Lists and For Loops Fundamentals (Executing Repeating Tasks)

Why is this important?

Looping and lists go hand in and hand. You can't really have one without the other. Lists normally store the data but you want to do something with each item in the list. That's where loops come in. They allow you to execute a task for each item in the list.

This is just like how you do things in real life. You get a grocery list, and when you get to the grocery store, for each item on the list you get the item. Same as todo list, you do each task on the list. Python allows you to do the same thing.

Loops in different programming languages have different syntax, but they all do the same thing. They allow you to execute a task for each item in a list.

What are we going to do?

Steps

1. Create a file called `lists_and_for_loops.py`

2. We're going to write the days of the week in a list and print them out using a for loop.

```
days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday'

print("Days of the week are")
for day in days_of_week:
    print(F"{day} is a day of the week")
```

- You see here below that we use `for` new variable name `in` list name. This is the syntax for a for loop in Python. The `for` keyword is used to start the loop.
  - The new variable name is the name of the variable that will hold the value of the current item in the list.
  - The `in` keyword is used to separate the new variable name from the list name.
  - The `:` is used to end the line.
  - The next line is indented. This is how Python knows that the next line is part of the loop. The next line is the code that you want to execute for each item in the list. The loop will execute the code for each item in the list. When it gets to the end of the list, it will stop.
- Let's take a look at the output.

```
$ python lists_and_for_loops.py
Days of the week are
Sunday is a day of the week
```

```
Monday is a day of the week
Tuesday is a day of the week
Wednesday is a day of the week
Thursday is a day of the week
Friday is a day of the week
Saturday is a day of the week
```

- A few notes on the above.
  - You can see that the indented code after the `for ... in` is executed

3. Let's use our knowledge of lists and for loops to see which days are weekdays and which are weekends.

- we're going to introduce the idea of a tuple. A tuple is a list that can't be changed. It's like a list but you can't add or remove items from it. You can only read the items in it.
  - to define a tuple you use `(` and `)` instead of `[` and `]`. If you have one item, you need to add an extra comma because your parenthesis will look like a math equation.
- Type the following code into your file.

```
days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
days_of_weekend = ("Saturday", "Sunday")

print("Days of the week are")
for day in days_of_week:
    print(F"{day} is a day of the week")
```

- the `days_of_weekend` is a tuple. We're going to use this data structure sometimes because it's a good way to store data that you don't want to change.

- Let's print out whether a day is a weekday separately depending on the day in our for loop.

```
days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
days_of_weekend = ("Saturday", "Sunday")

for day in days_of_week:
    if day in days_of_weekend:
        print(F"{day} is a weekend day")
    else:
        print(F"{day} is a weekday")
```

- You can see here that we have one indent so that we know which statements are part of our `for` loop. Here there 4 statements that are all part of the if statement.
  - Next you can see that we have to indent code again so that our if statement knows which code is part of it. Here we have one statement

    that is indented under the `if` and one statement that is indented under the `else`.

- Let's take a look at the output.

```
$ python lists_and_for_loops.py
Sunday is a weekend day
Monday is a weekday
Tuesday is a weekday
Wednesday is a weekday
Thursday is a weekday
Friday is a weekday
Saturday is a weekend day
```

- You can see here that our loop prints out something different depending on the day of the week. This is because we have an `if` statement inside of our loop.
  - The `if` statement checks to see if the current day is in the `days_of_weekend` tuple. So you can use the `in` keyword to check if an item is in a tuple just like a list!
  - If the `day` is in the `days_of_weekend`, then it prints out that it's a weekend day. If it's not, then it prints out that it's a weekday.
  -

4. Let's use the index, to print out the day number of the week.

- This executes for each item in the list.
- Sometimes you want to know the index of the item while you're looping over the list. There's a hand way to do this with the `enumerate` function.
- Let's use enumerate to print out the day number of the week.

```
days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday'
days_of_weekend = ("Saturday", "Sunday")

for index, day in enumerate(days_of_week):
    print(F"{day} is day number {index + 1} of the week")
    if day in days_of_weekend:
        print(F"{day} is a weekend day")
    else:
        print(F"{day} is a weekday")
```

