

# Installation guide for the FLAP package

7 April, 2020

S. Zoletnik ([zoletnik.sandor@energia.mta.hu](mailto:zoletnik.sandor@energia.mta.hu))

## Introduction

FLAP is a modular Python program package for the analysis of large multidimensional measurement or modelling datasets, mostly designed for magnetic fusion research. This guide describes the installation process of FLAP for interested users. We assume minimal knowledge in Python, but it is advised to read a general introduction about concepts.

## GIT


FLAP is maintained on GitHub, which is a community program archive and development platform. Programs are handled there using GIT a distributed version control system (<https://en.wikipedia.org/wiki/Git>). Git can be obtained for various operating systems from e.g. <https://git-scm.com/download>.

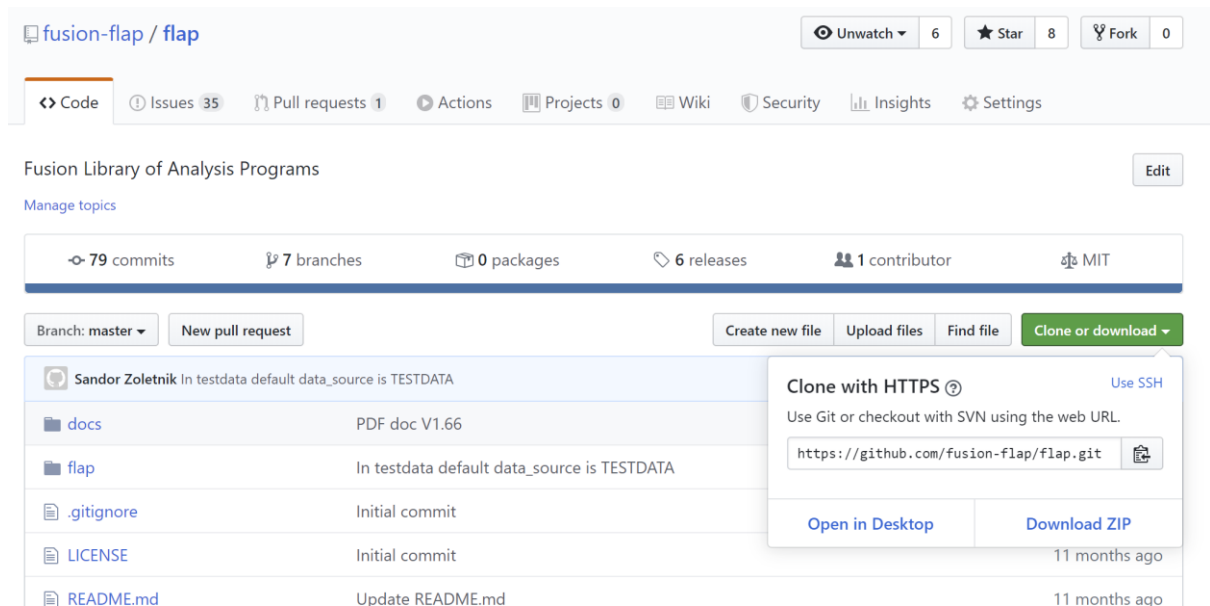
Data in git are stored in “repositories”. From a server a repository can be *cloned* to another machine. This creates a local repository. Each repository has different branches where different versions are stored. The top branch is called master. There might be different releases of the programs which are stored in branches. Also branches are used for program development. The normal procedure of changing the program is to create a new branch where the actual status is copied. There the modifications are done, first they need to be committed into the actual branch, tested and finally merged back to the original branch. A branch in the local repository can be pushed to the server which means the actual status is present both in the local and remote repository. Also branches can be pulled from the remote repository to copy changes present there. There are much more complex procedures in git but this is the basic knowledge necessary to start using git.

To use FLAP first install git. On windows it has a Graphical interface but it has only limited functionality, not recommended.

