SDEV 1001

Programming Fundamentals

Arrays and Loops - 3

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.

- Lists and For Loops in Python
- Creating a List and Looping Through It
- Using Tuples for Unchangeable Data
- Getting the Index with enumerate
- Skipping Items with continue
- Stopping a Loop with break
- Summary

Lists and For Loops in Python

Introduction to Lists and For Loops

- Lists store collections of data.
- For loops let you perform actions for each item in a list.
- Together, they help automate repetitive tasks in your code.

Real-Life Analogy

- Think of a packing list for a trip.
- For each item on the list, you pack it in your bag.
- In Python, you can do the same with a list and a for loop.

Creating a List and Looping Through It

Below is an example of creating a list of books and printing each one:

- The `for` loop goes through each item in the list and executes the indented code.
- `book` is a variable that takes the value of each item in the list during each iteration.
  - so the first time through the loop, `book` is "1984", then "Brave New World", and so on.

Example:

```
books = ["1984", "Brave New World", "Fahrenheit 451"]

print("Books to read:")
for book in books:
    print(f"Don't forget to read: {book}")
```

Using Tuples for Unchangeable Data

Tuples are similar to lists but are immutable, meaning they cannot be changed after creation. They are useful for fixed collections of items.

- we'll be using this less in the course, but it's important to know about them.

```
months = ["January", "February", "March"]
winter_months = ("December", "January", "February")

for month in months:
    if month in winter_months:
        print(f"{month} is a winter month")
    else:
        print(f"{month} is not a winter month")
```

Getting the Index with enumerate

Earlier we learned about indexes in lists. The `enumerate()` function allows you to loop through a list and get both the index and the value at the same time.

- in the examples below `student` is the value, and `idx` is the index.
  - For example, for the first iteration `idx` will be 0 and `student` will be "Alice".
  - The second iteration `idx` as 1 and `student` as "Bob", and so on.

Example:

```
students = ["Alice", "Bob", "Charlie"]

for idx, student in enumerate(students):
    print(f"Student {idx + 1}: {student}")
```

Skipping Items with continue

Sometimes you want to skip certain items in a loop. The `continue` statement allows you to skip the rest of the current iteration and move to the next item.

Example:

```
numbers = [1, 2, 3, 4, 5]
for n in numbers:
    if n % 2 == 0:
        continue  # Skip even numbers
    print(f"Odd number: {n}")
```

Above we're using modulo `%` to check if a number is even. If it is, we skip printing it because it hits the `continue` statement, and the loop moves to the next number.

Stopping a Loop with break

Sometimes you want to stop a loop early. The `break` statement allows you to exit the loop immediately when a certain condition is met.

Example:

```
pets = ["dog", "cat", "parrot", "hamster"]
for pet in pets:
    if pet == "parrot":
        print("Found a parrot! Stopping search.")
        break
    print(f"Checked: {pet}")
```

Here in the above example, we loop through the list of pets and stop searching when we find a `"parrot"`. The `break` statement exits the loop immediately, so we don't check any more pets after that.

Summary

- Use lists to store data and for loops to process each item.
- Use `enumerate` for indexes, `continue` to skip, and `break` to stop early.
- Practice with your own lists and loops to get comfortable!

Example

Let’s go run a few examples together