CERC hub reference manual

Release 0.1.7.29

CERC Next-Generation Cities

Jul 31, 2023

CONTENTS:

CHAPTER

ONE

CERC HUB' REFERENCE MANUAL

1.1 Authors

- Guillermo Gutierrez Morote
- Pilar Monsalvete Alvarez de Uribarri

1.2 Contributors

- Seyedehrabeeh Hosseinihaghighi
- Milad Aghamohamadnia
- Peter Yefi
- Koa Wells
- Sanam Dabirian
- Soroush Samareh Abolhassani

1.3 About the hub

This document contains the essential documentation for the CERC hub, a set of classes, factories, and helpers that simplifies the research at urban scale in multiples domains; these components are designed around three central axes, **extensibility**, **code clarity** and **consistency** as we intend to allow domain experts to perform urban scale simulations with multiple programs and enrich the city from several data sources. hub is composed of four main components: **city model structure**, **factories catalogs** and **helpers**.

- **City model structure** is the set of classes designed to be familiar to the domain experts; this familiarity will be possible thanks to using a *standard-based* approach in order to flatten the learning curve. These classes compose the CERC *data model* that provides a simple way to study cities at an urban scale after the enriching process.
- **Factories** are pieces of code in charge of import and export information in and out of the **data model** they will perform the needed conversions to read or write different formats such as epw weather files, insel files or citygml. these factories are mean to be extended, allowing the hub ecosystem to expand with new formats.
- **Catalogs** are datasets used in the enrichment of the city that can also be used by third party consumers like researchers or simulations software.
- **Helpers** are set's of general tools used by any of the other parts and doesn't fit in any of the previous categories.

CHAPTER

TWO

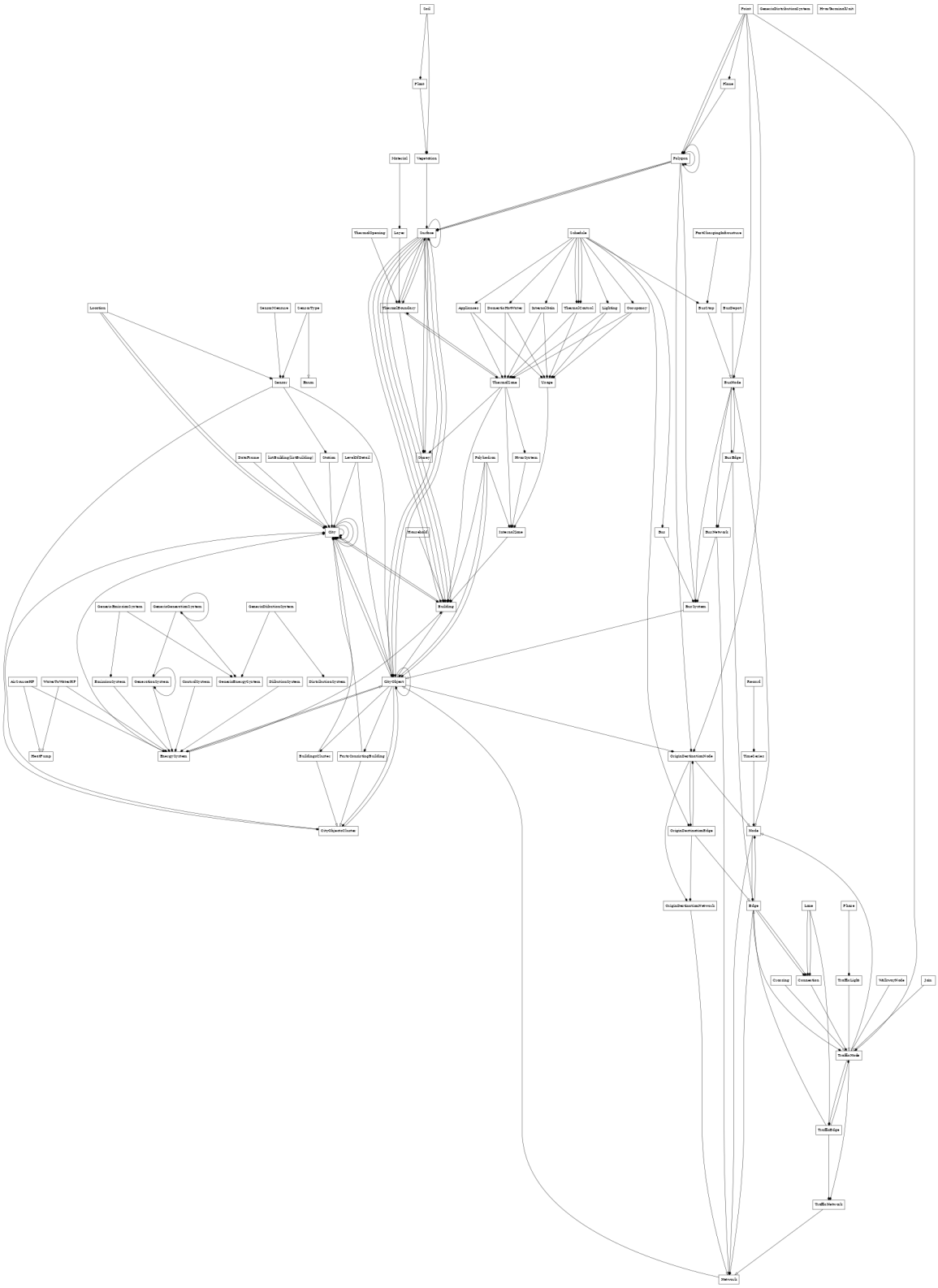
CITY MODEL STRUCTURE

The **city model structure** contains the common data model intended to represent a city digital twin, CERC team and contributors, will further extend these classes to include other domains, in the following sections, researchers and developers could find information about the methods and properties exposed by the city model structure classes.