## Obtaining FLAP

FLAP is a modular program. The core operations and structures are in the flap module, while device independent data access is provided in data access modules. All of these modules are in the FLAP organisation on GitHub at <https://github.com/fusion-flap> in separate repositories. . On your computer select a FLAP base directory where the flap modules will be stored. This should be separate from your working directories as it is not intended to modify the package contents unless you are a FLAP developer.

Each module has its own repository. In each of them the latest stable version is in branch master, the actual development version is in branch “development”. It is assumed that the development branch has more features than master but testing was not fully done. To obtain the flap core module master branch use a browser to visit the GitHub page <https://github.com/fusion-flap> and click the flap repository. On the Code page in the Branch button select master (this is the default). On the right hand side push the “Clone or download” button. A little window opens up below the button as shown below. Press the  button, this copies the command left to it to the paste buffer



On your computer navigate to your FLAP base directory and on Windows long click the right mouse button to open a menu on it. Select “Git Bash here” which opens a terminal. On Linux open a terminal in this directory. Check that in the prompt you see the FLAP base directory. Type “clone” and paste the contents of the paste buffer in the terminal. At the end of the command put the desired directory name where the flap module should be cloned. pressing enter clones the flap core repository to your machine as shown below.

```

MINGW64:/c/Users/Zoletnik/Python
zoletnik@DESKTOP-708118J MINGW64 ~/Python
$ git clone https://github.com/fusion-flap/flap.git flap_fusion
Cloning into 'flap_fusion'...
remote: Enumerating objects: 177, done.
remote: Counting objects: 100% (177/177), done.
remote: Compressing objects: 100% (132/132), done.
remote: Total 1193 (delta 88), reused 95 (delta 45), pack-reused 1016
Receiving objects: 100% (1193/1193), 15.60 MiB | 17.23 MiB/s, done.
Resolving deltas: 100% (769/769), done.
zoletnik@DESKTOP-708118J MINGW64 ~/Python
$ |

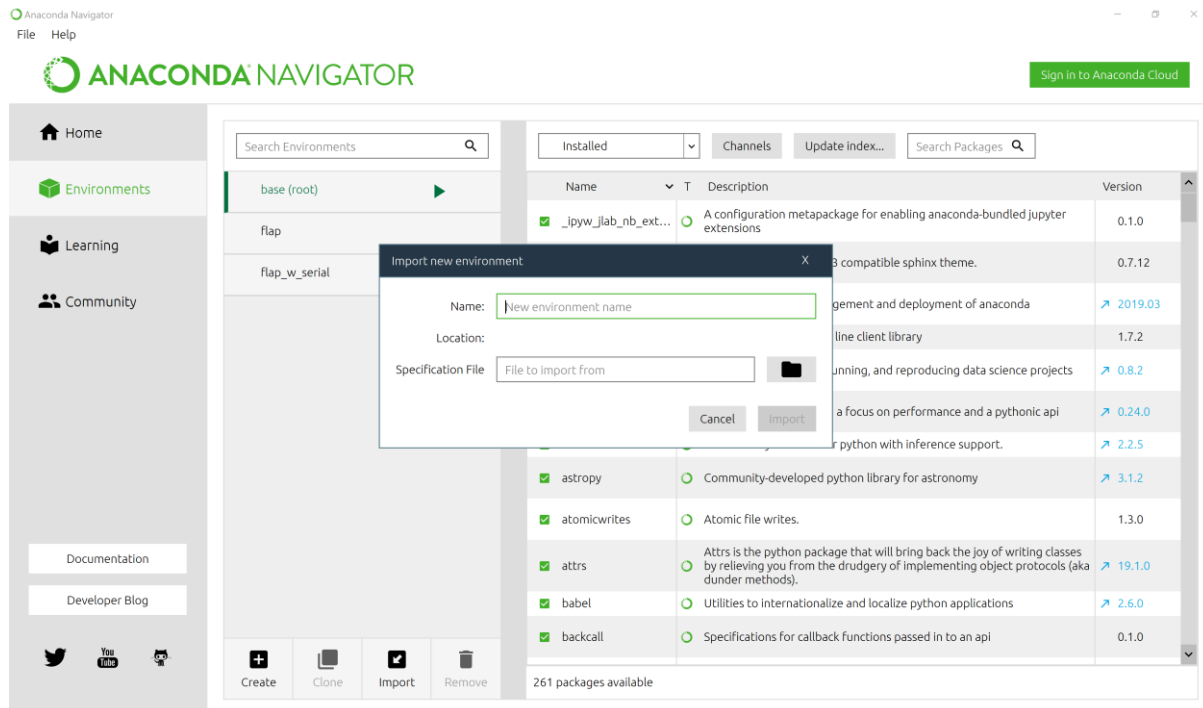
```

