

Sublists and Nested Loops

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 18, Fall 2020

Monday October 5

Let's see what sublists are in
`lecture_18_sublists.py`

Copying lists

The Independence Rule for list of lists

If slicing returns one or more of the original sublists, the resulting copy is NOT independent of the original list!

Independence Rule violations (1/3)

The following slices return 1 or more LISTS (**sublist**), NOT a basic data type!!!

```
>>> digits = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> middleRow = digits[1] # does this return a sublist?
>>> firstTwo = digits[0:2] # slice. Returns a SUBLIST
>>> whole = digits[:] # Full slice. Returns a LIST
```

- **ALL OF THE DERIVED VARIABLES/LISTS ABOVE ARE DEPENDENT ON digits!!!!**

Let's play around with copying digits in code
(lecture_18_copying_sublists.py sections 1-3).

When Independence Rule is not violated (2/3)

If you take **a slice of** an original sublist (**involving basic data types**), the resulting copy **IS** independent of the original because Python treats it as a different new list.

```
>>> digits = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> middleRow = digits[1][:2] # Items 1 and 2 from 2nd sublist
>>> middleRow[0] = -99 # does NOT affect digits
```

Let's play around with copying digits in code
(lecture_18_copying_sublists.py section 4).

Independence Rule violations (3/3)

When the sublists contain objects, the Independence Rule is also **violated!**

- Why are sublists so tricky? **Lists themselves are objects**, regardless of whether they have sublists or not.
- Let's play around with copying `Turtle` objects in code (`lecture_18_copying_sublists.py` section 5).

Nested loops

Let's explore how they work in `lecture_18_nested_loops.py`