SDEV 1001

Programming Fundamentals

Arrays and Loops - 2

A LEADING POLYTECHNIC COMMITTED TO YOUR SUCCESS

Expectations - What I expect from you

- No Late Assignments

- No Cheating

- Be a good classmate

- Don’t waste your time

- Show up to class

Agenda

On the right is what we will cover today.

- What is a List (Python) or Array (other languages)?
- Creating and Printing Lists
- Python List Methods: A Practical Overview
- Sorting Lists
- Adding Items: append and insert
- Removing Items: pop and remove
- Checking for Items: in Operator
- Avoiding Errors with remove
- More List Methods
- Summary

Creating and Printing Lists Review

You create a list with square brackets `[]` and separate items with commas.

Below is an example of creating and printing a list of strings

```
fruits = ["apple", "banana", "cherry"]
print("Available fruits:")
print(fruits)
```

- Lists can hold any data type.
- Use `print()` to display the contents of a list.

Python List Methods: A Practical Overview

- Lists are a fundamental data structure in Python.
- List methods let you add, remove, sort, and check for items efficiently.
- Mastering these methods will help you manage collections of data in your programs.
- There are also a ton of built in methods that you can use, take a look at the Python List Documentation for more information.

Sorting Lists

Sometimes you need to sort items in a list. Python provides a simple way to do this with the `sort()` method.

```
numbers = [42, 7, 19, 100, 3]
numbers.sort()
print("Sorted numbers:", numbers)
numbers.sort(reverse=True)
print("Sorted (descending):", numbers)
```

- `sort()` arranges items in ascending order by default.
- Use `reverse=True` for descending order.

Adding Items: append and insert

You’ll often need to add items to a list.

- `append()` adds an item to the end of the list.
- `insert(index, value)` adds an item at a specific position.

Here’s an example of both methods in action:

```
colors = ["red", "blue"]
colors.append("green")
colors.insert(1, "yellow")
print("Colors:", colors)
```

- `append()` adds to the end of the list.
- `insert(index, value)` adds at a specific position.

Removing Items: pop and remove

You also have the capability to remove items from a list.

```
animals = ["cat", "dog", "rabbit", "hamster"]
animals.pop(2)        # Removes "rabbit"
animals.remove("cat") # Removes "cat"
print("Animals:", animals)
```

- `pop(index)` removes by position.
- `remove(value)` removes by value (first occurrence).

Note: `pop()` returns the removed item, while `remove()` does not. Seoncdly , `remove()` raises a `ValueError` if the item is not found.

Checking for Items: in Operator

Many times you'll need to check if an item exists in a list. You can do this easily with the `in` operator.

```
cities = ["Edmonton", "Calgary", "Toronto"]
print("Is Calgary in the list?", "Calgary" in cities)
print("Is Vancouver in the list?", "Vancouver" in cities)
```

- The `in` operator returns `True` if the item exists, `False` otherwise.
- Useful for conditionals and avoiding errors when removing items.

Avoiding Errors with remove

With `remove()`, if you try to remove an item that doesn’t exist, Python raises a `ValueError`. To avoid this, always check if the item is in the list before removing it.

```
numbers = [1, 2, 3]
if 4 in numbers:
    numbers.remove(4)
else:
    print("4 is not in the list, nothing to remove.")
```

- Always check if an item exists before removing to avoid `ValueError`.

More List Methods

- `len(list)` — Get the number of items.
- `list.index(value)` — Find the index of a value.
- `list.count(value)` — Count occurrences of a value.
- `list.clear()` — Remove all items.

Summary

- Lists are versatile and essential in Python and all languages.
- Use methods like `append`, `insert`, `sort`, `pop`, `remove`, and the `in` operator to manage your data.
- Practice with different data and scenarios to become confident with list operations!

Example

Let’s go run a few examples together

A LEADING POLYTECHNIC COMMITTED TO YOUR SUCCESS