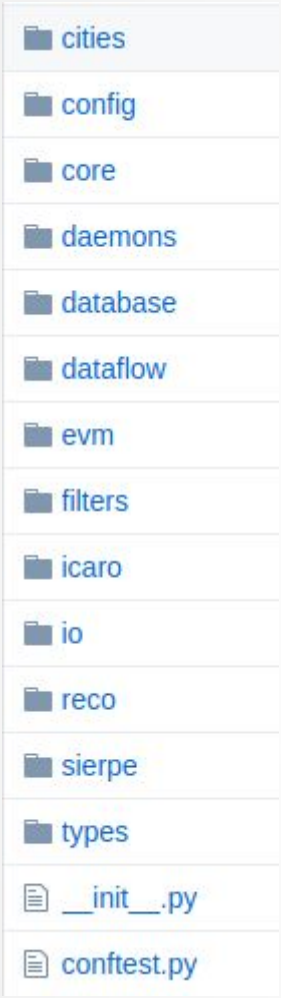
IC structure

What's on IC

bin	- Contains executable commands (currently only *city*)
doc	- Generic documentation, not updated nor reliable
invisible_cities	- The actual code!
.gitattributes	|
.gitconfig	|
.gitignore	- | Git stuff
.lfsconfig	|
.travis.yml	- Configuration for running test suite on travis
CONTRIBUTING.md	- It is supposed to describe how to contribute. I wouldn't trust it
README.rst	- Quickstart guide
bash_manage.sh	- You don't care
conftest.py	- Contains some configuration for hypothesis
ipynb_drop_output	- A script that I'm not sure what's doing here
manage.sh	- Script to manage the instalation and setup of the package
pytest.ini	- Pytest configuration
setup.py	- Script to install IC (basically cytonize)

What's on invisible_cities



- Cities and city components definitions
- Default and example config files
- Generic python functions. Not NEXT-specific (in theory)
- A corpse
- Functions to deal with the detector database
- Definition of the dataflow structure
- Definition of the event model
- Definition of data filters
- I don't know, don't ask me
- Input / output operations (i.e. functions to read and write data)
- Functions for data reconstruction
- Functions to deal with the energy plane electronics (BLR)
- Contains data type definitions (overlaps with evm)
- Python stuff. Ignore
- Pytest initialization and definition of stuff common to many tests

The most relevant modules: core

configure.py	- Stuff to deal with config files and command line
core_functions.py	- Generic python functions
exceptions.py	- Definitions of IC exceptions
fit_functions.py	- Functions to perform fits. Also frequently used distributions and more
random_sampling.py	- Functions to perform random sampling on any distribution
stat_functions.py	- Functions to deal with statistics
system_of_units.py	- Units definitions
testing_utils.py	- Useful functions to be used in tests

The most relevant modules: database

test_data	- Small data files to be used in tests
__init__.py	-
download.py	- Script to download database (deprecated)
download_test.py	-
load_db.py	- Functions to interact with the database
load_db_test.py	-
localdb.DEMOPPDB.sqlite3	- DEMO++ database
localdb.NEWDB.sqlite3	- NEW database
localdb.NEXT100DB.sqlite3	- NEXT100 database

The most relevant modules: evm

event_model.py	- Definition of transient data structures for NEXT (except PMaps)
event_model_test.py	-
histos.py	- I don’t know
histos_test.py	-
ic_containers.py	- Useful containers for data without physical meaning
nh5.py	- Definition of persistent data structures
pmaps.py	- Definition of Pmaps
pmaps_test.py	-

The most relevant modules: io

- channel_param_io.py
- channel_param_io_test.py
- dst_io.py
- dst_io_test.py
- event_filter_io.py
- event_filter_io_test.py
- fee_io.py
- hist_io.py
- hist_io_test.py
- hits_io.py
- hits_io_test.py
- kdst_io.py
- kdst_io_test.py
- mcinfo_io.py
- mcinfo_io_test.py
- pmaps_io.py
- pmaps_io_test.py
- run_and_event_io.py
- rwf_io.py
- table_io.py
- trigger_io.py
- voxels_io.py
- voxels_io_test.py

The most relevant modules: reco

calib_functions.py	- | Sensor calibration stuff
calib_sensors_functions.py	|
corrections.py	- Functions to perform corrections, old implementation
corrections_new.py	- Functions to perform corrections, new implementation
dst_functions.py	- Functions to deal with dsts, mostly deprecated
histogram_functions.py	- Functions to deal with monitoring histograms (not used in IC?)
hits_functions.py	- Functions to deal with Hits
monitor_functions.py	- Monitoring (ICAROS?)
paolina_functions.py	- Paolina stuff, track functions and so on
peak_functions.py	- Peak reconstruction functions
pmaps_functions.py	- Operations on Pmaps
sensor_functions.py	- Sensor simulation stuff
spe_response.py	- Functions to fit/describe single photon spectra
tbl_functions.py	- Pytables-related functions
wfm_functions.py	- Functions to manipulate waveforms.
xy_algorithms.py	- Functions for XY reconstruction

The city command

Read command line

Import city function

Pass arguments to city

```
#!/usr/bin/env python
import traceback
from sys       import argv
from importlib import import_module

from invisible_cities.core.configure import configure

_, *args  = argv
city_name = args[0]

try:
    module_name   = f'invisible_cities.cities.{city_name}'
    city_function = getattr(import_module(module_name), city_name)

except ModuleNotFoundError:
        print('available cities: <TODO look at contents of config directory>')
        print('usage TODO: <get this from our config parser>')
        traceback.print_exc()
        exit(1)
else:
    city_function(**configure(args))

```

Parse arguments combining config file with command line