

An Introduction To Prolog And Why You Should Care

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

untitled

```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5
6      cout << "Hello World!" << endl;
7
8  }
```

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).
```

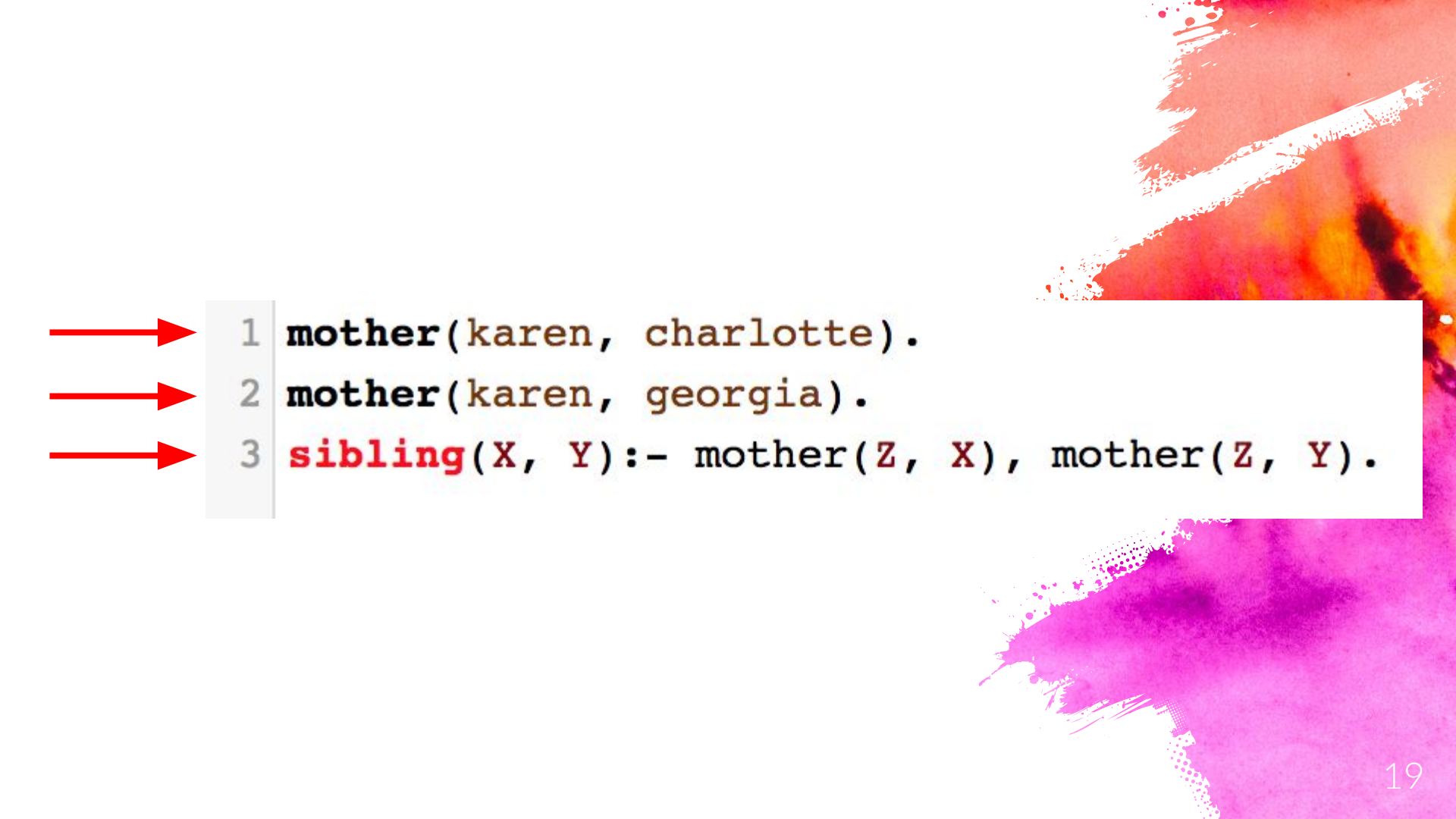
```
1 mother(karen, charlotte).  
2 mother(karen, georgia).  
3 sibling(X, Y) :- mother(Z, X), mother(Z, Y).
```

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

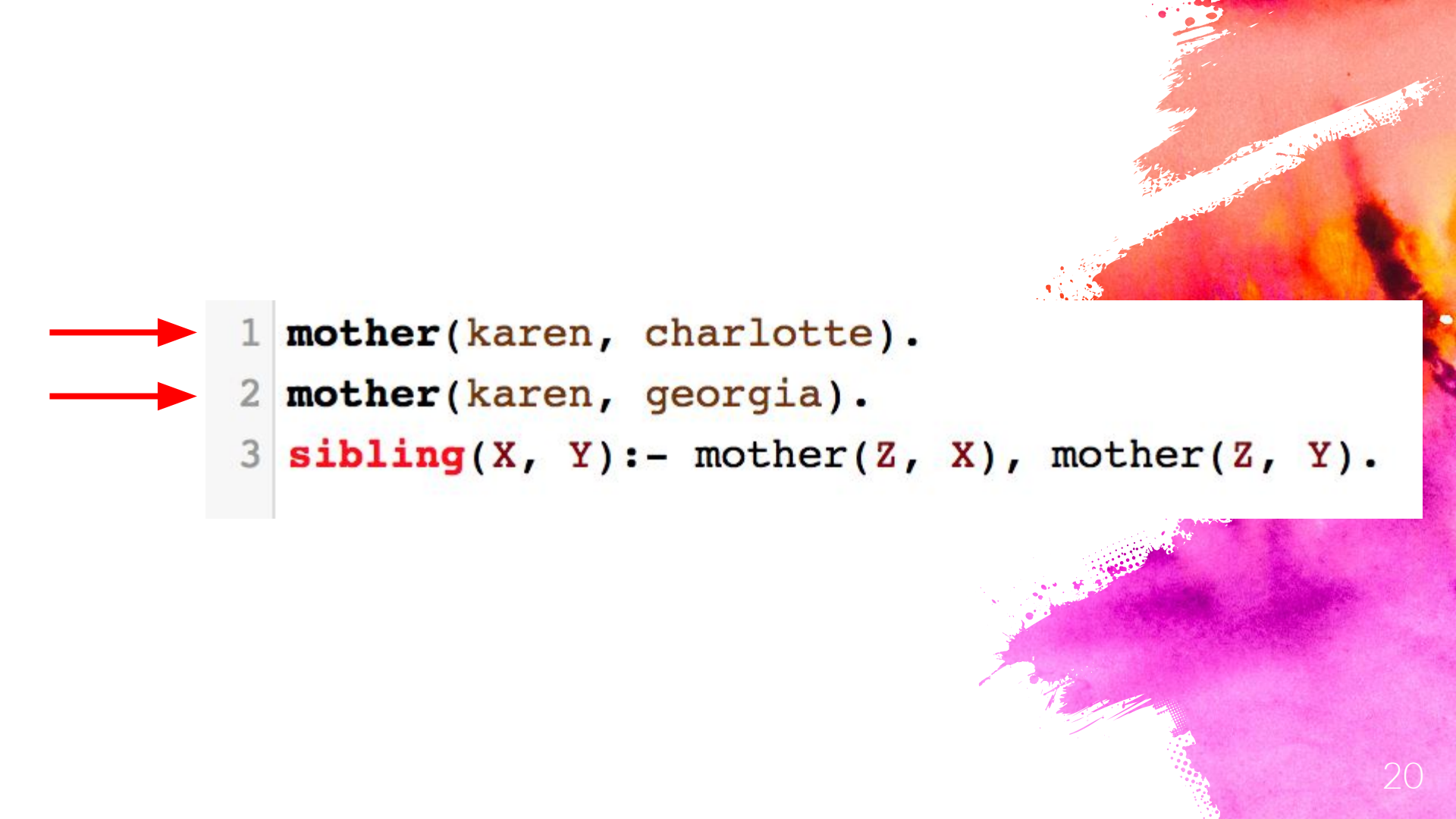


