



---

# INTERPRETE PROLOG

---



Elaborado por:

Christopher Vásquez.

Bryan Blanco.

## Contenido

Resumen Ejecutivo.....	2
Propósito.....	3
Descripción del proyecto:.....	3
Requerimientos:.....	3
Funcionalidades.....	3
Descripción detallada del programa.....	4
Decisiones de diseño:.....	4
Lenguajes de Programación usados:.....	4
Librerías usadas:.....	4
Problemas encontrados.....	4
Manual de Usuario.....	5
Conclusión personal.....	7

DE 2014

LENGUAJES DE PROGRAMACIÓN  
TI 3404



## Resumen Ejecutivo

El proyecto consiste en crear un intérprete que tenga la misma funcionalidad que el IDE SWI de Prolog.

Esto se pretende lograr desde el lenguaje Python definiendo una serie de instancias que permitan identificar cuando se trata de un hecho, una regla o una consulta.

Las reglas son las encargadas de definir el orden de los caracteres y su combinación lógica de acuerdo con el lenguaje de programación a utilizar, de esta forma se evaluará la correcta definición de los hechos por parte del usuario.

Posteriormente los hechos serán almacenados en memoria generando una base de conocimientos, la cual será utilizada para verificar las consultas que el usuario realice.

**¿Qué es lo que vamos hacer?** Desarrollar un intérprete para el lenguaje de programación Prolog implementado en el lenguaje Python.

**¿Quiénes lo vamos hacer?** Christopher Vásquez y Bryan Blanco Morales. Estudiantes de Administración de Tecnologías de Información del Instituto Tecnológico de Costa Rica.

**¿Cómo lo vamos hacer?** Creando un interprete que evalúe hechos, reglas y consultas acerca del lenguaje.

**¿Por qué lo vamos hacer?** Para generar un conocimiento a profundidad acerca de este lenguaje, demostrando que no solo con el IDE de Prolog es posible programar en este lenguaje.

## Propósito

### Descripción del proyecto:

Como se mencionó anteriormente el proyecto consiste en un intérprete de prolog el cual, a partir de reglas y hechos, permite evaluar las consultas que el usuario realice y posteriormente dar un resultado y mostrarle al usuario si la consulta hecha es verdadera o falsa.

### Requerimientos:

Para poder ejecutar el intérprete correctamente se deben de tener en cuenta los siguientes requerimientos:

- Python 2.7 o posterior.
- Librería **sys** (módulo que provee acceso a funciones y objetos mantenidos por el intérprete.)
- Librería **copy** (Este módulo proporciona operaciones de copia genéricas ya sea superficial o profunda).
- Librería **re** (modulo estándar que permite reemplazar múltiples substrings de una cadena de forma sencilla)
- Trabajar en Linux de 32bits.

## Funcionalidades

El proyecto cuenta con dos funcionalidades definidas las cuales son: definición de predicados y módulo de consulta.

A continuación se especifica cada una de las funciones.

**Definición de predicados:**

Para la definición de predicados se tomara en cuenta que toda clausula que no posea un “?” al final de la clausula sera tomada como una definición de predicados.

Los predicados serán evaluados para verificar que su sintaxis sea correcta y posteriormente serán almacenados en una base de conocimientos.

**Módulo de consulta:**

Al igual que en la definición de predicados se debe validar que la consulta tenga una sintaxis correcta. Tomando en cuenta que si gusta hacer una consulta debe ingresar el “?” al final de la clausula.

Posteriormente el motor de inferencia hará su trabajo recorriendo la base de conocimientos creada previamente y de esta manera determinar el resultado correcto y devolverlo al usuario.

## Descripción detallada del programa

### Decisiones de diseño:

En un principio los requerimientos fueron tomados en cuenta basados en un desarrollo con el lenguaje c++. Pero en el transcurso se decidió por cuestiones varias implementarlo en el lenguaje de Python, creando nuevamente una estructura que diera paso a la solución del problema planteado anteriormente.

