Nested Structures

Containers are objects that store other objects. For example, `list` stores a sequence of objects, and `dict` stores a mapping of objects to objects. Containers can also hold other containers, which makes it possible to represent any type (or shape) of data.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how rows and columns of data can be stored in lists.
- Write nested for loops to iterate data and compute functions.
- Navigate a complex data structure with dictionaries and lists.

Process Skill Goals

During the activity, students should make progress toward:

- Developing algorithms that loop through lists to compute a result. (Problem Solving)

Facilitation Notes

Given the complexity of the of examples, it's important to provide students with the source code in advance. See the accompanying `nested.py` file in the repository. The students should only have to type the expressions in the tables.

On Model 1, you might want to explain that `'R'` and `'Y'` represent Red and Yellow discs. It's less obvious what they mean when printed in black and white. You might also want to point out the optional comma after the last row. Python allows you to put a comma after every item in a list, including the last one, for convenience.

When reporting out Model 1 and Model 3, it's useful to show how these data structures look on Python Tutor (pythontutor.com). You might want to use the "render all objects on the heap (Python/Java)" option to discuss references.

Model 2 explores nested for loops in two ways: first with containers (`groceries` and `grid`) and then with the `range` function. Consider having teams write their solutions on the board, one team per question. If time permits, step through the solutions with a debugger or Python Tutor.


Copyright © 2019 T. Shepherd, C. Mayfield, and H. Hu. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Model 1 Lists of Lists

Connect Four (® Hasbro, Inc.) is a two-player game in which the players take turns dropping colored discs into a six-row by seven-column grid. The objective of the game is to be the first player to form a horizontal, vertical, or diagonal line of four of one's own discs. (paraphrased from https://en.wikipedia.org/wiki/Connect_Four)


```
# current state of the game
grid = [
    [' ', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', ' '],
    ['Y', ' ', ' ', ' ', 'Y', 'Y', ' '],
    ['R', ' ', ' ', 'Y', 'R', 'R', ' '],
    ['R', 'R', 'Y', 'R', 'Y', 'R', ' '],
    ['R', 'Y', 'R', 'Y', 'Y', 'Y', 'R'],
]
```

Enter the `grid` code above into a Python Shell, and run each line of the table below. If the output is longer than one line, summarize it with a few words.

Python code	Shell output
`print(grid)`	prints the grid without line breaks
`print(grid[5])`	['R', 'Y', 'R', 'Y', 'Y', 'Y', 'R']
`print(grid[5][0])`	R
`type(grid)`	<class 'list'>
`type(grid[5])`	<class 'list'>
`type(grid[5][0])`	<class 'str'>
`len(grid)`	6
`len(grid[5])`	7
`len(grid[5][0])`	1
`import pprint`	
`help(pprint)`	Pretty-print a Python object to a stream
`pprint.pprint(grid)`	prints the grid on multiple lines
`for item in grid:` `  print(item)`	prints each row on a separate line
`for i in range(len(grid)):` `  print(grid[i])`	prints each row on a separate line

Questions (15 min) Start time: _____

1. What does `grid` look like when you first `print` it? (How is the output different from the original format shown in Model 1?)

It is not formatted neatly; the entire list prints with no line breaks.

2. What does `grid` look like when you use `pprint` instead? Explain what `pprint` means.

It looks rectangular and is much easier to read. The word `pprint` stands for "pretty-print".

3. When viewed as a rectangle, how many "rows" and "columns" does `grid` have?

There are six rows and seven columns.

4. What type of object is `grid`? What type of objects does it contain?

`grid` is a list, and its elements are also lists.

5. What type of object is `grid[5]`? What type of objects does it contain?

`grid[5]` is a list, and its elements are strings.

6. In the expression `grid[5][0]`, which index corresponds to the row, and which index corresponds to the column?

The first index [5] is the row, and the second index [0] is the column.

7. Is `grid` a list of rows or a list of columns? Justify your answer.

It is a list of rows; `len(grid)` is 6, and the `for` loop prints rows.

8. Describe how to append one more row to `grid`.

Simply use `grid.append([...])` to add the entire row in one step.

9. What is necessary to append a "column" to `grid`?

We need a `for` loop to append one string at a time to the end of each row.

Model 2 Nested `for` Loops

Example A
We typically use a `for` loop to examine the contents of a list:

```
1 groceries = ["Apples", "Milk", "Flour", "Chips"]
2 for item in groceries:
3     print("Don't forget the", item)
```

Example B
If a list contains another list, we need a `for` loop that contains another `for` loop. For example, to count the "spaces" in the `grid` from Model 1:

```
4 count = 0
5 for row in grid:  # outer loop
6     print("row =", row)
7     for cell in row:  # inner loop
8         print("cell =", cell)
9         if cell == ' ':
10            count += 1
11 print(count, "spaces remaining")
```

Questions (15 min) Start time: _____

10. As a team, discuss the two examples from Model 2. Predict how many times each of the following lines will execute. Then run the code and check your answers based on the output.

a) How many times does Line 3 execute? Predicted: _____ Actual: _____ 4

b) How many times does Line 6 execute? Predicted: _____ Actual: _____ 6

c) How many times does Line 8 execute? Predicted: _____ Actual: _____ 42

d) How many times does Line 10 execute? Predicted: _____ Actual: _____ 22

11. What determined how many times the "`for item`" loop would run? Number of groceries

12. Answer the following questions in terms of `grid`.

    a) What determined how many times the "`for row`" loop would run?

