

Nested Structures

Containers like lists and dictionaries can hold other containers. This ability allows containers to represent any type (or shape) of data.

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'],
]
```

As a team, discuss the following examples based on the grid list. Run the last three examples in a shell to see the actual output.

Questions (15 min)

Start time:

1. What does grid look like when you first `print` the data? How is the output different from the original format shown in Model 1?
2. What does grid look like when you use `pprint` instead? Explain what `pprint` means.

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

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

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

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

6. In the expression `grid[5][0]`, which index corresponds to the row, and which index corresponds to the column?
7. Is `grid` a list of rows or a list of columns? Justify your answer.
8. Describe how to append one more row to `grid`.
9. What is necessary to append a “column” to `grid`?

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:
- b) How many times does Line 6 execute? Predicted: Actual:
- c) How many times does Line 8 execute? Predicted: Actual:
- d) How many times does Line 10 execute? Predicted: Actual:

11. What determined how many times the “`for item`” loop would run?

12. Answer the following questions in terms of grid.

- a) What determined how many times the “`for row`” loop would run?
- b) What determined how many times the “`for cell`” loop would run?

13. Predict how many times the `print` statement will execute in the example below. Then run the code to verify your answer. Predicted: Actual:

```
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.

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

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
```

Model 3 Nested Dictionaries

Data 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”.

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

As a team, discuss the following examples based on the `movies` dictionary. Run the last two examples in a shell to see the actual output.

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

Questions (15 min)

Start time:

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

18. Explain the `TypeError` encountered near the end of the table.

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

- Is it necessary that all movies have the same format?
- Name one advantage of storing data in the same format:

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”.

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

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

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

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