Important: Please take a look to hub tutorial to see how to use hub for your own research

[how to use the hub]

2.1 City model structure UML



2.2 Folder structure

2.2.1 city_model_structure

Main city objects.

```
../hub/hub/city_model_structure/
├── building.py
├── buildings_cluster.py
├── bus_system.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── energy_system.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── __init__.py
├── level_of_detail.py
├── network.py
└── parts_consisting_building.py

1 directory, 15 files
```

2.2.2 attributes

Geometrical and non physical components of the city.

```
../hub/hub/city_model_structure/attributes/
├── edge.py
├── __init__.py
├── node.py
├── plane.py
├── point.py
├── polygon.py
├── polyhedron.py
├── record.py
├── schedule.py
└── time_series.py

0 directories, 10 files
```

2.2.3 building_demand

Main classes to model building energy demand.

../hub/hub/city_model_structure/building_demand/
├── appliances.py
├── domestic_hot_water.py
├── household.py
├── __init__.py
├── internal_gain.py
├── internal_zone.py
├── layer.py
├── lighting.py
├── material.py
├── occupancy.py
├── storey.py
├── surface.py
├── thermal_boundary.py
├── thermal_control.py
├── thermal_opening.py
├── thermal_zone.py
└── usage.py

0 directories, 17 files

2.2.4 energy_systems

Main classes to model energy systems.

../hub/hub/city_model_structure/energy_systems/
├── air_source_hp.py
├── control_system.py
├── distribution_system.py
├── emission_system.py
├── energy_system.py
├── generation_system.py
├── generic_distribution_system.py
├── generic_emission_system.py
├── generic_energy_system.py
├── generic_generation_system.py
├── heat_pump.py
├── hvac_system.py
├── hvac_terminal_unit.py
├── __init__.py
└── water_to_water_hp.py

0 directories, 15 files

2.2.5 iot

Classes to model IoT devices.

../hub/hub/city_model_structure/iot/
├── __init__.py
├── sensor_measure.py
├── sensor.py
├── sensor_type.py
└── station.py

0 directories, 5 files

2.2.6 transport

Classes to model transport.

```
../hub/hub/city_model_structure/transport/
├── bus_depot.py
├── bus_edge.py
├── bus_network.py
├── bus_node.py
├── bus.py
├── bus_stop.py
├── connection.py
├── crossing.py
├── __init__.py
├── join.py
├── lane.py
├── origin_destination_edge.py
├── origin_destination_network.py
├── origin_destination_node.py
├── phase.py
├── traffic_edge.py
├── traffic_light.py
├── traffic_network.py
├── traffic_node.py
└── walkway_node.py

0 directories, 20 files
```

2.2.7 full schema

```
../hub/hub/city_model_structure/
├── attributes
│   ├── edge.py
│   ├── __init__.py
│   ├── node.py
│   ├── plane.py
│   ├── point.py
│   ├── polygon.py
│   ├── polyhedron.py
│   ├── record.py
│   ├── schedule.py
│   └── time_series.py
├── building_demand
│   ├── appliances.py
│   ├── domestic_hot_water.py
│   ├── household.py
│   ├── __init__.py
│   ├── internal_gain.py
│   ├── internal_zone.py
│   ├── layer.py
│   ├── lighting.py
│   ├── material.py
│   ├── occupancy.py
│   ├── storey.py
│   ├── surface.py
│   ├── thermal_boundary.py
│   ├── thermal_control.py
│   ├── thermal_opening.py
│   ├── thermal_zone.py
│   └── usage.py
├── energy_systems
│   ├── air_source_hp.py
│   ├── control_system.py
│   ├── distribution_system.py
│   ├── emission_system.py
│   ├── energy_system.py
│   ├── generation_system.py
│   ├── generic_distribution_system.py
│   ├── generic_emission_system.py
│   ├── generic_energy_system.py
│   ├── generic_generation_system.py
│   ├── heat_pump.py
│   ├── hvac_system.py
│   ├── hvac_terminal_unit.py
│   ├── __init__.py
│   └── water_to_water_hp.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── iot
│   ├── __init__.py
│   ├── sensor_measure.py
│   ├── sensor.py
│   ├── sensor_type.py
│   └── station.py
├── transport
│   ├── bus_depot.py
│   ├── bus_edge.py
│   ├── bus_network.py
│   ├── bus_node.py
│   ├── bus.py
│   ├── bus_stop.py
│   ├── connection.py
│   ├── crossing.py
│   ├── __init__.py
│   ├── join.py
│   ├── lane.py
│   ├── origin_destination_edge.py
│   ├── origin_destination_network.py
│   ├── origin_destination_node.py
│   ├── phase.py
│   ├── traffic_edge.py
│   ├── traffic_light.py
│   ├── traffic_network.py
│   ├── traffic_node.py
│   └── walkway_node.py
├── building.py
├── buildings_cluster.py
├── bus_system.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── energy_system.py
├── __init__.py
├── level_of_detail.py
├── network.py
└── parts_consisting_building.py

6 directories, 82 files
```

2.3 Classes

2.3.1 City

class City (*lower_corner*, *upper_corner*, *srs_name*)

City class

add_building_alias (*building*, *alias*)

Add an alias to the building

add_city_object (*new_city_object*)

Add a CityObject to the city

Parameter new_city_object CityObject

Returns None or not implemented error

add_city_objects_cluster (*new_city_objects_cluster*)

Add a CityObject to the city

Parameter new_city_objects_cluster CityObjectsCluster

Returns None or NotImplementedError

building_alias (*alias*) → list[Building | list[Building]] | None

Retrieve the city CityObject with the given alias alias

⚠ Building alias is not guaranteed to be unique

Parameter alias str

Returns None or [CityObject]

city_object (*name*) → Optional[*CityObject*]

Retrieve the city CityObject with the given name

Parameter name str

Returns None or CityObject

static load (*city_filename*) → *City*

Load a city saved with city.save(city_filename)

Parameter city_filename city filename

Returns City

merge (*city*) → *City*

Return a merged city combining the current city and the given one

Returns City

region (*center*, *radius*) → *City*

Get a region from the city

Parameter center specific point in space [x, y, z]

Parameter radius distance to center of the sphere selected in meters

Returns selected_region_city

remove_city_object (*city_object*)

Remove a CityObject from the city

Parameter city_object CityObject

Returns None

save(*city_filename*)
Save a city into the given filename

Parameter city_filename destination city filename

Returns None

save_compressed(*city_filename*)
Save a city into the given filename

Parameter city_filename destination city filename

Returns None

2.3.2 CityObject

class CityObject(*name*, *surfaces*)
class CityObject

surface(*name*) → Optional[*Surface*]
Get the city object surface with a given name

