An Introduction To Prolog And Why You Should Care

Charlotte Shreve
@littlechrob

Charlotte Shreve
@littlechrob

ELECTRICAL ENGINEERING &
COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

Charlotte Shreve
@littlechrob

Agenda

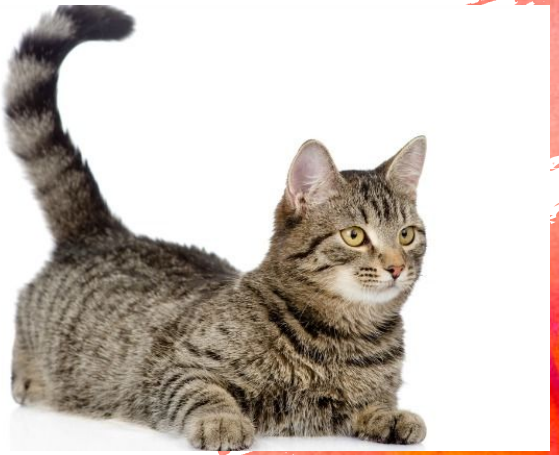
- An introduction to prolog
- Why you should care

Agenda

- What is Prolog
- Prolog Terminology
- How does a Prolog program work
- Practice problem
- What is Prolog used for today
- How Prolog made me a better developer

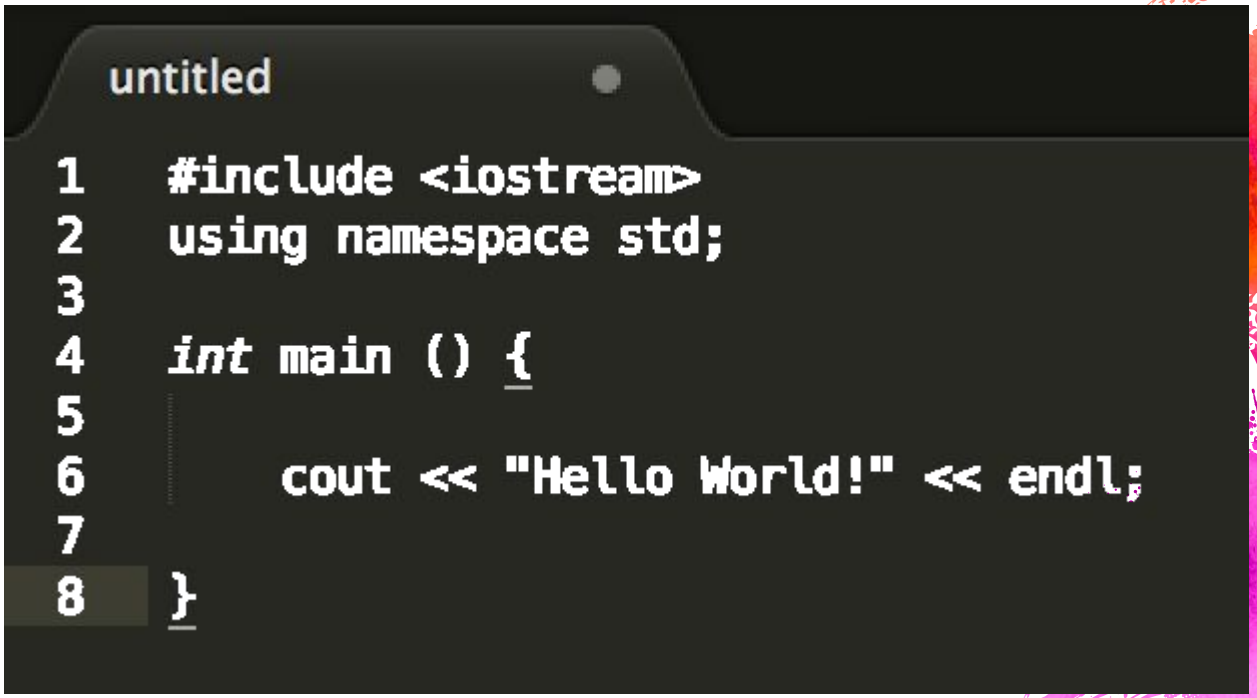
Objective

- Have a basic knowledge of Prolog and how it is similar and different to the languages you already know
- Understand why learning new languages is important to your development and teaching different types of languages from the beginning is important to the development of the community

