html" \l "the-console-scripts-entry-point)

For a project organised like this:

D:\CESI\DEV\INHARD-TREATMENT

| .gitignore

| LICENSE

| Makefile

| MANIFEST.in

| MarkdownTemplate.md

| README - HOW TO USE TEMPLATE.md

| README.md

| **setup.py**

| tree.txt

| Version.md

|

+---inhardtreatment

| | appConsole.py

| | helpers.py

| | inhard.py

| | **inhardtreatment\_cli.py**

| | vhd-usage.txt

| |

| +---pymo

+---tests

Here is an example of setup tools usable

# -\*- coding: utf-8 -\*-

from setuptools import setup, find\_packages

with open('README.md') as f:

readme = f.read()

with open('LICENSE') as f:

license = f.read()

requirements = [

"nose",

"sphinx",

"opencv-python >= 4.2.0.34",

"click",

"funcy",

"pandas >= 1.0.5"

]

setup(

name='inhardtreatment',

version='0.1.0',

install\_requires = requirements,

description='Manage inhard data treatment',

long\_description=readme,

author='Vincent Havard',

author\_email='vhavard@cesi.fr',

url='TO DO',

license=license,

packages=find\_packages(exclude=('tests', 'docs')),

entry\_points='''

[console\_scripts]

inhardt=inhardtreatment.inhardtreatment\_cli:cli

inhardtreatment=inhardtreatment.inhardtreatment\_cli:cli

''',

)

**entry\_points** set the name of the command (before = sign) and the python module and function to execute when the command is called (after = sign). You can use several command name if you want.

In that example:

* Command name is **inhardt** and **inhardtreatment**
* Code executed is located at

### Test PIP package

[https://click.palletsprojects.com/en/7.x/setuptools/#testing-the-script](https://click.palletsprojects.com/en/7.x/setuptools/" \l "testing-the-script)

[https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/#activating-a-virtual-environment](https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/" \l "activating-a-virtual-environment)

First, install **virtualenv** in your anaconda terminal console

pip install virtualenv

Then, create a virtual environment for testing deployment

virtualenv venv1

#LINUX activation of environment

#%. venv1/bin/activate

# WINDOWS activation of environment

.\venv1\Scripts\activate

# go into the virtual environment for putting data in it when installing

cd venv1

# WINDOWS deactivation of environment

.\venv1\Scripts\deactivate

and install it

#

pip install --editable path/to/package

path/to/package is the folder containing the **setup.py** file

pip install –-editable "D:\CESI\DEV\inhard-treatment"

Validate that all the command are working as expected. The best way is to use the tests contains in the test folder.

### Deploy on public repository Pypi

<https://medium.com/@joel.barmettler/how-to-upload-your-python-package-to-pypi-65edc5fe9c56>

A LIRE

<https://packaging.python.org/tutorials/packaging-projects/>

A TESTER PAR VINCENT

### Install from a private repository

Use this command on windows command line

pip install –e git+http://gitrecherche.cesi.fr/HumanActivityRecognition/inhard-treatment.git#egg=inhardtreatment

A voir pour le support ssh git pour l’installation depuis un repository privé

<https://adamtheautomator.com/ssh-with-powershell/#ssh-client-windows-10-setup>

<http://www.kevinsubileau.fr/informatique/astuces-tutoriels/windows-10-client-serveur-ssh-natif.html>

### Various test to Deploy on private

<https://medium.com/@arocketman/creating-a-pip-package-on-a-private-repository-using-setuptools-fff608471e39>

<https://stackoverflow.com/questions/4830856/is-it-possible-to-use-pip-to-install-a-package-from-a-private-github-repository>

<https://packaging.python.org/tutorials/installing-packages/>

I have installed SourceTree and I have opened the cmd and went on:

cd C:\Users\vhavard\AppData\Local\Atlassian\SourceTree\git\_local\usr\bin

Start ssh agent

ssh-agent -s

Activate ssh-agent service and set it to manual in order to make it launch when needed.

Get-Service -Name ssh-agent | Set-Service -StartupType Manual

Clone with username in http. It works

git clone http://vhavard@gitrecherche.cesi.fr/HumanActivityRecognition/inhard-treatment.git ./TEMPTEST

%USERPROFILE% = c:/users/username

Clone with SSH

git config --global user.name "your\_username"

git config --global user.email your\_email\_address@example.com

git config --global –list

git clone git@gitrecherche.cesi.fr:HumanActivityRecognition:vhavard/inhard-treatment.git ./TEMPTEST

pip install git+ssh://git@repo-address.git#egg=name

DOES NOT WORK, you need to replace the :HumanActivityRecognition to **/HumanActivityRecognition**

pip install git+ssh://git@gitrecherche.cesi.fr:HumanActivityRecognition/inhard-treatment.git

pip install git+ssh://git@gitrecherche.cesi.fr/HumanActivityRecognition/inhard-treatment.git

pip install -e git+ssh://vhavard@gitrecherche.cesi.fr/HumanActivityRecognition/inhard-treatment.git#egg=inhardtreatment

pip install git+http://gitrecherche.cesi.fr/HumanActivityRecognition/inhard-treatment.git

### Known error

#### Setuptools: ModuleNotFoundError: No module named <modulename> after pip install

<https://stackoverflow.com/questions/59305504/modulenotfounderror-no-module-named-modulename-after-pip-install>

**Version du document**

**V1.0 - jj/mm/aaaa**

Détail

**V6.0 – 26/06/2019**

Ajout d’équation numérotée

**V8.0 – 05/11/2019**

Style « Code » et « Console » en police « Courrier New »