Parameter name str

Returns None or Surface

surface_by_id(*identification_number*) → Optional[*Surface*]
Get the city object surface with a given name

Parameter identification_number str

Returns None or Surface

2.3.3 City Objects Cluster

class CityObjectsCluster(*name*, *cluster_type*, *city_objects*)
Inherit: ABC, CityObject
CityObjectsCluster(ABC) class

add_city_object(*city_object*) → [*CityObject*]
add new object to the cluster

Returns [CityObjects]

2.3.4 Building

class Building(*name*, *surfaces*, *year_of_construction*, *function*, *terrains=None*, *city=None*)
Inherit: CityObject
Building(CityObject) class

add_alias(*value*)
Add a new alias for the building

2.3.5 Parts Consisting Building

class PartsConsistingBuilding(*name*, *city_objects*)
Inherit: CityObjectsCluster
PartsConsistingBuilding(CityObjectsCluster) class

2.3.6 Buildings Cluster

class BuildingsCluster(*name*, *city_objects*)
Inherit: CityObjectsCluster
BuildingsCluster(CityObjectsCluster) class

2.3.7 Network

class Network(*name*, *edges=None*, *nodes=None*)
Inherit: CityObject
Generic network class to be used as base for the network models

2.3.8 Bus System

class BusSystem(*name*, *surfaces*)
Inherit: CityObject
BusSystem(CityObject) class

2.3.9 Energy System

class EnergySystem(*name*, *surfaces*)
Inherit: CityObject
EnergySystem(CityObject) class

2.3.10 Level of detail

class LevelOfDetail
Level of detail for the city class

2.3.11 Edge

class Edge(*name*, *nodes=None*)
Generic edge class to be used as base for the network edges

2.3.12 Node

class Node (*name*, *edges=None*)
Generic node class to be used as base for the network nodes

2.3.13 Plane

class Plane (*origin*, *normal*)
Plane class

distance_to_point (*point*)
Distance between the given point and the plane

Returns float

2.3.14 Point

class Point (*coordinates*)
Point class

distance_to_point (*other_point*)
Calculates distance between points in an n-D Euclidean space

Parameter other_point point or vertex

Returns float

2.3.15 Polygon

class Polygon (*coordinates*)
Polygon class

divide (*plane*)
Divides the polygon in two by a plane

Parameter plane plane that intersects with self to divide it in two parts (Plane)

Returns Polygon, Polygon, [Point]

static triangle_mesh (*vertices*, *normal*) → Trimesh
Get the triangulated mesh for the polygon

Returns Trimesh

2.3.16 Polyhedron

class Polyhedron (*polygons*)
Polyhedron class

obj_export (*full_path*)
Export the polyhedron to obj given file

Parameter full_path str

Returns None

show()
Auxiliary function to render the polyhedron

Returns None

stl_export(*full_path*)
Export the polyhedron to stl given file

Parameter full_path str

Returns None

2.3.17 Record

class Record(*time=None*, *value=None*, *flag=None*)
Record class

2.3.18 Schedule

class Schedule
Schedule class

2.3.19 Time Series

class TimeSeries(*time_series_type=None*, *records=None*)
TimeSeries class

2.3.20 Appliances

class Appliances
Appliances class

2.3.21 Household

class Household
Household class

2.3.22 Internal Gain

class InternalGain
InternalGain class

2.3.23 Internal Zone

Note: The internal zone class represents each of the internal zones described in the geometry when imported.

This imported geometry can be later on divided in different thermal zones in a workflow (e.g. if the building with no interiors defined (LoD up to 3), it will produce one interior zone. Later on, this can be divided by storey and create one thermal zone per each).

Also, several usages can be associated with that internal zone. This usages are described in the Usage class, which has not only the parameters that describe each usage, but also the percentage of the internal zone volume that is affected by that specific use.

class InternalZone(*surfaces*, *area*)
InternalZone class

2.3.24 Layer

class Layer
Layer class

2.3.25 Lighting

class Lighting
Lighting class

2.3.26 Material

class Material
Material class

2.3.27 Occupancy

class Occupancy
Occupancy class

2.3.28 Storey

class Storey(*name*, *storey_surfaces*, *neighbours*, *volume*, *internal_zone*, *floor_area*)
Storey class

2.3.29 Surface

class Surface(*solid_polygon*, *perimeter_polygon*, *holes_polygons=None*, *name=None*, *surface_type=None*)
Surface class

divide(*z*)
Divides a surface at Z plane

Returns Surface, Surface, Any

2.3.30 Thermal Boundary

class ThermalBoundary(*parent_surface*, *opaque_area*, *windows_areas*)
ThermalBoundary class

2.3.31 Thermal Control

class ThermalControl
ThermalControl class

2.3.32 Thermal Opening

class ThermalOpening
ThermalOpening class

2.3.33 Thermal Zone

class ThermalZone(*thermal_boundaries*, *parent_internal_zone*, *volume*, *footprint_area*, *usage_name=None*)
ThermalZone class

2.3.34 Usage

class Usage
Usage class

2.3.35 Air Source

class AirSourceHP
Inherit: HeatPump
AirSourceHP class

2.3.36 Heat Pump

class HeatPump
HeatPump class

2.3.37 HVAC System

class HvacSystem
HvacSystem class

2.3.38 HVAC Terminal Unit

class HvacTerminalUnit
HvacTerminalUnit class

2.3.39 Water to Water HP

class WaterToWaterHP
Inherit: HeatPump
WaterToWaterHP class

2.3.40 Plant

class Plant(*name*, *height*, *leaf_area_index*, *leaf_reflectivity*, *leaf_emissivity*, *minimal_stomatal_resistance*, *co2_sequestration*, *grows_on_soils*)
Plant class

2.3.41 Soil

class Soil(*name*, *roughness*, *dry_conductivity*, *dry_density*, *dry_specific_heat*, *thermal_absorptance*, *solar_absorptance*, *visible_absorptance*, *saturation_volumetric_moisture_content*, *residual_volumetric_moisture_content*)
Soil class

