Challenge September 2026
Zebra Puzzle

A solution with DT5GL/ Python/Claude
by Jack Jansonius – 23 September 2026

Problem Statement (from the website):

Zebra Puzzle
September 2026

The following puzzle often called **Einstein's Riddle.** It appeared in *Life International* in 1962:

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Question: Where is the zebra?
Source: https://en.wikipedia.org/wiki/Zebra_Puzzle

Problem analysis

This is typically the sort of puzzle I wouldn't have attempted in the past, that is, before the arrival of AI/LLMs. Now, it takes very little effort to ask an LLM such as Claude, using a very short prompt, to generate a solution to this challenge and then see if I can convert that code into decision tables.

The prompt was in Dutch and consisted of nothing more than:

"Please provide a solution in Python for the challenge you can find here: https://dmcommunity.org/challenges/2026-09-zebra-puzzle/

Can you solve this using a traditional Python programme with lots of nested loops, but without using all sorts of complicated libraries? "

The great thing about the Python programme generated by Claude[1] was that, after some puzzling, I also worked out how the programme arrives at a solution to the challenge.

The key trick behind the solution is to create a list of all possible permutations for the 5 houses, which are then applied in turn to the 5 categories in the following order: colour → nationality → drink → cigarette → pet.

Now, 3 houses would yield 3! = 6 permutations:

```
from itertools import permutations

list(permutations([1,2,3]))
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

and 4 houses 4! = 24 permutations:

```
list(permutations([1,2,3,4]))
[(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), (1, 3, 4, 2), (1, 4, 2, 3), (1, 4, 3, 2), (2, 1, 3, 4),
(2, 1, 4, 3), (2, 3, 1, 4), (2, 3, 4, 1), (2, 4, 1, 3), (2, 4, 3, 1), (3, 1, 2, 4), (3, 1, 4, 2),
(3, 2, 1, 4), (3, 2, 4, 1), (3, 4, 1, 2), (3, 4, 2, 1), (4, 1, 2, 3), (4, 1, 3, 2), (4, 2, 1, 3),
(4, 2, 3, 1), (4, 3, 1, 2), (4, 3, 2, 1)]
```

Next, the required 5 houses with 5! = 120 permutations, and only the first 5:

```
list(permutations([1,2,3,4,5]))
[(1, 2, 3, 4, 5), (1, 2, 3, 5, 4), (1, 2, 4, 3, 5), (1, 2, 4, 5, 3), (1, 2, 5, 3, 4), …
```

Based on this, the pseudocode for the algorithm is easy to understand:

[1] See: https://github.com/JackJansonius/DT5GL/blob/main/zebra.py

```
# ALGORITHM (pseudocode)

ALL_PERMS = every possible way to assign the 5 house numbers {1,2,3,4,5}
        to 5 "slots", i.e. all 120 orderings of [1,2,3,4,5]

FOR each permutation assigned to (red, green, ivory, yellow, blue):

   IF  green is immediately right of ivory:       # prune early, clue 6
      check next category

   FOR each permutation assigned to (englishman, spaniard, ukrainian,
                              norwegian, japanese):
      IF     englishman's house-number = red's house-number    # clue 2
      AND  norwegian's house-number =   1                      # clue 10
      AND  Norwegian and blue are next to each other:          # clue 15
           check next category

      FOR each permutation assigned to (coffee, tea, milk, oj, water):
         IF     coffee's house-number = green's house-number      # clue 4
         AND  tea's house-number    = ukrainian's house-number  # clue 5
         AND  milk's house-number   = 3:                         # clue 9
              check next category

         FOR each permutation assigned to (old_gold, kools,chesterfields,
                                           lucky_strike, parliaments):
            IF     kools's house-number = yellow's house-number              # clue 8
            AND  lucky_strike's house-number = oj's house-number             # clue 13
            AND  parliaments's house-number = japanese's house-number:       # clue 14
                 check next category