Para el desarrollo del proyecto se crearon las siguientes funciones:

- unify (srcTerm, srcEnv, destTerm, destEnv) : seria la funcion que valida la unificación. Toma cuatro argumentos; un term fuente y el entorno actual, y un term de destino y el entorno destino. Si la unificación es exitosa el entorno de destino puede ser actualizada. Ambos ambientes son sólo diccionarios con enlaces de variables. Al unificar devolverá 1 si tiene éxito, de lo contrario 0.
- def search (term) : seria la funcion encargada no solo de buscar los hechos y reglas, sino que da respuesta al usuario y manejo de backtracking.  
La búsqueda se da mediante una pila de metas. La búsqueda del Term se convierte en una regla que luego se utiliza para crear un objeto Goal que es a su vez usada para inicializar la pila.
- fatal (mesg) : es la funcion encargada del manejo de errores
- procFile (f, prompt) : seria la funcion que se encarga del todo el proceso que se le hace a las clausulas ingresadas por el usuario
- los métodos \_\_init\_\_ utiliza la función de 'dividir' en strings para fragmentar los campos en caracteres y eliminar espacios en blanco .
-

también se definieron las clases que interactúan con estas funciones descritas:

- La clase Goal : en este caso define la clase “Meta” y se puede definir que una meta se puede interpretar como una regla, en donde las su-metas pueden ser unificados. En el caso de un diagrama de árbol se encontraría arriba, y cada rama o nodo representaría una meta. A excepción de la “búsqueda original” que sería encontrar propiamente en la actual, cada meta posee una meta principal. Los atributos de un objeto Meta son la regla, los padres, el medio y un índice que indica que necesita entrar en una submeta de la regla para ser unificado.
- La clase Regla: Una regla se compone de un Head Term ( term de cabecera) y una lista de Terms tipo Goal (lista de metas).
- La clase Term: Este programa está diseñado y construido para procesar sentencias (y consultas) de los archivos en la línea de comandos y luego interactuar de la misma manera con el usuario para dar una respuesta.

## Lenguajes de Programación usados:

Los lenguajes utilizados son:

**Prolog:** presenta un tipo de paradigma lógico el cual trabaja con predicados, hechos y reglas, esto permite un proceso de inferencia. Este será el lenguaje que podrá ser ejecutado en el intérprete creado en Python.

**Python:** Aquí es donde se realizó el proyecto creando funciones que permiten determinar cuáles serán las reglas permitidas para la declaración de los hechos y posteriormente realizar las consultas deseadas

## Librerías usadas:

Las librerías que se utilizaron fueron:

**Sys:** módulo que provee acceso a funciones y objetos mantenidos por el intérprete.

**Copy:** este módulo proporciona operaciones de copia genéricas ya sea superficial o profunda.

**Re:** modulo estándar que permite reemplazar múltiples substrings de una cadena de forma sencilla.

## Problemas encontrados

1. El primer problema que tuvimos fue la mala organización por parte del grupo ya que teníamos diferentes responsabilidades, tanto laborales como de otros cursos y choque de horarios por lo que se nos dificultó reunirnos para trabajar juntos.
2. Built-in-predicates no se definieron.
3. En un principio se hicieron esfuerzos por realizar el proyecto en C++ y a pesar de que se avanzó, al final no logramos hacer que tanto el generador de parser como las clases y funciones creadas compilaran bien y funcionara el trabajo realizado por lo que tuvimos que empezar desde cero con Python.