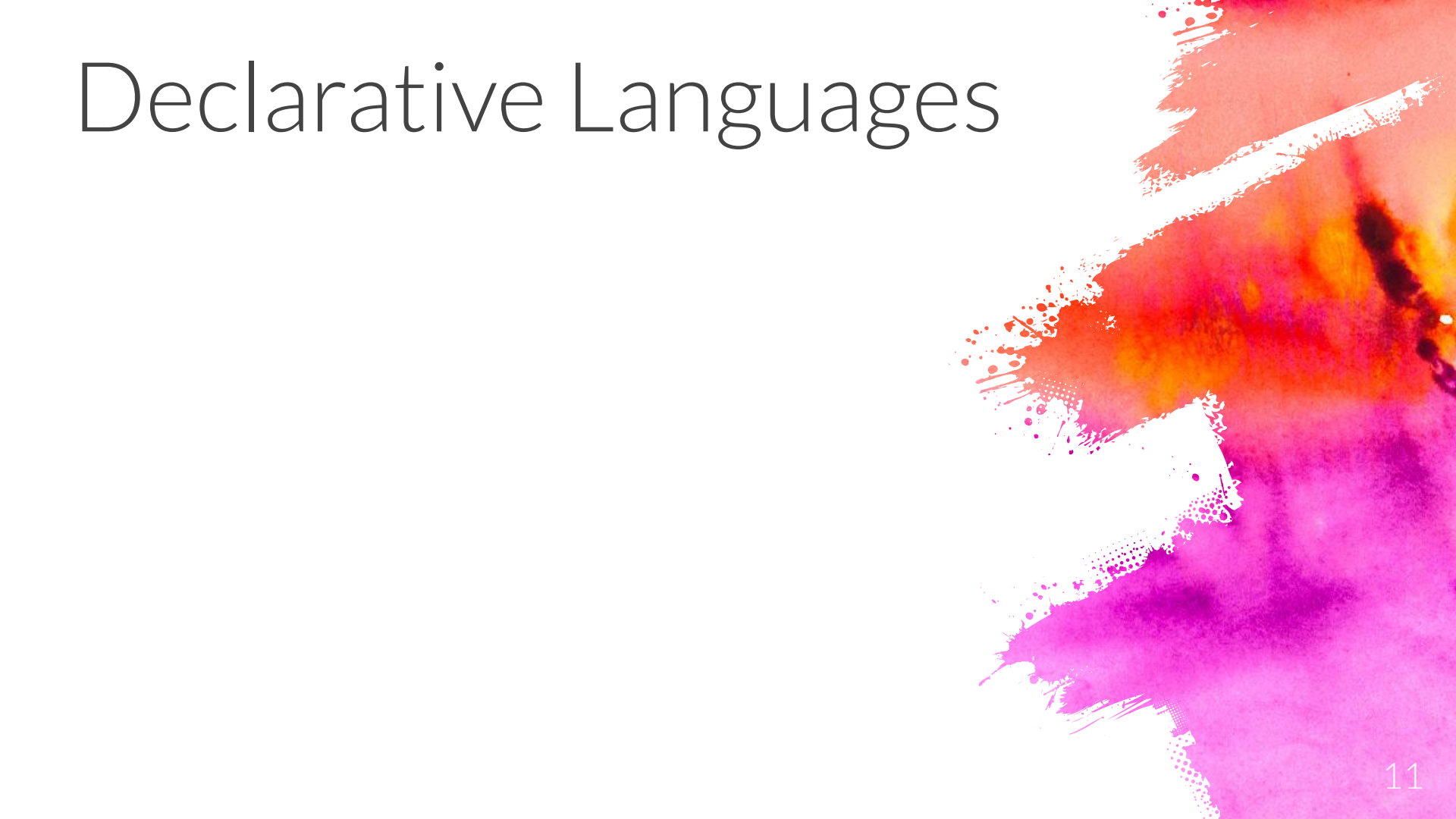


Imperative Languages

Imperative Languages



Declarative Languages



Declarative Languages

|

Logic Languages

Declarative Languages

|

Logic Languages

|

Prolog

```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

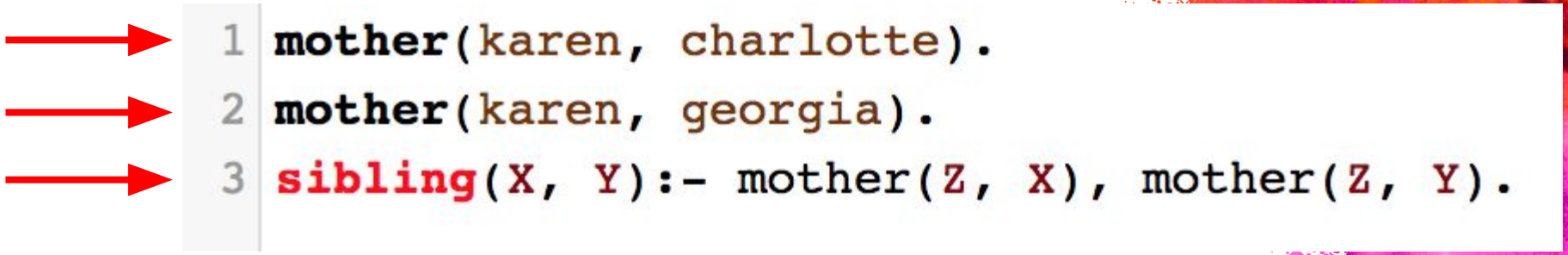
https://swish.swi-prolog.org/

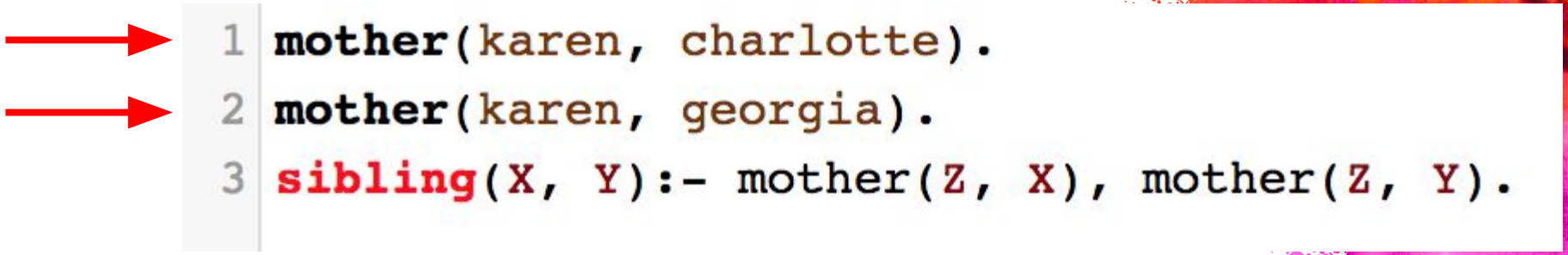
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

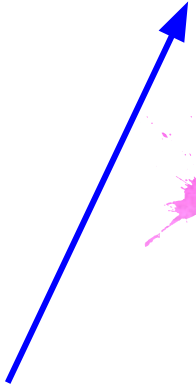
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```


1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).


1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).

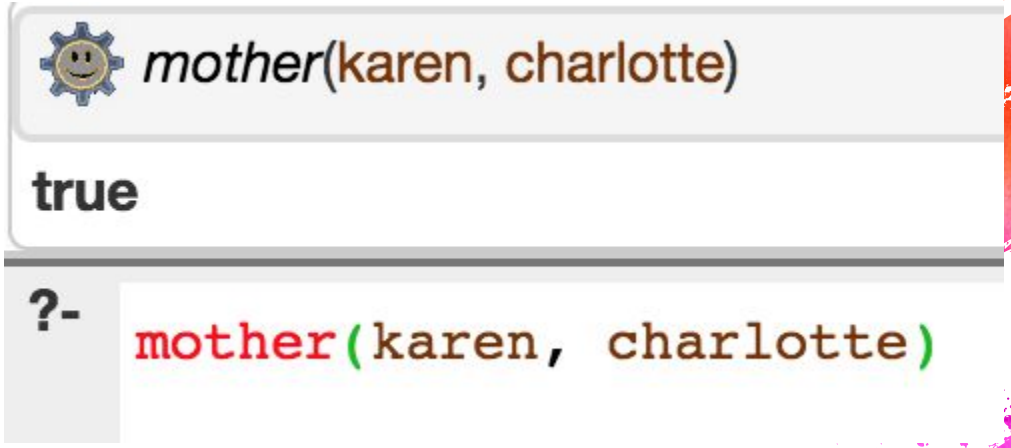
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

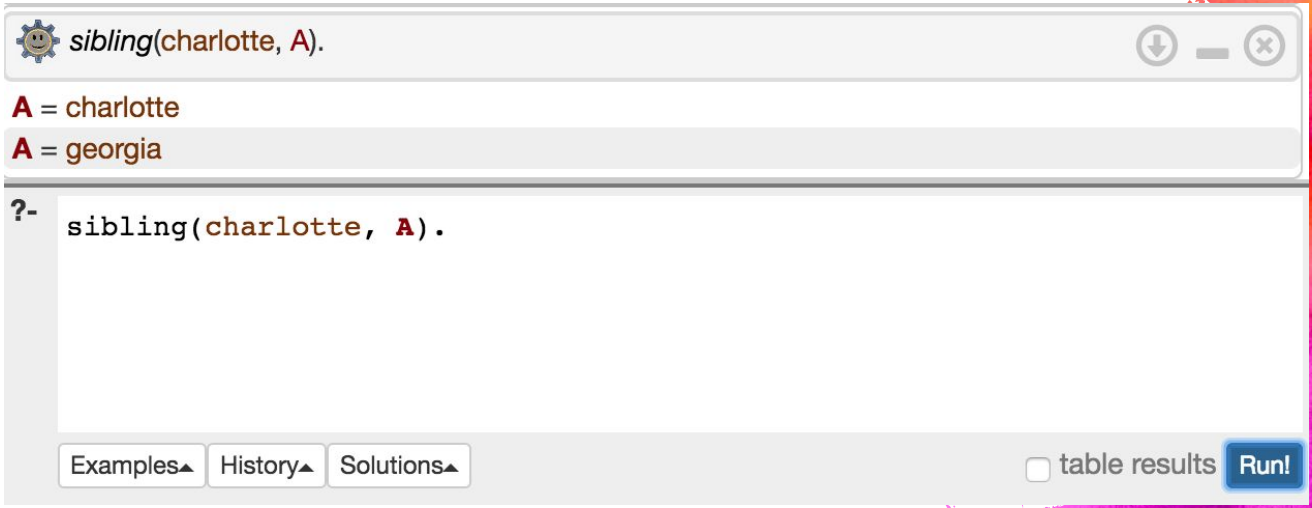
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



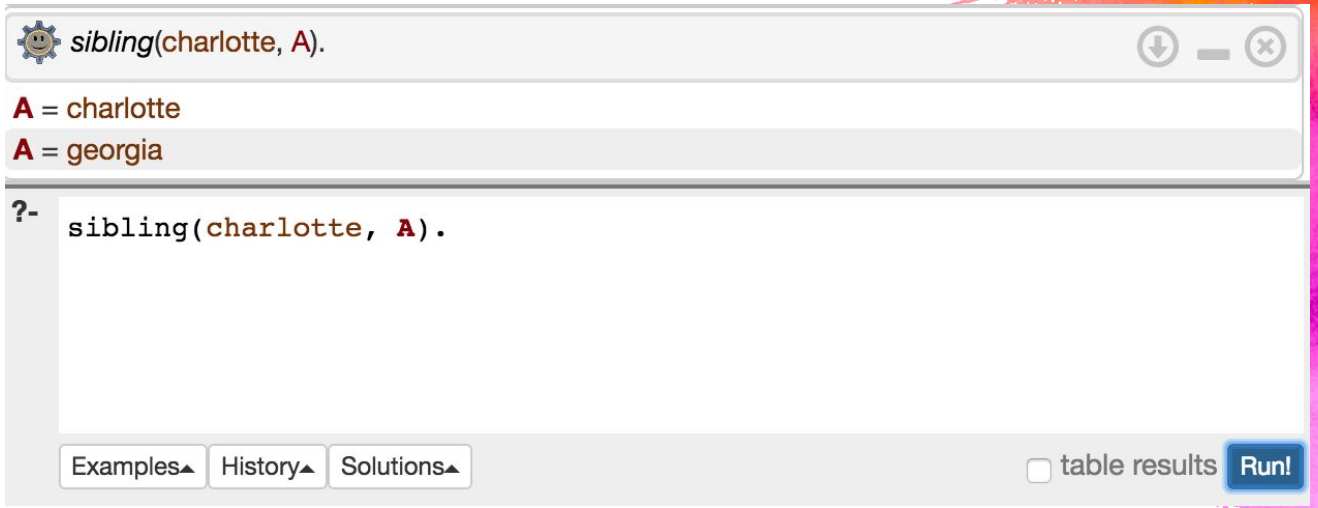
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```


Phew.


mother(karen, charlotte)
true
?-
mother(karen, charlotte)


sibling(charlotte, A).
A = charlotte
A = georgia
?-
sibling(charlotte, A).
Examples
History
Solutions
table results
Run!

