
py-galactic Documentation

Release 0.0.2

The Galactic Organization

Feb 27, 2018

Contents:

1	Install py-galactic	3
1.1	Getting Help	3
2	Context	5
2.1	Mixins	14
2.2	MemoryContext	17
3	Types	23
3.1	Categories	23
3.2	Boolean	23
3.3	Number	23
4	Indices and tables	27
	Python Module Index	29

py-galactic is a package for studying Formal Concept Analysis.

CHAPTER 1

Install py-galactic

py-galactic requires [python 3](#), a programming language that comes pre-installed on linux and Mac OS X, and which is easily installed [on Windows](#).

Install *py-galactic* using the bash command

```
pip install py-galactic
```

To upgrade to the most recent release, use

```
pip install --upgrade py-galactic
```

pip is a script that downloads and installs modules from the Python Package Index, [PyPI](#). It should come installed with your python distribution. If you are running linux, *pip* may be bundled separately. On a Debian-based system (including Ubuntu), you can install it using

```
apt-get update
apt-get install python-pip
```

1.1 Getting Help

If you have any difficulties with *py-galactic*, please feel welcome to [file an issue](#) on github so that we can help.

The *galactic.context* package defines generic classes for using contexts:

- *Context*: a context is composed by a population and a model
- *Population* a population is a container for individuals
- *Model* a model is a container for attributes
- *Individual* an individual has an identifier and values
- *Attribute* an attribute has a name and a type

```
class galactic.context.C
    Generic subclass of the Context class
```

```
class galactic.context.P
    Generic subclass of the Population class
```

```
class galactic.context.M
    Generic subclass of the Model class
```

```
class galactic.context.X
    Generic subclass of the Individual class
```

```
class galactic.context.A
    Generic subclass of the Attribute class
```

```
class galactic.context.Context
    A Context handles a model and a population.
```

It's possible to access to the context *population* attribute or to the context *model* attribute.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
```

```
... )
>>> print(context.population)
['0', '1']
>>> print(context.model)
{'mybool': <class 'bool'>, 'myint': <class 'int'>}
```

It's possible to check if a context is not empty (both *model* and *population* are not empty) using the python builtin `bool()` function.

It's possible to get a readable representation of a context using the python builtin `str()` function.

Example

```
>>> print(context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↪ 'int'>}}
```

Example

```
>>> bool(context)
True
```

Contexts are container for individuals and attributes. It's possible to know if an individual or an attribute belongs to a context using the python `in` keyword.

Example

```
>>> context.model['mybool'] in context
True
>>> context.population['0'] in context
True
```

New in version 0.0.1.

population

Get the population for this context.

Returns the underlying population

Return type *P*

New in version 0.0.1.

model

Get the underlying model.

Returns the underlying model

Return type *M*

New in version 0.0.1.

class `galactic.context.Population`

A *Population* is a container for individuals.

It's possible to access to the population `context` attribute.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> population = context.population
>>> print(population.context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↳ 'int'>}}
```

It's possible to get a readable representation of a population.

Example

```
>>> print(population)
['0', '1']
```

It's possible to check if a population is not empty using the python builtin `bool()` function.

Example

```
>>> bool(population)
True
```

It's possible to access to an individual with its identifier using the python array access construct.

Example

```
>>> print(population['0'])
{'mybool': False, 'myint': 0}
```

It's possible to check if an individual belongs to a population using the python `in` keyword.

Example

```
>>> '0' in population
True
```

It's possible to iterate over a population using the python `for` keyword.

Example

```
>>> {ident: str(individual) for ident, individual in population.items()}
{'0': '{"mybool": False, "myint": 0}', '1': '{"mybool": False, "myint": 0}'}
```

It's possible to get the length of a population using the python builtin `len()` function.

Example

```
>>> len(population)
2
```

New in version 0.0.1.

context

Get the underlying context.