```
1 mother(karen, charlotte).  
2 mother(karen, georgia).  
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```


```
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).
```

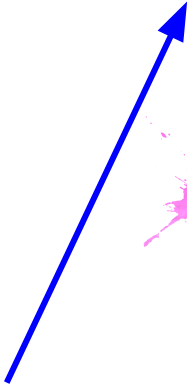


```
1 mother(karen, charlotte).  
2 mother(karen, georgia).  
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```

```
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).
```









mother(karen, charlotte)

true

?-

mother(karen, charlotte)

 `sibling(charlotte, A).`

`A = charlotte`
`A = georgia`

?-

`sibling(charlotte, A).`

Examples▲

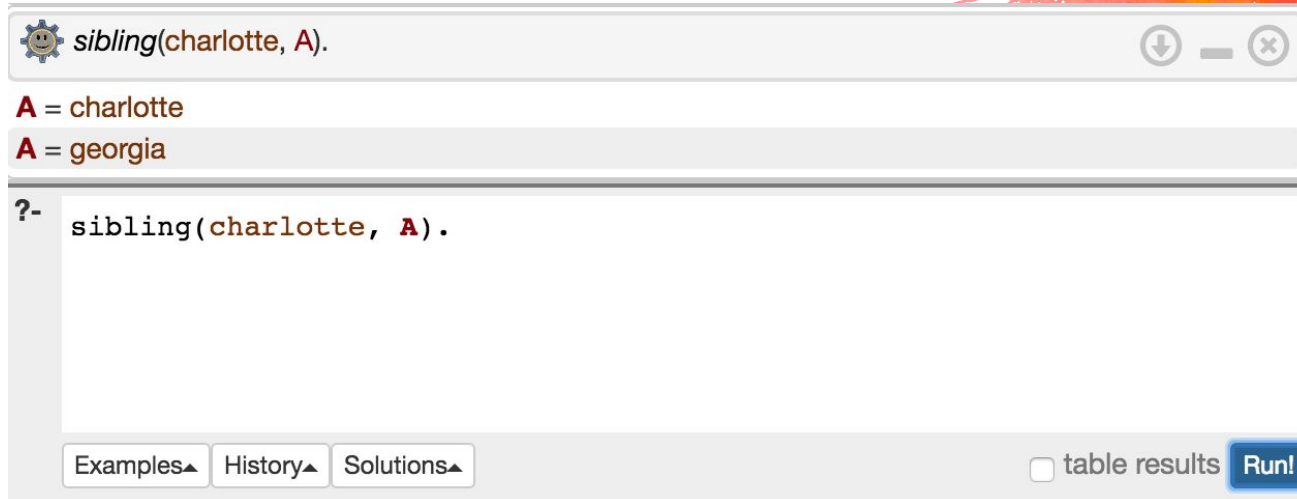
History▲

Solutions▲





☐ table results

Run!

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



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon, the text `sibling(charlotte, A).`, and standard window controls (download, minimize, close). Below the title bar, there are two lines of output: `A = charlotte` and `A = georgia`. The main area of the window contains the prompt `?-` followed by the query `sibling(charlotte, A).`. At the bottom, there are three buttons labeled "Examples▲", "History▲", and "Solutions▲". To the right of these buttons is a checkbox labeled "table results" which is currently unchecked, and a blue button labeled "Run!".

 `sibling(charlotte, A).`   

`A = charlotte`
`A = georgia`

`?- sibling(charlotte, A).`

Examples▲ History▲ Solutions▲ ☐ table results **Run!**

```
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▲

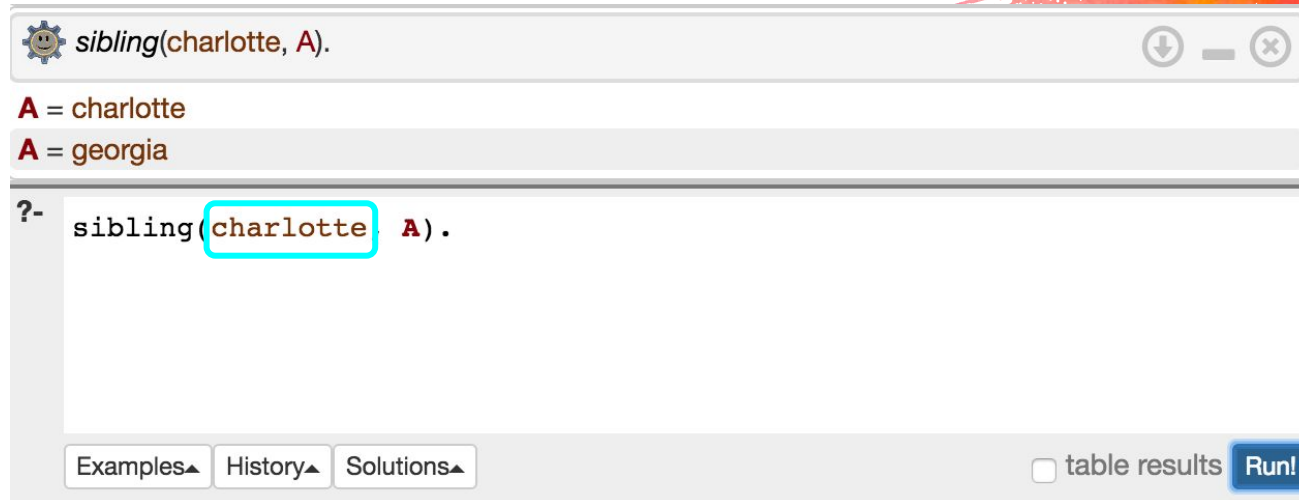
Solutions▲

☐ table results

Run!