```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



```
1 mother(karen, georgia).
2 mother(karen, charlotte).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

sibling(charlotte, A).

A = georgia

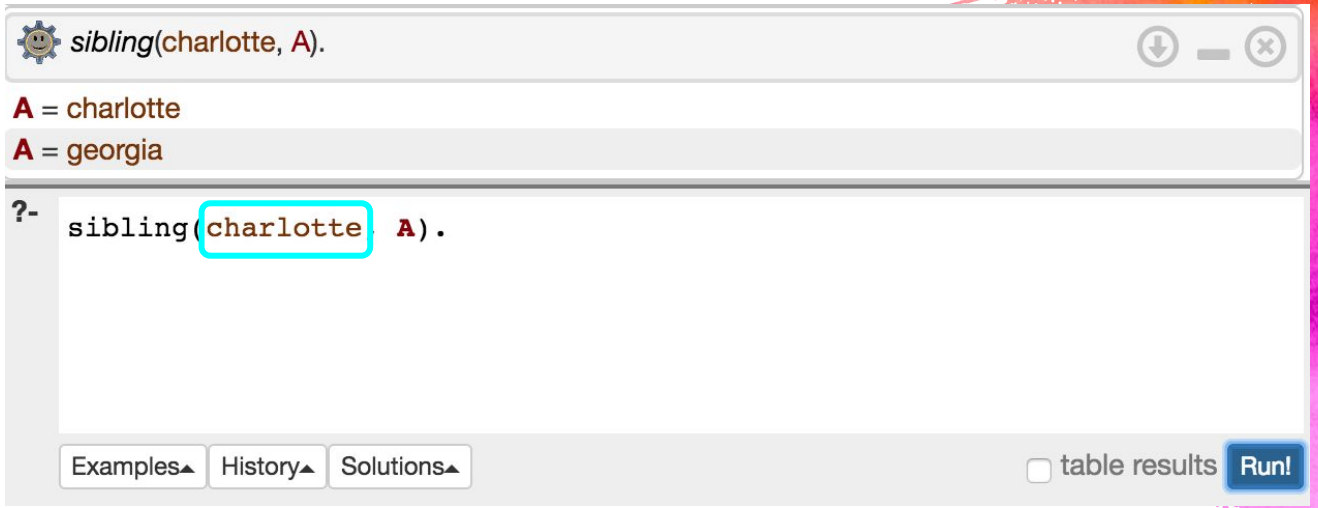
A = charlotte