            FOR each permutation assigned to (dog, snails, fox,horse, zebra):
               IF    dog's house-number = spaniard's house-number            # clue 3
               AND snails's house-number = old_gold's house-number           # clue 7
               AND chesterfields and fox are next to each other              # clue 11
               AND kools and horse are next to each other:                   # clue 12

               # All 15 clues satisfied -> we have a full solution.
               # 'solution' is a flat record of 25 numbers, each
               # answering "which house does this attribute live in?"
               RECORD solution = {
                  red, green, ivory, yellow, blue,
                  englishman, spaniard, ukrainian, norwegian, japanese,
                  coffee, tea, milk, oj, water,
                  old_gold, kools, chesterfields,
                  lucky_strike, parliaments,
                  dog, snails, fox, horse, zebra
               }
```

Solution in DT5GL/Python:

```
Initial_instructions:
>>: last_perm = get_num_perms() - 1
>>: True =  1
End_Instructions


rTable 1: Check first categorie (colours) or done
If:                                          | 0| 1| 2|
next i1 in [0 - last_perm]                   | Y| Y| N|
True = right_of(green, ivory)                | Y| N| -|
Then:
Category1 is selected                        | X|  |  |
Category1 is check_next                      |  | X|  |
Category1 is finished                        |  |  | X|
# .......
# 6. The green house is immediately to the right of the ivory house


rTable 2: Check second categorie (nationalities)
If:                                          | 0| 1| 2| 3|
Category1.getvalue is selected               | Y| Y| Y| N|[2]
next i2 in [0 - last_perm]                   | Y| Y| N| -|
True = englishman == red                     | Y| -| -| -|
True = norwegian == 1                        | Y| -| -| -|
True = next_to(norwegian, blue)              | Y| -| -| -|
Then:
Category2 is selected                        | X|  |  |  |
Category2 is check_next                      |  | X|  |  |
Category2 is finished                        |  |  | X| X|

# .......
# 2. The Englishman lives in the red house
# 10. The Norwegian lives in the first house
# 15. The Norwegian lives next to the blue house


rTable 3: Check third categorie (drinks)
If:                                          | 0| 1| 2| 3|
Category2.getvalue is selected               | Y| Y| Y| N|
next i3 in [0 - last_perm]                   | Y| Y| N| -|
True = coffee == green                       | Y| -| -| -|
True = tea == ukrainian                      | Y| -| -| -|
True = milk == 3                             | Y| -| -| -|
Then:
Category3 is selected                        | X|  |  |  |
Category3 is check_next                      |  | X|  |  |
Category3 is finished                        |  |  | X| X|
# .......
# 4. Coffee is drunk in the green house
# 5. The Ukrainian drinks tea
# 9. Milk is drunk in the middle house
```

2 A goal attribute cannot appear in a condition (as this would mean it ceases to be a goal attribute). This is prevented by adding .getvalue to the attribute name.

```
rTable 4: Check fourth categorie (smokes)
If:                                          | 0| 1| 2| 3|
Category3.getvalue is selected               | Y| Y| Y| N|
next i4 in [0 - last_perm]                   | Y| Y| N| -|
True = kools == yellow                       | Y| -| -| -|
True = lucky_strike == oj                    | Y| -| -| -|
True = parliaments == japanese               | Y| -| -| -|
Then:
Category4 is selected                        | X|  |  |  |
Category4 is check_next                      |  | X|  |  |
Category4 is finished                        |  |  | X| X|
# .......
# 8. Kools are smoked in the yellow house
# 13. The Lucky Strike smoker drinks orange juice
# 14. The Japanese smokes Parliaments


rTable 5: Check fifth categorie (pets)
If:                                          | 0| 1| 2| 3|
Category4.getvalue is selected               | Y| Y| Y| N|
next i5 in [0 - last_perm]                   | Y| Y| N| -|
True = dog == spaniard                       | Y| -| -| -|
True = snails == old_gold                    | Y| -| -| -|
True = next_to(chesterfields, fox)           | Y| -| -| -|
True = next_to(kools, horse)                 | Y| -| -| -|
Then:
Category5 is selected                        | X|  |  |  |
Category5 is check_next                      |  | X|  |  |
Category5 is finished                        |  |  | X| X|
# .......
# 3. The Spaniard owns the dog
# 7. The Old Gold smoker owns snails
# 11. The man who smokes Chesterfields lives in the house next to the man with the
fox
# 12. Kools are smoked in the house next to the house where the horse is kept

