# Quickieee Prolog

Bruno Henriques

## Goals

- Essence of PL
- Prolog PL
- Projecto LP

## Programming Languages

# Programming Paradigms

**IMPERATIVE LANGUAGES**

- **Procedural**
  - C
  - C++
  - Python
- **Object-Oriented**
  - C++
  - Java
  - Python

**DECLARATIVE LANGUAGES**

- **Functional**
  - Lisp
  - Python
- **Logic**
  - Prolog
- **Constraint**
  - Prolog
- **Dataflow**
  - TensorFlow

WORK WITH THE LANGUAGE,
NOT AGAINST IT.

- Adolf Hitler

## Prolog

# Prolog

- Installation
- Data Types
- Code Structure
- Code Examples
- Debugging

## Installing Prolog

**Linux/Debian**

```
$   apt-add-repository ppa:swi-prolog/stable
$   apt update
$   apt install swi-prolog
```

**MacOS**

```
$   brew install swi-prolog
```

## Prolog Data Types

- Variables (uppercase letters)
- Atoms (lowercase letters, “strings”)
- Numbers
- Lists (& Tuples)

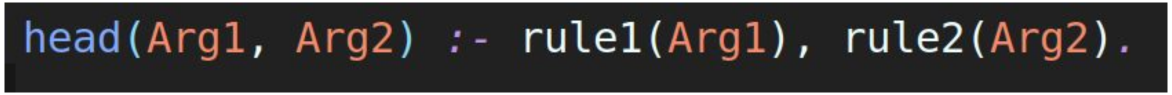
## Prolog Operators

| | |
|---|---|
| `+` | Addition |
| `-` | Subtraction |
| `*` | Multiplication |
| `/` | Division |
| `<, =<` | Less than (or equal) |
| `>, >=` | Greater than (or equal) |
| `mod` | Modulus |

| | |
|---|---|
| `=` | Variable assignment |
| `is` | Variable assignment (eval) |
| `=` | Equal to |
| `\=` | Not Equal to |
| `==` | Equal to (with eval) |
| `\==` | Not Equal to (with eval) |

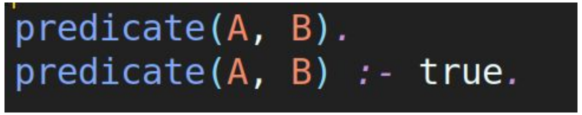
## Rules and Facts

**Rule:** head :- body.



```
head(Arg1, Arg2) :- rule1(Arg1), rule2(Arg2).
```

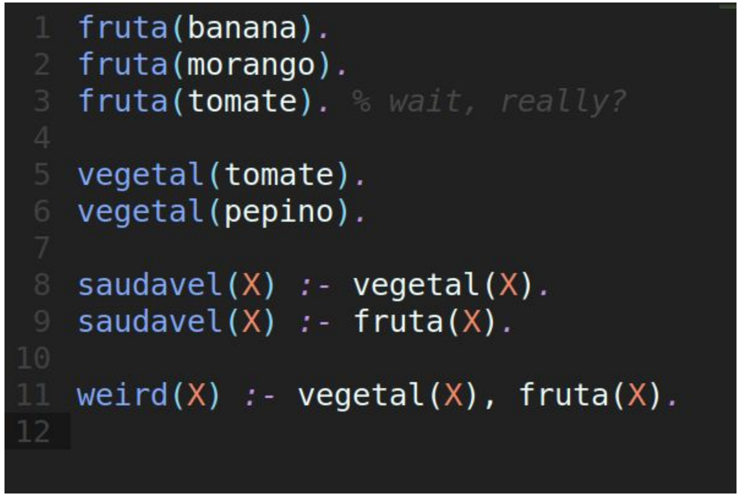
- Head “like” function signature. Body “like” function body
- “Functions” in Prolog called **predicates**
- Predicates can only evaluated to **true** or **false**
- Rules without body are called **Facts**.



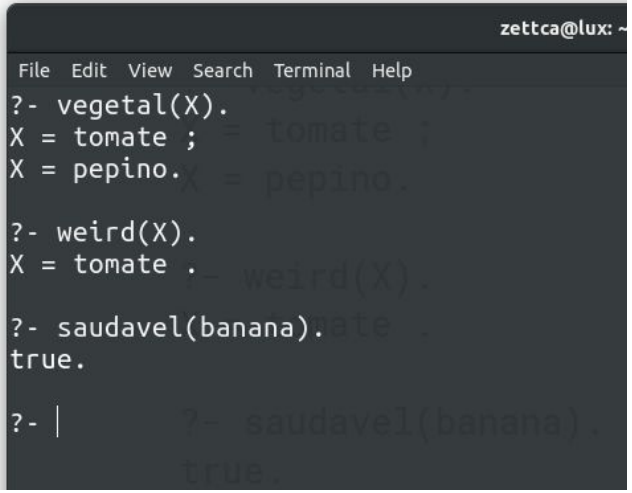
```
predicate(A, B).
predicate(A, B) :- true.
```

