**Array Declaration and Access**

a[0]="zara"

a[1]="askdhkasdh"

a[2]="adakjhsdj"

echo "${a[0]}"

echo "${a[1]}"

**Explanation:**

* a[0]="zara" → Declares an array and assigns "zara" to index **0**.
* a[1]="askdhkasdh" → Assigns "askdhkasdh" to index **1**.
* a[2]="adakjhsdj" → Assigns "adakjhsdj" to index **2**.
* echo "${a[0]}" → Prints the **first element** (zara).
* echo "${a[1]}" → Prints the **second element** (askdhkasdh).

**Comparison Operators**

| **Operator** | **Meaning** |
| --- | --- |
| -eq | Equal to |
| -ne | Not equal to |
| -gt | Greater than |
| -lt | Less than |
| -ge | Greater than or equal to |
| -le | Less than or equal to |
| -a | Logical AND |
| -o | Logical OR |

**Bank Transaction Conditions**

balance=500

withdrawl=1200

daily\_limit=1000

account\_type="savings"

description=""

* Declares **variables** with account details.

**Conditional Checks**

if [ $balance -eq 5000 ]; then

echo "Balance is exactly 5000"

fi

* Checks if balance is **exactly 5000**, then prints the message.

if [ $withdrawl -ne 1000 ]; then

echo "Withdrawn amount is not 1000"

fi

* If withdrawl is **not** 1000, it prints the message.

if [ $balance -gt $withdrawl ]; then

echo "You have a valid balance to withdraw money"

fi

* If balance is **greater than** withdrawl, it confirms **sufficient balance**.

if [ $withdrawl -le $balance -a $withdrawl -le $daily\_limit ]; then

echo "Transaction approved"

else

echo "Transaction not approved"

fi

* Checks **both conditions**:
  1. withdrawl ≤ balance
  2. withdrawl ≤ daily\_limit
* If both are **true**, transaction is approved; otherwise, rejected.

if [ $withdrawl -le $balance -o $balance -ge 500 ]; then

echo "Customer is valuable to bank"

fi

* Checks **at least one** condition:
  1. withdrawl ≤ balance
  2. balance ≥ 500
* If **any one** is true, prints **valuable customer** message.

if [[ ! $withdrawl -le $balance || $balance -ge 500 ]]; then

echo "Customer is valuable to bank"

fi

* **Same as previous**, but uses [[ ]] notation.
* ! negates the first condition.

if [ "$account\_type" = "savings" ]; then

echo "This is a savings account"

fi

* Checks if account\_type is **"savings"**.

if [ "$account\_type" != "savings" ]; then

echo "This is a false savings account"

fi

* Checks if account\_type **is NOT** "savings".

if [ -z "$description" ]; then

echo "Description is not provided"

fi

* -z checks if **description is empty**.

**File Permission Checks**

| **Option** | **Meaning** |
| --- | --- |
| -r | File has **read** permission |
| -w | File has **write** permission |
| -x | File has **execute** permission |
| -s | File **is not empty** |

Example usage:

if [ -r filename ]; then

echo "File is readable"

fi

* Checks if **file is readable**.

**Timed Input and Secure Input**

read -t 5 -p "Quick 5 sec: " pin

* Waits **5 seconds** for user input, then proceeds.

echo "Enter your name"

read name

echo "$name"

* Reads **name** input and prints it.

read -p "Enter account number and password: " acn password

echo $acn

echo $password

* Reads **account number** and **password** in one line.

read -s -p "Enter password: " p

* -s makes input **silent** (useful for passwords).

**Case Statement for Account Type Selection**

read -p "Enter selection [1-3]: " selection

case $selection in

1) account\_type="checking"; echo "You have selected checking";;

2) account\_type="saving"; echo "You have selected saving";;

3) account\_type="current"; echo "You have selected current";;

\*) account\_type="random"; echo "Random selection";;

esac

**How It Works**

* **Reads** a number between **1-3**.
* Matches it to corresponding account type.
* If input is **not** 1, 2, or 3, it assigns "random".

**grep Commands for Pattern Matching**

| **Command** | **Explanation** |
| --- | --- |
| grep "selection$" case.sh | Finds lines where "selection" appears **at the end**. |
| grep -Ril "selection" case.sh | Recursively searches **all files** and prints **filenames** containing "selection". |
| grep "s.lection$" case.sh | Matches "sXlection" (where X is **any character**) at the end. |
| grep "[0-9]" case.sh | Finds lines containing **any digit** (0-9). |
| grep "[a-zA-Z]" case.sh | Finds lines with **any letter** (A-Z or a-z). |
| grep "[aeiou]" case.sh | Finds lines containing **any vowel** (a, e, i, o, u). |
| grep "s\*n" case.sh | Matches "sn", "sssn", "snnn", etc. |
| grep "se\*n" case.sh | Matches "sn", "sen", "seen", "seeeeen", etc. |
| grep "selecti\*n" case.sh | Matches "selection", "selectn", "selectiiin", etc. |
| grep "sel.n" case.sh | Matches "selan", "selbn", "selZn", etc. |
| grep "selicti.n" case.sh | Matches "selictiXn" (where X is any character). |
| grep "selecti.n" case.sh | Matches "selection", "selectiYn", etc. |
| grep "s\*on" case.sh | Matches "on", "son", "sssson", etc. |