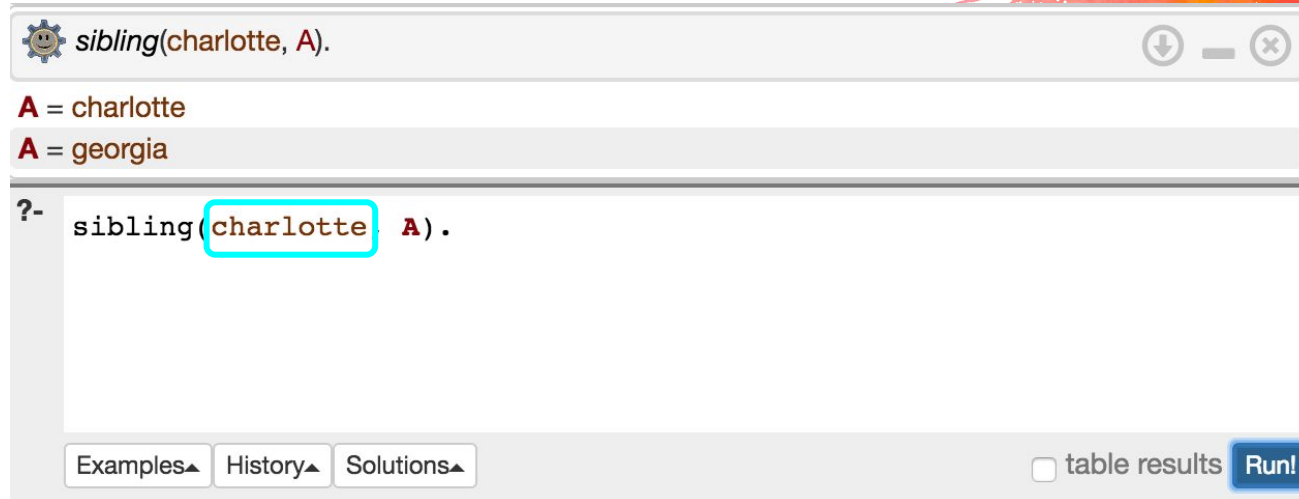
Unification


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



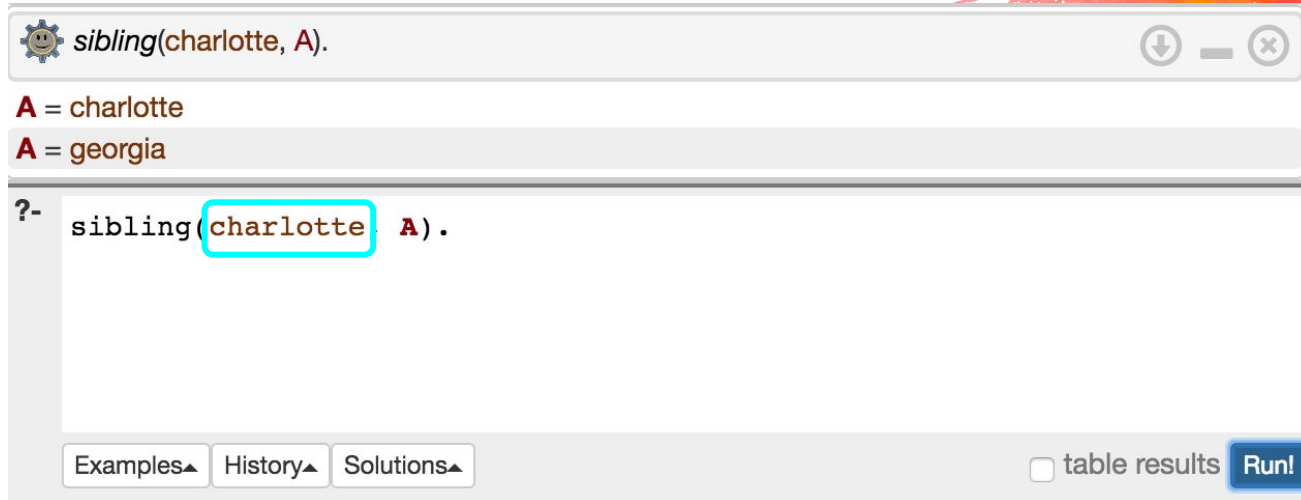
The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `sibling(charlotte, A).`. The window has standard minimize, maximize, and close buttons. Below the title bar, the results of the query are displayed as `A = charlotte` and `A = georgia`. The main area of the window shows the prompt `?- sibling(charlotte, A).` with `charlotte` highlighted by a red box. At the bottom of the window, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