2.3.42 Vegetation

class Vegetation(*name*, *soil*, *soil_thickness*, *plants*)
Vegetation class

2.3.43 Sensor

class Sensor
Sensor abstract class

2.3.44 Sensor Measure

class SensorMeasure(*latitude*, *longitude*, *utc_timestamp*, *value*)
Sensor measure class

2.3.45 Sensor Type

class SensorType(*value*)
Inherit: Enum
Sensor type enumeration

2.3.46 Station

class Station(*station_id=None*, *_mobile=False*)
Station class

2.3.47 Connection

class Connection
Connection class

2.3.48 Crossing

class Crossing(*name*, *coordinates*, *priority*, *width*, *shape=None*, *edges=None*)
Inherit: TrafficNode
Crossing class

2.3.49 Join

class Join(*name*, *coordinates*, *nodes*)
Inherit: TrafficNode
Join class

2.3.50 Lane

class Lane
Lane class

2.3.51 Phase

class Phase
Phase class

2.3.52 Traffic Edge

class TrafficEdge(*name*, *nodes*, *priority*, *speed*, *lanes*, *length*, *allows=None*, *disallows=None*, *sidewalk_width=None*, *edge_type='TrafficEdge'*)
Inherit: Edge
TrafficEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

2.3.53 Traffic Light

class TrafficLight(*name*, *coordinates*, *offset*, *phases=None*, *edges=None*, *right_on_red=False*)
Inherit: TrafficNode
Traffic light class

2.3.54 Traffic Network

class TrafficNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network
TrafficNetwork(Network) class

2.3.55 Traffic Node

class TrafficNode(*name*, *coordinates*, *node_type='TrafficNode'*, *edges=None*, *prohibitions=None*, *connections=None*)
Inherit: Node
TrafficNode class

2.3.56 Walkway Node

class WalkwayNode(*name*, *coordinates*, *edges=None*, *shape=None*)
Inherit: TrafficNode
WalkwayNode class

2.3.57 Bus

class Bus
Bus class

2.3.58 Bus Depot

class BusDepot(*name*, *coordinates*, *edges=None*)
Inherit: BusNode
BusDepot class

2.3.59 Bus Edge

class BusEdge(*name*, *nodes*, *edge_type='BusEdge'*)
Inherit: Edge
BusEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

2.3.60 Bus Network

class BusNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network
BusNetwork(Network) class

2.3.61 Bus Node

class BusNode(*name*, *coordinates*, *node_type='BusNode'*, *edges=None*)
Inherit: Node
BusNode class

2.3.62 Bus Stop

2.3.63 Fast Charging Infrastructure

2.3.64 Origin Destination Network

class OriginDestinationNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network
OriginDestinationNetwork(Network) class

2.3.65 Origin Destination Edge

class OriginDestinationEdge(*name*, *nodes*, *edge_type='OriginDestinationEdge'*)
Inherit: Edge
OriginDestinationEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

2.3.66 Origin Destination Node

class OriginDestinationNode(*name*, *coordinates*, *node_type='OriginDestinationNode'*, *edges=None*, *polygon=None*)

Inherit: Node

OriginDestinationNode class

CHAPTER

THREE

FACTORIES

Factories are divided into Imports and Exports, depending on if they are used to enrich (Import) the *city model structure* or to deliver third party defined formats (Export) such as **INSEL** or **IDF** file, the factories could be extended to include new imports and outputs providing an additional level of abstraction to researchers.

⚠ Please, note that the private methods, the ones starting with an underscore character (_), documented in the factories are mean to be called by using the **handler** parameter; this parameter must contain the method name without the _ character.

Note: For instance, to use _citygml handler in the Geometry factory, the handler parameter value needs to be 'citygml'

⚠ **This documentation includes only the base factories classes as these are the intended entry points for the Import/Export functionality.**

Important: Please refer to the development manual if you want to create your own factories.

[development manual]

3.1 Folder structure

3.1.1 Imports

```
../hub/hub/imports/
├── energy_systems
│   ├── helpers
│   │   ├── energy_systems_helper.py
│   │   └── __init__.py
│   ├── air_source_hp_parameters.py
│   ├── __init__.py
│   ├── montreal_custom_energy_system_parameters.py
│   └── water_to_water_hp_parameters.py
├── results
│   ├── __init__.py
│   ├── insel_heatpump_energy_demand.py
│   ├── insel_monthly_energy_balance.py
│   └── simplified_radiosity_algorithm.py
├── construction_factory.py
├── energy_systems_factory.py
├── geometry_factory.py
├── __init__.py
├── results_factory.py
├── usage_factory.py
└── weather_factory.py

3 directories, 17 files
```

3.1.2 Exports

```
../hub/hub/exports/
├── building_energy
│   ├── idf_files
│   │   ├── Energy+.idd
│   │   └── Minimal.idf
│   ├── insel
│   │   ├── __init__.py
│   │   └── insel_monthly_energy_balance.py
│   ├── energy_ade.py
│   ├── idf.py
│   └── __init__.py
├── energy_systems
│   ├── air_source_hp_export.py
│   ├── heat_pump_export.py
│   └── water_to_water_hp_export.py
├── energy_building_exports_factory.py
├── energy_systems_factory.py
├── exports_factory.py
└── __init__.py

4 directories, 14 files
```

3.2 Imports Classes

3.2.1 Construction Factory

class ConstructionFactory(*handler*, *city*)
ConstructionFactory class

_eilat()
Enrich the city by using Eilat information

_nrcan()
Enrich the city by using NRCAN information

_nrel()
Enrich the city by using NREL information

enrich()
Enrich the city given to the class using the class given handler

Returns None

3.2.2 Geometry Factory

class GeometryFactory(*file_type*, *path=None*, *data_frame=None*, *aliases_field=None*, *height_field=None*, *year_of_construction_field=None*, *function_field=None*, *function_to_hub=None*)
GeometryFactory class

3.2.3 Usage Factory

class UsageFactory(*handler*, *city*)
UsageFactory class

_comnet()
Enrich the city with COMNET usage library

_eilat()
Enrich the city with Eilat usage library

_nrcan()
Enrich the city with NRCAN usage library

enrich()
Enrich the city given to the class using the usage factory given handler

Returns None

3.2.4 Weather Factory

class WeatherFactory(*handler*, *city: hub.city_model_structure.city.City*)
WeatherFactory class

_epw()
Enrich the city with energy plus weather file

enrich()
Enrich the city given to the class using the given weather handler

Returns None

3.3 Exports

3.3.1 Export Factory

class ExportsFactory(*handler*, *city*, *path*, *target_buildings=None*, *adjacent_buildings=None*)

Exports factory class

export()

Export the city given to the class using the given export type handler

Returns None

CHAPTER

FOUR

CATALOGS

In its simplest form, a catalogue is a file or group of files that provide technical and/or commercial information regarding components that form a system within any domain. The components are listed with relevant details and associated data is tabulated. Also listed are the dominant/standard configurations in which the components may be used to satisfy use-cases/output requirements (as supplied by component manufacturer/standard organisations).