- taking a look at the output, we know that our `index` will begin at 0 (just like when we did arrays) so we need to add 1 to it to get the day number of the week. This is shown below.

```
$ python lists_and_for_loops.py
Sunday is day number 1 of the week
Sunday is a weekend day
Monday is day number 2 of the week
Monday is a weekday
Tuesday is day number 3 of the week
Tuesday is a weekday
Wednesday is day number 4 of the week
Wednesday is a weekday
Thursday is day number 5 of the week
Thursday is a weekday
Friday is day number 6 of the week
Friday is a weekday
Saturday is day number 7 of the week
Saturday is a weekend day
```

5. Let's check to see if the day contains the letter u and skip the day if it does with the `continue` keyword.

- Sometimes you want to skip an item in the list if it meets a certain condition. You can do this with the `continue` keyword.
- So what we're going to do is right after the `for` statement, we're going to check to see if the day contains the letter `u`. If it does, we're going to skip the day.
  - As well this bring us that strings in python are lists of characters. So we can use the `in` keyword to check if a character is in a string.
    - In other languages, you need to use character arrays to do this, but in python you can just use some list methods on strings.
- Let's take a look at the code.

```python
days_of_week = ["Sunday", "Monday", "Tuesday",
                "Wednesday", "Thursday", "Friday",
                "Saturday"]
days_of_weekend = ("Saturday", "Sunday")

for index, day in enumerate(days_of_week):
    # check if the day has a u
    if "u" in day:
        continue

    print(F"{day} is day number {index + 1} of the week")
    if day in days_of_weekend:
        print(F"{day} is a weekend day")
    else:
        print(F"{day} is a weekday")
```

- You can see that that output is much shorter because `Sunday`, `Tuesday`, `Thursday`, and `Saturday` all contain the letter `u`.

```
$ python lists_and_for_loops.py
Monday is day number 2 of the week
Monday is a weekday
Wednesday is day number 4 of the week
Wednesday is a weekday
Friday is day number 6 of the week
Friday is a weekday
```

6. Let's stop the loop once you hit Wednesday with the `break` keyword.

- Sometimes you want to stop the loop once you hit a certain condition. You can do this with the `break` keyword.
- Take a look at the code below.
  - right before we check for the day containing the letter `u`, we're going to check to see if the day is `"Wednesday"`. If it is, we're going to stop the loop.
  - "'python days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"] days_of_weekend = ("Saturday", "Sunday")

for index, day in enumerate(days_of_week): # stop the loop if it's wednesday if day == "Wednesday": break # check if the day has a u if "u" in day: continue

```
print(F"{day} is day number {index + 1} of the week")
if day in days_of_weekend:
    print(F"{day} is a weekend day")
else:
    print(F"{day} is a weekday")
```

```
- Let's take a look at the output
```

$ python lists_and_for_loops.py Monday is day number 2 of the week Monday is a weekday "`- this is interesting   - We see that we're looping over the list, but we only see one day being printed out. Let's explain what's going on.     - In the first iteration:        - First,`Sunday`is not`Wednesday`so it continues to execute.       - Second, the program checks to see if`Sunday`contains the letter`u`. It does, so it skips the rest of the code in the loop and goes to the next iteration.     - In the second Iteration:        - First,`Monday`is not`Wednesday`so it continues to execute.       - Second, the program checks to see if`Monday`contains the letter`u`. It does not, so it prints out the day and the day number. Then it checks to see if`Monday`is in the`days_of_weekend`tuple. It is not, so it prints out that it's a weekday. Then it goes to the next iteration.     - In the third iteration:        - First,`Tuesday`is not`Wednesday`so it continues to execute.       - Second, the program checks to see if`Tuesday`contains the letter`u`. It does, so it skips the rest of the code in the loop and goes to the next iteration.     - In the fourth Iteration:        - First,`Wednesday`is`Wednesday' so it stops the loop. - There are no more iterations after this so the program ends.

Conclusion

This is the fundamentals of using a list with a for loop. We're going to use this a lot in the future of all of our programs.