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



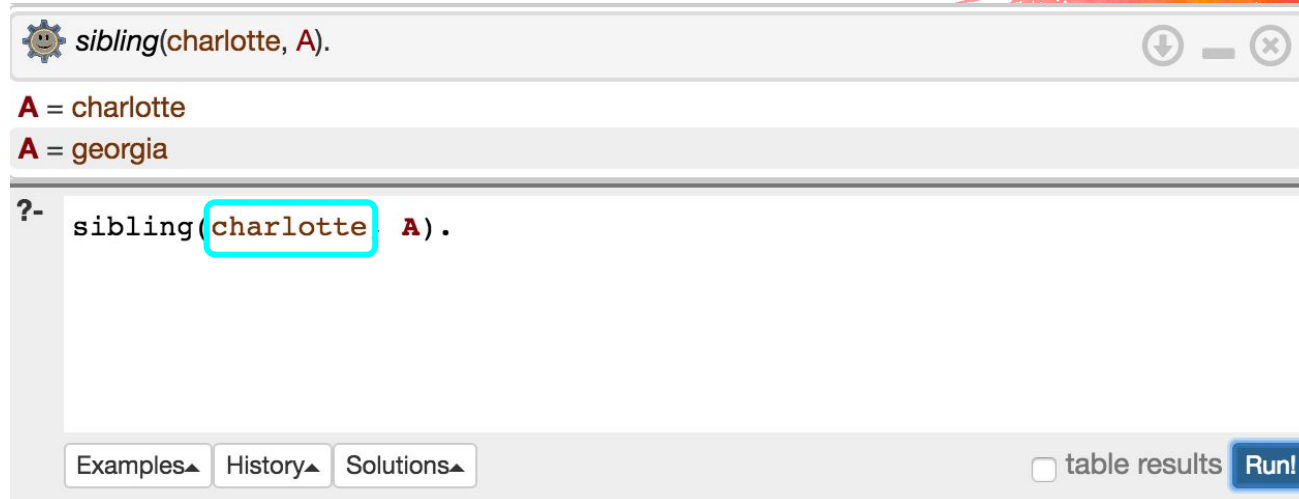
The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `sibling(charlotte, A).`. The window has standard window controls (download, minimize, close) on the right. The main area displays the results of the query: `A = charlotte` and `A = georgia`. Below the results, the prompt `?-` is followed by the query `sibling(charlotte, A).`, where `charlotte` is highlighted with a red box. At the bottom, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

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



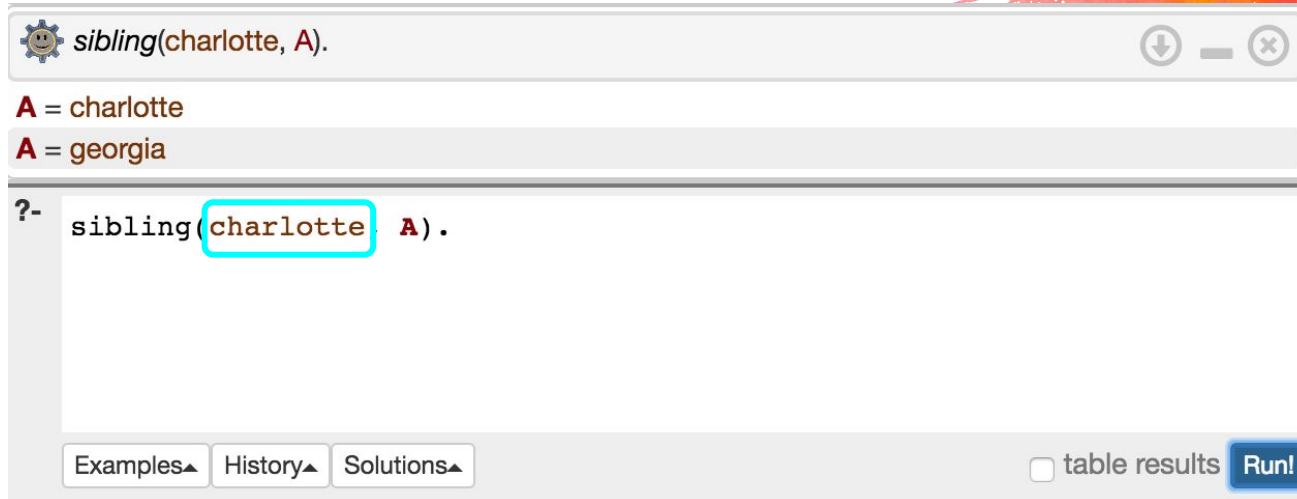
The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `sibling(charlotte, A).`. The window has standard window controls (download, minimize, close) on the right. Below the title bar, the results of the query are displayed: `A = charlotte` and `A = georgia`. The main area of the window shows the query `?- sibling(charlotte, A).` with `charlotte` highlighted in a cyan box. At the bottom of the window, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

