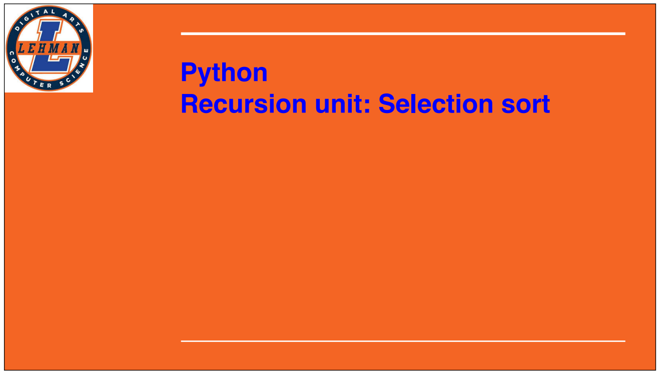
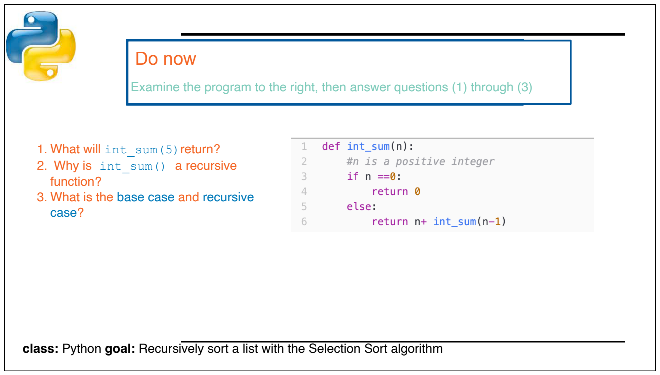


Python
Recursion unit: Selection sort

Material adapted from BJC curriculum: https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/optional-project-sorting/2-selection-sort.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment

New York State Standard:
9-12.CT.6
Demonstrate how at least two classic algorithms work and analyze the trade-offs related to two or more algorithms for completing the same task.

NOTE: This lesson is envisioned as part of a larger unit in which other sorting algorithms are introduced and compared.



Do now

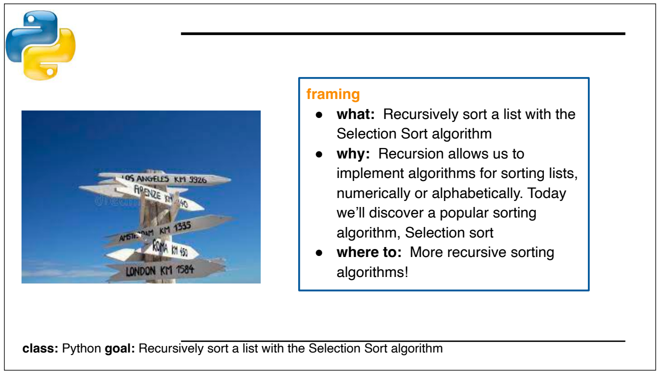
Examine the program to the right, then answer questions (1) through (3)

1. What will `int_sum(5)` return?
2. Why is `int_sum()` a recursive function?
3. What is the base case and recursive case?

```
def int_sum(n):
    #n is a positive integer
    if n ==0:
        return 0
    else:
        return n+ int_sum(n-1)
```

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

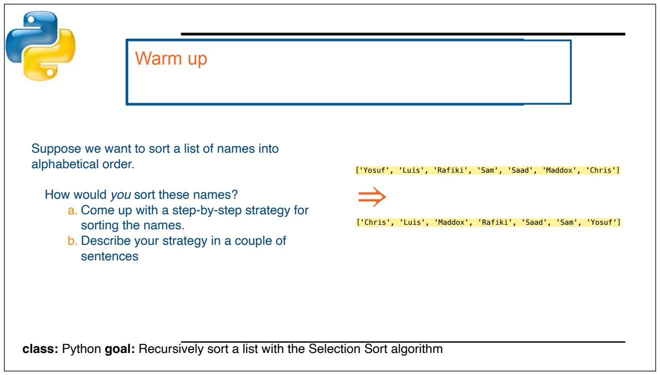
1. 5+4+3+2+1=15
2. because the definition of the function calls itself in the recursive step.
3. the base case is if n==0, then 0 is returned. the recursive step is to add int_sum(n-1) to n.
+Why is it crucial to have a base case? because otherwise the function would continue to call itself.