Note: Examples, Heat Pump catalogue should consist of the heat pump models produced, heat pump type, manufacturer name, output temperatures, nominal capacities, typical configurations for the heat pumps (e.g., configurations when used for space heating only, Domestic Hot Water/DHW purposes only, both space heating and DHW, combinations with solar thermal/PV), storage tank data, circulation pump data, compressor type and associated technical data, valve types etc.

Important: Please refer to the catalogs manual if you want to create or extend your own catalogs.

[catalogs manual]

4.1 Folder structure

4.1.1 Catalogs

```
../hub/hub/catalog_factories/
├── construction
│   ├── construction_helper.py
│   ├── eilat_catalog.py
│   ├── __init__.py
│   ├── nrcan_catalog.py
│   └── nrel_catalog.py
├── cost
│   ├── __init__.py
│   └── montreal_custom_catalog.py
├── data_models
│   ├── construction
│   │   ├── archetype.py
│   │   ├── construction.py
│   │   ├── content.py
│   │   ├── __init__.py
│   │   ├── layer.py
│   │   ├── material.py
│   │   └── window.py
│   ├── cost
│   │   ├── archetype.py
│   │   ├── capital_cost.py
│   │   ├── chapter.py
│   │   ├── content.py
│   │   ├── fuel.py
│   │   ├── income.py
│   │   ├── __init__.py
│   │   ├── item_description.py
│   │   └── operational_cost.py
│   ├── energy_systems
│   │   ├── archetype.py
│   │   ├── content.py
│   │   ├── distribution_system.py
│   │   ├── emission_system.py
│   │   ├── generation_system.py
│   │   ├── __init__.py
│   │   └── system.py
│   ├── greenery
│   │   ├── content.py
│   │   ├── __init__.py
│   │   ├── plant_percentage.py
│   │   ├── plant.py
│   │   ├── soil.py
│   │   └── vegetation.py
│   ├── usages
│   │   ├── appliances.py
│   │   ├── content.py
│   │   ├── domestic_hot_water.py
│   │   ├── __init__.py
│   │   ├── lighting.py
│   │   ├── occupancy.py
│   │   ├── schedule.py
│   │   ├── thermal_control.py
│   │   └── usage.py
│   └── __init__.py
├── energy_systems
│   ├── __init__.py
│   └── montreal_custom_catalog.py
├── greenery
│   ├── ecore_greenery
│   │   ├── greenerycatalog.ecore
│   │   ├── greenerycatalog_no_quantities.ecore
│   │   └── greenerycatalog.py
│   ├── greenery_catalog.py
│   └── __init__.py
├── usage
│   ├── comnet_catalog.py
│   ├── eilat_catalog.py
│   ├── __init__.py
│   ├── nrcan_catalog.py
│   └── usage_helper.py
├── catalog.py
├── construction_catalog_factory.py
├── costs_catalog_factory.py
├── energy_systems_catalog_factory.py
├── greenery_catalog_factory.py
├── __init__.py
└── usage_catalog_factory.py

12 directories, 65 files
```

4.2 Catalog Base Class

4.2.1 Catalog

class Catalog

Catalogs base class catalog_factories will inherit from this class.

entries(*category=None*)

Base property to return the catalog entries

Returns Catalog content filter by category if provided

get_entry(*name*)

Base property to return the catalog entry matching the given name

Returns Catalog entry with the matching name

names(*category=None*)

Base property to return the catalog entries names.

Returns Catalog names filter by category if provided

4.3 Greenery

4.3.1 Greenery Catalog Factory

class GreeneryCatalogFactory(*handler*, *base_path=None*)
GreeneryCatalogFactory class

4.3.2 Greenery Content Data Model

class Content(*vegetations*, *plants*, *soils*)
Content class

to_dictionary()
Class content to dictionary

4.3.3 Greenery Plant Data Model

class Plant(*category*, *plant*)
Plant class

to_dictionary()
Class content to dictionary

4.3.4 Greenery Plant Percentage Data Model

class PlantPercentage(*percentage*, *plant_category*, *plant*)
Inherit: Plant
Plant percentage class

to_dictionary()
Class content to dictionary

4.3.5 Greenery Plant Soil Data Model

class Soil(*soil*)
Soil class

to_dictionary()
Class content to dictionary

4.3.6 Greenery Vegetation Data Model

class Vegetation(*category*, *vegetation*, *plant_percentages*)
Vegetation class

to_dictionary()
Class content to dictionary

4.4 Construction

4.4.1 Construction Catalog Factory

class ConstructionCatalogFactory(*handler*, *base_path=None*)
Construction catalog factory class

4.4.2 Construction Content Data Model

class Content(*archetypes*, *constructions*, *materials*, *windows*)
Content class

to_dictionary()
Class content to dictionary

4.4.3 Construction Archetype Data Model

class Archetype(*archetype_id*, *name*, *function*, *climate_zone*, *construction_period*, *constructions*, *average_storey_height*, *thermal_capacity*, *extra_loses_due_to_thermal_bridges*, *indirect_heated_ratio*, *infiltration_rate_for_ventilation_system_off*, *infiltration_rate_for_ventilation_system_on*)
Archetype class

to_dictionary()
Class content to dictionary

4.4.4 Construction Construction Data Model

class Construction(*construction_id*, *construction_type*, *name*, *layers*, *window_ratio=None*, *window=None*)
Construction class

to_dictionary()
Class content to dictionary

4.4.5 Construction Layer Data Model

class Layer(*layer_id*, *name*, *material*, *thickness*)
Layer class

to_dictionary()
Class content to dictionary

4.4.6 Construction Material Data Model

class Material(*material_id*, *name*, *solar_absorptance*, *thermal_absorptance*, *visible_absorptance*, *no_mass=False*, *thermal_resistance=None*, *conductivity=None*, *density=None*, *specific_heat=None*)

