PROLOG

Programing in **Log**ic

Benoit Viguier
@b_viguier

Afup Lyon
13/12/2017


STONE ET
CHARDEN
L'avventura.
mfp

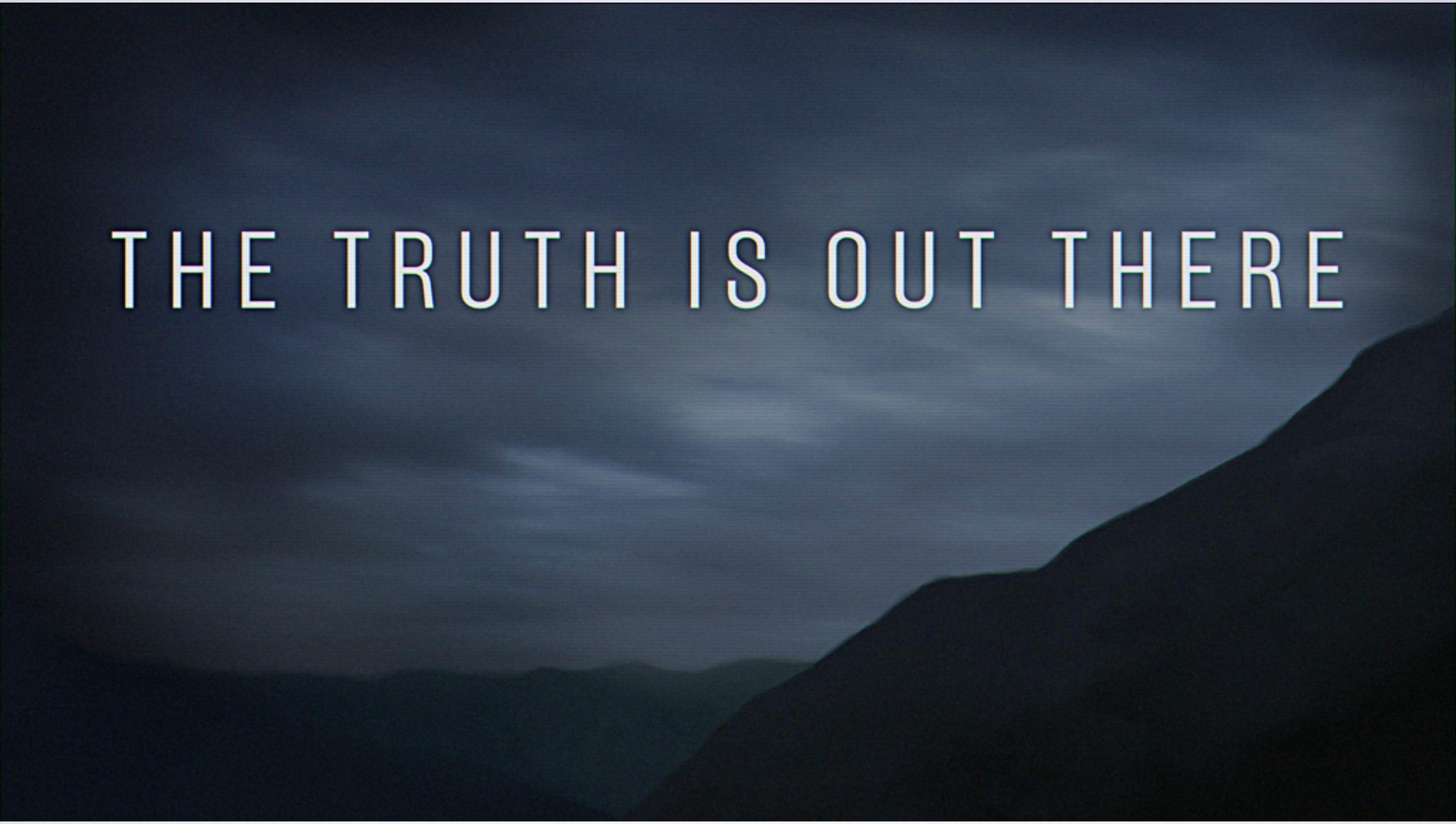

FABRIQUÉ
en
FRANCE


MIKE
BRANT
mfp
QUI SAURA
C'EST MA PRIÈRE
RIEN QU'UNE LARME
LAISSE.MOI T'AIMER

1972

SWI Prolog

http://www.swi-prolog.org/


THE TRUTH IS OUT THERE

```
?-  length([a,b,c],3).
true.
```

```
?-  length([a,b,c],2).
false.
```

End of instruction

```
?-  length([a,b,c],3).
true.
```

```
?-  length([a,b,c],2).
false.
```

Result

Unification

```
?-  length([a,b,c],S).
S  =  3.
```

```
?-  length(L,3).
L  =  [_642, _648, _654].
```

axiome

Variable

```
?-  length([a,b,c],S).
S  =  3.
```

```
?-  length(L,3).
L  =  [_642  _648,  _654].
```

