**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**Topic 19 - Lists: Popping Elements**  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What Does Popping Elements Mean?**

When you pop an element from a list in Python, you're removing it but not discarding it entirely. Instead, you save the popped element in a variable so it can be used for another purpose—like appending it to a different list or processing it further. Unlike del or remove(), which delete elements without retaining them, pop() allows you to "capture" the element as you remove it.

**Example Task List Setup:**

Let's say you have a list of tasks:

tasks = ["email Frank", "call Sarah", "meet with Zach"]

In this list:

* tasks[0] is "email Frank"
* tasks[1] is "call Sarah"
* tasks[2] is "meet with Zach"

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Why Use pop() Instead of del or remove()?**

1. **Keep Track of Removed Items**:  
   pop() lets you hold on to an element after removing it, which is useful if you need it for later processing or for another list.
2. **Flexibility**:  
   With pop(), you can specify an index to remove any item you want, or you can leave the parentheses empty to pop the last item automatically.
3. **Transfer Elements Easily Between Lists**:  
   pop() makes it simple to shift elements from one list to another without losing any data.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How to Use pop() to Remove and Store Elements in Python**

**1. Popping an Element by Index and Storing It in a Variable**

Use pop() with an index to remove an element from a specific position in the list. Python then returns the element so you can assign it to a variable.

**Example**:

latest\_task\_accomplished = tasks.pop(1)

# Result: tasks = ["email Frank", "meet with Zach"]

# latest\_task\_accomplished = "call Sarah"

After popping tasks[1], the list tasks now has only two items, and "call Sarah" is stored in the variable latest\_task\_accomplished.

**2. Syntax of the pop() Method**

* **Variable Name**: Start by naming the variable that will hold the popped value.
* **Equal Sign**: Use the equal sign (=) to assign the popped value to this variable.
* **List Name**: Specify the list you’re popping from.
* **Dot and Keyword pop**: Add a dot (.) and the keyword pop.
* **Index in Parentheses**: Inside the parentheses, specify the index of the item you want to pop.

**Example**:

latest\_task\_accomplished = tasks.pop(1)

**3. Transferring a Popped Element to Another List**

To remove an item from one list and add it to another list, you can combine pop() and append(). This approach moves the element directly to the new list.

**Example of Popping and Appending**:

tasks\_accomplished = []

tasks\_accomplished.append(tasks.pop(1))

# Result: tasks = ["email Frank", "meet with Zach"]

# tasks\_accomplished = ["call Sarah"]

In this example, "call Sarah" is removed from tasks and immediately added to tasks\_accomplished.

**4. Popping an Element and Inserting It into a Specific Position in Another List**

You can also use insert() with pop() to remove an element from one list and insert it into a specific position in another list.

**Example of Popping and Inserting**:

tasks\_accomplished.insert(1, tasks.pop(1))

# Result: tasks = ["email Frank", "meet with Zach"]

# tasks\_accomplished = [existing elements, "call Sarah"]

In this code, "call Sarah" is removed from tasks and inserted at index 1 in tasks\_accomplished.

**5. Popping the Last Element of a List Automatically**

To quickly pop the last item in a list, leave the parentheses empty. Python automatically removes the last item in the list and stores it in your chosen variable.

**Example**:

latest\_task\_accomplished = tasks.pop()

# This pops "meet with Zach" if it’s the last item in the list

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Important Points to Remember**

1. **Why Use pop()**:  
   pop() is useful for keeping an element after removal, especially when you want to use or store it elsewhere.
2. **Automatic Re-indexing**:  
   When you pop an element, Python automatically re-indexes the list, so there are no gaps.
3. **Using pop() without an Index**:  
   Leaving pop() empty targets the last element in the list, making it a convenient shortcut.
4. **Flexibility Across Lists**:  
   Combine pop() with append() or insert() to move elements easily between lists, streamlining your code.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Examples of Using pop()**

1. **Popping and Storing in a Variable**:

latest\_task\_accomplished = tasks.pop(0)

# Now tasks = ["call Sarah", "meet with Zach"]

# latest\_task\_accomplished = "email Frank"

1. **Popping and Appending to Another List**:

tasks\_accomplished = []

tasks\_accomplished.append(tasks.pop(1))

# Result: tasks = ["email Frank", "meet with Zach"]

# tasks\_accomplished = ["call Sarah"]

1. **Popping the Last Item**:

last\_task = tasks.pop()

# Removes the last task and assigns it to last\_task

1. **Popping and Inserting in Another List**:

tasks\_accomplished = ["initial task"]

tasks\_accomplished.insert(1, tasks.pop(0))

# Moves "email Frank" to tasks\_accomplished at position 1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Summary**

Using pop() is an efficient way to manage list elements when you need to remove but not lose them. By specifying an index or leaving it empty to pop the last item, pop() helps maintain data you may want to reuse or transfer to another list. This technique allows you to keep track of changes in your lists, and Python’s automatic re-indexing ensures lists stay neatly ordered.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**