## Project Structure

- Write **facts** and **rules** on a file (*code.pl*)
- Load the file and ask questions



```prolog
fruta(banana).
fruta(morango).
fruta(tomate). % wait, really?

vegetal(tomate).
vegetal(pepino).

saudavel(X) :- vegetal(X).
saudavel(X) :- fruta(X).

weird(X) :- vegetal(X), fruta(X).
```

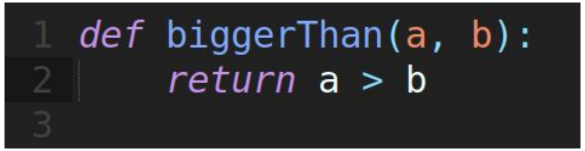


```
?- vegetal(X).
X = tomate ;
X = pepino.

?- weird(X).
X = tomate .

?- saudavel(banana).
true.

?- 
```

## Python vs Prolog

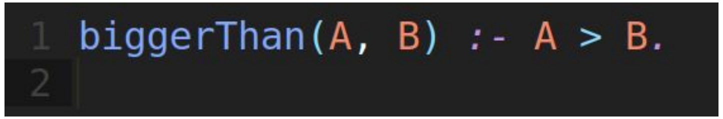
## Basic Comparison

**Python**



```python
def biggerThan(a, b):
    return a > b
```

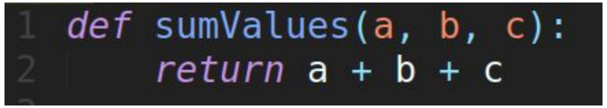
**Prolog**



```prolog
biggerThan(A, B) :- A > B.
```

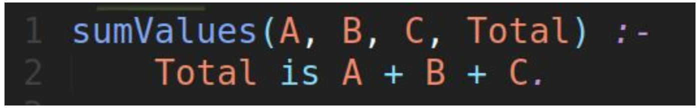
# “Returning” a Number

**Python**



```python
def sumValues(a, b, c):
    return a + b + c
```

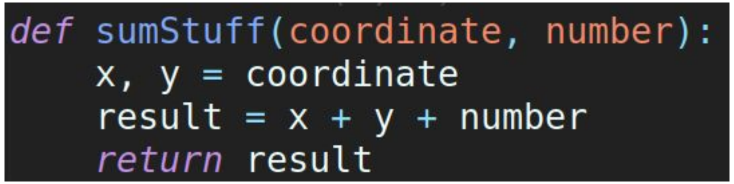
**Prolog**



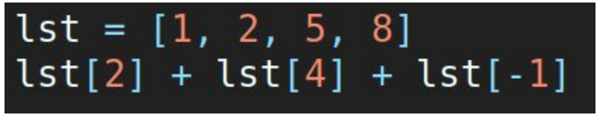
```prolog
sumValues(A, B, C, Total) :-
    Total is A + B + C.
```

## Handling Lists and Tuples

**Python**

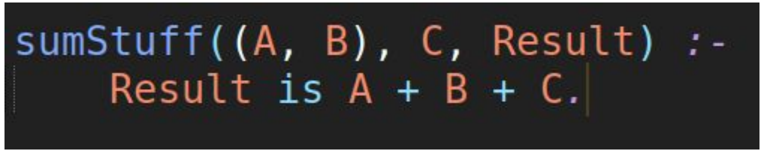


```python
def sumStuff(coordinate, number):
    x, y = coordinate
    result = x + y + number
    return result
```

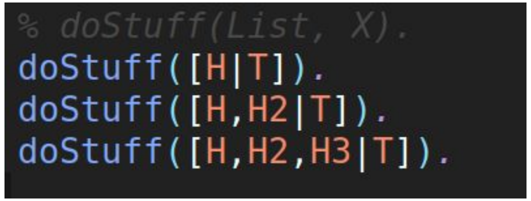


```python
lst = [1, 2, 5, 8]
lst[2] + lst[4] + lst[-1]
```

**Prolog**



```prolog
sumStuff((A, B), C, Result) :-
    Result is A + B + C.
```



```prolog
% doStuff(List, X).
doStuff([H|T]).
doStuff([H,H2|T]).
doStuff([H,H2,H3|T]).
```

## Examples

- List Sum
- List Member
- Factorial

## Projecto Termometros