Material class

to_dictionary()

Class content to dictionary

4.4.7 Construction Window Data Model

class Window(*window_id*, *frame_ratio*, *g_value*, *overall_u_value*, *name*, *window_type=None*)

Window class

to_dictionary()

Class content to dictionary

4.5 Costs

4.5.1 Costs Catalog Factory

class CostsCatalogFactory(*file_type*, *base_path=None*)
CostsCatalogFactory class

4.5.2 Costs Content Data Model

class Content(*archetypes*)
Content class

to_dictionary()
Class content to dictionary

4.5.3 Costs Archetype Data Model

class Archetype(*lod*, *function*, *municipality*, *country*, *currency*, *capital_cost*, *operational_cost*, *end_of_life_cost*, *income*)
Archetype class

to_dictionary()
Class content to dictionary

4.5.4 Costs Capital Cost Data Model

class CapitalCost(*general_chapters*, *design_allowance*, *overhead_and_profit*)
Capital cost class

chapter(*name*) → *Chapter*
Get specific chapter by name

Returns Chapter

to_dictionary()
Class content to dictionary

4.5.5 Costs Chapter Data Model

class Chapter(*chapter_type*, *items*)
Chapter class

item(*name*) → *ItemDescription*
Get specific item by name

Returns ItemDescription

to_dictionary()
Class content to dictionary

4.5.6 Costs Fuel Data Model

class Fuel(*fuel_type*, *fixed_monthly=None*, *fixed_power=None*, *variable=None*, *variable_units=None*)
Fuel class

to_dictionary()
Class content to dictionary

4.5.7 Costs Income Data Model

class Income(*construction_subsidy=None*, *hvac_subsidy=None*, *photovoltaic_subsidy=None*, *electricity_export=None*, *reductions_tax=None*)
Income class

to_dictionary()
Class content to dictionary

4.5.8 Costs Item Description Data Model

class ItemDescription(*item_type*, *initial_investment=None*, *initial_investment_unit=None*, *refurbishment=None*, *refurbishment_unit=None*, *reposition=None*, *reposition_unit=None*, *lifetime=None*)
Item description class

to_dictionary()
Class content to dictionary

4.5.9 Costs Operational Cost Data Model

class OperationalCost(*fuels*, *maintenance_heating*, *maintenance_cooling*, *maintenance_pv*, *co2*)
Operational cost class

to_dictionary()
Class content to dictionary

4.6 Energy Systems

4.6.1 Energy Systems Catalog Factory

class EnergySystemsCatalogFactory(*handler*, *base_path=None*)
Energy system catalog factory class

4.6.2 Energy Systems Content Data Model

class Content(*archetypes*, *systems*, *generations*, *distributions*, *emissions*)
Content class

to_dictionary()
Class content to dictionary

4.6.3 Energy Systems Archetype Data Model

class Archetype(*lod*, *name*, *systems*)
Archetype class

to_dictionary()
Class content to dictionary

4.6.4 Energy Systems Distribution System Data Model

class DistributionSystem(*system_id*, *name*, *system_type*, *supply_temperature*, *distribution_consumption_fix_flow*, *distribution_consumption_variable_flow*, *heat_losses*)
Distribution system class

to_dictionary()
Class content to dictionary

4.6.5 Energy Systems Emission System Data Model

class EmissionSystem(*system_id*, *name*, *system_type*, *parasitic_energy_consumption*)
Emission system class

to_dictionary()
Class content to dictionary

4.6.6 Energy Systems Generation System Data Model

class GenerationSystem(*system_id*, *name*, *system_type*, *fuel_type*, *source_types*, *heat_efficiency*, *cooling_efficiency*, *electricity_efficiency*, *source_temperature*, *source_mass_flow*, *storage*, *auxiliary_equipment*)
Generation system class

to_dictionary()
Class content to dictionary

4.6.7 Energy Systems Energy Systems Data Model

class System(*lod*, *system_id*, *name*, *demand_types*, *generation_system*, *distribution_system*, *emission_system*)

System class

to_dictionary()

Class content to dictionary

4.7 Usage

4.7.1 Usage Catalog Factory

class UsageCatalogFactory(*handler*, *base_path=None*)
Usage catalog factory class

4.7.2 Usage Content Data Model

class Content(*usages*)
Content class

to_dictionary()
Class content to dictionary

4.7.3 Usage Appliances Data Model

class Appliances(*density*, *convective_fraction*, *radiative_fraction*, *latent_fraction*, *schedules*)
Appliances class

to_dictionary()
Class content to dictionary

4.7.4 Usage Content Data Model

class Content(*usages*)
Content class

to_dictionary()
Class content to dictionary

4.7.5 Usage Domestic Hot Water Data Model

class DomesticHotWater(*density*, *peak_flow*, *service_temperature*, *schedules*)
DomesticHotWater class

to_dictionary()
Class content to dictionary

4.7.6 Usage Internal Gain Data Model

4.7.7 Usage Lighting Data Model

class Lighting(*density*, *convective_fraction*, *radiative_fraction*, *latent_fraction*, *schedules*)
Lighting class

to_dictionary()
Class content to dictionary

4.7.8 Usage Occupancy Data Model

class Occupancy(*occupancy_density*, *sensible_convective_internal_gain*, *sensible_radiative_internal_gain*, *latent_internal_gain*, *schedules*)

Occupancy class

to_dictionary()

Class content to dictionary

4.7.9 Usage Schedule Data Model

class Schedule(*schedule_type*, *values*, *data_type*, *time_step*, *time_range*, *day_types*)

Schedule class

to_dictionary()

Class content to dictionary

4.7.10 Usage Thermal Control Data Model

class ThermalControl(*mean_heating_set_point*, *heating_set_back*, *mean_cooling_set_point*, *hvac_availability_schedules*, *heating_set_point_schedules*, *cooling_set_point_schedules*)

ThermalControl class

to_dictionary()

Class content to dictionary

4.7.11 Usage Usage Data Model

class Usage(*name*, *hours_day*, *days_year*, *mechanical_air_change*, *ventilation_rate*, *occupancy*, *lighting*, *appliances*, *thermal_control*, *domestic_hot_water*)

Usage class

to_dictionary()

Class content to dictionary

CHAPTER

FIVE

HELPERS

CERC hub provides a set of *helpers* that will simplify certain operations; these helpers are mean to be freely used at any point and therefore could be consumed from several places.