Attribute: fn         Type: Integer
Attribute: last_perm  Type: Integer


# red, green, ivory, yellow, blue
Attribute: red             Type: Integer
Equals: get_perm(i1,0)
Attribute: green           Type: Integer
Equals: get_perm(i1,1)
Attribute: ivory           Type: Integer
Equals: get_perm(i1,2)
Attribute: yellow          Type: Integer
Equals: get_perm(i1,3)
Attribute: blue            Type: Integer
Equals: get_perm(i1,4)

# englishman, spaniard, ukrainian, norwegian, japanese
Attribute: englishman      Type: Integer
Equals: get_perm(i2,0)
Attribute: spaniard        Type: Integer
Equals: get_perm(i2,1)
Attribute: ukrainian       Type: Integer
Equals: get_perm(i2,2)
Attribute: norwegian       Type: Integer
Equals: get_perm(i2,3)
Attribute: japanese        Type: Integer
Equals: get_perm(i2,4)
```

```
# coffee, tea, milk, oj, water
Attribute: coffee          Type: Integer
Equals: get_perm(i3,0)
Attribute: tea             Type: Integer
Equals: get_perm(i3,1)
Attribute: milk            Type: Integer
Equals: get_perm(i3,2)
Attribute: oj              Type: Integer
Equals: get_perm(i3,3)
Attribute: water           Type: Integer
Equals: get_perm(i3,4)

# old_gold, kools, chesterfields, lucky_strike, parliaments
Attribute: old_gold        Type: Integer
Equals: get_perm(i4,0)
Attribute: kools           Type: Integer
Equals: get_perm(i4,1)
Attribute: chesterfields   Type: Integer
Equals: get_perm(i4,2)
Attribute: lucky_strike    Type: Integer
Equals: get_perm(i4,3)
Attribute: parliaments     Type: Integer
Equals: get_perm(i4,4)

# dog, snails, fox, horse, zebra
Attribute: dog             Type: Integer
Equals: get_perm(i5,0)
Attribute: snails          Type: Integer
Equals: get_perm(i5,1)
Attribute: fox             Type: Integer
Equals: get_perm(i5,2)
Attribute: horse           Type: Integer
Equals: get_perm(i5,3)
Attribute: zebra           Type: Integer
Equals: get_perm(i5,4)


GOALATTRIBUTE: Category1
Repeat_until: finished

Case: finished
>>: fn = print_solutions()

Case: selected
Print: "#REM# - "
Case: check_next
Print: "#REM# - i1 = %s"  i1

GOALATTRIBUTE: Category2
Repeat_until: finished

Case: finished
Print: "#REM# - "
Case: selected
Print: "#REM# - "
Case: check_next
Print: "#REM# - i1 = %s; i2= %s "  i1  i2


GOALATTRIBUTE: Category3
Repeat_until: finished

Case: finished
Print: "#REM# - "
Case: selected
Print: "#REM# - "
Case: check_next
Print: "#REM# - i1 = %s; i2= %s; i3= %s "  i1  i2  i3
```

```
GOALATTRIBUTE: Category4
Repeat_until: finished

Case: finished
Print: "#REM# - "
Case: selected
Print: "#REM# - "
Case: check_next
Print: "#REM# - i1 = %s; i2= %s; i3= %s; i4= %s "  i1  i2  i3 i4


GOALATTRIBUTE: Category5
Repeat_until: finished

Case: finished
Print: "#REM# - "