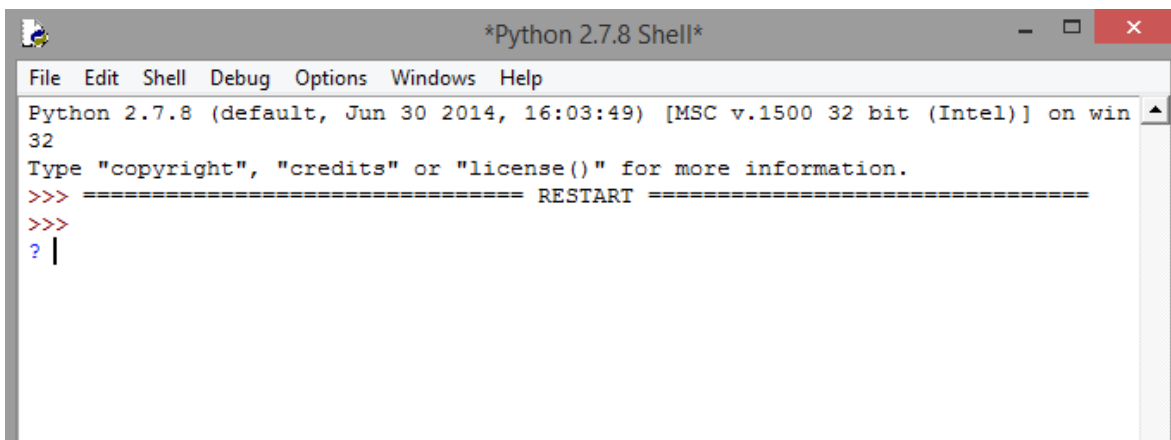


## Manual de Usuario

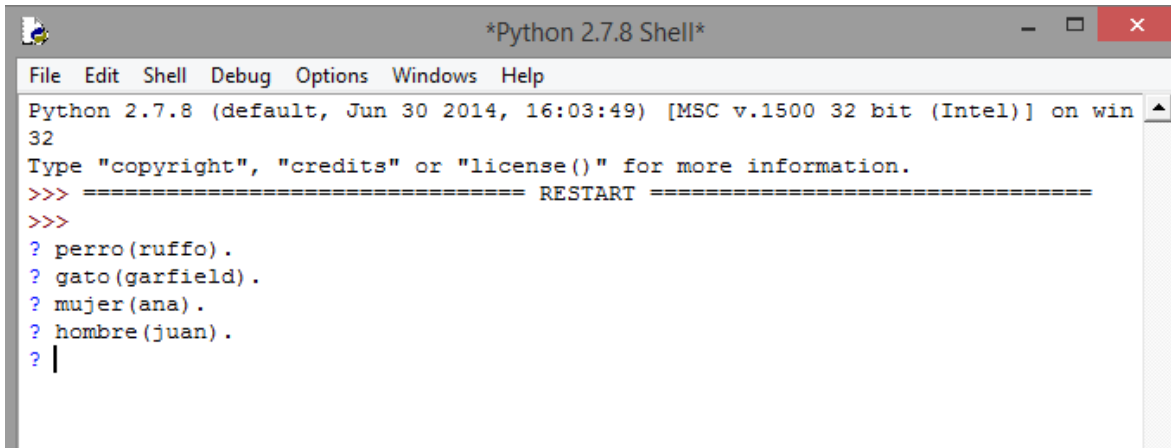
1. Instale el programa Python 2.7 en su computadora. (En caso de tenerlo omita esta instrucción).
2. Si desea puede instalar el IDLE de Python2.7 para generar el código desde allí, esto lo podrá encontrar en el centro de software de Ubuntu, ingrese y siga los pasos de instalación.