5.1 Folder structure

```
../hub/hub/helpers/
├── data
│   ├── alkis_function_to_hub_function.py
│   ├── eilat_function_to_hub_function.py
│   ├── hft_function_to_hub_function.py
│   ├── hub_function_to_eilat_construction_function.py
│   ├── hub_function_to_montreal_custom_costs_function.py
│   ├── hub_function_to_nrcan_construction_function.py
│   ├── hub_function_to_nrel_construction_function.py
│   ├── hub_usage_to_comnet_usage.py
│   ├── hub_usage_to_eilat_usage.py
│   ├── hub_usage_to_hft_usage.py
│   ├── hub_usage_to_nrcan_usage.py
│   ├── __init__.py
│   ├── montreal_demand_type_to_hub_energy_demand_type.py
│   ├── montreal_function_to_hub_function.py
│   ├── montreal_system_to_hub_energy_generation_system.py
│   └── pluto_function_to_hub_function.py
├── peak_calculation
│   ├── __init__.py
│   └── loads_calculation.py
├── auth.py
├── configuration_helper.py
├── constants.py
├── dictionaries.py
├── geometry_helper.py
├── __init__.py
├── location.py
├── monthly_values.py
├── peak_loads.py
└── utils.py

2 directories, 28 files
```

5.2 Configuration Helper

class ConfigurationHelper
Configuration class

5.3 Constants

KELVIN = 273.15
HOUR_TO_MINUTES = 60
MINUTES_TO_SECONDS = 60
HOUR_TO_SECONDS = 3600
METERS_TO_FEET = 3.28084
BTU_H_TO_WATTS = 0.29307107
KILO_WATTS_HOUR_TO_JULES = 3600000
GALLONS_TO_QUBIC_METERS = 0.0037854117954011185
SECOND = 'second'
MINUTE = 'minute'
HOUR = 'hour'
DAY = 'day'
WEEK = 'week'
MONTH = 'month'
YEAR = 'year'
MONDAY = 'monday'
TUESDAY = 'tuesday'
WEDNESDAY = 'wednesday'
THURSDAY = 'thursday'
FRIDAY = 'friday'
SATURDAY = 'saturday'
SUNDAY = 'sunday'
HOLIDAY = 'holiday'
WINTER_DESIGN_DAY = 'winter_design_day'
SUMMER_DESIGN_DAY = 'summer_design_day'
WEEK_DAYS = 'Weekdays'
WEEK_ENDS = 'Weekends'
ALL_DAYS = 'Alldays'
DAYS_A_MONTH = {'monday': [5, 4, 4, 5, 4, 4, 5, 4, 4, 5, 4, 5],
'tuesday': [5, 4, 4, 4, 5, 4, 5, 4, 4, 5, 4, 4],
'wednesday': [5, 4, 4, 4, 5, 4, 4, 5, 4, 5, 4, 4],
'thursday': [4, 4, 5, 4, 5, 4, 4, 5, 4, 4, 5, 4],
'friday': [4, 4, 5, 4, 4, 5, 4, 5, 4, 4, 5, 4],
'saturday': [4, 4, 5, 4, 4, 5, 4, 4, 5, 4, 4, 5],
'sunday': [4, 4, 4, 5, 4, 4, 5, 4, 5, 4, 4, 5],
'holiday': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}
DAYS_A_YEAR = {'monday': 51,
'tuesday': 50,
'wednesday': 50,
'thursday': 50,
'friday': 50,
'saturday': 52,
'sunday': 52,
'holiday': 10}
ANY_NUMBER = 'any_number'

FRACTION = ‘fraction’
ON_OFF = ‘on_off’
TEMPERATURE = ‘temperature’
HUMIDITY = ‘humidity’
CONTROL_TYPE = ‘control_type’
CONTINUOUS = ‘continuous’
DISCRETE = ‘discrete’
CONSTANT = ‘constant’
INTERNAL_GAINS = ‘internal_gains’
WALL = ‘Wall’
GROUND_WALL = ‘Ground wall’
GROUND = ‘Ground’
ATTIC_FLOOR = ‘Attic floor’
ROOF = ‘Roof’
INTERIOR_SLAB = ‘Interior slab’
INTERIOR_WALL = ‘Interior wall’
VIRTUAL_INTERNAL = ‘Virtual internal’
WINDOW = ‘Window’
DOOR = ‘Door’
SKYLIGHT = ‘Skylight’
RESIDENTIAL = ‘residential’
SINGLE_FAMILY_HOUSE = ‘single family house’
MULTI_FAMILY_HOUSE = ‘multifamily house’
ROW_HOUSE = ‘row house’
MID_RISE_APARTMENT = ‘mid rise apartment’
HIGH_RISE_APARTMENT = ‘high rise apartment’
OFFICE_AND_ADMINISTRATION = ‘office and administration’
SMALL_OFFICE = ‘small office’
MEDIUM_OFFICE = ‘medium office’
LARGE_OFFICE = ‘large office’
COURTHOUSE = ‘courthouse’
FIRE_STATION = ‘fire station’
PENITENTIARY = ‘penitentiary’
POLICE_STATION = ‘police station’
POST_OFFICE = ‘post office’
LIBRARY = ‘library’
EDUCATION = ‘education’
PRIMARY_SCHOOL = ‘primary school’
PRIMARY_SCHOOL_WITH_SHOWER = ‘school with shower’
SECONDARY_SCHOOL = ‘secondary school’
UNIVERSITY = ‘university’
LABORATORY_AND_RESEARCH_CENTER = ‘laboratory and research centers’
STAND_ALONE_RETAIL = ‘stand alone retail’
HOSPITAL = ‘hospital’
OUT_PATIENT_HEALTH_CARE = ‘out-patient health care’
HEALTH_CARE = ‘health care’
RETIREMENT_HOME_OR_ORPHANAGE = ‘retirement home or orphanage’