    The number of rows in the grid

    b) What determined how many times the "`for cell`" loop would run?

    The total number of cells in the grid (i.e., number of rows * number of cols)

13. In the example below, predict how many times the `print` statement will execute. Then run the code to verify your answer. 6 * 7 = 42 times

```
for i in range(6):
    for j in range(7):
        print(i, '+', j, '=', i + j)
```

14. Rewrite the nested `for` loops in Model 2 Lines 4–10 using the `range` function. Replace the variables `row` and `cell` with `i` and `j`, respectively. For simplicity, you may omit the `print` statements in your answer.

```
count = 0
for i in range(len(grid)):  # outer loop
    for j in range(len(grid[i])):  # inner loop
        if grid[i][j] == ' ':
            count += 1
```

15. Write a `for` loop (using `range`) that computes the factorial of a given integer n. Recall that $n! = n * (n-1) * (n-2) * \ldots * 1$. Store your result in a variable named `fact`.

```
fact = 1
for i in range(n, 0, -1):
    fact *= i
```

16. Write nested loops that compute and display the factorial of each integer from 1 to 20. Use your code from the previous question as the inner loop. Your output should be in this format:

```
The factorial of 1 is 1
The factorial of 2 is 2
The factorial of 3 is 6
The factorial of 4 is 24
The factorial of 5 is 120
```

```
for n in range(1, 21):
    fact = 1
    for i in range(n, 0, -1):
        fact *= i
    print("The factorial of", n, "is", fact)
```

Model 3 Nested Dictionaries

Containers can be nested in arbitrary ways. For example, the following data could be described as a "dictionary of dictionaries of integers and lists of strings".

Enter the following code into a Python Shell, and complete the table. If the output is longer than one line, summarize it with a few words.

```
movies = {
    "Casablanca": {
        "year": 1942,
        "genres": ["Drama", "Romance", "War"],
    },
    "Star Wars": {
        "year": 1977,
        "genres": ["Action", "Adventure", "Fantasy"],
    },
    "Groundhog Day": {
        "year": 1993,
        "genres": ["Comedy", "Fantasy", "Romance"],
    },
}
```

Python code	Shell output
`movies`	prints all of `movies` without any formatting
`movies["Casablanca"]`	{'genres': ['Drama', 'Romance', 'War'], 'year': 1942}
`movies["Casablanca"]["year"]`	1942
`movies["Casablanca"]["genres"]`	['Drama', 'Romance', 'War']
`type(movies)`	<class 'dict'>
`type(movies["Casablanca"])`	<class 'dict'>
`type(movies["Casablanca"]["year"])`	<class 'int'>
`type(movies["Casablanca"]["genres"])`	<class 'list'>
`len(movies)`	3
`len(movies["Casablanca"])`	2
`len(movies["Casablanca"]["year"])`	TypeError: object of type 'int' has no len()
`len(movies["Casablanca"]["genres"])`	3
`for key in movies:` `  print(key)`	prints the keys: Casablanca, Groundhog Day, Star Wars
`for key, val in movies.items():` `  print(key, val)`	prints each individual movie (the inner dictionaries)

Questions (15 min) Start time: _____

17. Explain the `TypeError` you encountered.

The expression `movies["Casablanca"]["year"]` is an integer, so you can't get the length of it.

18. In the expression `movies["Casablanca"]["genres"]`, describe the purpose of the strings `"Casablanca"` and `"genres"`.

They are keys to their corresponding dictionaries. The first string selects a particular movie, and the second string selects the corresponding movie data.

19. When iterating a dictionary using a `for` loop (i.e., `for x in movies`), what gets assigned to the variable?

The keys of the dictionary.

20. What is wrong with the following code that attempts to `print` each movie?

```
for i in range(len(movies)):
    print(movies[i])
```

You cannot iterate a dictionary by index number; it is not a sequence. Running this code results in `KeyError: 0`.

21. Write nested loops that output every *genre* found under the `movies` dictionary. You should have nine total lines of output.

```
for key in movies:
    movie = movies[key]
    for genre in movie["genres"]:
        print(genre)
```

22. Each movie in Model 3 has a title, a year, and three genres.

a) Is it necessary that all movies have the same format? No

b) Name one advantage of storing data in the same format: It simplifies the code

c) Show how you would represent The LEGO Movie (2014) with a runtime of 100 min and the plot keywords "construction worker" and "good cop bad cop".

```
"The LEGO Movie": {
    "year": 2014,
    "runtime": "100 min",
    "keywords": ["construction worker", "good cop bad cop"],
},
```