# Help file for pyDCCB

This file contains info and settings for the different aspectos of pyDCCB.
It is assumed that we have intalled Aaconda in the case of development. Otherwise the programs are self contained as an .exe file and then can be runned as a stand-alone.

Each app has a \requirements\prod.txt containing all packages required. It is advisable to produce and initiate a new virtual enviroment and then run:
`cd` into the app directory and install its dependencies in a virtual environment in the following way:

```
conda create myEnviroment python== @python-version
conda activate myEnviroment
pip install -r requirements/dev.txt
```

then run the app:

```
python app.py
```

This will start the app, to exit: `CTRL` + `PAUSE`
Once runnig visit http://127.0.0.1:8050/ in your web browser.

# pyDCCB_abrir_mediciones_Dash_app

Dash program to open and parse the output files of the MI6010D DCCB. This program will take an Excel protocol file and fill it with the measurments outputs.

Assumptions:
RELLENAR

Each app has a requirements.txt, install the dependecies in a virtual environment.

## Dash app project structure

### Data

- All data (csv, json, txt, etc) should be in a data folder
- `/apps/{DASH_APP_NAME}/data/`

### Assets

- All stylesheets and javascript should be in an assets folder
- `/apps/{DASH_APP_NAME}/assets/`

### Requirements

- `dev.txt` calls `prod.txt` to install dependencies.
- `prod.txt` production file, contains all dependencies of the project

### app folder.

- `python-version` python version used to create the app.

## Project boilerplate

```
apps
├── ...
├── {DASH_APP_NAME}         # app project level
│   ├── assets/             # all stylesheets and javascript files
│   ├── data/               # all data (csv, json, txt, etc)
│   ├── requirements/       # contains all dependencies, recommended to be installed in a new virt
│   ├── app.py              # dash application entry point
│   ├── python-version      # project python version used during development.
│   └── ...
└── ...
```

## Handle relative path

Assets should never use a relative path, as this will fail when deployed to Dash Enterprise due to use of subdirectories for serving apps.

Reading from assets and data folder

```
Img(src="./assets/logo.png") will fail at root level
```

Tips

- Use get_asset_url()
- Use Pathlib for more flexibility

```
import pathlib
import pandas as pd

# get relative assets
html.Img(src=app.get_asset_url('logo.png'))       # /assets/logo.png

# get relative data

DATA_PATH = pathlib.Path(__file__).parent.joinpath("data")  # /data
df = pd.read_csv(DATA_PATH.joinpath("sample-data.csv"))    # /data/sample-data.csv

with open(DATA_PATH.joinpath("sample-data.csv")) as f:  # /data/sample-data.csv
    some_string = f.read()
```

# Developer Guide

## Creating a new project

```
# branch off master
git checkout -b "{YOUR_CUSTOM_BRANCH}"

# create a new folder in apps/
mkdir /apps/{DASH_APP_NAME}

# push new branch
git push -u origin {YOUR_CUSTOM_BRANCH}
```

## Before committing

```
# make sure your code is linted (we use black)
black . --exclude=venv/ --check

# if black is not installed
pip install black
```

## App is ready to go!

```
# once your branch is ready, make a PR into master!

PR has two checkers.
1. make sure your code passed the black linter
2. make sure your project is deployed on dns playground
```