```
?- sibling(charlotte, A).
```

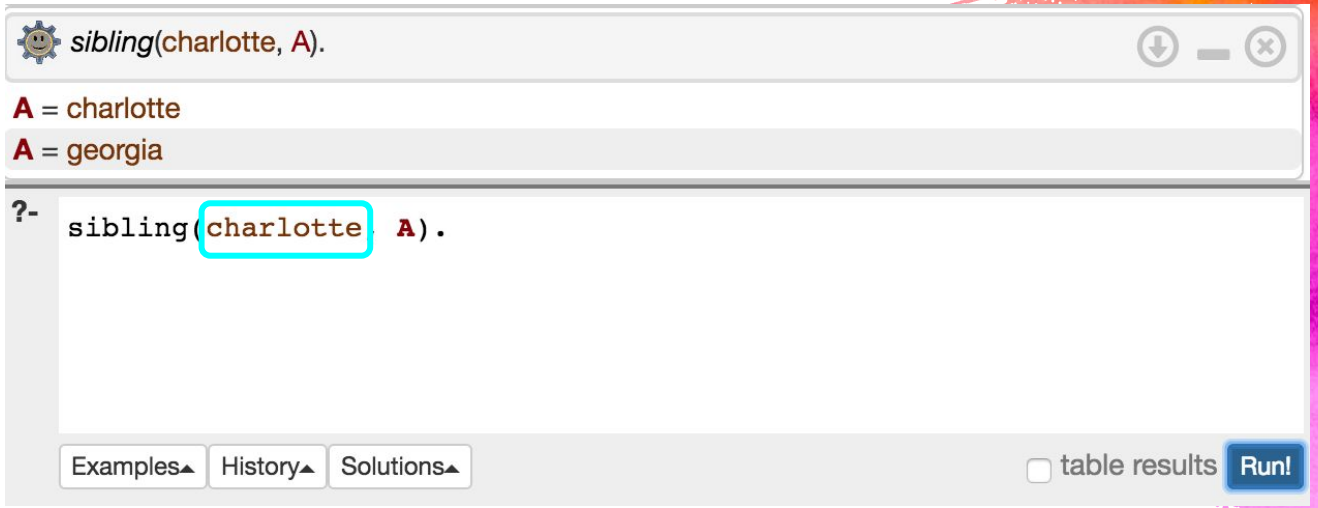
Examples▲ History▲ Solutions▲ table results Run!

Unification

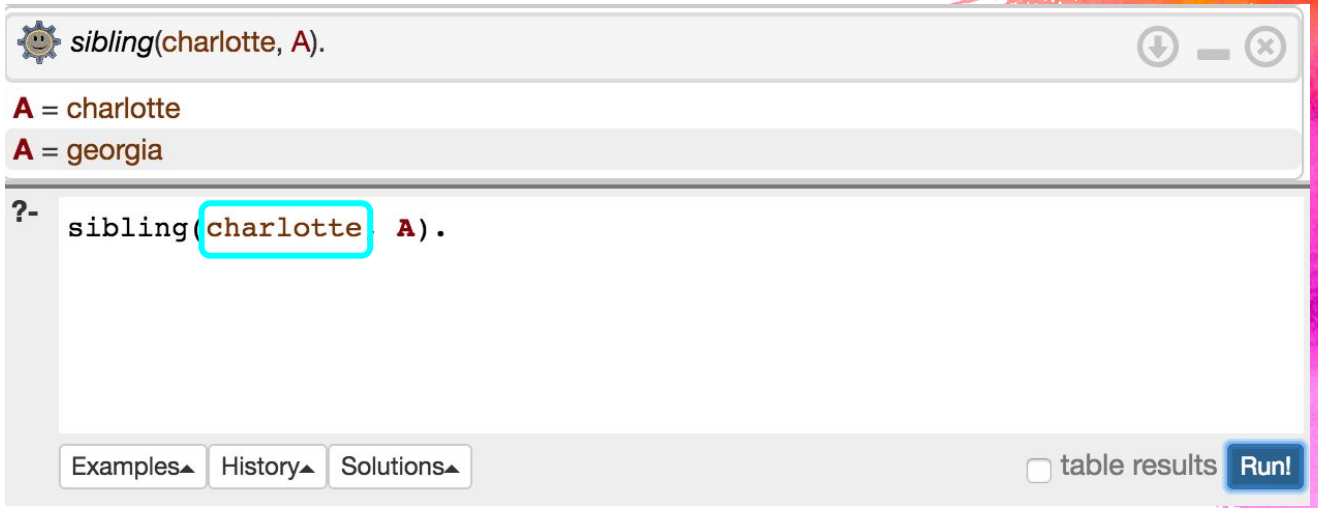
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



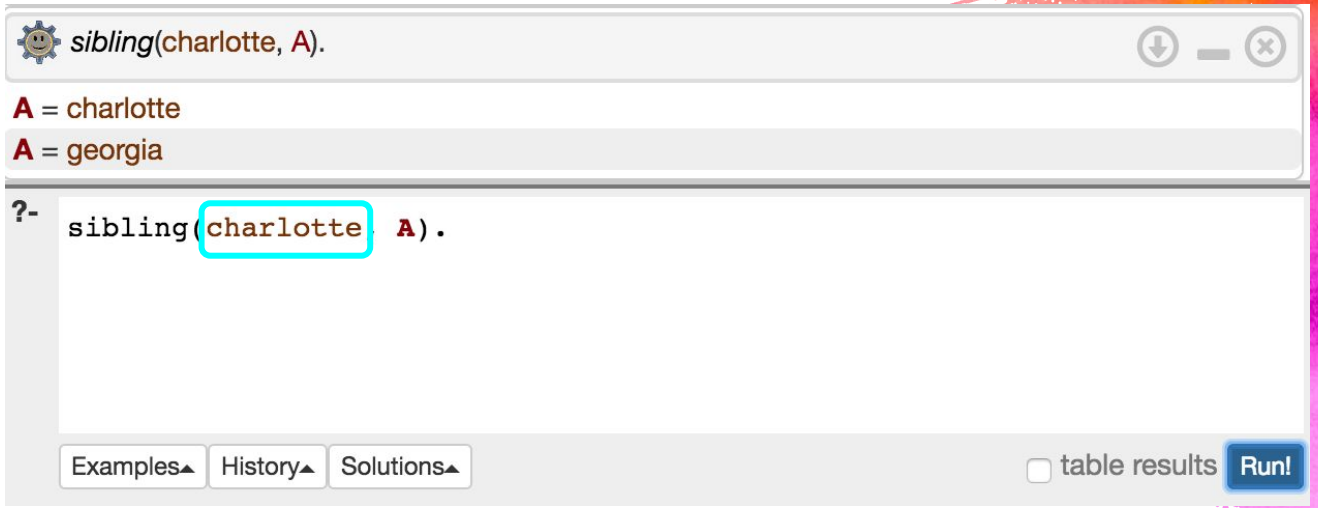
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



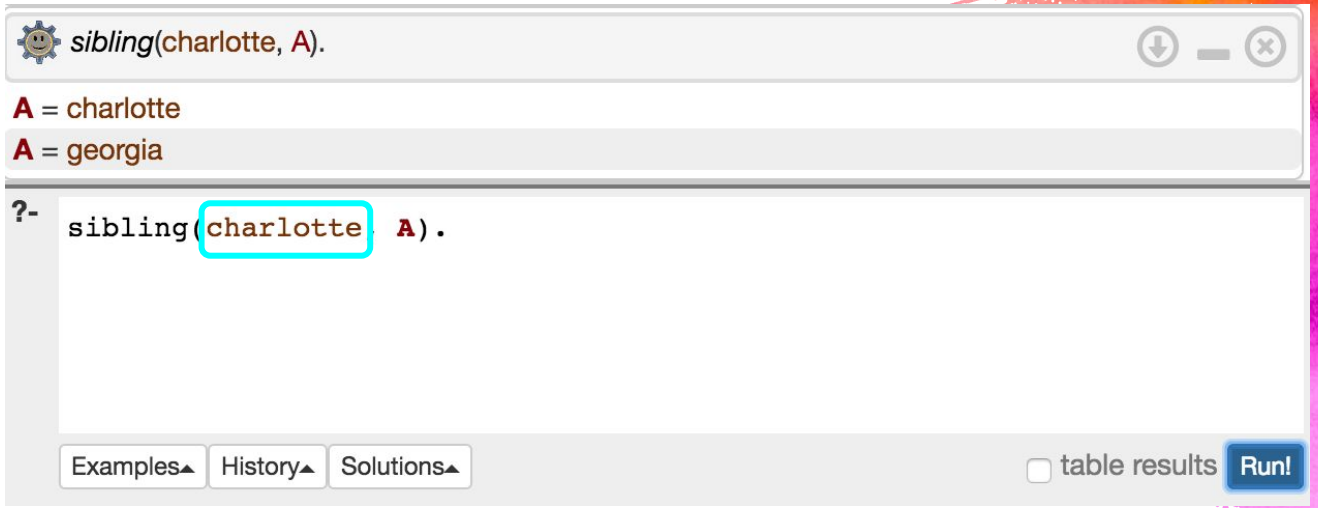
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



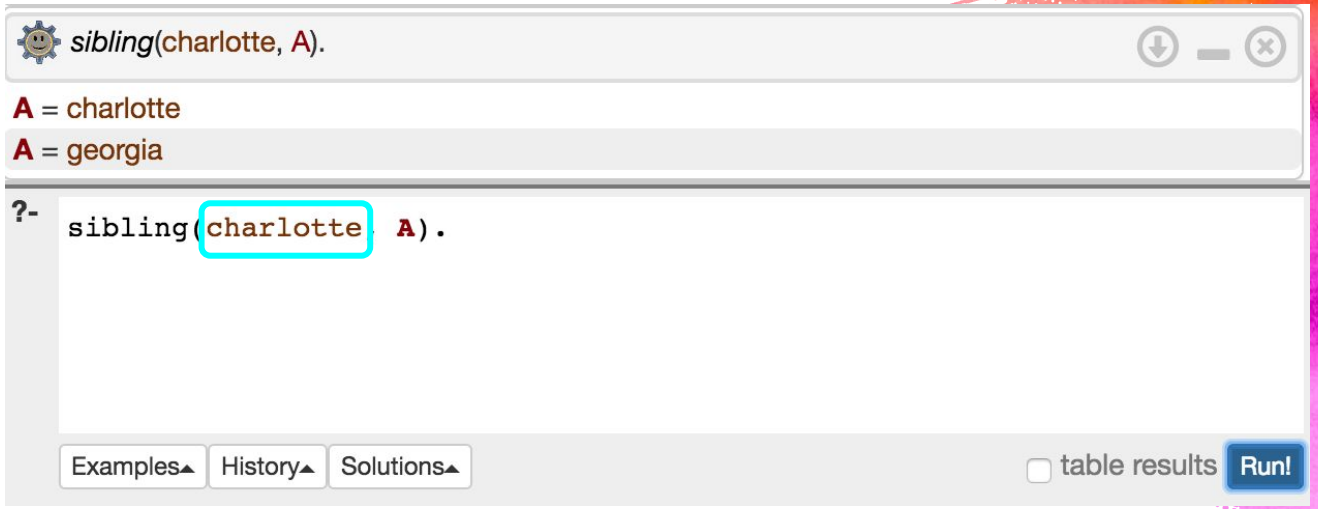
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

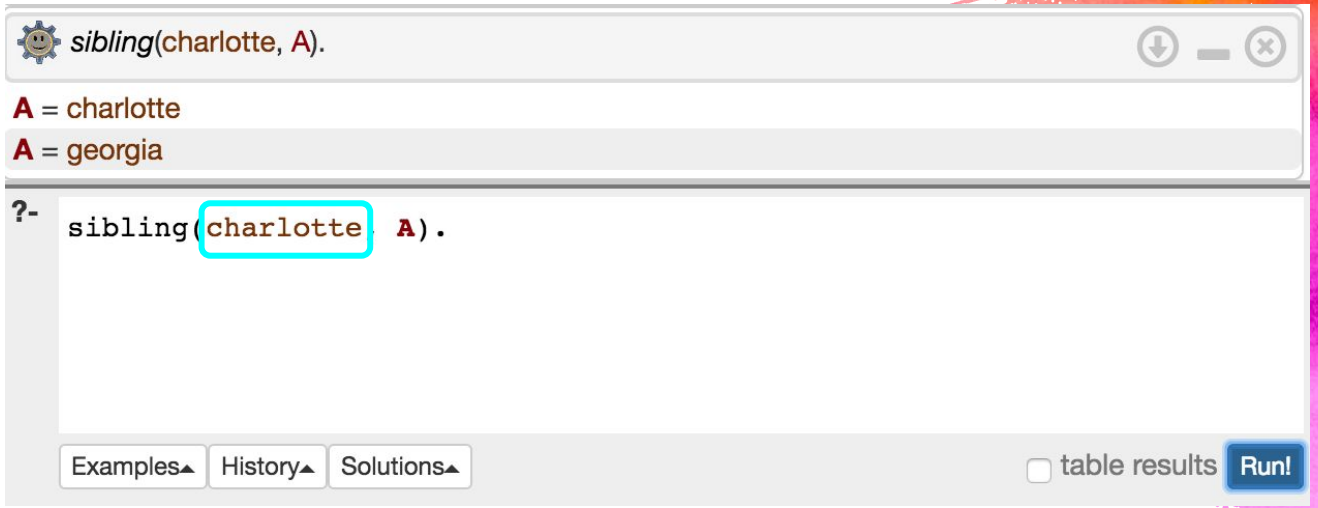


```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

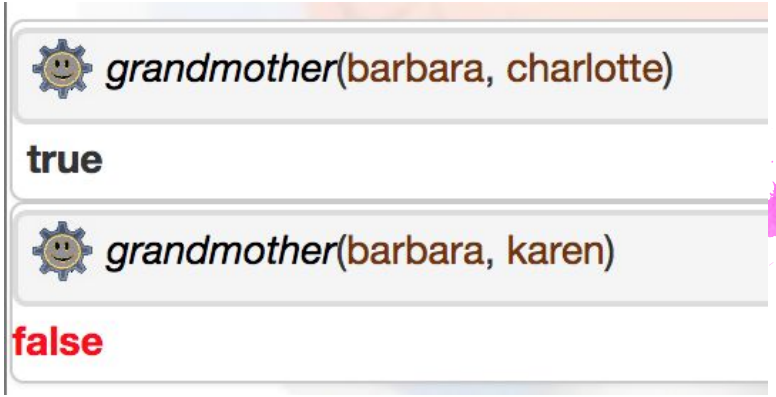


Backtracking

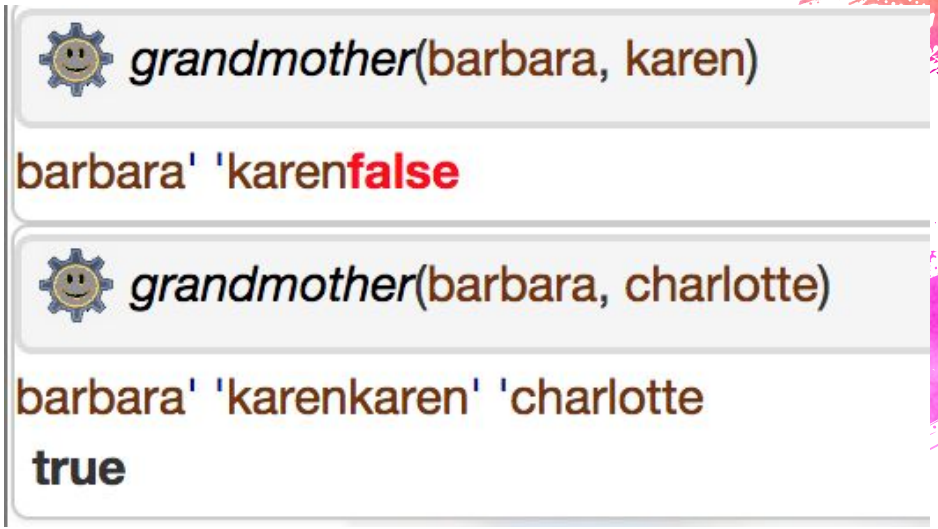
```
1 mother(karen, charlotte).
2 mother(karen, georgia).
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