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



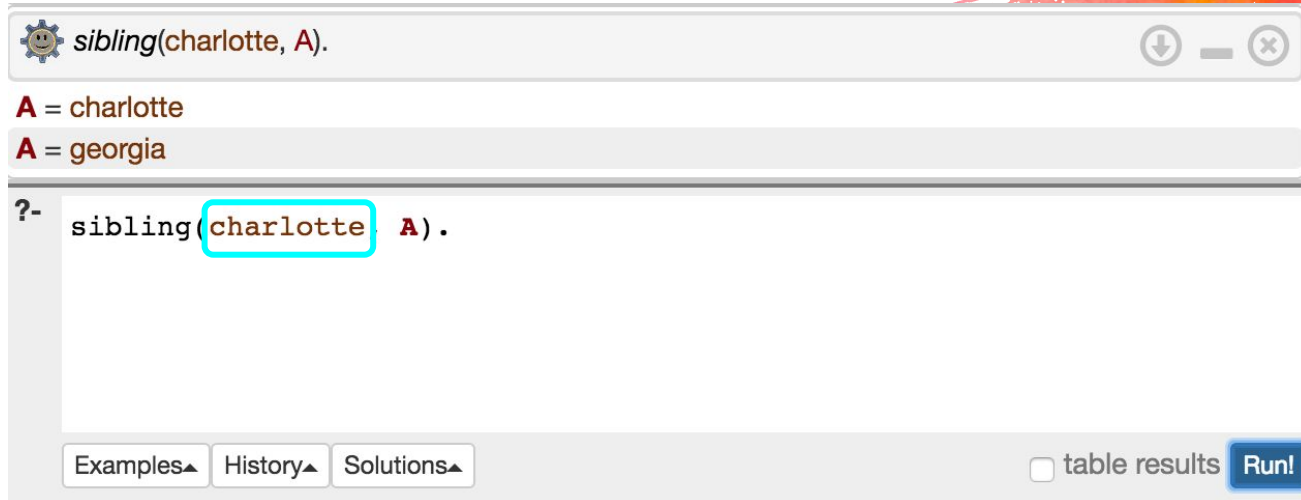
The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `sibling(charlotte, A).`. The window has standard window controls (download, minimize, close) on the right. The main area displays the results of the query: `A = charlotte` and `A = georgia`. Below the results, there is a prompt `?-` followed by the query `sibling(charlotte, A).`, where `charlotte` is highlighted with a red box. At the bottom of the window, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

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



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `sibling(charlotte, A).`. The window has standard window controls (download, minimize, close) on the right. Below the title bar, the results of the query are displayed: `A = charlotte` and `A = georgia`. The main area of the window shows the query `?- sibling(charlotte, A).` with `charlotte` highlighted in a cyan box. At the bottom of the window, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

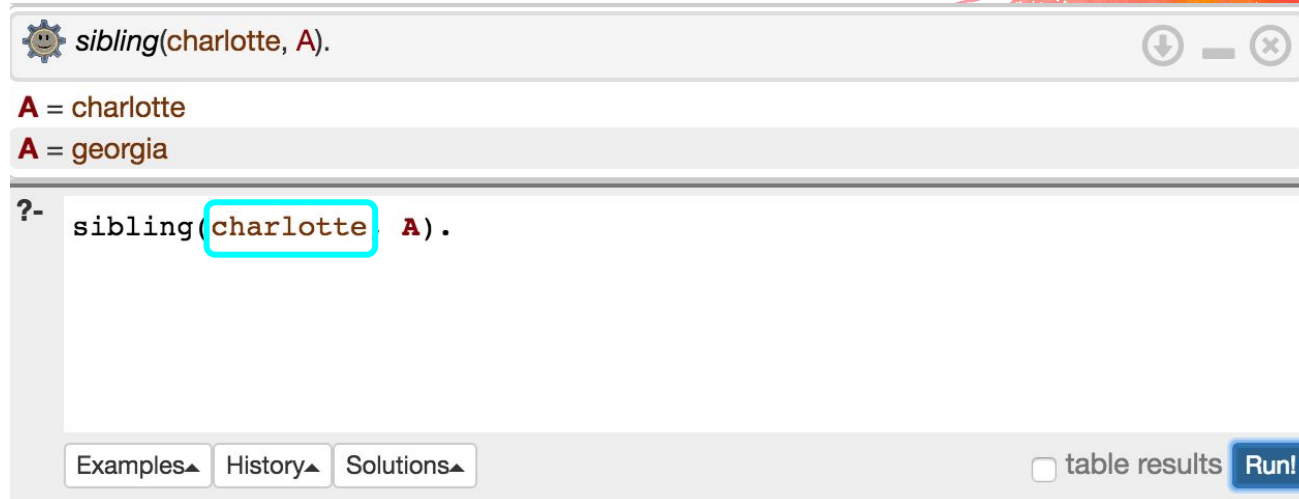

```
1 mother(karen, charlotte).  
2 mother(karen, georgia).  
3 sibling(X, Y):- mother(Z, X), mother(Z, Y).
```