Case: selected
Print: "Solution found at i1 = %s; i2= %s; i3= %s; i4= %s; i5= %s "  i1 i2 i3 i4 i5
>>: fn = collect_solution_cat1(red, green, ivory, yellow, blue)
>>: fn = collect_solution_cat2(englishman, spaniard, ukrainian, norwegian,
japanese)
>>: fn = collect_solution_cat3(coffee, tea, milk, oj, water)
>>: fn = collect_solution_cat4(old_gold, kools, chesterfields, lucky_strike,
parliaments)
>>: fn = collect_solution_cat5(dog, snails, fox, horse, zebra)
>>: fn = append_solution()

Case: check_next
Print: "#REM# - i1 = %s; i2= %s; i3= %s; i4= %s; i5= %s "  i1 i2 i3 i4 i5
```

Added to DTFunctions.py[3]:

```
######################################################################
# functions for challenge: September 2026 Zebra Puzzle       ######
# https://dmcommunity.org/challenges/2026-09-zebra-puzzle/   ######
######################################################################

from itertools import permutations

# Houses are numbered 1 through 5 (left to right)
HOUSES = [1, 2, 3, 4, 5]

# Precompute all 120 permutations as an array, so we can loop through
# them with an index i instead of iterating directly over the
# permutations object.

ALL_PERMS = list(permutations(HOUSES))
NUM_PERMS = len(ALL_PERMS)

def get_perm(x, y):
    return ALL_PERMS[x][y]

def get_num_perms():
    return NUM_PERMS

def right_of(a, b):
    """a is directly to the right of b"""
    return a == b + 1
```

[3] Almost all the code was taken from Claude.

```
def next_to(a, b):
    """a is directly next to b (left or right)"""
    return abs(a - b) == 1


_cats = {}
solutions = []


def collect_solution_cat1(red, green, ivory, yellow, blue):
    _cats[1] = (red, green, ivory, yellow, blue)
    return 1

def collect_solution_cat2(englishman, spaniard, ukrainian, norwegian, japanese):
    _cats[2] = (englishman, spaniard, ukrainian, norwegian, japanese)
    return 1

def collect_solution_cat3(coffee, tea, milk, oj, water):
    _cats[3] = (coffee, tea, milk, oj, water)
    return 1

def collect_solution_cat4(old_gold, kools, chesterfields, lucky_strike,
parliaments):
    _cats[4] = (old_gold, kools, chesterfields, lucky_strike, parliaments)
    return 1

def collect_solution_cat5(dog, snails, fox, horse, zebra):
    _cats[5] = (dog, snails, fox, horse, zebra)
    return 1

def append_solution():
    # Restore the five tuples to their original variable names...
    red, green, ivory, yellow, blue = _cats[1]
    englishman, spaniard, ukrainian, norwegian, japanese = _cats[2]
    coffee, tea, milk, oj, water = _cats[3]
    old_gold, kools, chesterfields, lucky_strike, parliaments = _cats[4]
    dog, snails, fox, horse, zebra = _cats[5]

    # All 15 clues have now been checked -> valid solution
    solution = {
        'house': HOUSES,
        'red': red, 'green': green, 'ivory': ivory,
        'yellow': yellow, 'blue': blue,
        'englishman': englishman, 'spaniard': spaniard,
        'ukrainian': ukrainian, 'norwegian': norwegian,
        'japanese': japanese,
        'coffee': coffee, 'tea': tea, 'milk': milk,
        'oj': oj, 'water': water,
        'old_gold': old_gold, 'kools': kools,
        'chesterfields': chesterfields,
        'lucky_strike': lucky_strike,
        'parliaments': parliaments,
        'dog': dog, 'snails': snails, 'fox': fox,
        'horse': horse, 'zebra': zebra,
    }
    solutions.append(solution)
    return 1
```