```
1 mother(karen, georgia).
2 mother(karen, charlotte).
3 mother(barbara, karen).
4 sibling(X, Y):- mother(Z, X), mother(Z, Y).
5 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```



```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```



```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara

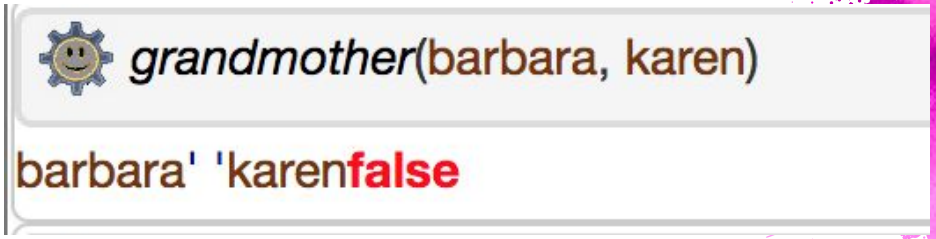
Karen

grandmother(barbara, karen)

barbara' 'karen**false**

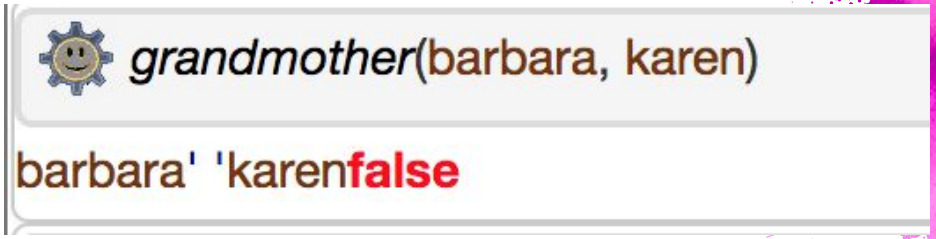
```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara Karen Barbara Karen



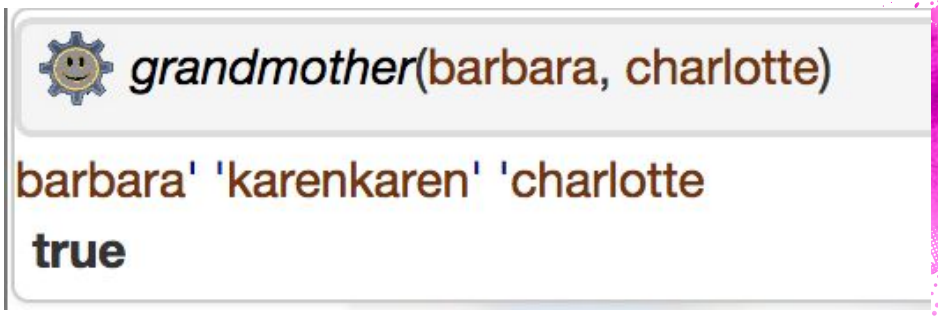
```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara Karen Barbara Karen Karen Karen



```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara Charlotte



```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara Charlotte Barbara Karen

```
grandmother(barbara, charlotte)
barbara' 'karenkaren' 'charlotte
true
```

```
1 mother(karen, georgia):- print(karen), print(' '), print(georgia).
2 mother(karen, charlotte):- print(karen), print(' '), print(charlotte).
3 mother(barbara, karen):- print(barbara), print(' '), print(karen).
4 grandmother(X, Z):- mother(X, Y), mother(Y, Z).
```

Barbara Charlotte Barbara Karen Karen Charlotte

grandmother(barbara, charlotte)

barbara' 'karenkaren' 'charlotte

true

2 last things...

Negation

```
+\ grandmother(barbara, charlotte)
```

Annoymous variables

```
grandmother(barbara, _)
```

Let’s Write Code

There is a neighborhood that has 4 houses that are different colors (red, blue, yellow and pink). The families that live in the houses each have a different type of pet (cat, dog, bunny, bird). Each house has a different toy (a trampoline, a pool, a scooter, a basketball hoop).

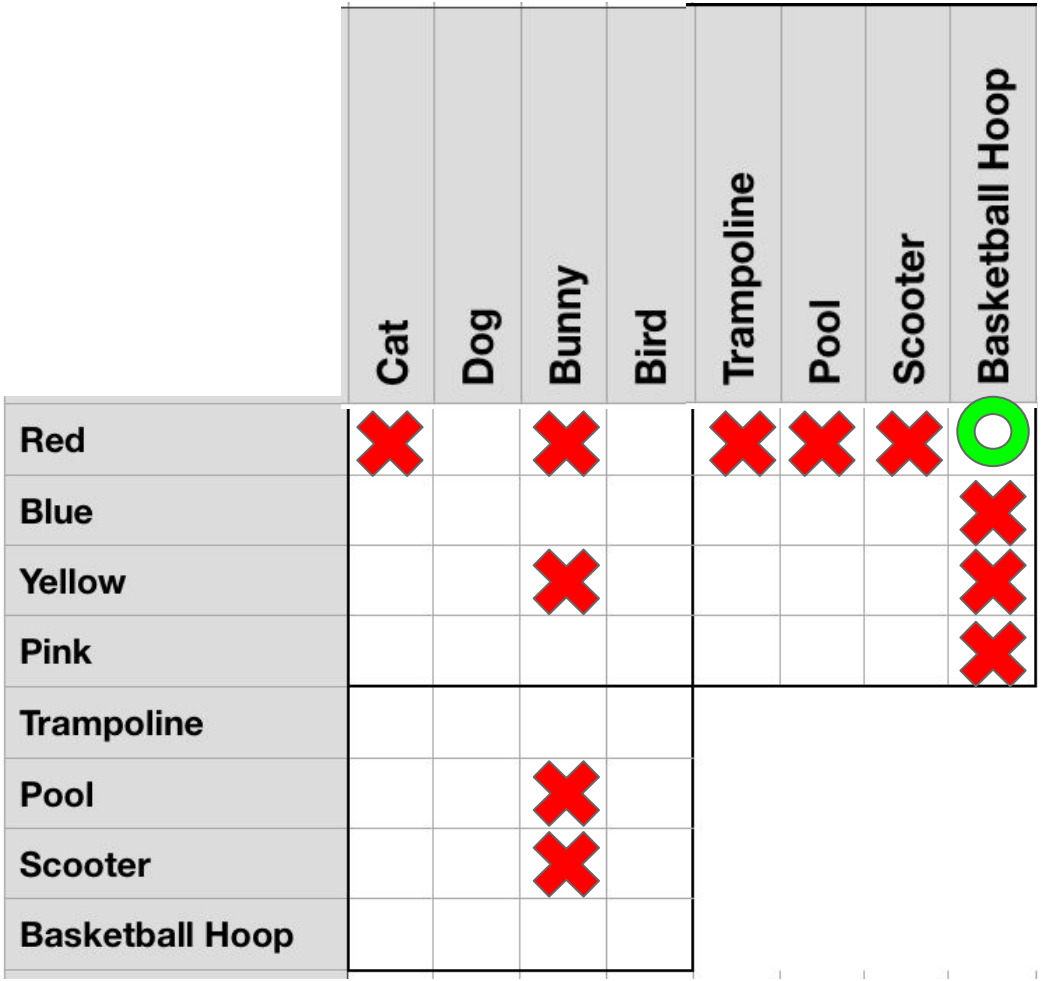
Goal: find out which house has which pet and which toy.

