#38

J'ACCUSE!

You are the world-famous detective Mathilde Camus. Zophie the cat has gone missing, and you must sift through the clues. Suspects either always tell lies or always tell the truth. Will you find Zophie the cat in time and accuse the guilty party?

In this game, you take a taxi to different locations around the city. At each location is a suspect and an item. You can ask suspects about other suspects and items, compare their answers with your own exploration notes, and determine if they are lying or telling the truth. Some will know who has catnapped Zophie (or where she is, or what item is found at the location of the kidnapper), but you must determine if you can believe them. You have five minutes to find the criminal but will lose if you make three wrong accusations. This game is inspired by Homestar Runner's "Where's an Egg?" game.

The Program in Action

When you run *jaccuse.py*, the output will look like this:

```
J'ACCUSE! (a mystery game)
--snip--
Time left: 5 min, 0 sec
  You are in your TAXI. Where do you want to go?
(A)LBINO ALLIGATOR PIT
(B)OWLING ALLEY
(C)ITY HALL
(D)UCK POND
(H)IPSTER CAFE
(O)LD BARN
(U)NIVERSITY LIBRARY
(V)IDEO GAME MUSEUM
(Z)OO
> a

Time left: 4 min, 48 sec
  You are at the ALBINO ALLIGATOR PIT.
  ESPRESSA TOFFEEPOT with the ONE COWBOY BOOT is here.

(J) "J'ACCUSE!" (3 accusations left)
(Z) Ask if they know where ZOPHIE THE CAT is.
(T) Go back to the TAXI.
(1) Ask about ESPRESSA TOFFEEPOT
(2) Ask about ONE COWBOY BOOT
> z
  They give you this clue: "DUKE HAUTDOG"
Press Enter to continue...
--snip--
```

How It Works

To fully understand this program, you should pay close attention to the `clues` dictionary, which is set up on lines 51 to 109. You can uncomment lines 151 to 154 to display it on the screen. This dictionary has strings from the `SUSPECTS` list for the keys and "clue dictionaries" for the values. Each of these clue dictionaries contains strings from `SUSPECTS` and `ITEMS`. The original suspect will answer with these strings when asked about another suspect or item. For example, if `clues['DUKE HAUTDOG']['CANDLESTICK']` is set to `'DUCK POND'`, then when the player asks Duke Hautdog about the Candlestick, they'll say it is at the Duck Pond. The suspects, items, locations, and culprit get shuffled each time the game is played.

