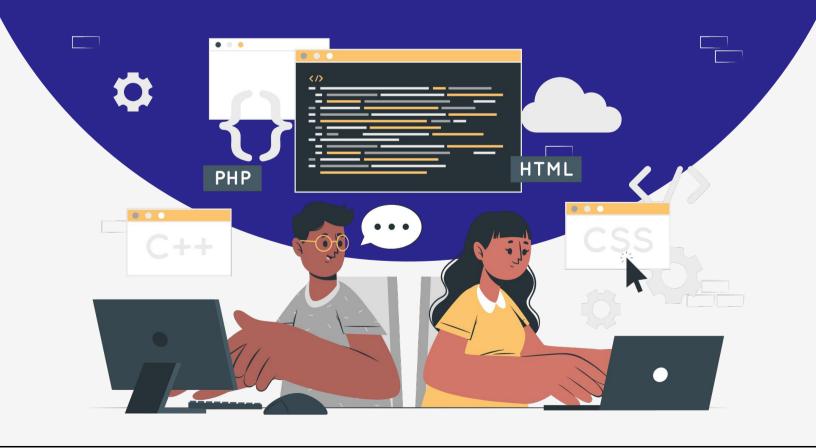
Lesson Plan

Bash variables and parameters







Bash variables and parameters are fundamental concepts in shell scripting. They allow you to store and manipulate data, control script execution, and pass values into and out of functions or scripts. Here's a comprehensive overview:

1. Bash Variables

Types of Variables

- Local Variables: Defined in a script or function and are only accessible within that scope.
- Environment Variables: Available system-wide and can be accessed by any script or process.
- Special Variables: Built-in variables provided by Bash, often used to manage script behavior.

Declaring Variables

- Variables are declared by assigning a value to a name without spaces.
- By convention, variable names are usually uppercase for environment variables and lowercase for local variables.

Syntax:

```
VARIABLE_NAME=value
```

Example:

```
#!/bin/bash
greeting="Hello, World!"
echo $greeting
```

Variable Assignment and Usage

- Assignment: No spaces around '='.
- Usage: Precede the variable name with '\$' to access its value.
- Double Quotes: Use double quotes around variables to prevent word splitting or globbing.



Example:

```
#!/bin/bash
name="John Doe"
echo "Welcome, $name"
```

2. Parameter Expansion

Parameter expansion allows you to manipulate the value of variables.

Basic Expansion

• Syntax: \${variable} or \$variable

Example:

```
#!/bin/bash
name="John"
echo "Hello, ${name}!"
```

Default values

- \${variable:-default}: Uses default if variable is not set.
- \${variable:=default}: Assigns default to variable if it is not set.

Example:

```
#!/bin/bash
echo "Username: ${USER:-defaultuser}"
```

String Length

• \${#variable}: Returns the length of the variable's value.

Example:

```
#!/bin/bash
name="John"
echo "Length of name: ${#name}"
```



Substring Extraction

• \${variable:position:length}: Extracts a substring.

Example:

```
#!/bin/bash
greeting="Hello, World!"
echo "${greeting:7:5}" # Output: World
```

Substring Replacement

- \${variable/pattern/replacement}: Replaces the first occurrence of a pattern.
- \${variable//pattern/replacement}: Replaces all occurrences.

Example:

```
#!/bin/bash

text="apples are red"
echo "${text/apples/oranges}" # Output: oranges are red
```