```
def print_solution(sol):
    colors = {v: k for k, v in [('red', sol['red']), ('green', sol['green']),
                                ('ivory', sol['ivory']), ('yellow', sol['yellow']),
                                ('blue', sol['blue'])]}
    nationalities = {v: k for k, v in [('Englishman', sol['englishman']),
                                       ('Spaniard', sol['spaniard']),
                                       ('Ukrainian', sol['ukrainian']),
                                       ('Norwegian', sol['norwegian']),
                                       ('Japanese', sol['japanese'])]}
    drinks = {v: k for k, v in [('coffee', sol['coffee']), ('tea', sol['tea']),
                                ('milk', sol['milk']), ('orange juice', sol['oj']),
                                ('water', sol['water'])]}
    smokes = {v: k for k, v in [('Old Gold', sol['old_gold']),
                                   ('Kools', sol['kools']),
                                   ('Chesterfields', sol['chesterfields']),
                                   ('Lucky Strike', sol['lucky_strike']),
                                   ('Parliaments', sol['parliaments'])]}
    pets = {v: k for k, v in [('dog', sol['dog']), ('snails', sol['snails']),
                                ('fox', sol['fox']), ('horse', sol['horse']),
                                ('zebra', sol['zebra'])]}

    print(f"{'House':<7}{'Color':<8}{'Nationality':<13}{'Drink':<14}{'Smoke':<15}
                                {'Pet':<8}")
    for h in HOUSES:
        print(f"{h:<7}{colors[h]:<8}{nationalities[h]:<13}{drinks[h]:<14}
                {smokes[h]:<15}{pets[h]:<8}")

    zebra_house = sol['zebra']
    nationality = nationalities[zebra_house]
    print(f"\n>>> The {nationality} owns the zebra, in house {zebra_house}.")



def print_solutions():
    global solutions
    print(f"Number of solutions found: {len(solutions)}\n")
    for sol in solutions:
        print_solution(sol)
    return 1
```

Testruns Claude/Python versus DT5GL/Python

python zebra.py

```
Number of solutions found: 1

House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          fox
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    zebra

>>> The Japanese owns the zebra, in house 5.
```

.\dt.exe[4] -source:zebra.txt -nti

```
Solution found at i1 = 70; i2= 62; i3= 105; i4= 48; i5= 84
Number of solutions found: 1

House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          fox
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    zebra

>>> The Japanese owns the zebra, in house 5.
Time elapsed: 0:00:01.886655
```

Asking Claude:
If I disable one clue in the Python programme (by temporarily commenting it out), this yields more solutions. Could you find a clue for me which, when disabled, yields between 2 and 8 solutions?

Answer:
I've found three that fall exactly within that range (2 to 8):

Clue Text Number of solutions when eliminated
4 Coffee is drunk in the green house 8
9 Milk is drunk in the middle house 6
11 The Chesterfield smoker lives next door to the man with the fox 2

[4] DT.exe is Python code (Version 3.6), compiled to C, and runs directly under Windows (without pre-installation of Python). Download and all necessary files available at: https://github.com/JackJansonius/DT5GL

```
# 11. The Chesterfield smoker lives next to the man with the fox
# if not next_to(chesterfields, fox):
    # continue
```

python zebra.py

```
Number of solutions found: 2

House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          fox
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    zebra

>>> The Japanese owns the zebra, in house 5.
House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          zebra
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    fox

>>> The Norwegian owns the zebra, in house 1.



In decision table 5:
True = next_to(chesterfields, fox)             | -| -| -| -|
```

.\dt.exe -source:zebra.txt -nti

```
Solution found at i1 = 70; i2= 62; i3= 105; i4= 48; i5= 84
Solution found at i1 = 70; i2= 62; i3= 105; i4= 48; i5= 89
Number of solutions found: 2

House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          fox
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    zebra

>>> The Japanese owns the zebra, in house 5.
House  Color   Nationality  Drink         Smoke          Pet
1      yellow  Norwegian    water         Kools          zebra
2      blue    Ukrainian    tea           Chesterfields  horse
3      red     Englishman   milk          Old Gold       snails
4      ivory   Spaniard     orange juice  Lucky Strike   dog
5      green   Japanese     coffee        Parliaments    fox

>>> The Norwegian owns the zebra, in house 1.
Time elapsed: 0:00:01.780485
```