Day 14-Bash Scripting Part 2

Conditional Logic, Loops, and Functions

Shebang Line

Always start a shell script with the shebang line to specify the interpreter:

```
#!/bin/bash
```

Conditional Statements

- Conditional logic defines which part of code to run based on given conditions.

if statement:

```
if [ condition ]
then
    <commands>
fi
```

- Example:

```
if [ $age -ge 18 ]
then
    echo "You are eligible to vote."
fi
```

if-else statement:

- Gives two choices depending on condition.

```
if [ condition ]
then
    <commands if true>
else
    <commands if false>
fi
```

- Example:

```
if [ $number -gt 0 ]
then
    echo "Positive number"
else
    echo "Zero or Negative number"
fi
```

If-Elif-Else statement:

- Allows decisions based on multiple conditions.

```
if [ condition1 ]
then
```

```
    <commands for condition1>
elif [ condition2 ]
then
    <commands for condition2>
else
    <commands if none match>
fi
```

- Example:

```
if [ $marks -ge 90 ]
then
    echo "Grade A"
elif [ $marks -ge 75 ]
then
    echo "Grade B"
else
    echo "Grade C"
fi
```

Enhanced Test with [[ ]]

- `[[ ]]` is a more flexible, safer version of `[ ]`.
- Example:

```
if [[ $str == "hello" ]]
then
    echo "Strings match"
fi
```

- No escaping needed for `&&` and `||`
- Supports pattern matching (`==`, `!=`, regex)
- Example:

```
if [[ $age -ge 18 && $citizen == "yes" ]]
then
    echo "Eligible voter"
fi
```

Loops

For Loop

- Iterating Over Items

```
for var in item1 item2 item3
do
    <commands>
done
```

- Example:

  -
    ```
    for color in red blue green
    do
        echo "Color: $color"
    done
    ```

- Reading list from a file:

```
for i in $(cat list_of_items.txt)
do
    echo "Item: $i"
done
```

- *Better than using backticks* `` ` ``, which can break
- Generating Sequence

```
for i in {0..5}
do
    echo "Number: $i"
done
```

- C/CPP Style For Loop

```
for (( i=0; i<5; i++ ))
do
    echo "Counter: $i"
done
```

While Loop

- Execute While Condition True

```
while [ condition ]
do
    <commands>
done
```

- Example:

```
counter=0
while [ $counter -lt 3 ]
do
    echo "Counter: $counter"
    ((counter++))
done
```

Case Statement

- Simplifies checking multiple conditions.
- Syntax:

```
case $variable in
    pattern1)
        commands ;;
    pattern2)
        commands ;;
    *)
        default commands ;;
esac
```

- Example:

```
read -p "Enter a number between 1 and 3: " num
case $num in
```

```
    1)
        echo "One";;
    2)
        echo "Two";;
    3)
        echo "Three";;
    *)
        echo "Invalid choice";;
esac
```

Functions

- Functions group reusable code blocks.
- Syntax:

```
function_name() {
    commands
}
```

- or

```
function function_name {
    commands
}
```

- Example:

```
greet_user() {
    echo "Hello, $1!"
}

greet_user Fatima
```

- Passing arguments to functions:
- `$1`, `$2`, etc. are used inside functions to refer to arguments.
- Example:

```
add_numbers() {
    sum=$(( $1 + $2 ))
    echo "Sum is: $sum"
}

add_numbers 5 7
```