3. Ejecute el archivo llamado prolog.py. Al ejecutarlo podrá ver esta pantalla:

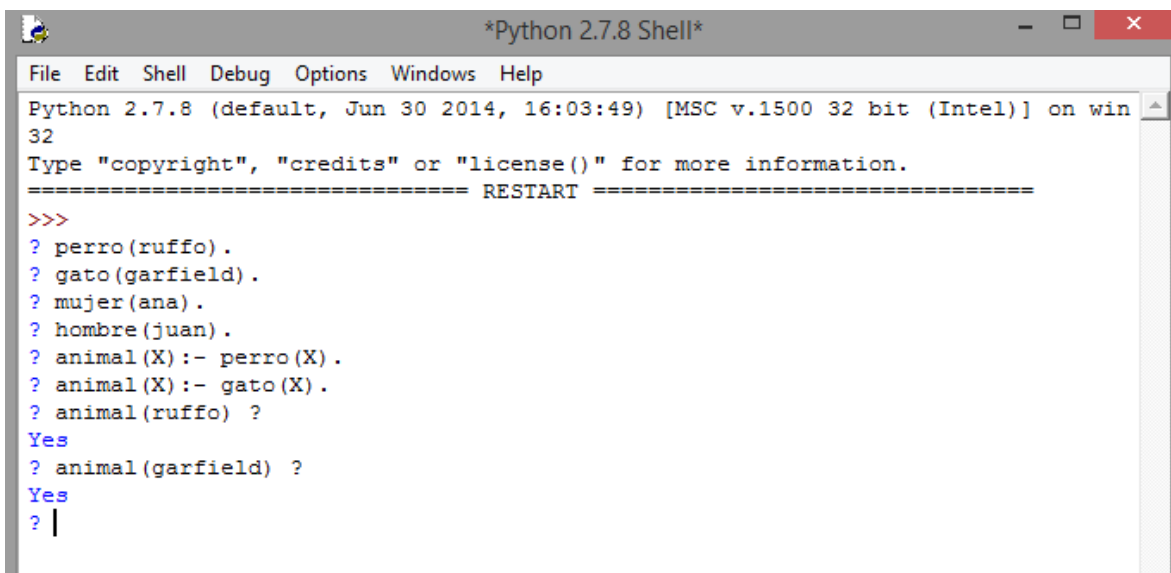


4. Aquí podrá definir todos los hechos o predicados que desee que sean agregados a la base de conocimientos. Los predicados se definen como se muestra en la siguiente imagen.



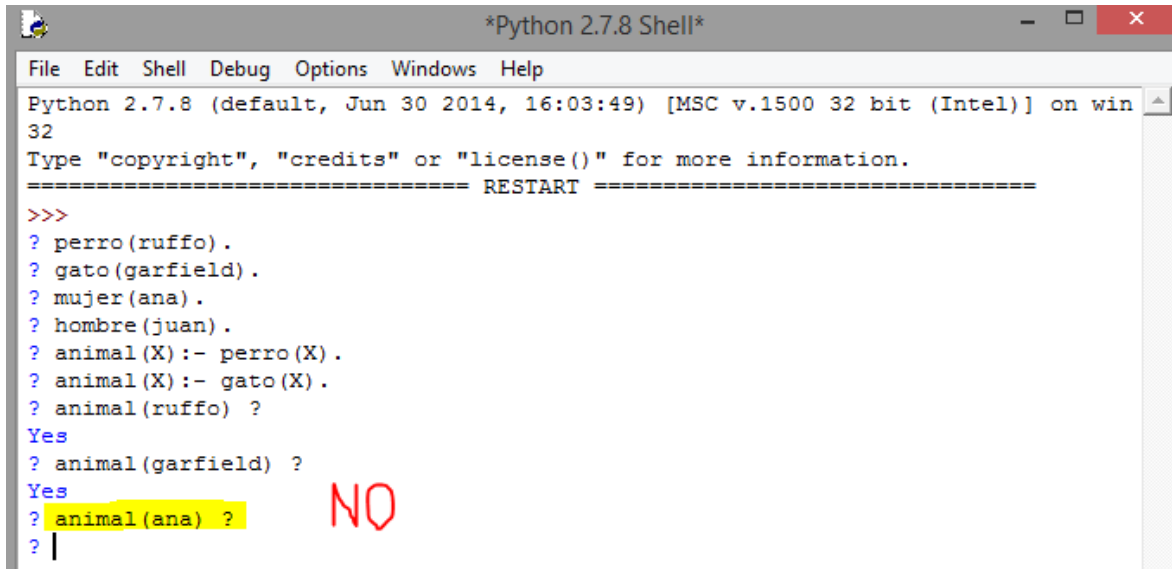
```
*Python 2.7.8 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
? perro(ruffo).
? gato(garfield).
? mujer(ana).
? hombre(juan).
? |
```