To start testing and learning FLAP no other modules are needed, a test data source module is included in the core module. If you intend to use some data source, select the repository in the GitHub FLAP organisation and follow the same procedure as for the core flap module.

## Obtaining and setting up Python

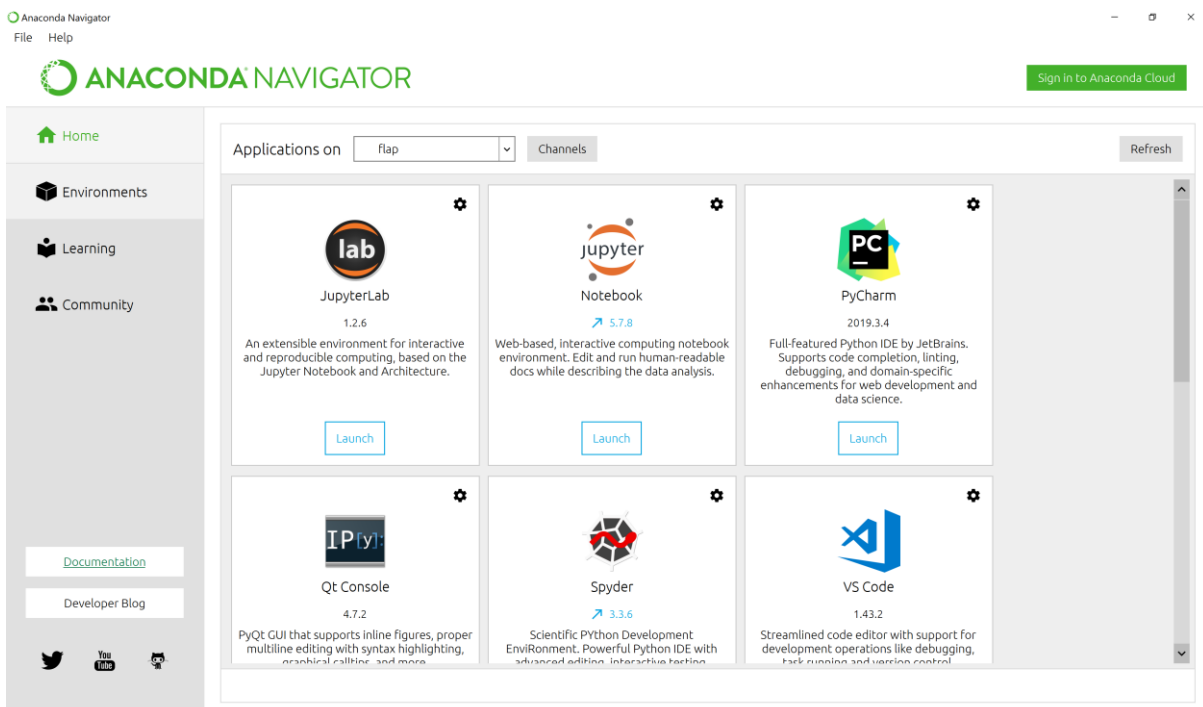
The best way to obtain Python is to use the Anaconda environment management program. Download Anaconda from <https://www.anaconda.com/distribution/>. On Linux systems Anaconda is installed for each user separately into the home directory, therefore no

root access is needed. After installation run Anaconda navigator. In its window select the “Environments” page and there at the bottom “Import”. In the screen below enter flap into the Name field and select the conda\_setup.yml file in the flap\_fusion/docs directory under your FLAP base directory.



