4-2-1: Examining Scripting Logic

After completing this episode, you should be able to:

- Identify and explain common scripting logic, given a scenario.

Description: In this episode, the learner will examine the components and structure of scripting logic. We will explore the basics of variables, conditionals, integers, Boolean operators, conditionals, and more.

- Describe variables
  - Storage locations identified by a name that containing data
  - Examples:
    - PowerShell:

```
# Variables store data
# "18" is a data type called an "integer"

$age = 18
```

    - Bash:

```
# Variable storing an integer value

age=18
```

- Describe data types
  - A classification of data
  - Examples:
    - PowerShell:

```
# 'string' data type
$name = "Student01"
# 'decimal' or 'floating-point' data type
$height = 5.9
# 'Boolean' data type
$is_student = $true
```

- Describe conditionals
  - Allow the execution of different code blocks based on specified conditions
  - Examples:
    - PowerShell:

```
# "Please enter your age:" is a data type
called a "string"

# Prompt the user to enter their age
$age = Read-Host "Please enter your age:"

# Check to see if the user is 18 or older
if ($age -ge 18) {
    Write-Output "You are eligible to vote."
} else {
    Write-Output "You are not eligible to vote."
}
```

    - Bash:

```
!/bin/bash

# Prompt the user to enter their age
echo "Please enter your age:"
read age

# Check to see if the user is older than 18
if [ $age -ge 18 ]; then
  echo "You are eligible to vote."
else
  echo " You are not eligible to vote."
fi
```

- Describe operators
  - Symbols that perform specific operations on one or more operands, such as arithmetic, logical, and comparative.
  - Examples:
    - PowerShell:

```
  # Prompt the user to input their age
Write-Host "Please enter your age:"
$age_input = Read-Host

# Perform the arithmetic operation
$age = 10 + 5

# Perform the comparison operation
$is_adult = $age_input -ge 18

# Check if the user is eligible to vote
if ($is_adult) {
    Write-Output "You are eligible to vote."
} else {
    Write-Output "You are not eligible to vote."
}
```

- Functions:
  - Reusable blocks of code that perform tasks
  - Examples:
    - PowerShell:

```
# Define a function named "Greet"
function Greet {
  param (
      [string]$name
  )
  Write-Output "Hello, $name!"
}

# Call the function
Greet -name "John"
```

    - Bash:

```
# Define a function named "greet"
greet() {
  name=$1
  echo "Hello, $name!"
}

# Call the function
greet "John"
```

    - Python

```
# Define a function named "greet"
def greet(name):
  return "Hello, " + name + "!"

# Call the function
print(greet("John"))
```

Additional References

- Variable - a storage location in memory used to store data.
- Integer - a data type in programming used to represent whole numbers, both positive and negative, without any fractional or decimal part.
- Float (floating-point) - is a data type used in programming to represent real numbers (numbers with a fractional part).
- Boolean - a data type in programming that represents one of two possible values: true or false.
- Array - a fundamental data structure used in programming to store a collection of elements.
- Conditional (conditional statements) - programming constructs that allow the execution of different code blocks based on certain conditions.
- Operator - symbols or keywords in programming that perform operations on one or more operands to produce a result (examples: arithmetic, comparison, logical).
- Operand - values or variables that are manipulated or operated on by operators to produce a result.
- Examples of arrays
  - Bash

    ```
    numbers=(1 2 3 4 5)
    echo "The third number is: ${numbers[2]}"
    ```

  - Python

    ```
    fruits = ["apple", "banana", "orange"]
    print("The first fruit is:", fruits[0])
    ```