5. Así que el usuario tiene la base de conocimientos creada con los respectivos hechos, puede proceder a realizar las consultas que desee. La consulta debe finalizar un signo de pregunta “?” como se muestra en la siguiente imagen:



```
*Python 2.7.8 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
===== RESTART =====
>>>
? perro(ruffo).
? gato(garfield).
? mujer(ana).
? hombre(juan).
? animal(X):- perro(X).
? animal(X):- gato(X).
? animal(ruffo) ?
Yes
? animal(garfield) ?
Yes
? |
```

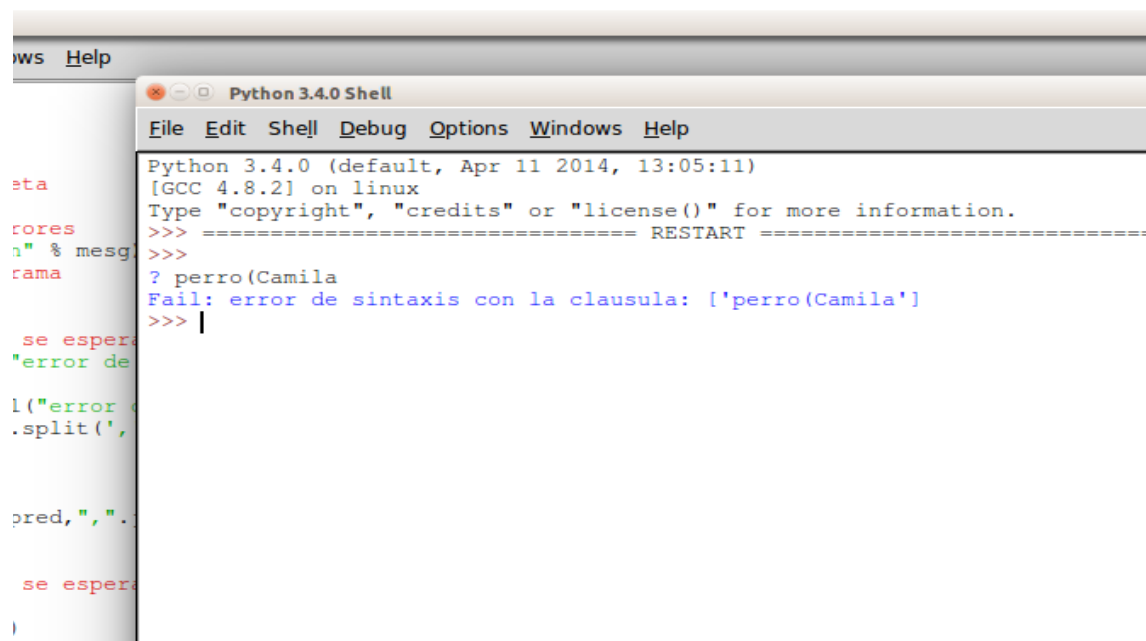
6. En caso de la consulta sea errónea o no exista en la base de conocimientos el programa solicitará una nueva consulta con un espacio en blanco como se muestra a continuación:



```

Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
===== RESTART =====
>>>
? perro(ruffo).
? gato(garfield).
? mujer(ana).
? hombre(juan).
? animal(X):- perro(X).
? animal(X):- gato(X).
? animal(ruffo) ?
Yes
? animal(garfield) ?
Yes
? animal(ana) ?
? |
  
```

7. El manejo de Errores, utilizado para la comprobación de la sintaxis básica.



```

Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
? perro(Camila
Fail: error de sintaxis con la clausula: ['perro(Camila']
>>> |
  
```

## Conclusión personal

Tomando en cuenta tanto la parte positiva como la negativa que tuvo este proyecto para nosotros, podemos concluir que para un futuro proyecto necesitamos una mejor organización y compromiso, al mismo tiempo tener conocimiento sobre el campo en el que se está trabajando e investigar sobre las herramientas que nos pueden ser de ayuda para concluir con éxito el trabajo realizado y no tener que empezar desde cero como nos ocurrió aquí.

Sin embargo pensamos que a pesar de todo el resultado fue positivo y logramos aprender lecciones que en un futuro serán de gran ayuda, no solo para este curso, sino para todos los demás y la vida en general.