framing

- **what:** Recursively sort a list with the Selection Sort algorithm
- **why:** Recursion allows us to implement algorithms for sorting lists, numerically or alphabetically. Today we'll discover a popular sorting algorithm, Selection sort
- **where to:** More recursive sorting algorithms!

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm



Warm up

Suppose we want to sort a list of names into alphabetical order.

How would *you* sort these names?

a. Come up with a step-by-step strategy for sorting the names.
b. Describe your strategy in a couple of sentences

```
['Yosuf', 'Luis', 'Rafiki', 'Sam', 'Saad', 'Maddox', 'Chris']
⇒
['Chris', 'Luis', 'Maddox', 'Rafiki', 'Saad', 'Sam', 'Yosuf']
```

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

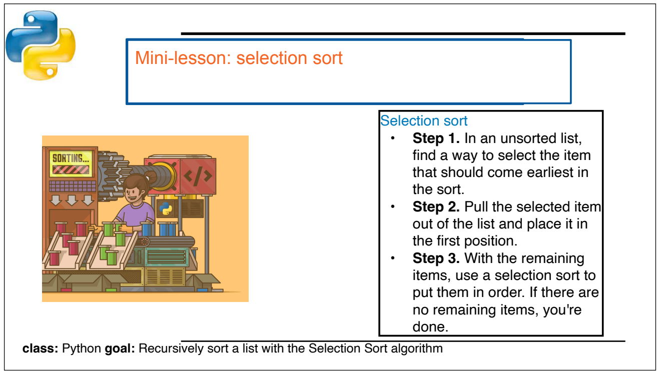
The lists on the screen are Python lists, but don't think about it that way for now. Just think about how you would sort the list in general.

Possible solution: Find the first name in the alphabet. Move it to the front, then continue with the rest of the list (this is basically selection sort).

Another solution: Take the first name ('Yosuf'),
[Yosuf]
then take the next name ('Luis')and order it w/r/t Yosuf
[Luis, Yosuf]

And so on until you get to the end of the list.

This is basically insertion sort.



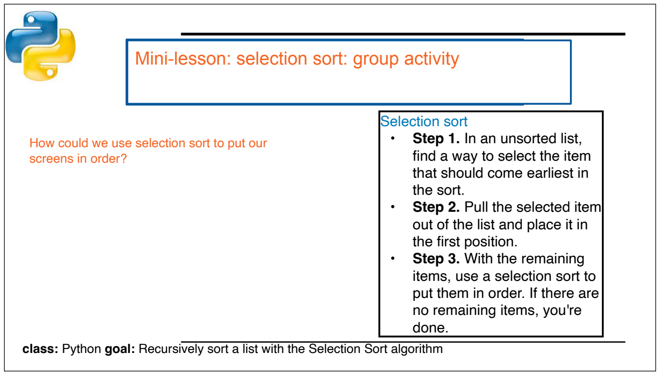
Mini-lesson: selection sort

Selection sort

- **Step 1.** In an unsorted list, find a way to select the item that should come earliest in the sort.
- **Step 2.** Pull the selected item out of the list and place it in the first position.
- **Step 3.** With the remaining items, use a selection sort to put them in order. If there are no remaining items, you're done.

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

Students read aloud.



Mini-lesson: selection sort: group activity

How could we use selection sort to put our screens in order?

Selection sort

- **Step 1.** In an unsorted list, find a way to select the item that should come earliest in the sort.
- **Step 2.** Pull the selected item out of the list and place it in the first position.
- **Step 3.** With the remaining items, use a selection sort to put them in order. If there are no remaining items, you're done.