Singleton Variable

```
?-  length(L,S).
L  =  [],
S  =  0
```

Backtracking

```
?- length(L,S).
L = [],
S = 0 ;
L = [_676],
S = 1 ;
L = [_676, _682],
S = 2 ;
L = [_676, _682, _688],
S = 3 ;
L = [_676, _682, _688, _694],
S = 4 .
```

```
?-  length(L,S).
L  =  [],
S  =  0 ;
L  =  [_676],
S  =  1 ;
L  =  [_676, _682],
S  =  2 ;
L  =  [_676, _682, _688],
S  =  3 ;
L  =  [_676, _682, _688, _694],
S  =  4 .
```

Creating predicats

```
swipl  -f  myfile.pl
```

```
?  make.
```

```
language(php).
language(cplusplus).
language(prolog).
```

```
?-  language(php).
true.

?-  language(MyVar).
MyVar  =  php  ;
MyVar  =  cplusplus  ;
MyVar  =  prolog.
```

```
language(php).
language(cplusplus).
language(prolog).
```

```
?-  language(js).
false.
```

```
?-  language(💩).
false.
```

```
fruit(🍎).
fruit(🍌).
fruit(🍉).
fruit(🍍).
```

```
?- fruit(🍎).
true.

?- fruit(🚗).
false.

?- fruit(F).
F = 🍎 ;
F = 🍌 .
```

Rules

```
smoothie([A,B],2) :-
 fruit(A),
 fruit(B),
 A @> B.
```

```
?- smoothie(L,2).
L = [🍎, 🍌] ;
L = [🍎, 🍉] ;
L = [🍎, 🍍] ;
L = [🍌, 🍉] ;
L = [🍍, 🍌] ;
L = [🍍, 🍉] ;
false.
```

« Is implied by... »

```
smoothie([A,B],2) :-
 fruit(A),
 fruit(B),
 A @> B.
```

term comparison

```
?- smoothie(L,2).
L = [🍎, 🍌] ;
L = [🍎, 🍉] ;
L = [🍎, 🍍] ;
L = [🍌, 🍉] ;
L = [🍍, 🍌] ;
L = [🍍, 🍉] ;
false.
```

```
smoothie([A,B],2)  :-
 fruit(A),
fruit(B),
A  @>  B.
```

```
?-  smoothie(L,N).
L  =  [🍎,  🍌],
N  =  2  ;
L  =  [🍎,  🍉],
N  =  2  ;
L  =  [🍎,  🍍],
N  =  2  ;
L  =  [🍌,  🍉],
N  =  2  ;
L  =  [🍍,  🍌],
N  =  2  ;
L  =  [🍍,  🍉],
N  =  2  ;
false.
```

Recursion

```
smoothie([F1|S],N):-
  fruit(F1),
  length([F1|S], N),
  Nm1 is N-1,
  smoothie(S, Nm1),
  [F2|_] = S,
  F1 @> F2.
```

```
?- smoothie(L,3).
L = [🍎, 🍌, 🍉] ;
L = [🍎, 🍍, 🍌] ;
L = [🍎, 🍍, 🍉] ;
L = [🍍, 🍌, 🍉] ;
false.
```

Head | Tail

```
smoothie([F|S],N):-
  fruit(F),
  length([F|S], N),
  Nm1 is N-1,
  smoothie(S, Nm1),
  [F2|_] = S,
  F1 @< F2.
```

Anonymous variable

```
?- smoothie(L,3).
L = [🍎, 🍌, 🍉] ;
L = [🍎, 🍍, 🍌] ;
L = [🍎, 🍍, 🍉] ;
L = [🍍, 🍌, 🍉] ;
false.
```