Returns the underlying context

Return type *C*

New in version 0.0.1.

model

Get the underlying model.

Returns the underlying model

Return type *M*

New in version 0.0.1.

class `galactic.context.Model`

A *Model* is a container for attributes.

It's possible to access to the model *context* attribute.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> model = context.model
>>> print(model.context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↪ 'int'>}}
```

It's possible to get a readable representation of a model using the python builtin `str()` function.

Example

```
>>> print(model)
{'mybool': <class 'bool'>, 'myint': <class 'int'>}
```

It's possible to check if a model is not empty using the python builtin `bool()` function.

Example

```
>>> bool(model)
True
```

It's possible to access to an attribute with its name using the python array access construct.

Example

```
>>> print(model['mybool'])
{'name': 'mybool', 'type': <class 'bool'>}
```

It's possible to check if an attribute belongs to a model using the python `in` keyword.

Example

```
>>> 'mybool' in model
True
```

It's possible to iterate over a model using the python `for` keyword.

Example

```
>>> {name: attribute.type for name, attribute in model.items()}
{'mybool': <class 'bool'>, 'myint': <class 'int'>}
```

It's possible to get the length of a population using the python builtin `len()` function.

Example

```
>>> len(model)
2
```

New in version 0.0.1.

context

Get the underlying context.

Returns the underlying context

Return type `C`

New in version 0.0.1.

population

Get the underlying population.

Returns the underlying population

Return type `P`

New in version 0.0.1.

class galactic.context.Individual

A *Individual* is a container for values.

It's possible to access to the individual *identifier* attribute.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> individual = context.population['0']
>>> individual.identifier
'0'
```

It's possible to access to the individual *context* attribute.

Example

```
>>> print(individual.context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↪ 'int'>}}
```

It's possible to access to the individual *model* attribute.

Example

```
>>> print(individual.model)
{'mybool': <class 'bool'>, 'myint': <class 'int'>}
```

It's possible to access to the individual *population* attribute.

Example

```
>>> print(individual.population)
['0', '1']
```

It's possible to get a readable representation of an individual using the python builtin `str()` function.

Example

```
>>> print(individual)
{'mybool': False, 'myint': 0}
```

It's possible to access to the individual values using the *value()* method.

Example

```
>>> attribute = individual.model['mybool']
>>> individual.value(attribute)
False
```

It's possible to access to the individual values using the python array access construct.

Example

```
>>> individual['mybool']
False
```

It's possible to get the length of an individual using the python builtin `len()` function.

Example

```
>>> len(individual)
2
```

It's possible to iterate over an individual using the python `for` keyword.

Example

```
>>> {name: value for name, value in individual.items()}
{'mybool': False, 'myint': 0}
```

New in version 0.0.1.

identifier

Get this individual identifier.

Returns the individual identifier

Return type `str`

New in version 0.0.1.

population

Get the underlying population.

Returns the underlying population

Return type `P`

New in version 0.0.1.

context

Get the underlying context.

Returns the underlying context

Return type `C`

New in version 0.0.1.

model

Get the underlying model.

Returns the underlying model

Return type *M*

New in version 0.0.1.

value (*attribute: A*)

Get the attribute value for this individual.

Parameters **attribute** (*A*) – the attribute

Returns the value

Return type *object*

Raises *ValueError* – if the attribute does not belong to the underlying model.

New in version 0.0.1.

class `galactic.context.Attribute`

A *Attribute* is described by a *name* and a *type*.