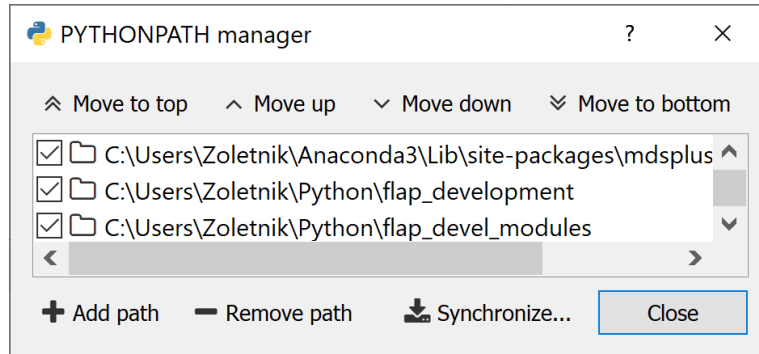
Pressing “Import” will load from the network all modules and Python components needed for FLAP.

When this is done select the Home page in Anaconda and on the top select the “flap” environment. This loads the packages and program versions for FLAP.



Select one of the programming environments: Spyder, VS Code or Pycharm. For beginners I recommend Spyder. After selecting the flap environment on the home page of Anaconda navigator the Install button should be pressed in the Spyder block. Then it should be launched.

In Spyder the Python path should be set. There is a blue-yellow Python logo button on the top of Spyder which opens the Pythonpath manager window as shown below.



Add the flap\_fusion directory to the list. The synchronize button need not be pressed. If you are intending to use flap input modules the directory where they are installed should also be added. After closing the Pythonpath manager window Spyder should be closed and restarted to take the new path effect. Usually Spyder will not restart unless Anaconda navigator is restarted first.

## Running tests

Once Spyder is up and running set the graphics output. As default it is “Inline” which means the graphics is intermixed with commands in the IPython terminal and not interactive. To do so go to Tools->Preferences->IPython console->Graphics and set to Automatic or some of the Qt versions. (Automatic is usually OK.)

Open the flap\_tests program under the test subdirectory in your FLAP directory:

```

C:/Users/Zoletnik/Root/Experiments/W7-X/A-beam/Measurement - Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:/Users/Zoletnik/Root/Experiments/W7-X/A-beam/Measurement
Editor - C:/Users/Zoletnik/Python/flap_fusion/flap/tests/flap_tests.py
flap_tests.py
1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 6 22:11:28 2018
4
5@author: Sandor Zoletnik (zoletnik.sandor@wigner.mta.hu)
6
7Example/test programs for FLAP.
8
9"""
10
11import matplotlib.pyplot as plt
12from matplotlib.gridspec import GridSpec
13import os
14import numpy as np
15from scipy import signal

```

The program can be started with the green right arrow on the top. The test program runs through several steps. The first part prints messages to the IPython console. After each step

the Enter should be pressed. Later graphics windows are opened, there might be multiple windows in one test, which appear on top of each other normally. To continue a key should be pressed when the mouse is in any of the graphics windows. After about 20 test procedures the test program stops. If no error was found the installation of FLAP is ready.

One could look into the test program. The individual tests are contained in functions. At the bottom of the program these are called. Towards the bottom of the program the `test_all` variable is set to `True` as shown below. Setting this to `False` inhibits running all tests and individual tests can be run by changing one or more `False` in the chain of if commands.

```
692# Reading configuration file in the test directory
693thisdir = os.path.dirname(os.path.realpath(__file__))
694fn = os.path.join(thisdir,"flap_tests.cfg")
695flap.config.read(file_name=fn)
696
697test_all = True
698
699# Running tests
700plt.close('all')
701
702if (False or test_all):
703    test_storage()
704    input("Press Enter to continue...")
705if (False or test_all):
706    test_saveload()
707    input("Press Enter to continue...")
708if (False or test_all):
709    test_coordinates()
710    input("Press Enter to continue...")
711if (False or test_all):
712    test_arithmetic()
```