A CLOUD GURU

Lesson Description - Parsing Command Line Parameters

Many times scripts are more useful if we can pass in arguments when we type the command rather than having a second step to get user input.

Python Documentation For This Video

- The `sys` module
- The `sys.argv` attribute

Accepting Simple Positional Arguments

Most of the scripts and utilities that we work with accept positional arguments instead of prompting us for information after we've run the command. The simplest way for us to do this in Python is to use the `sys` module's `argv` attribute. Let's try this out by writing a small script that echoes our first argument back to us:

~/bin/param_echo

```
#!/usr/bin/env python3.6

import sys

print(f"First argument {sys.argv[0]}")
```

After we make this executable and give it a shot, we see that the first argument is the script itself:

```
$ chmod u+x ~/bin/param_echo
$ param_echo testing
First argument /home/user/bin/param_echo
```

That's not quite what we wanted, but now we know that `argv` will contain the script and we'll need to get the index of `1` for our first argument. Let's adjust our script to echo all of the arguments except the script name and then echo the first positional argument by itself:

~/bin/param_echo

```
#!/usr/bin/env python3.6

import sys

print(f"Positional arguments: {sys.argv[1:]}")
print(f"First argument: {sys.argv[1]}")
```

Trying the same command again, we get a much different result:

```
$ param_echo testing
Positional arguments: ['testing']
First argument: testing
$ param_echo testing testing12 'another argument'
Positional arguments: ['testing', 'testing12', 'another argument']
First argument: testing
$ param_echo
Positional arguments: []
Traceback (most recent call last):
  File "/home/user/bin/param_echo", line 6, in
    print(f"First argument: {sys.argv[1]}")
IndexError: list index out of range
```

This shows us a few things about working with `argv`:

1. Positional arguments are based on spaces unless we explicitly wrap the argument in quotes.
2. We can get a slice of the first index and after without worrying about it being empty.
3. We risk an `IndexError` if we assume that there will be an argument for a specific position and one isn't given.

Using `sys.argv` is the simplest way to allow our scripts to accept positional arguments. In the next video, we'll explore a standard library package that will allow us to provide a more robust command line experience with help text, named arguments, and flags.