It's possible to access to the attribute identifier and *type* attributes.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> attribute = context.model['mybool']
>>> attribute.name
'mybool'
>>> attribute.type
<class 'bool'>
```

It's possible to access to the individual *context* attribute.

Example

```
>>> print(attribute.context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↪ 'int'>}}
```

It's possible to access to the individual *model* attribute.

Example

```
>>> print(attribute.model)
{'mybool': <class 'bool'>, 'myint': <class 'int'>}
```

It's possible to access to the individual *population* attribute.

Example

```
>>> print(attribute.population)
['0', '1']
```

It's possible to get a readable representation of an attribute using the python builtin `str()` function.

Example

```
>>> print(attribute)
{'name': 'mybool': 'type': <class 'bool'>}
```

It's possible to access to the attribute values using the `value()` method.

Example

```
>>> individual = attribute.population['0']
>>> attribute.value(individual)
False
```

It's possible to access to the attribute values using the python array access construct.

Example

```
>>> attribute['0']
False
```

It's possible to get the length of an attribute using the python builtin `len()` function.

Example

```
>>> len(attribute)
2
```

It's possible to iterate over an attribute using the python `for` keyword.

Example

```
>>> {identifier: value for identifier, value in attribute.items()}
{'0': False, '1': False}
```

New in version 0.0.1.

name

Get the attribute name.

Returns the attribute name

Return type `str`

New in version 0.0.1.

type

Get the attribute type.

Returns the attribute type

Return type `type`

New in version 0.0.1.

model

Get the underlying model.

Returns the underlying model

Return type `M`

New in version 0.0.1.

context

Get the underlying context.

Returns the underlying context

Return type `C`

New in version 0.0.1.

population

Get the underlying population.

Returns the underlying population

Return type `P`

New in version 0.0.1.

value (*individual: X*)

Get the individual value for this attribute.

Parameters **individual** (*X*) – the individual

Returns the value

Return type `object`

Raises `ValueError` – if the individual does not belong to the underlying population.

New in version 0.0.1.

2.1 Mixins

The `galactic.context.mixins` package defines mixins classes for defining new types of contexts:

- `ConcreteIndividual` for defining individuals that own their identifier as a field
- `ConcreteAttribute` for defining attributes that own their name and their type as fields
- `ContextHolder` for defining elements that own their context as a field
- `PopulationHolder` for defining elements that own their population as a field
- `ModelHolder` for defining elements that own their model as a field
- `AttributesHolder` for defining models that own their attributes as a field
- `IndividualsHolder` for defining population that own their individuals as a field

- *ValuesHolder* for defining individuals that own their values as a field

They are widely used for defining

- *MemoryContext*
- *MemoryModel*
- *MemoryPopulation*
- *MemoryAttribute*
- *MemoryIndividual*

in the *galactic.context.memory* package.

New in version 0.0.1.

class *galactic.context.mixins.ConcreteIndividual* (**kwargs)

The *ConcreteIndividual* class is a mixin used in subclassing the *Individual* class for storing their identifier as a field.

New in version 0.0.1.

__init__ (**kwargs)

Initialise an individual.

Keyword Arguments **identifier** (*str*) – the individual identifier

New in version 0.0.1.

identifier

Get the individual identifier.

Returns the individual identifier

Return type *str*

New in version 0.0.1.

class *galactic.context.mixins.ConcreteAttribute* (**kwargs)

The *ConcreteAttribute* class is a mixin used in subclassing the *Attribute* class for storing their name and their type as a class.

New in version 0.0.1.

__init__ (**kwargs)

Initialise an attribute

Keyword Arguments

- **name** (*str*) – the attribute name
- **type** (*type*) – the attribute type

New in version 0.0.1.

name

Get the attribute name.

Returns the attribute name

Return type *str*

New in version 0.0.1.

type

Get the attribute type.

Returns the attribute type

Return type `type`

New in version 0.0.1.

class `galactic.context.mixins.ContextHolder` (***kwargs*)

The `ContextHolder[C]` is a mixin used for storing an element context as a field.

It's a generic class that depends of a `Context` subclass C.

New in version 0.0.1.

__init__ (***kwargs*)

Initialise an element by setting its context.

Keyword Arguments `context` (C) – the context

New in version 0.0.1.

context

Get the context.

Returns the context

Return type C

New in version 0.0.1.

class `galactic.context.mixins.PopulationHolder` (***kwargs*)

The `PopulationHolder[P]` is a mixin used for storing an element population as a field.

It's a generic class that depends of a `Population` subclass P.

New in version 0.0.1.

__init__ (***kwargs*)

Initialise an element by setting its population.

Keyword Arguments `population` (P) – the population

New in version 0.0.1.

population

Get the population.

Returns the population

Return type P

New in version 0.0.1.

class `galactic.context.mixins.ModelHolder` (***kwargs*)

The `ModelHolder[M]` class is a mixin used for storing an element model as a field.

It's a generic class that depends of a `Model` subclass M.

New in version 0.0.1.

__init__ (***kwargs*)

Initialise an element by setting its model.

Keyword Arguments `model` (M) – the model

New in version 0.0.1.

model

Get the model.

Returns the model

Return type M

New in version 0.0.1.