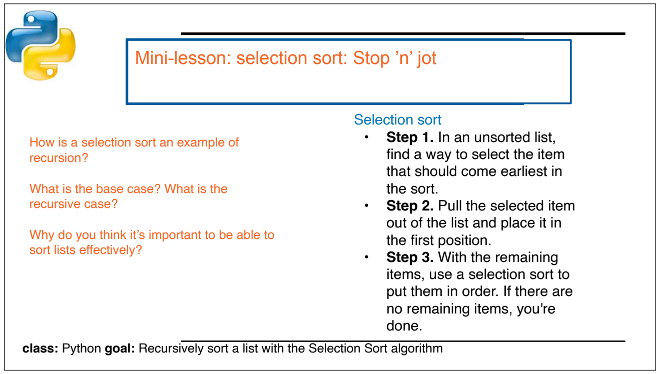
class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

Give students a chance to try on their own.

Make sure Zoom is in Gallery view.

Then talk through how they did it.

(1) Look at your classmates in Gallery view. Select the lowest alphabetically.
(2) Move to the leftmost position.
(3) Repeat (1) and (2) with the whole class minus the moved students
(4) stop when list is empty



Mini-lesson: selection sort: Stop 'n' jot

How is a selection sort an example of recursion?

What is the base case? What is the recursive case?

Why do you think it's important to be able to sort lists effectively?

Selection sort

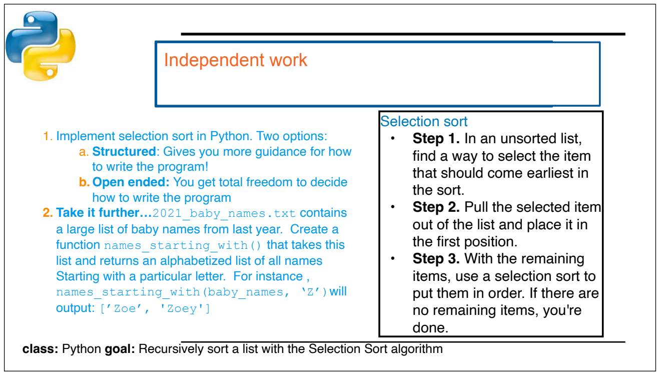
- **Step 1.** In an unsorted list, find a way to select the item that should come earliest in the sort.
- **Step 2.** Pull the selected item out of the list and place it in the first position.
- **Step 3.** With the remaining items, use a selection sort to put them in order. If there are no remaining items, you're done.

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

How is a selection sort an example of recursion? Because the same procedure is continuously applied to smaller and smaller sublists.

What is the base case? What is the recursive case? Base case is when there is zero or one item in the list.

Why do you think it's important to be able to sort lists effectively? Lots of real world examples where this is important.



Independent work

1. Implement selection sort in Python. Two options:
   a. **Structured**: Gives you more guidance for how to write the program!
   b. **Open ended:** You get total freedom to decide how to write the program
2. **Take it further…**`2021_baby_names.txt` contains a large list of baby names from last year. Create a function `names_starting_with()` that takes this list and returns an alphabetized list of all names Starting with a particular letter. For instance , `names_starting_with(baby_names, 'Z')` will output: `['Zoe', 'Zoey']`

Selection sort

- **Step 1.** In an unsorted list, find a way to select the item that should come earliest in the sort.
- **Step 2.** Pull the selected item out of the list and place it in the first position.
- **Step 3.** With the remaining items, use a selection sort to put them in order. If there are no remaining items, you're done.

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm

See selection_sort_structured.py for structure assignment.

See selection_sort_open_ended.py for openended assignment.

See selection_sort_solution.py for solution.

See baby_names.py for extension assignment and baby_names_solution.py for solution.

Reflection:

be sure to: Answer each question below with a complete sentence. Be prepared to share out!

1. What are some unexpected challenges that you ran into while working on the activities for today's class?
2. What's one thing you understand better about sorting?
3. What lingering questions do you have?

class: Python **goal:** Recursively sort a list with the Selection Sort algorithm