COMMERCIAL = 'commercial'
STRIP_MALL = 'strip mall'
SUPERMARKET = 'supermarket'
RETAIL_SHOP_WITHOUT_REFRIGERATED_FOOD = 'retail shop without refrigerated food'
RETAIL_SHOP_WITH_REFRIGERATED_FOOD = 'retail shop with refrigerated food'
RESTAURANT = 'restaurant'
QUICK_SERVICE_RESTAURANT = 'quick service restaurant'
FULL_SERVICE_RESTAURANT = 'full service restaurant'
HOTEL = 'hotel'
HOTEL_MEDIUM_CLASS = 'hotel medium class'
SMALL_HOTEL = 'small hotel'
LARGE_HOTEL = 'large hotel'
DORMITORY = 'dormitory'
EVENT_LOCATION = 'event location'
CONVENTION_CENTER = 'convention center'
HALL = 'hall'
GREEN_HOUSE = 'green house'
INDUSTRY = 'industry'
WORKSHOP = 'workshop'
WAREHOUSE = 'warehouse'
WAREHOUSE_REFRIGERATED = 'warehouse refrigerated'
SPORTS_LOCATION = 'sports location'
SPORTS_ARENA = 'sports arena'
GYMNASIUM = 'gymnasium'
MOTION_PICTURE_THEATRE = 'motion picture theatre'
MUSEUM = 'museum'
PERFORMING_ARTS_THEATRE = 'performing arts theatre'
TRANSPORTATION = 'transportation'
AUTOMOTIVE_FACILITY = 'automotive facility'
PARKING_GARAGE = 'parking garage'
RELIGIOUS = 'religious'
NON_HEATED = 'non-heated'
DATACENTER = 'datacenter'
LIGHTING = 'Lights'
OCCUPANCY = 'Occupancy'
APPLIANCES = 'Appliances'
HVAC_AVAILABILITY = 'HVAC Avail'
INFILTRATION = 'Infiltration'
VENTILATION = 'Ventilation'
COOLING_SET_POINT = 'ClgSetPt'
HEATING_SET_POINT = 'HtgSetPt'
EQUIPMENT = 'Equipment'
ACTIVITY = 'Activity'
PEOPLE_ACTIVITY_LEVEL = 'People Activity Level'
DOMESTIC_HOT_WATER = 'Domestic Hot Water'
HEATING = 'Heating'
COOLING = 'Cooling'

ELECTRICITY = 'Electricity'
RENEWABLE = 'Renewable'
WOOD = 'Wood'
GAS = 'Gas'
DIESEL = 'Diesel'
COAL = 'Coal'
AIR = 'Air'
WATER = 'Water'
GEOTHERMAL = 'Geothermal'
DISTRICT_HEATING_NETWORK = 'District Heating'
GRID = 'Grid'
ONSITE_ELECTRICITY = 'Onsite Electricity'
PHOTOVOLTAIC = 'Photovoltaic'
BOILER = 'Boiler'
HEAT_PUMP = 'Heat Pump'
BASEBOARD = 'Baseboard'
CHILLER = 'Chiller'
EPSILON = 0.0000001
ONLY_HEATING = 'Heating'
ONLY_COOLING = 'Colling'
ONLY_VENTILATION = 'Ventilation'
HEATING_AND_VENTILATION = 'Heating and ventilation'
COOLING_AND_VENTILATION = 'Cooling and ventilation'
HEATING_AND_COLLING = 'Heating and cooling'
FULL_HVAC = 'Heating and cooling and ventilation'
MAX_FLOAT = float('inf')
MIN_FLOAT = float('-inf')
SRA = 'sra'
INSEL_MEB = 'insel meb'
CURRENCY_PER_SQM = 'currency/m2'
CURRENCY_PER_CBM = 'currency/m3'
CURRENCY_PER_KW = 'currency/kW'
CURRENCY_PER_KWH = 'currency/kWh'
CURRENCY_PER_MONTH = 'currency/month'
CURRENCY_PER_LITRE = 'currency/l'
CURRENCY_PER_KG = 'currency/kg'
CURRENCY_PER_CBM_PER_HOUR = 'currency/(m3/h)'
PERCENTAGE = '%'
SUPERSTRUCTURE = 'B_shell'
ENVELOPE = 'D_services'
ALLOWANCES_OVERHEAD_PROFIT = 'Z_allowances_overhead_profit'

5.4 Geometry Helper

class GeometryHelper(*delta=0*, *area_delta=0*)
Geometry helper class

static city_mapping(*city*, *building_names=None*, *plot=False*) → Dict
Parameter city city to be mapped

Parameter building_names list of building names to be mapped or None

Parameter plot True if minimap image should be displayed

Returns shared_information dictionary

static coordinate_to_map_point(*coordinate*, *city*)
Transform a real world coordinate to a minimap one

Parameter coordinate real world coordinate

Parameter city current city

Returns None

static distance_between_points(*vertex1*, *vertex2*)
distance between points in an n-D Euclidean space

Parameter vertex1 point or vertex

Parameter vertex2 point or vertex

Returns float

static divide_mesh_by_plane(*trimesh*, *normal_plane*, *point_plane*)
Divide a mesh by a plane

Parameter trimesh Trimesh

Parameter normal_plane [x, y, z]

Parameter point_plane [x, y, z]

Returns [Trimesh]

static factor()
Set minimap resolution

Returns None

static get_location(*latitude*, *longitude*) → *Location*
Get Location from latitude and longitude

Parameter latitude Latitude

Parameter longitude Longitude

Returns Location

static segment_list_to_trimesh(*lines*) → Trimesh
Parameter lines lines

Returns Transform a list of segments into a Trimesh

5.5 Location

class Location(*country*, *city*, *region_code*)

5.6 Dictionaries

class Dictionaries
Dictionaries class

CHAPTER

SIX

ADDITIONAL FILES

6.1 Readme

README.md

6.2 License

LICENSE.md

6.3 Code of conduct

CODE_OF_CONDUCT.md

6.4 How to contribute

CONTRIBUTING.md

6.5 Coding style

PYGUIDE.md

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W