```
class galactic.context.mixins.AttributesHolder (**kwargs)
```

The *AttributesHolder*[A] class is a mixin used for storing the model attributes in memory.

It's a generic class that depends of an *Attribute* subclass A.

New in version 0.0.1.

```
__init__ (**kwargs)
```

Initialise an attribute holder.

Keyword Arguments **attributes** (*Iterable*[A]) – the attributes

New in version 0.0.1.

```
class galactic.context.mixins.IndividualsHolder (**kwargs)
```

The *IndividualsHolder*[X] class is a mixin used for storing the population individuals in memory.

It's a generic class that depends of an *Individual* subclass X.

New in version 0.0.1.

```
__init__ (**kwargs)
```

Initialise an individuals holder.

Keyword Arguments **individuals** (*Iterable*[A]) – the individuals

New in version 0.0.1.

```
class galactic.context.mixins.ValuesHolder (**kwargs)
```

The *ValuesHolder*[A] class is a mixin for storing the individual values in memory.

It's a generic class that depends of an *Attribute* subclass A.

New in version 0.0.1.

```
__init__ (**kwargs)
```

Initialise a values holder.

Keyword Arguments **values** (*Mapping*[str, object]) – the initial (name, value) pairs

New in version 0.0.1.

2.2 MemoryContext

The *galactic.context.memory* module give the ability to define *Context* that resides in memory.

```
class galactic.context.memory.MemoryContext (**kwargs)
```

The *MemoryContext* class is designed to define contexts in memory. It inherits of all the behavior from the *Context* class and allows direct creation and modification of a context.

It's possible to create a context without nothing.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext()
>>> print(context)
{'population': [], 'model': {}}
```

It's possible to create a context specifying the model definition.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(definition={'mybool': bool, 'myint': int})
>>> print(context)
{'population': [], 'model': {'mybool': <class 'bool'>, 'myint': <class 'int'>}}
```

It's possible to create a context specifying the model definition and the list of individual identifiers.

Example

```
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> print(context)
{'population': ['0', '1'], 'model': {'mybool': <class 'bool'>, 'myint': <class
↪ 'int'>}}
```

It's possible to create a context specifying the model definition and the individual values.

Example

```
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals={'0': {'mybool': True}, '1': {'myint': 1}}
... )
>>> {ident: str(context.population[ident]) for ident in context.population}
{'0': '{"mybool": True, "myint": 0}', '1': '{"mybool": False, "myint": 1}'}
```

New in version 0.0.1.

__init__(**kwargs)

Initialise a context in memory.

Keyword Arguments

- **definition** (Mapping[str, type]) – definition of the context by a mapping from name of attributes to their type
- **individuals** (Union[Iterable[str], Mapping[str, Mapping[str, object]]]) – initial iterable of individual identifiers or a mapping from individual identifiers to individual values

Raises

- `KeyError` – if an attribute is not in the definition
- `ValueError` – if a value does not correspond to an attribute type
- `TypeError` – if the definition or if the individuals parameter are not of the correct type

New in version 0.0.1.