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon, the text `sibling(charlotte, A).`, and standard window controls (download, minimize, close). The main area displays the results of the query: `A = charlotte` and `A = georgia`. Below this, the prompt `?-` is followed by the query `sibling(charlotte, A).`, where `charlotte` is highlighted with a red box. At the bottom, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

Backtracking

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



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon, the text `sibling(charlotte, A).`, and standard window controls (download, minimize, close). The main area displays the results of a query: `A = charlotte` and `A = georgia`. Below this, the query prompt `?- sibling(charlotte, A).` is shown, with `charlotte` highlighted in a red box. At the bottom, there are three buttons: `Examples▲`, `History▲`, and `Solutions▲`. To the right of these buttons is a checkbox labeled `table results` and a blue button labeled `Run!`.

```
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).
```



grandmother(barbara, charlotte)

true



grandmother(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).
```



grandmother(barbara, karen)

barbara' 'karen**false**



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

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



`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

Karen

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

Charlotte



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



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)
```

Anonymous
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	✗	◯	✗	✗	✗	✗	✗	◯
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	✖	○	✖	✖				






```
1 class house:
2     def __init__(self, color):
3         self.color = color
4         self.possible_pets = ['cat', 'dog', 'bird', 'bunny']
5         self.possible_toys = ['trampoline', 'pool', 'scooter', 'basketball hoop']
6
7     def is_done(self):
8         if len(self.possible_pets) == 1 and len(self.possible_toys) == 1:
9             return True
10        else:
11            return False
12
13    def remove_pet(self, pet):
14        if pet in self.possible_pets:
15            self.possible_pets.remove(pet)
16
17    def remove_toy(self, toy):
18        if toy in self.possible_toys:
19            self.possible_toys.remove(toy)
20
```



```
21 class neighborhood:
22     def __init__(self):
23         self.houses = [house('red'), house('blue'), house('yellow'), house('pink')]
24
25     def is_valid(self):
26         for house in self.houses:
27             if not house.is_done():
28                 return False
29
30         for i in self.houses:
31             for j in self.houses:
32                 if i.color != j.color:
33                     if i.possible_pets[0] == j.possible_pets[0]:
34                         return False
35
36                     if i.possible_toys[0] == j.possible_pets[0]:
37                         return False
38         return True
39
40     def normalize(self):
41         # if you've found the correct pet or toy for a house
42         # remove that from others
43         for house in self.houses:
44             if len(house.possible_pets) == 1:
45                 for otherhouse in self.houses:
46                     if house.color != otherhouse.color:
47                         otherhouse.remove_pet(house.possible_pets[0])
48
49         for house in self.houses:
50             if len(house.possible_toys) == 1:
51                 for otherhouse in self.houses:
52                     if house.color != otherhouse.color:
53                         otherhouse.remove_toy(house.possible_toys[0])
```

```
61 neighborhood = neighborhood()
62 cycle_count = 0
63
64 while not neighborhood.is_valid():
65     print("Cycle count: " + str(cycle_count))
66     cycle_count += 1
67     # clue 1 A - the bunny doesn't live in the red or yellow houses
68     for house in neighborhood.houses:
69         if house.color in ['red', 'yellow']:
70             house.remove_pet('bunny')
71
72     # clue 1 B - the bunny doesn't live with either the pool or the scooter
73     for house in neighborhood.houses:
74         if len(house.possible_pets) == 1 and house.possible_pets[0] == 'bunny':
75             house.remove_toy('pool')
76             house.remove_toy('scooter')
77
78     # clue 2 - the red house doesn't have the cat but has the basketball hoop
79     for house in neighborhood.houses:
80         if house.color == 'red':
81             house.remove_pet('cat')
82             house.possible_toys = ['basketball hoop']
83         else:
84             house.remove_toy('basketball hoop')
```

```
85
86     # clue 3 - the pink house has the bird
87     for house in neighborhood.houses:
88         if house.color == 'pink':
89             house.possible_pets = ['bird']
90
91     # clue 4 - the cat lives with the pool
92     for house in neighborhood.houses:
93         if len(house.possible_pets) == 1 and house.possible_pets[0] == 'cat':
94             house.possible_toys = ['pool']
95
96
97 neighborhood.normalize()
98
99 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/>




