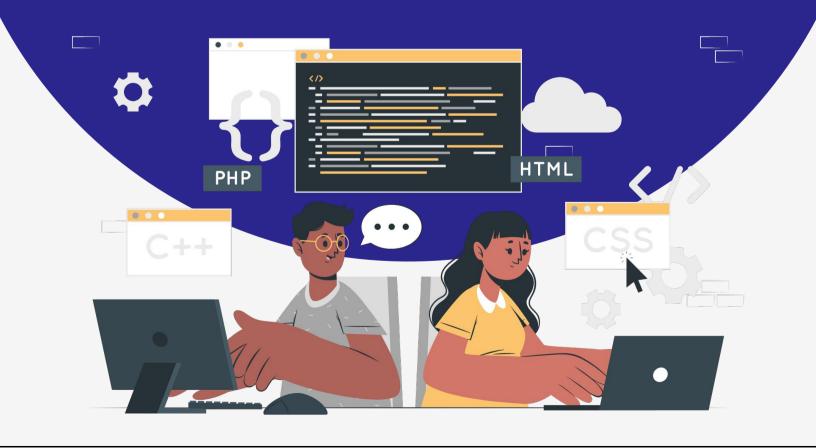
## Lesson Plan Shell Scripting Basics







## **Shell Scripting Basics**

• Shell scripting involves writing a series of commands for the shell to execute. Here are the basics to get you started:

### 1. Creating a Script

- Open a Text Editor: Use any text editor like 'nano', 'vim', or 'gedit'.
- Write the Script: Start with a shebang (#!) followed by the path to the interpreter.

```
#!/bin/bash
echo "Hello, World!"
```

• Save the File: Save it with a '.sh' extension, e.g., 'script.sh'.

### 2. Making the Script Executable

• Change the file permissions to make it executable:

```
chmod +x script.sh
```

### 3. Running the Script

· Execute the script from the terminal:

```
./script.sh
```

### 4. Basic Components

· Comments: Use '#' for comments.

```
# This is a comment
echo "Comments are ignored by the interpreter"
```

· Variables: Store and use data.

```
NAME="Alice"
echo "Hello, $NAME"
```



• User Input: Read input from the user.

```
echo "Enter your name:"
read NAME
echo "Hello, $NAME"
```

- Control Structures:
  - If Statement:

```
if [ "$NAME" == "Alice" ]; then
    echo "Hi, Alice!"
else
    echo "You are not Alice."
fi
```

• For Loop:

```
for i in 1 2 3
do
echo "Number $i"
done
```

### 5. Functions

• Define and use functions to organize code.

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

# Call the function
greet "Alice"
greet "Bob"
```

### 6. File Operations

• Create a directory:

```
mkdir new_directory
```



• Navigate to a directory:

cd new\_directory

· List files:

ls

### 7. String Operations

• String Length:

```
STR="Hello"
echo ${#STR}
```

• Substring:

```
echo ${STR:1:3}
```

### 8. Arithmetic Operations

• Perform arithmetic operations using double parentheses.

```
result=$((3 + 5))
echo $result
```

### 9. Debugging

• Run the script with -x to enable debugging.

bash -x script.sh



# THANK YOU!