```
class galactic.context.memory.MemoryModel (context: galactic.context.memory.MemoryContext, definition: typing.Mapping[str, typing.type])
```

The `MemoryModel` class is designed to define models that resides in memory. It inherits of all the behavior from the `Model` class.

It's possible to change or to set attribute values.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> model = context.model
>>> model['mybool'] = int
>>> model['myint2'] = int
>>> print(context.population['0'])
{'mybool': 0, 'myint': 0, 'myint2': 0}
```

It's possible to delete an attribute using its name.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> model = context.model
>>> del model['mybool']
>>> {ident: str(context.population[ident]) for ident in context.population}
{'0': '{"myint": 0}', '1': '{"myint": 0}'}
```

New in version 0.0.1.

```
__init__(context: galactic.context.memory.MemoryContext, definition: typing.Mapping[str, typing.type])
```

Initialise a model in memory.

Parameters

- **context** (`MemoryContext`) – the underlying context
- **definition** (`Mapping[str, type]`) – the attributes definition

New in version 0.0.1.

```
class galactic.context.memory.MemoryPopulation (context: galactic  
                                                    tic.context.memory.MemoryContext,  
                                                    identifiers: typing.Iterable[str])
```

The *MemoryPopulation* class is designed to define populations that resides in memory. It inherits of all the behavior from the *Population* class.

It's possible to change or to set individual values.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> population = context.population
>>> population['0'] = {'mybool': True}
>>> population['2'] = {'myint': 1}
>>> print(population['0'])
{'mybool': True, 'myint': 0}
>>> print(population['2'])
{'mybool': False, 'myint': 1}
```

It's possible to delete an individual using its identifier.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> population = context.population
>>> del population['0']
>>> {ident: str(context.population[ident]) for ident in context.population}
{'1': '{"mybool": False, "myint": 0}'}
```

New in version 0.0.1.

```
__init__ (context: galactic.context.memory.MemoryContext, identifiers: typing.Iterable[str])  
    Initialise a population in memory.
```

Parameters

- **context** (*MemoryContext*) – the underlying context
- **identifiers** (*Iterable[str]*) – an iterable of identifiers

New in version 0.0.1.

```
class galactic.context.memory.MemoryIndividual (population: galactic  
                                                    tic.context.memory.MemoryPopulation,  
                                                    identifier: str)
```

The *MemoryIndividual* is designed to define individuals that resides in memory. It inherits of all the behavior from the *Individual* class.

It's possible to modify a value for an individual using an attribute name.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> individual = context.population['0']
>>> individual['mybool'] = True
>>> individual['myint'] = 1
>>> {ident: str(context.population[ident]) for ident in context.population}
{'0': '{"mybool": True, "myint": 1}', '1': '{"mybool": False, "myint": 0}'}
```

New in version 0.0.1.

__init__ (*population: galactic.context.memory.MemoryPopulation, identifier: str*)
 Initialise an individual.

Parameters

- **population** (*MemoryPopulation*) – the population
- **identifier** (*str*) – the individual identifier

New in version 0.0.1.

class galactic.context.memory.**MemoryAttribute** (*model: galactic.context.memory.MemoryModel, name: str, cls: type*)

The *MemoryAttribute* is designed to define attributes that resides in memory. It inherits of all the behavior from the *Attribute* class.

It's possible to modify a value for an attribute using an individual identifier.

Example

```
>>> from galactic.context.memory import MemoryContext
>>> context = MemoryContext(
...     definition={'mybool': bool, 'myint': int},
...     individuals=['0', '1']
... )
>>> attribute = context.model['myint']
>>> attribute['0'] = 3
>>> attribute['1'] = 4
>>> {ident: str(context.population[ident]) for ident in context.population}
{'0': '{"mybool": False, "myint": 3}', '1': '{"mybool": False, "myint": 4}'}
```

New in version 0.0.1.