The code for this program revolves around this data structure, so understanding it is necessary to unlocking your understanding of the rest of the program.

```
1. """J'ACCUSE!, by Al Sweigart al@inventwithpython.com
2. A mystery game of intrigue and a missing cat.
3. View this code at https://nostarch.com/big-book-small-python-projects
```

```
 4. Tags: extra-large, game, humor, puzzle"""
 5.
 6. # Play the original Flash game at:
 7. # https://homestarrunner.com/videlectrix/wheresanegg.html
 8. # More info at: http://www.hrwiki.org/wiki/Where's_an_Egg%3F
 9.
10. import time, random, sys
11.
12. # Set up the constants:
13. SUSPECTS = ['DUKE HAUTDOG', 'MAXIMUM POWERS', 'BILL MONOPOLIS', 'SENATOR SCHMEAR',
    'MRS. FEATHERTOSS', 'DR. JEAN SPLICER', 'RAFFLES THE CLOWN', 'ESPRESSA TOFFEEPOT',
    'CECIL EDGAR VANDERTON']
14. ITEMS = ['FLASHLIGHT', 'CANDLESTICK', 'RAINBOW FLAG', 'HAMSTER WHEEL', 'ANIME VHS TAPE',
    'JAR OF PICKLES', 'ONE COWBOY BOOT', 'CLEAN UNDERPANTS', '5 DOLLAR GIFT CARD']
15. PLACES = ['ZOO', 'OLD BARN', 'DUCK POND', 'CITY HALL', 'HIPSTER CAFE', 'BOWLING ALLEY',
    'VIDEO GAME MUSEUM', 'UNIVERSITY LIBRARY', 'ALBINO ALLIGATOR PIT']
16. TIME_TO_SOLVE = 300  # 300 seconds (5 minutes) to solve the game.
17.
18. # First letters and longest length of places are needed for menu display:
19. PLACE_FIRST_LETTERS = {}
20. LONGEST_PLACE_NAME_LENGTH = 0
21. for place in PLACES:
22.     PLACE_FIRST_LETTERS[place[0]] = place
23.     if len(place) > LONGEST_PLACE_NAME_LENGTH:
24.         LONGEST_PLACE_NAME_LENGTH = len(place)
25.
26. # Basic sanity checks of the constants:
27. assert len(SUSPECTS) == 9
28. assert len(ITEMS) == 9
29. assert len(PLACES) == 9
30. # First letters must be unique:
31. assert len(PLACE_FIRST_LETTERS.keys()) == len(PLACES)
32.
33.
34. knownSuspectsAndItems = []
35. # visitedPlaces: Keys=places, values=strings of the suspect & item there.
36. visitedPlaces = {}
37. currentLocation = 'TAXI'  # Start the game at the taxi.
38. accusedSuspects = []  # Accused suspects won't offer clues.
39. liars = random.sample(SUSPECTS, random.randint(3, 4))
40. accusationsLeft = 3  # You can accuse up to 3 people.
41. culprit = random.choice(SUSPECTS)
42.
43. # Common indexes link these; e.g. SUSPECTS[0] and ITEMS[0] are at PLACES[0].
44. random.shuffle(SUSPECTS)
45. random.shuffle(ITEMS)
46. random.shuffle(PLACES)
47.
48. # Create data structures for clues the truth-tellers give about each
49. # item and suspect.
50. # clues: Keys=suspects being asked for a clue, value="clue dictionary".
51. clues = {}
52. for i, interviewee in enumerate(SUSPECTS):
53.     if interviewee in liars:
54.         continue  # Skip the liars for now.
```

```
55.
56.     # This "clue dictionary" has keys=items & suspects,
57.     # value=the clue given.
58.     clues[interviewee] = {}
59.     clues[interviewee]['debug_liar'] = False  # Useful for debugging.
60.     for item in ITEMS:  # Select clue about each item.
61.         if random.randint(0, 1) == 0:  # Tells where the item is:
62.             clues[interviewee][item] = PLACES[ITEMS.index(item)]
63.         else:  # Tells who has the item:
64.             clues[interviewee][item] = SUSPECTS[ITEMS.index(item)]
65.     for suspect in SUSPECTS:  # Select clue about each suspect.
66.         if random.randint(0, 1) == 0:  # Tells where the suspect is:
67.             clues[interviewee][suspect] = PLACES[SUSPECTS.index(suspect)]
68.         else:  # Tells what item the suspect has:
69.             clues[interviewee][suspect] = ITEMS[SUSPECTS.index(suspect)]
70.
71. # Create data structures for clues the liars give about each item
72. # and suspect:
73. for i, interviewee in enumerate(SUSPECTS):
74.     if interviewee not in liars:
75.         continue  # We've already handled the truth-tellers.
76.
77.     # This "clue dictionary" has keys=items & suspects,
78.     # value=the clue given:
79.     clues[interviewee] = {}
80.     clues[interviewee]['debug_liar'] = True  # Useful for debugging.
81.
82.     # This interviewee is a liar and gives wrong clues:
83.     for item in ITEMS:
84.         if random.randint(0, 1) == 0:
85.             while True:  # Select a random (wrong) place clue.
86.                 # Lies about where the item is.
87.                 clues[interviewee][item] = random.choice(PLACES)
88.                 if clues[interviewee][item] != PLACES[ITEMS.index(item)]:
89.                     # Break out of the loop when wrong clue is selected.
90.                     break
91.         else:
92.             while True:  # Select a random (wrong) suspect clue.
93.                 clues[interviewee][item] = random.choice(SUSPECTS)
94.                 if clues[interviewee][item] != SUSPECTS[ITEMS.index(item)]:
95.                     # Break out of the loop when wrong clue is selected.
96.                     break
97.     for suspect in SUSPECTS:
98.         if random.randint(0, 1) == 0:
99.             while True:  # Select a random (wrong) place clue.
100.                clues[interviewee][suspect] = random.choice(PLACES)