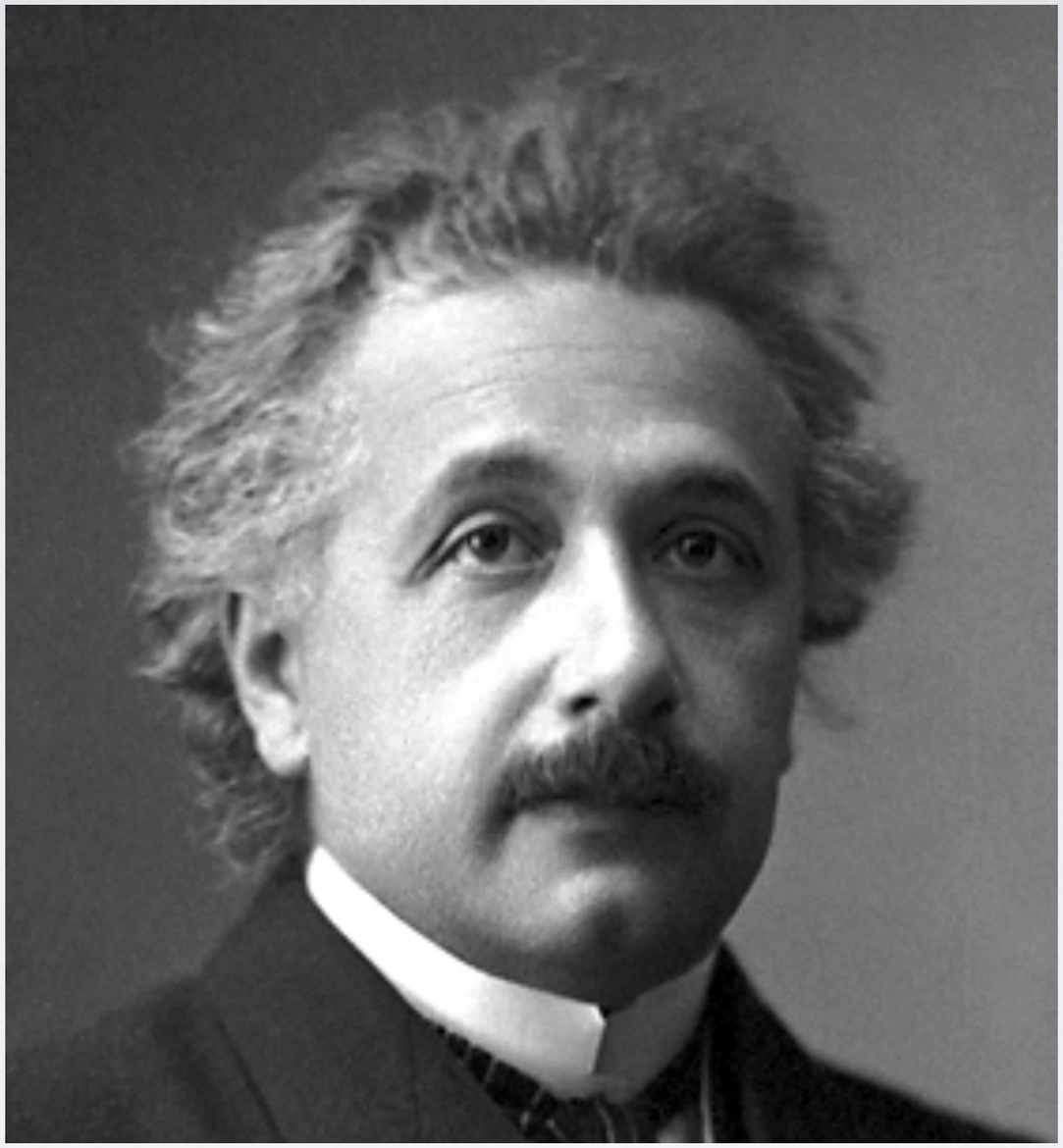
```
smoothie([F1|S],N):-
  fruit(F1),
  length([F1|S], N),
  Nm1 is N-1,
  smoothie(S, Nm1),
  [F2|_] = S,
  F1 @> F2.
```

```
?- smoothie(L,3).
L = [🍎, 🍌, 🍉] ;
L = [🍎, 🍍, 🍌] ;
L = [🍎, 🍍, 🍉] ;
L = [🍍, 🍌, 🍉] ;
false.
```

```
smoothie([F 1 |S],N):-
  fruit(F 1),
  length([F 1 |S], N),
  Nm 1 is N- 1,
  smoothie(S, Nm 1),
  [F 2 |_] = S,
  F 1 @> F 2.
```

```
?- smoothie(L,4).
L = [🍎, 🍍, 🍌, 🍉]
;
false.

?- smoothie(L,5).
false.
```



Einstein's Puzzle

or Zebra Puzzle (1962)

- There are five houses.
- The Englishman lives in the red house.
- The Spaniard owns the dog.
- Coffee is drunk in the green house.
- The Ukrainian drinks tea.
- The green house is immediately to the right of the ivory house.
- The Old Gold smoker owns snails.
- Kools are smoked in the yellow house.
- Milk is drunk in the middle house.
- The Norwegian lives in the first house.
- The man who smokes Chesterfields lives in the house next to the man with the fox.
- Kools are smoked in the house next to the house where the horse is kept.
- The Lucky Strike smoker drinks orange juice.
- The Japanese smokes Parliaments.
- The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra?

It is often claimed that only 2% of the population can solve the puzzle.