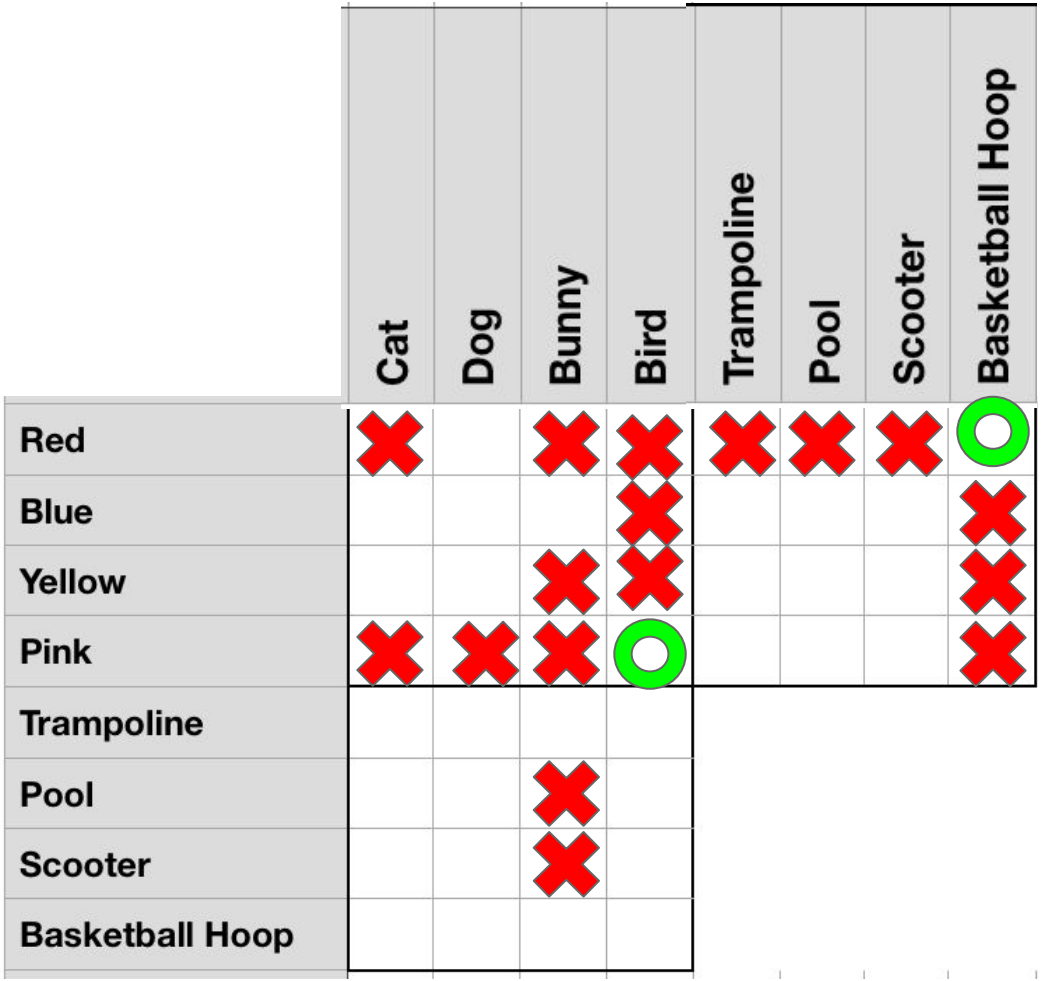
Clues:

1. The bunny (who doesn't live at the red or yellow house) doesn't live at the house with the pool or the scooter.
2. The red house (which doesn't have the cat) has the basketball hoop.
3. The pink house has the bird.
4. The cat lives at the house with the pool.

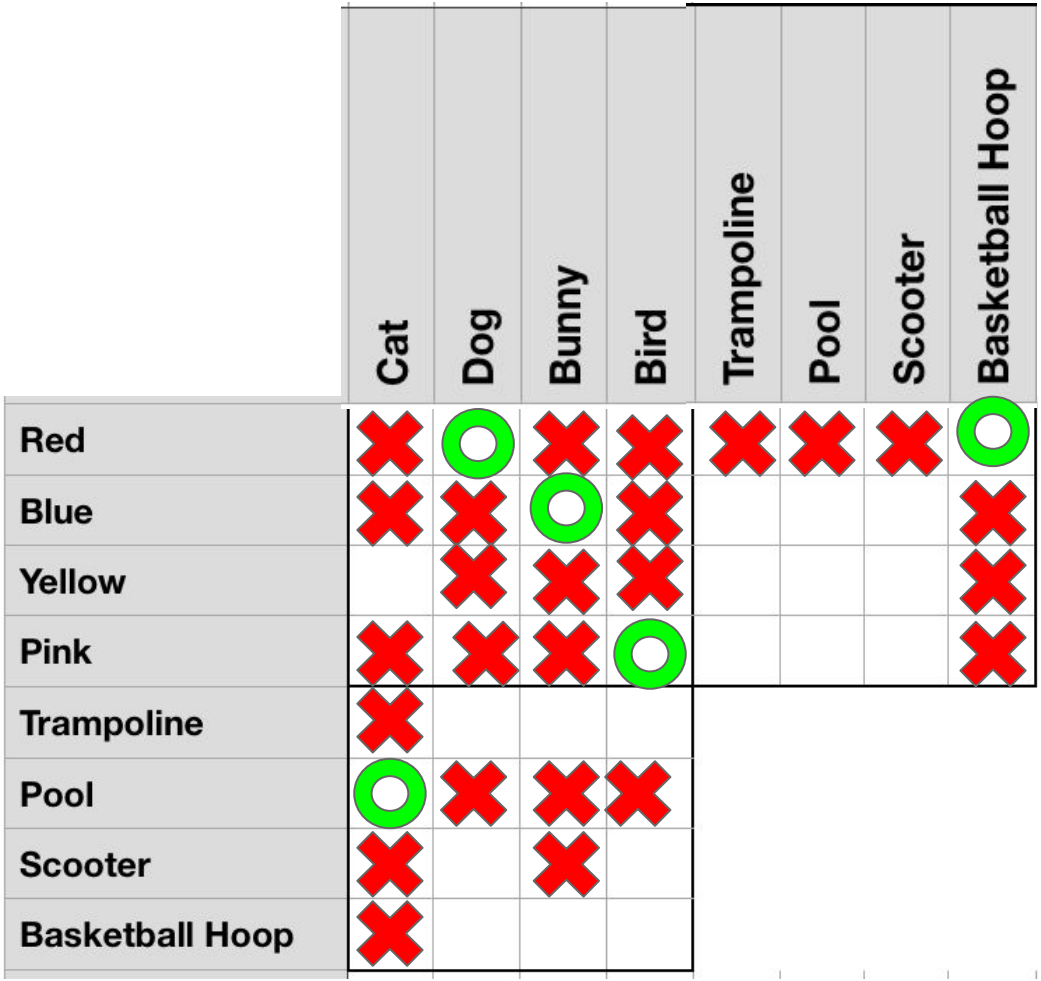
	Cat	Dog	Bunny	Bird	Trampoline	Pool	Scooter	Basketball Hoop
Red								
Blue								
Yellow								
Pink								
Trampoline								
Pool								
Scooter								
Basketball Hoop								

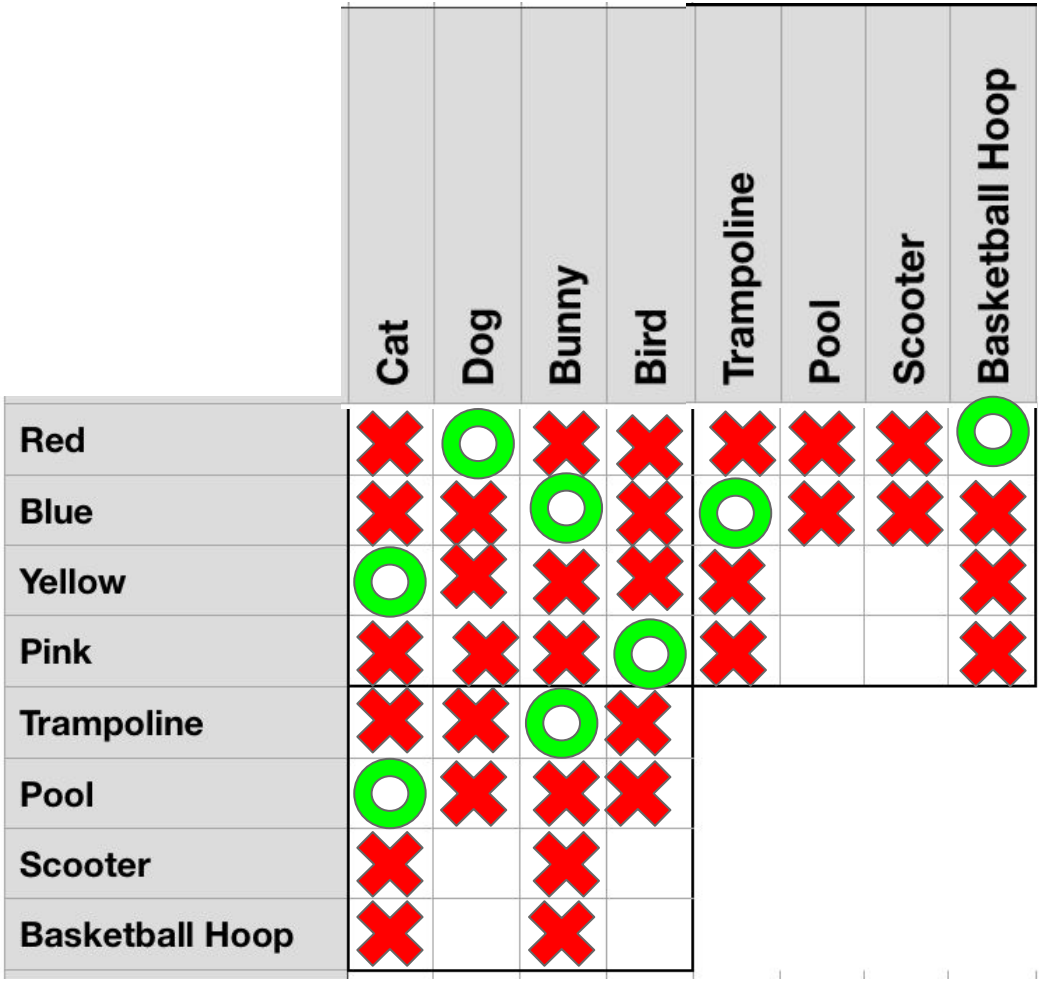
	Cat	Dog	Bunny	Bird	Trampoline	Pool	Scooter	Basketball Hoop
Red			✖					
Blue								
Yellow			✖					
Pink								
Trampoline								
Pool			✖					
Scooter			✖					
Basketball Hoop								