101.                if clues[interviewee][suspect] != PLACES[ITEMS.index(item)]:
102.                    # Break out of the loop when wrong clue is selected.
103.                    break
104.        else:
105.            while True:  # Select a random (wrong) item clue.
106.                clues[interviewee][suspect] = random.choice(ITEMS)
107.                if clues[interviewee][suspect] != ITEMS[SUSPECTS.index(suspect)]:
108.                    # Break out of the loop when wrong clue is selected.
109.                    break
```

```
110.
111. # Create the data structures for clues given when asked about Zophie:
112. zophieClues = {}
113. for interviewee in random.sample(SUSPECTS, random.randint(3, 4)):
114.     kindOfClue = random.randint(1, 3)
115.     if kindOfClue == 1:
116.         if interviewee not in liars:
117.             # They tell you who has Zophie.
118.             zophieClues[interviewee] = culprit
119.         elif interviewee in liars:
120.             while True:
121.                 # Select a (wrong) suspect clue.
122.                 zophieClues[interviewee] = random.choice(SUSPECTS)
123.                 if zophieClues[interviewee] != culprit:
124.                     # Break out of the loop when wrong clue is selected.
125.                     break
126.
127.     elif kindOfClue == 2:
128.         if interviewee not in liars:
129.             # They tell you where Zophie is.
130.             zophieClues[interviewee] = PLACES[SUSPECTS.index(culprit)]
131.         elif interviewee in liars:
132.             while True:
133.                 # Select a (wrong) place clue.
134.                 zophieClues[interviewee] = random.choice(PLACES)
135.                 if zophieClues[interviewee] != PLACES[SUSPECTS.index(culprit)]:
136.                     # Break out of the loop when wrong clue is selected.
137.                     break
138.     elif kindOfClue == 3:
139.         if interviewee not in liars:
140.             # They tell you what item Zophie is near.
141.             zophieClues[interviewee] = ITEMS[SUSPECTS.index(culprit)]
142.         elif interviewee in liars:
143.             while True:
144.                 # Select a (wrong) item clue.
145.                 zophieClues[interviewee] = random.choice(ITEMS)
146.                 if zophieClues[interviewee] != ITEMS[SUSPECTS.index(culprit)]:
147.                     # Break out of the loop when wrong clue is selected.
148.                     break
149.
150. # EXPERIMENT: Uncomment this code to view the clue data structures:
151. #import pprint
152. #pprint.pprint(clues)
153. #pprint.pprint(zophieClues)
154. #print('culprit =', culprit)
155.
156. # START OF THE GAME
157. print("""J'ACCUSE! (a mystery game)")
158. By Al Sweigart al@inventwithpython.com
159. Inspired by Homestar Runner\'s "Where\'s an Egg?" game
160.
161. You are the world-famous detective Mathilde Camus.
162. ZOPHIE THE CAT has gone missing, and you must sift through the clues.
163. Suspects either always tell lies, or always tell the truth. Ask them
164. about other people, places, and items to see if the details they give are
```

```
truthful and consistent with your observations. Then you will know if
their clue about ZOPHIE THE CAT is true or not. Will you find ZOPHIE THE
CAT in time and accuse the guilty party?
""")
input('Press Enter to begin...')


startTime = time.time()
endTime = startTime + TIME_TO_SOLVE

while True:  # Main game loop.
    if time.time() > endTime or accusationsLeft == 0:
        # Handle "game over" condition:
        if time.time() > endTime:
            print('You have run out of time!')
        elif accusationsLeft == 0:
            print('You have accused too many innocent people!')
        culpritIndex = SUSPECTS.index(culprit)
        print('It was {} at the {} with the {} who catnapped her!'.format(culprit,
        PLACES[culpritIndex], ITEMS[culpritIndex]))
        print('Better luck next time, Detective.')
        sys.exit()

    print()
    minutesLeft = int(endTime - time.time()) // 60
    secondsLeft = int(endTime - time.time()) % 60
    print('Time left: {} min, {} sec'.format(minutesLeft, secondsLeft))

    if currentLocation == 'TAXI':
        print('  You are in your TAXI. Where do you want to go?')
        for place in sorted(PLACES):
            placeInfo = ''
            if place in visitedPlaces:
                placeInfo = visitedPlaces[place]
            nameLabel = '(' + place[0] + ')' + place[1:]
            spacing = " " * (LONGEST_PLACE_NAME_LENGTH - len(place))
            print('{} {}{}'.format(nameLabel, spacing, placeInfo))
        print('(Q)UIT GAME')
        while True:  # Keep asking until a valid response is given.
            response = input('> ').upper()
            if response == '':
                continue  # Ask again.
            if response == 'Q':
                print('Thanks for playing!')
                sys.exit()
            if response in PLACE_FIRST_LETTERS.keys():
                break
        currentLocation = PLACE_FIRST_LETTERS[response]