__init__ (*model: galactic.context.memory.MemoryModel, name: str, cls: type*)
 Initialise an attribute.

Parameters

- **model** (*MemoryModel*) – the underlying model
- **name** (*str*) – the attribute name
- **cls** (*type*) – the attribute type

New in version 0.0.1.

A type is associated to an *Attribute*. A type is valid if it is a class that can be called

- without argument: in that case a default value is returned for that type
- with an argument: in that case, the class try to convert it to an acceptable value for that type

For example the `int` and `bool` classes are acceptable types as they accept to be called without arguments or with an argument which is converted to the desired type.

3.1 Categories

3.2 Boolean

3.3 Number

This module one type useful in Context values:

- *ImpreciseFloat* for representing imprecise numbers as intervals.

class `galactic.type.number.ImpreciseFloat` (***kwargs*)

The *ImpreciseFloat* class ca be used to represent intervals on the real line.

__init__ (***kwargs*)

Initialise an *ImpreciseFloat*.

Keyword Arguments

- **inf** (*float*) – the lower limit of the interval (default to `-math.inf`)
- **sup** (*float*) – the upper limit of the interval (default to `math.inf`)

New in version 0.0.2.

inf

Get the lower limit.

Returns the lower limit**Return type** `float`

New in version 0.0.2.

sup

Get the upper limit.

Returns the upper limit**Return type** `float`

New in version 0.0.2.

isdisjoint (*other*: `galactic.type.number.ImpreciseFloat`) → `bool`

Return True if the imprecise float has no elements in common with the other. Imprecise floats are disjoint if and only if their intersection is the empty imprecise float.

Parameters *other* (`ImpreciseFloat`) – the other imprecise float**Returns** True if the imprecise float is disjoint from the other**Return type** `bool`

New in version 0.0.2.

issubset (*other*: `galactic.type.number.ImpreciseFloat`) → `bool`

Test if the imprecise float is included (or equal) to the other.

Parameters *other* (`ImpreciseFloat`) –**Returns** True if this imprecise float is included or equal to the other**Return type** `bool`

New in version 0.0.2.

issuperset (*other*: `galactic.type.number.ImpreciseFloat`) → `bool`

Test if the imprecise float includes (or is equal to) the other.

Parameters *other* (`ImpreciseFloat`) –**Returns** True if this imprecise float includes (or is equal to) the other**Return type** `bool`

New in version 0.0.2.

union (**others*) → `galactic.type.number.ImpreciseFloat`

Compute the union between this imprecise float and the others.

Parameters **others* – Variable length argument list**Returns** the union between this imprecise float and the others**Return type** `ImpreciseFloat`

New in version 0.0.2.

intersection (**others*) → `galactic.type.number.ImpreciseFloat`

Compute the intersection between this imprecise float and the others.

Parameters **others* – Variable length argument list**Returns** the intersection between this imprecise float and the others

Return type *ImpreciseFloat*

New in version 0.0.2.

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

g

- `galactic.context`, [5](#)
- `galactic.context.memory`, [17](#)
- `galactic.context.mixins`, [14](#)
- `galactic.type`, [23](#)
- `galactic.type.number`, [23](#)

Symbols

[__init__\(\)](#) (galactic.context.memory.MemoryAttribute method), 21
[__init__\(\)](#) (galactic.context.memory.MemoryContext method), 18
[__init__\(\)](#) (galactic.context.memory.MemoryIndividual method), 21
[__init__\(\)](#) (galactic.context.memory.MemoryModel method), 19
[__init__\(\)](#) (galactic.context.memory.MemoryPopulation method), 20
[__init__\(\)](#) (galactic.context.mixins.AttributesHolder method), 17
[__init__\(\)](#) (galactic.context.mixins.ConcreteAttribute method), 15
[__init__\(\)](#) (galactic.context.mixins.ConcreteIndividual method), 15
[__init__\(\)](#) (galactic.context.mixins.ContextHolder method), 16
[__init__\(\)](#) (galactic.context.mixins.IndividualsHolder method), 17
[__init__\(\)](#) (galactic.context.mixins.ModelHolder method), 16
[__init__\(\)](#) (galactic.context.mixins.PopulationHolder method), 16
[__init__\(\)](#) (galactic.context.mixins.ValuesHolder method), 17
[__init__\(\)](#) (galactic.type.number.ImpreciseFloat method), 23