Cat
Dog
Bunny
Bird
Trampoline
Pool
Scooter
Basketball Hoop
Red
Blue
Yellow
Pink
Trampoline
Pool
Scooter
Basketball Hoop


Cat
Dog
Bunny
Bird
Trampoline
Pool
Scooter
Basketball Hoop
Red
Blue
Yellow
Pink
Trampoline
Pool
Scooter
Basketball Hoop

	Cat	Dog	Bunny	Bird	Trampoline	Pool	Scooter	Basketball Hoop
Red	✗		✗	✗	✗	✗	✗	○
Blue				✗				✗
Yellow			✗	✗				✗
Pink	✗	✗	✗	○				✗
Trampoline	✗							
Pool	○	✗	✗	✗				
Scooter	✗		✗					
Basketball Hoop	✗							


Cat
Dog
Bunny
Bird
Trampoline
Pool
Scooter
Basketball Hoop
Red
Blue
Yellow
Pink
Trampoline
Pool
Scooter
Basketball Hoop


Cat
Dog
Bunny
Bird
Trampoline
Pool
Scooter
Basketball Hoop
Red
Blue
Yellow
Pink
Trampoline
Pool
Scooter
Basketball Hoop

	Cat	Dog	Bunny	Bird	Trampoline	Pool	Scooter	Basketball Hoop
Red	X	O	X	X	X	X	X	O
Blue	X	X	O	X	O	X	X	X
Yellow	O	X	X	X	X			X
Pink	X	X	X	O	X			X
Trampoline	X	X	O	X				
Pool	O	X	X	X				
Scooter	X	X	X	O				
Basketball Hoop	X	O	X	X				

	Cat	Dog	Bunny	Bird	Trampoline	Pool	Scooter	Basketball Hoop
Red	X	O	X	X	X	X	X	O
Blue	X	X	O	X	O	X	X	X
Yellow	O	X	X	X	X	O	X	X
Pink	X	X	X	O	X	X	O	X
Trampoline	X	X	O	X				
Pool	O	X	X	X				
Scooter	X	X	X	O				
Basketball Hoop	X	O	X	X				

```python
class house:
    def __init__(self, color):
        self.color = color
        self.possible_pets = ['cat', 'dog', 'bird', 'bunny']
        self.possible_toys = ['trampoline', 'pool', 'scooter', 'basketball hoop']

    def is_done(self):
        if len(self.possible_pets) == 1 and len(self.possible_toys) == 1:
            return True
        else:
            return False

    def remove_pet(self, pet):
        if pet in self.possible_pets:
            self.possible_pets.remove(pet)

    def remove_toy(self, toy):
        if toy in self.possible_toys:
            self.possible_toys.remove(toy)
```

```
class neighborhood:
    def __init__(self):
        self.houses = [house('red'), house('blue'), house('yellow'), house('pink')]

    def is_valid(self):
        for house in self.houses:
            if not house.is_done():
                return False

        for i in self.houses:
            for j in self.houses:
                if i.color != j.color:
                    if i.possible_pets[0] == j.possible_pets[0]:
                        return False

                    if i.possible_toys[0] == j.possible_pets[0]:
                        return False
        return True

    def normalize(self):
        # if you've found the correct pet or toy for a house
        # remove that from others
        for house in self.houses:
            if len(house.possible_pets) == 1:
                for otherhouse in self.houses:
                    if house.color != otherhouse.color:
                        otherhouse.remove_pet(house.possible_pets[0])

        for house in self.houses:
            if len(house.possible_toys) == 1:
                for otherhouse in self.houses:
                    if house.color != otherhouse.color:
                        otherhouse.remove_toy(house.possible_toys[0])
```

```
neighborhood = neighborhood()
cycle_count = 0

while not neighborhood.is_valid():
    print("Cycle count: " + str(cycle_count))
    cycle_count += 1
    # clue 1 A - the bunny doesn't live in the red or yellow houses
    for house in neighborhood.houses:
        if house.color in ['red', 'yellow']:
            house.remove_pet('bunny')

    # clue 1 B - the bunny doesn't live with either the pool or the scooter
    for house in neighborhood.houses:
        if len(house.possible_pets) == 1 and house.possible_pets[0] == 'bunny':
            house.remove_toy('pool')
            house.remove_toy('scooter')

    # clue 2 - the red house doesn't have the cat but has the basketball hoop
    for house in neighborhood.houses:
        if house.color == 'red':
            house.remove_pet('cat')
            house.possible_toys = ['basketball hoop']
        else:
                house.remove_toy('basketball hoop')
```

```
    # clue 3 - the pink house has the bird
    for house in neighborhood.houses:
        if house.color == 'pink':
            house.possible_pets = ['bird']

    # clue 4 - the cat lives with the pool
    for house in neighborhood.houses:
        if len(house.possible_pets) == 1 and house.possible_pets[0] == 'cat':
            house.possible_toys = ['pool']


    neighborhood.normalize()

    print(neighborhood)
```

```
~/Workspace/conference
▶ python3 SAMPLE.py
Cycle count: 0
red ['dog'] ['basketball hoop']
blue ['cat', 'dog', 'bunny'] ['trampoline', 'pool', 'scooter']
yellow ['cat', 'dog'] ['trampoline', 'pool', 'scooter']
pink ['bird'] ['trampoline', 'pool', 'scooter']

Cycle count: 1
red ['dog'] ['basketball hoop']
blue ['bunny'] ['trampoline', 'pool', 'scooter']
yellow ['cat'] ['trampoline', 'pool', 'scooter']
pink ['bird'] ['trampoline', 'pool', 'scooter']

Cycle count: 2
red ['dog'] ['basketball hoop']
blue ['bunny'] ['trampoline']
yellow ['cat'] ['pool']
pink ['bird'] ['scooter']
```

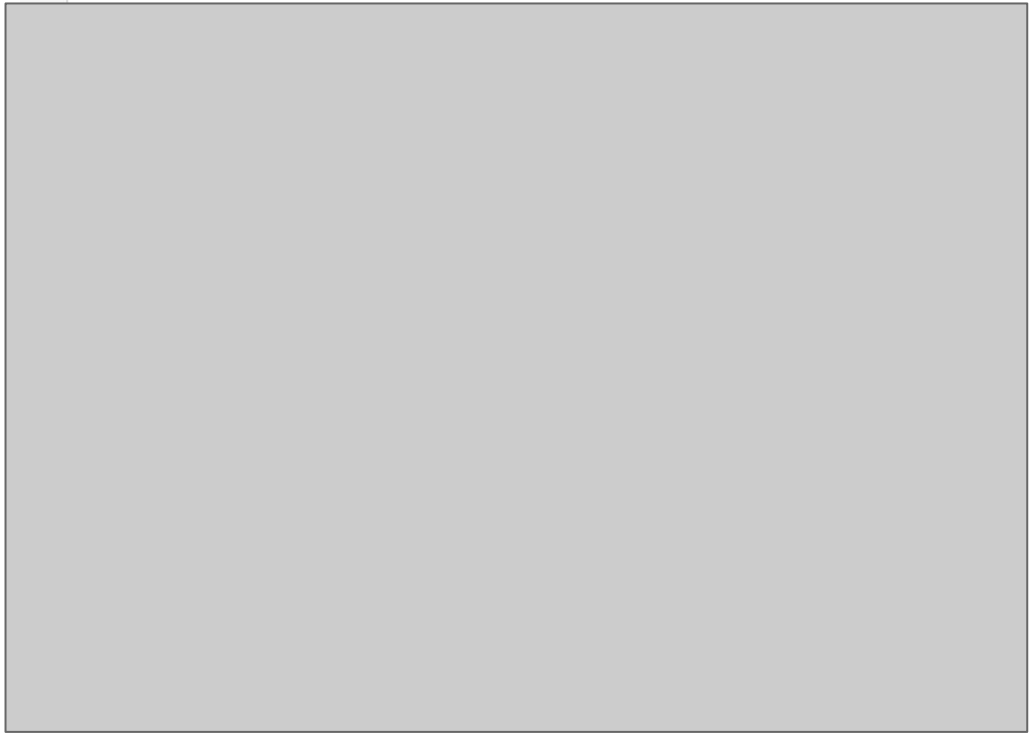
https://swish.swi-prolog.org/

```
pet(Pet):- member(Pet, [cat, dog, bird, bunny]).
valid_pets(P1, P2, P3, P4):- pet(P1), pet(P2), pet(P3), pet(P4),
    all_different(P1, P2, P3, P4).

toy(Toy):- member(Toy, [trampoline, pool, scooter, 'basketball hoop']).
valid_toys(T1, T2, T3, T4):- toy(T1), toy(T2), toy(T3), toy(T4),
    all_different(T1, T2, T3, T4).

all_different(A, B, C, D):-
    A \= B, A \= C, A \= D,
    B \= C, B \= D,
    C \= D.
```

```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
```



```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
```