Solution

with Emojis 😁

House = List

[🇬🇧, 🐍, 🍓, 🍼, 🚗],

```
einstein(X):-  length(X,5),
        member([🇬🇧,_,_,_,🚗],  X),
        member([🇪🇸,🐶,_,_,_],  X),
        member([_,_,_,🍺,🚲],  X),
        member([🇺🇦,_,_,🍵,_],  X),
        nextto([_,_,_,_,🚲],  [_,_,_,_,🚐],  X),
        member([_,🐍,🍓,_,_],  X),
        member([_,_,🍋,_,🚕],  X),
        X  =  [_,_,[_,_,_,🍼,_],_,_],
        X  =  [[🇳🇴,_,_,_,_],_,_,_,_],
        next([_,_,🍍,_,_],  [_,🐕,_,_,_],  X),
        next([_,_,🍋,_,_],  [_,🐴,_,_,_],  X),
        member([_,_,🍇,🍹,_],  X),
        member([🇯🇵,_,🍒,_,_],  X),
        nextto([🇳🇴,_,_,_,_],  [_,_,_,_,🚙],  X),
        member([_,_,_,🍷,_],  X),
        member([_,🐘,_,_,_],  X).
```

Five houses.

```
length(X,5)
```

The Englishman owns the red car.

```
member([🇬🇧,_,_,_,🚗],X)
```

member([🇬🇧,_,_,_,🚗],X)

member([🇪🇸,🐶,_,_,_],X)

member([_,_,_,🍺,🚲],X)

member([🇺🇦,_,_,🍵,_],X)

member([_,🐍,🍓,_,_],X)

member([_,_,🍋,_,🚕],X)

member([_,_,🍇,🍹,_],X)

member([🇯🇵,_,🍒,_,_],X)

The bicycle owner lives immediately to the right of the white car owner.

```
nextto([_,_,_,_,🚲],[_,_,_,_,🚐],  X)
```

The Norwegian lives next to the blue car.

```
nextto([🇳🇴,_,_,_,_], [_,_,_,_,🚙], X)
```

The man who eats pineapples lives in the house next to the man with the fox.

next([_,_,🍍,_,_], [_,🦊,_,_,_], X)

```
next(X,Y,L)  :-
    nextto(X,Y,L),  !.

next(X,Y,L)  :-
    nextto(Y,X,L).
```

```
?-  next(b,c,[a,b,c]).
true.

?-  next(c,b,[a,b,c]).
true  ;
false.
```

Lemons are eaten in the house next to the house where the horse is kept.

next([_,_,🍋,_,_], [_,🐴,_,_,_], X)

Milk is drunk in the middle house.

$$X = [\_, \_, [\_, \_, \_, 🍼, \_], \_, \_]$$

The Norwegian lives in the first house.

$$X = [[🇳🇴,\_,\_,\_,\_],\_,\_,\_,\_]$$

Who drinks wine?

```
member([_,_,_,🍷,_], X)
```

Who owns the elephant?

```
member([_,🐘,_,_,_],  X)
```

```
einstein(X):-  length(X,5),
        member([🇬🇧,_,_,_,🚗],  X),
        member([🇪🇸,🐶,_,_,_],  X),
        member([_,_,_,🍺,🚲],  X),
        member([🇺🇦,_,_,🍵,_],  X),
        nextto([_,_,_,_,🚲],  [_,_,_,_,🚐],  X),
        member([_,🐍,🍓,_,_],  X),
        member([_,_,🍋,_,🚕],  X),
        X  =  [_,_,[_,_,_,🍼,_],_,_],
        X  =  [[🇳🇴,_,_,_,_],_,_,_,_],
        next([_,_,🍍,_,_],  [_,🐕,_,_,_],  X),
        next([_,_,🍋,_,_],  [_,🐴,_,_,_],  X),
        member([_,_,🍇,🍹,_],  X),
        member([🇯🇵,_,🍒,_,_],  X),
        nextto([🇳🇴,_,_,_,_],  [_,_,_,_,🚙],  X),
        member([_,_,_,🍷,_],  X),
        member([_,🐘,_,_,_],  X).
```

```
?- einstein(X).
X = [
  [🇳🇴, 🦊, 🍋, 🍷, 🚕],
  [🇺🇦, 🐴, 🍍, 🍵, 🚙],
  [🇬🇧, 🐍, 🍓, 🍼, 🚗],
  [🇯🇵, 🐘, 🍒, 🍺, 🚲],
  [🇪🇸, 🐶, 🍇, 🍹, 🚐]
] ;
false.
```

Try it!

http://www.swi-prolog.org/

Thanks

@b_viguier

```
?- [X|L] = [a,b,c].
X = a,
L = [b, c].

?- [X|L] = [a].
X = a,
L = [].

?- [X|L] = [].
false.
```

Cut

```
next(X,Y,L)  :-
    nextto(X,Y,L), !.
```

```
?-  next(b,c,[a,b,c]).
true.
```

```
next(X,Y,L)  :-
    nextto(Y,X,L).
```

```
?-  next(c,b,[a,b,c]).
true  ;
false.
```