```
1 pet(Pet):- member(Pet, [cat, dog, bird, bunny]).
2 valid_pets(P1, P2, P3, P4):- pet(P1), pet(P2), pet(P3), pet(P4),
3     all_different(P1, P2, P3, P4).
4
5 toy(Toy):- member(Toy, [trampoline, pool, scooter, 'basketball hoop']).
6 valid_toys(T1, T2, T3, T4):- toy(T1), toy(T2), toy(T3), toy(T4),
7     all_different(T1, T2, T3, T4).
8
9 all_different(A, B, C, D):-
10     A \= B, A \= C, A \= D,
11     B \= C, B \= D,
12     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).

```



solution(Houses).



```

Houses = [[red, dog, 'basketball hoop'], [blue, bunny, trampoline], [yellow, cat, pool], [pink, bird,
scooter]]
false

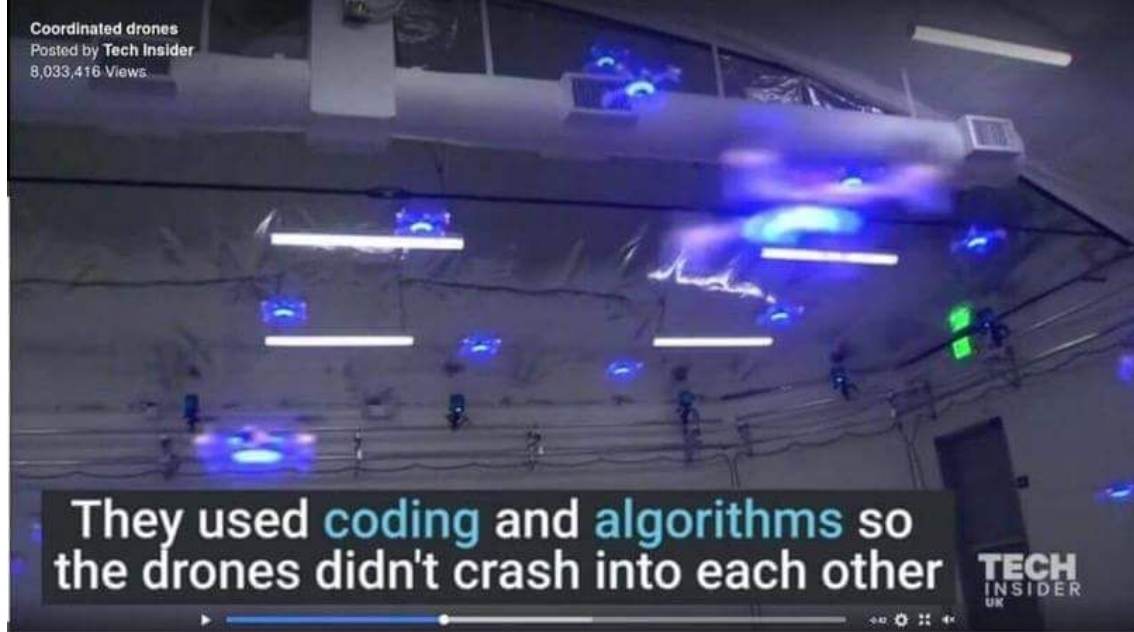
```



?- solution(Houses).



Coordinated drones
Posted by Tech Insider
8,033,416 Views



They used **coding** and **algorithms** so
the drones didn't crash into each other

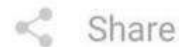
TECH
INSIDER
UK



16k



485



Share

≡ BEST

u/Skizm • 2mo

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



So why don't we teach
Prolog?

EXAMPLE



Hey!



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



The background of the slide features abstract, painterly splatters in shades of orange, red, and purple, primarily concentrated on the right side and bottom right corner, creating a dynamic and artistic feel.

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://tylermcginis.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