A

[A](#) (class in galactic.context), 5
[Attribute](#) (class in galactic.context), 12
[AttributesHolder](#) (class in galactic.context.mixins), 17

C

[C](#) (class in galactic.context), 5
[ConcreteAttribute](#) (class in galactic.context.mixins), 15
[ConcreteIndividual](#) (class in galactic.context.mixins), 15

[Context](#) (class in galactic.context), 5
[context](#) (galactic.context.Attribute attribute), 14
[context](#) (galactic.context.Individual attribute), 11
[context](#) (galactic.context.mixins.ContextHolder attribute), 16
[context](#) (galactic.context.Model attribute), 9
[context](#) (galactic.context.Population attribute), 8
[ContextHolder](#) (class in galactic.context.mixins), 16

G

[galactic.context](#) (module), 5
[galactic.context.memory](#) (module), 17
[galactic.context.mixins](#) (module), 14
[galactic.type](#) (module), 23
[galactic.type.number](#) (module), 23

I

[identifier](#) (galactic.context.Individual attribute), 11
[identifier](#) (galactic.context.mixins.ConcreteIndividual attribute), 15
[ImpreciseFloat](#) (class in galactic.type.number), 23
[Individual](#) (class in galactic.context), 9
[IndividualsHolder](#) (class in galactic.context.mixins), 17
[inf](#) (galactic.type.number.ImpreciseFloat attribute), 23
[intersection\(\)](#) (galactic.type.number.ImpreciseFloat method), 24
[isdisjoint\(\)](#) (galactic.type.number.ImpreciseFloat method), 24
[issubset\(\)](#) (galactic.type.number.ImpreciseFloat method), 24
[issuperset\(\)](#) (galactic.type.number.ImpreciseFloat method), 24

M

[M](#) (class in galactic.context), 5
[MemoryAttribute](#) (class in galactic.context.memory), 21
[MemoryContext](#) (class in galactic.context.memory), 17
[MemoryIndividual](#) (class in galactic.context.memory), 20
[MemoryModel](#) (class in galactic.context.memory), 19

MemoryPopulation (class in galactic.context.memory),
19

Model (class in galactic.context), 8

model (galactic.context.Attribute attribute), 14

model (galactic.context.Context attribute), 6

model (galactic.context.Individual attribute), 11

model (galactic.context.mixins.ModelHolder attribute),
16

model (galactic.context.Population attribute), 8

ModelHolder (class in galactic.context.mixins), 16

N

name (galactic.context.Attribute attribute), 13

name (galactic.context.mixins.ConcreteAttribute at-
tribute), 15

P

P (class in galactic.context), 5

Population (class in galactic.context), 6

population (galactic.context.Attribute attribute), 14

population (galactic.context.Context attribute), 6

population (galactic.context.Individual attribute), 11

population (galactic.context.mixins.PopulationHolder at-
tribute), 16

population (galactic.context.Model attribute), 9

PopulationHolder (class in galactic.context.mixins), 16

S

sup (galactic.type.number.ImpreciseFloat attribute), 24

T

type (galactic.context.Attribute attribute), 13

type (galactic.context.mixins.ConcreteAttribute at-
tribute), 15

U

union() (galactic.type.number.ImpreciseFloat method),
24

V

value() (galactic.context.Attribute method), 14

value() (galactic.context.Individual method), 12

ValuesHolder (class in galactic.context.mixins), 17

X

X (class in galactic.context), 5