SRE TRAINING (DAY 8) - SHELL SCRIPTING

DATA STRUCTURES (ARRAYS)

Arrays in shell scripting, like other programming languages, store multiple values in a single variable.

SCRIPT-1 - STORING DATA IN AN ARRAY AND PRINTING

```
a[0]="Rhea"
a[1]="Rebecca"
a[2]="Robinson"
fullname="${a[0]} ${a[1]} ${a[2]}"
echo "Hi, $fullname!"
~
```

```
root@RheaAlisha:/mnt/c/mthree_practice# ./arrays.sh
Hi, Rhea Rebecca Robinson!
```

SCRIPT-2 - BANK ACCOUNT (CONDITIONAL STATEMENT IF)

root@RheaAlisha: /mnt/c/mthree_practice

```
balance=500
withdrawl=1200
daily_limit=1000
account_type="savings"
description=""

if [ $balance -eq 5000 ]; then
        echo "Balance is exactly 5000"
fi

if [ $withdrawl -ne 1000 ]; then
        echo "Withdrawn amount is not 1000"
fi

if [ $balance -gt $withdrawl ]; then
        echo "You have a valid balance to withdraw money"
fi

if [ $withdrawl -le $balance -a $withdrawl -le $daily_limit ]; then
        echo "Transaction approved"
  else
          echo "Transaction not approved"
fi

if [ $withdrawl -le $balance -o $balance -ge 500 ]; then
        echo "Customer is valuable to the bank"
fi

if [[ ! $withdrawl -le $balance || $balance -ge 500 ]]; then
        echo "Customer is valuable to the bank"
fi

if [ "$account_type" = "savings" ]; then
        echo "This is a saving account"
fi

if [ "$account_type" != "savings" ]; then
        echo "This is a false saving account"
fi

if [ -z "$description" ]; then
        echo "Description is not provided"
fi
```

```
root@RheaAlisha:/mnt/c/mthree_practice# ./balance.sh
Withdrawn amount is not 1000
Transaction not approved
Customer is valuable to the bank
Customer is valuable to the bank
This is a saving account
Description is not provided
```

SCRIPT-3 - OPERATORS (Arithmetic, Relational, Assignment, String)

```
read -p "Enter the first number (a): " a
read -p "Enter the second number (b): " b
read -p "Enter a string: " str

echo "Arithmetic Operator (Addition):"
echo "$a + $b = $((a + b))"

echo "Relational Operator (Greater Than):"
if [ $a -gt $b ]; then
          echo "$a is greater than $b"
  else
            echo "$a is not greater than $b"
fi

echo "Logical Operator (AND):"
if [ $a -gt 0 ] && [ $b -gt 0 ]; then
          echo "Both $a and $b are positive numbers."
  else
            echo "One or both numbers are not positive."
fi

echo "Assignment Operator (+=):"
c=$a
((c += b))
echo "After c += b, c = $c"

echo "String Operator (-z: Check if string is empty):"
if [ -z "$str" ]; then
          echo "The string is empty."
  else
            echo "The string is not empty."
fi
~
```

```
root@RheaAlisha:/mnt/c/mthree_practice# ./operators.sh
Enter the first number (a): 20
Enter the second number (b): 45
Enter a string: Rhea
Arithmetic Operator (Addition):
20 + 45 = 65
Relational Operator (Greater Than):
20 is not greater than 45
Logical Operator (AND):
Both 20 and 45 are positive numbers.
Assignment Operator (+=):
After c += b, c = 65
String Operator (-z: Check if string is empty):
The string is not empty.
```

SCRIPT-5 - PASSWORDS (REGEX)

root@RheaAlisha: /mnt/c/mthree_practice

```
read -sp "Enter your password: " password
pattern='^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[@#$%&*]).{8,}$'
if [[ $password =~ $pattern ]]; then
          echo "Password is valid."
    else
              echo "Password is invalid."
                  echo "Requirements:"
                      echo "- At least 8 characters"
                          echo "- At least one uppercase letter"
                              echo "- At least one lowercase letter"
                                  echo "- At least one digit"
                                      echo "- At least one special character (@, #, $, %, &, *)"
fi
~
```

```
root@RheaAlisha:/mnt/c/mthree_practice# ./password.sh
Enter your password: Password is invalid.
Requirements:
- At least 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one digit
- At least one special character (@, #, $, %, &, *)
root@RheaAlisha:/mnt/c/mthree_practice#
```

BREAKOUT ROOM ACTIVITY -> REAL-TIME CALCULATOR (CASES)

root@RheaAlisha: /mnt/c/mthree_practice

```
read -p "Enter your expression: " input

IFS=' ' read -ra tokens <<< "$input"
result=${tokens[0]}

for ((i = 1; i < ${#tokens[@]}; i+=2)); do
    op=${tokens[i]}
    num=${tokens[i+1]}

    case $op in
        +) result=$(echo "$result + $num" | bc) ;;
        -) result=$(echo "$result - $num" | bc) ;;
        \*) result=$(echo "$result * $num" | bc) ;;
        /) result=$(echo "scale=2; $result / $num" | bc) ;;
        *) echo "Invalid operator: $op"; exit 1 ;;
    esac
done

echo "Result: $result"
~
~
```