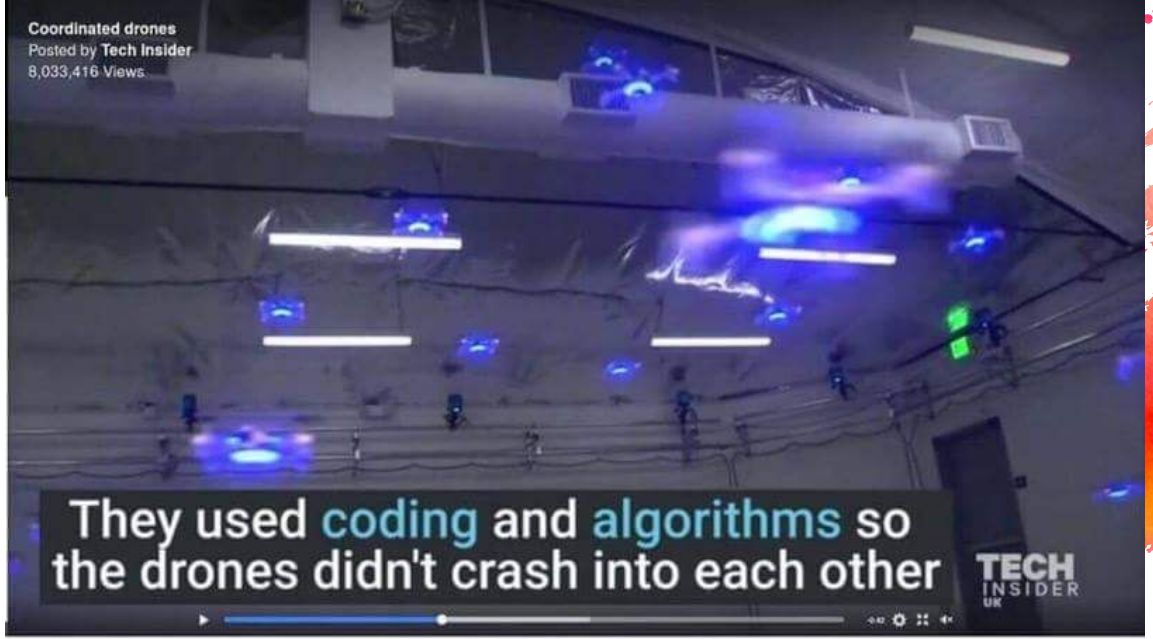
```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
23
24     % Clue 1 %
25     member(bunny, [BluePet, PinkPet]),
26     \+ member([_, bunny, pool], Houses),
27     \+ member([_, bunny, scooter], Houses),
```

```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
23
24     % Clue 1 %
25     member(bunny, [BluePet, PinkPet]),
26     \+ member([_, bunny, pool], Houses),
27     \+ member([_, bunny, scooter], Houses),
28
29     % Clue 2 %
30     member([red, _, 'basketball hoop'], Houses),
31     \+ member([red, cat, _], Houses),
```

```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
23
24     % Clue 1 %
25     member(bunny, [BluePet, PinkPet]),
26     \+ member([_, bunny, pool], Houses),
27     \+ member([_, bunny, scooter], Houses),
28
29     % Clue 2 %
30     member([red, _, 'basketball hoop'], Houses),
31     \+ member([red, cat, _], Houses),
32
33     % Clue 3 %
34     member([pink, bird, _], Houses),
35
```

```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
23
24     % Clue 1 %
25     member(bunny, [BluePet, PinkPet]),
26     \+ member([_, bunny, pool], Houses),
27     \+ member([_, bunny, scooter], Houses),
28
29     % Clue 2 %
30     member([red, _, 'basketball hoop'], Houses),
31     \+ member([red, cat, _], Houses),
32
33     % Clue 3 %
34     member([pink, bird, _], Houses),
35
36     % Clue 4 %
37     member([_, cat, pool], Houses).
```

```
14 solution(Houses):-
15     Houses = [
16         [red, RedPet, RedToy],
17         [blue, BluePet, BlueToy],
18         [yellow, YellowPet, YellowToy],
19         [pink, PinkPet, PinkToy]
20     ],
21     valid_pets(RedPet, BluePet, YellowPet, PinkPet),
22     valid_toys(RedToy, BlueToy, YellowToy, PinkToy),
23
24     % Clue 1 %
25     member(bunny, [BluePet, PinkPet]),
26     \+ member([_, bunny, pool], Houses),
27     \+ member([_, bunny, scooter], Houses
28
29     % Clue 2 %
30     member([red, _, 'basketball hoop'], Houses),
31     \+ member([red, cat, _], Houses),
32
33     % Clue 3 %
34     member([pink, bird, _], Houses),
35
36     % Clue 4 %
37     member([_, cat, pool], Houses).
```



16k 485 Share

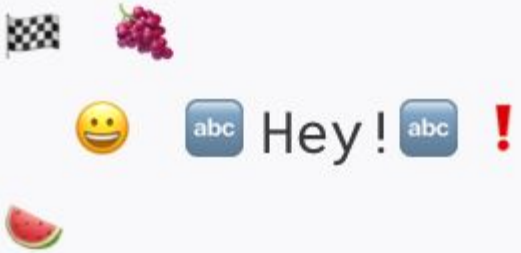
BEST

u/Skizm • 2mo

```
if(goingToCrashIntoEachOther)
{ dont(); }
```

So why don’t we teach Prolog?

EXAMPLE



Languages to solve problems

“Languages do not limit our ability to perceive the world or to think about the world, but they focus our perception, attention, and thought on specific aspects of the world”

Programming Languages Affect Thought

Learning Prolog made me a better developer

Soo...

- Prolog could be used to engage people who would have otherwise been turned off from programming
  - People think differently
- Understanding languages that are similar and different to your “primary language” is just as critical to your abilities as a programmer as being an expert in just that one

- http://www.dai.ed.ac.uk/groups/ssp/bookpages/quickprolog/quickprolog.html
- http://www.dipmat.unict.it/~barba/PROG-LANG/PROGRAMMI-TESTI/READING-MATERIAL/MapColoringIII.html
- http://www.cse.unsw.edu.au/~billw/dictionaries/prolog/
- https://www.bluej.org/papers/1999-08-JOOP1-languages.pdf
- http://www.learnprolognow.org/slides/official/LPNchapter1.pdf
- https://www.cis.upenn.edu/~matuszek/Concise%20Guides/Concise%20Prolog.html
- https://www.psychologytoday.com/us/blog/the-biolinguistic-turn/201702/how-the-language-we-speak-affects-the-way-we-think
- https://www.quora.com/Does-programming-affect-the-way-programmers-think
- https://www.cs.drexel.edu/~ac993/papers/caliskan_deanonymizing.pdf
- https://www.cs.utexas.edu/users/EWD/ewd04xx/EWD498.PDF
- https://tylermcginnis.com/imperative-vs-declarative-programming/
- https://medium.com/front-end-hacking/imperative-versus-declarative-code-whats-the-difference-adc7dd6c8380
- http://tenbitbyte.com/posts/2012-08-22-logic-vs-function.html
- EECS 490 @ University of Michigan by Dr. Amir Kamil

Code PaLOUsa 2019 Sponsors

An Introduction To Prolog And Why You Should Care

Charlotte Shreve

@littlechrob
charlotteshreve.com