        continue  # Go back to the start of the main game loop.

    # At a place; player can ask for clues.
    print('  You are at the {}.'.format(currentLocation))
    currentLocationIndex = PLACES.index(currentLocation)
    thePersonHere = SUSPECTS[currentLocationIndex]
    theItemHere = ITEMS[currentLocationIndex]
```

```
    print('  {} with the {} is here.'.format(thePersonHere, theItemHere))

    # Add the suspect and item at this place to our list of known
    # suspects and items:
    if thePersonHere not in knownSuspectsAndItems:
        knownSuspectsAndItems.append(thePersonHere)
    if ITEMS[currentLocationIndex] not in knownSuspectsAndItems:
        knownSuspectsAndItems.append(ITEMS[currentLocationIndex])
    if currentLocation not in visitedPlaces.keys():
        visitedPlaces[currentLocation] = '({}, {})'.format(thePersonHere.lower(),
        theItemHere.lower())

    # If the player has accused this person wrongly before, they
    # won't give clues:
    if thePersonHere in accusedSuspects:
        print('They are offended that you accused them,')
        print('and will not help with your investigation.')
        print('You go back to your TAXI.')
        print()
        input('Press Enter to continue...')
        currentLocation = 'TAXI'
        continue  # Go back to the start of the main game loop.

    # Display menu of known suspects & items to ask about:
    print()
    print('(J) "J\'ACCUSE!" ({} accusations left)'.format(accusationsLeft))
    print('(Z) Ask if they know where ZOPHIE THE CAT is.')
    print('(T) Go back to the TAXI.')
    for i, suspectOrItem in enumerate(knownSuspectsAndItems):
        print('({}) Ask about {}'.format(i + 1, suspectOrItem))

    while True:  # Keep asking until a valid response is given.
        response = input('> ').upper()
        if response in 'JZT' or (response.isdecimal() and 0 < int(response) <=
        len(knownSuspectsAndItems)):
            break

    if response == 'J':  # Player accuses this suspect.
        accusationsLeft -= 1  # Use up an accusation.
        if thePersonHere == culprit:
            # You've accused the correct suspect.
            print('You\'ve cracked the case, Detective!')
            print('It was {} who had catnapped ZOPHIE THE CAT.'.format(culprit))
            minutesTaken = int(time.time() - startTime) // 60
            secondsTaken = int(time.time() - startTime) % 60
            print('Good job! You solved it in {} min, {} sec.'.format(minutesTaken,
            secondsTaken))
            sys.exit()
        else:
            # You've accused the wrong suspect.
            accusedSuspects.append(thePersonHere)
            print('You have accused the wrong person, Detective!')
            print('They will not help you with anymore clues.')
            print('You go back to your TAXI.')
            currentLocation = 'TAXI'
```

```
271.
272.     elif response == 'Z':  # Player asks about Zophie.
273.         if thePersonHere not in zophieClues:
274.             print('"I don\'t know anything about ZOPHIE THE CAT."')
275.         elif thePersonHere in zophieClues:
276.             print('  They give you this clue: "{}"'.format(zophieClues[thePersonHere]))
277.             # Add non-place clues to the list of known things:
278.             if zophieClues[thePersonHere] not in knownSuspectsAndItems and
                 zophieClues[thePersonHere] not in PLACES:
279.                 knownSuspectsAndItems.append(zophieClues[thePersonHere])
280.
281.     elif response == 'T':  # Player goes back to the taxi.
282.         currentLocation = 'TAXI'
283.         continue  # Go back to the start of the main game loop.
284.
285.     else:  # Player asks about a suspect or item.
286.         thingBeingAskedAbout = knownSuspectsAndItems[int(response) - 1]
287.         if thingBeingAskedAbout in (thePersonHere, theItemHere):
288.             print('  They give you this clue: "No comment."')
289.         else:
290.             print('  They give you this clue:
                 "{}"'.format(clues[thePersonHere][thingBeingAskedAbout]))
291.             # Add non-place clues to the list of known things:
292.             if clues[thePersonHere][thingBeingAskedAbout] not in knownSuspectsAndItems
                 and clues[thePersonHere][thingBeingAskedAbout] not in PLACES:
293.                 knownSuspectsAndItems.append(clues[thePersonHere][thingBeingAskedAbout])
294.
295.     input('Press Enter to continue...')
```

Exploring the Program

Try to find the answers to the following questions. Experiment with some modifications to the code and rerun the program to see what effect the changes have.

1. What happens if you change `TIME_TO_SOLVE = 300` on line 16 to `TIME_TO_SOLVE = 0`?
2. What happens if you change `time.time() > endTime or accusationsLeft == 0` on line 176 to `time.time() > endTime and accusationsLeft == 0`?
3. What happens if you change `place[1:]` on line 198 to `place`?
4. What happens if you change `startTime + TIME_TO_SOLVE` on line 173 to `startTime * TIME_TO_SOLVE`?