# Objects Creation: FactBases



This document introduces Py4Graal environments and the creation of fact bases using DLGP syntax. It explains the relationship between fact bases, facts, and atoms, demonstrates how variables are handled, and clarifies the semantics of commas and periods in DLGP.

All code examples shown here can be found in the accompanying Python file: ex1_objects_creation.py.

## Creating a Graal Environment

Py4Graal objects are created inside a *Graal environment*. Each environment manages its own set of fact bases, rule bases, and queries, and is backed by the Java Py4Graal server.

To start, create a `Graal` instance:

```
from py4graal.graal import Graal

graal = Graal()
```

Creating a Graal object will:

- Launch the Java Py4Graal server (if not already running).
- Create a new environment on the server.
- Tie this Python object to that environment.

All objects you create through this graal instance will reside in its environment and cannot be shared with other Graal instances.

## Creating a factbase

A fact base stores a set of logical *facts* in DLGP syntax, a Prolog-like format used by Py4Graal. Each fact contains one or more *atoms* (written as a predicate followed by its arguments), separated by commas, ending with a period. For example, the atoms:

- `human(socrates).` → declares that Socrates is human.
- `parent(sophroniscus, socrates).` → declares that Sophroniscus is the parent of Socrates.

You can combine multiple facts in a single DLGP string. The following DLGP string considers two facts (the first one containing two atoms), and the `factbase` created combines them all in a single structure:

```
factbase_dlgp = "human(socrates), human(sophroniscus). parent(sophroniscus,
socrates)."
factbase = graal.create_factbase(factbase_dlgp)
```

Under the hood, this creates a Java fact base in the current Graal environment and returns a Python object that acts as a reference to it.

You can inspect the internal state of the Python object via `__dict__`:

```
print(factbase.__dict__)
```

Typical output might look like:

```
{'env': 'ENV:1753723909248_2', 'name': 'FB:1753723909410_3'}
```

Indeed, the `factbase` python object contains only the name of the object in an environment of the java server. In particular, it knows nothing about the atoms it contains.

When you print the object itself:

```
print(factbase)
```

Py4Graal asks the Java side to export the fact base back to DLGP, producing:

```
parent(sophroniscus, socrates), human(sophroniscus), human(socrates).
```

## Facts, variables, and commas

Contrary to what might happen, for instance, in a relational database, atoms in a factbase may have *variables* as arguments. In DLGP, constants always begin with a lowercase, while variables begin with an uppercase.

The following fact asserts that Sophroniscus has a(n unknown) parent.

```
factbase_dlgp2 = "parent(X, socrates)."
factbase2 = graal.create_factbase(factbase_dlgp2)
```

Please note that a variable is local to the fact it appears in. This is why the Java Server automatically rewrites them, to ensure that variables in different facts are treated as independent, even if they share the same name.

This behaviour is shown by printing `factbase2`:

```
print(factbase2)
```

That produces the following, where `Graal_EE1` is a generated variable name ensuring uniqueness inside the fact base.

```
parent(Graal_EE1, socrates).
```

With that in mind, we can now understand the subtle difference between periods and commas when creating a factbase.

```
print(graal.create_factbase("parent(X, socrates), parent(X, patrocles)."))
print(graal.create_factbase("parent(X, socrates). parent(X, patrocles)."))
```

In the first version, the two atoms are part of the same fact. This asserts that Socrates and Patrocles share a common parent (a single variable X).

In the second version, the atoms are in separate facts. This asserts that Socrates has some parent and Patrocles has some parent, but they may not be the same individual.

The output makes the difference clear:

```
parent(Graal_EE2, socrates), parent(Graal_EE2, patrocles).
parent(Graal_EE4, socrates), parent(Graal_EE5, patrocles).
```

Of course, when the facts contain only constants, there is no difference when separating atoms with periods and commas.