UNIX

- **Accepting user input and displaying outputs**

  ```
  #reading input
  read n

  #displaying output
  echo $n;
  ```

- **Operators**
  - Mathematical

    ```
    n=5
    m=2

    echo $(($n + $m));   # 5 + 2 = 7
    echo $(($n - $m));   # 5 - 2 = 3
    echo $(($n * $m));   # 5 * 2 = 10
    echo $(($n / $m));   # 5 / 2 = 2
    echo $(($n % $m));   # 5 % 2 = 1
    ```

  - Relational

    ```
    # -gt -> greater than
    # -ge -> greater than or equal to
    # -lt -> greater than
    # -le -> greater than or equal to
    # -eq -> equal to
    # -ne -> not equal to
    ```

  - Logical

    ```
    # && -> AND
    # || -> OR
    ```

```
# !  -> NOT
```

- **Conditional statements**
  - if statement
    - single if

      ```
      read n

      # longer version
      if [ $n -ge 0 ]; then
              echo "$n is positive";
      fi

      # shorter version
      [ $n -ge 0 ] && ( echo "$n is positive" )
      ```

    - if-else

      ```
      read n

      if [ $n -ge 0 ]; then
              echo "$n is positive";
      else
          echo "$n is negative"
      fi
      ```

    - if, elif and else

      ```
      read n

      if [ $n -gt 0 ]; then
              echo "$n is positive";
      elif [ $n -eq 0 ]; then
          echo "$n is zero"
      else
      ```

```
    echo "$n is negative"
fi
```

        - nested if statement

```
read n

if [ $n -gt 0 ]; then
    if [ $(($n % 2)) == 0 ]; then
    echo "$n is a positive even"
  fi
fi
```

    - case statement

```
read symbol

case "$symbol" in
  "H") echo "Hydrogen";;
  "O") echo "Oxygen";;
  "N") echo "Nitrogen";;
  (*)  echo "Other element";;
esac
```

- **Loops**
    - for loop

```
# range based iteration

for var in 1 2 3 4 5
do
  echo "$var";
done

# condition based iteration
```

```
read n

for (( i=0; i<n; i++ ));
do
  echo "$i";
done
```

- while loop

```
read n

i=0

# using while keyword
while [ $i -lt $n ]
do
   echo "$i"
   i=$(($i + 1))
done

# using until keyword
until [ $i -lt $n ]
do
   echo "$i"
   i=$(($i + 1))
done
```

- seq keyword

```
read n

# for loop using seq init step end

for i in $(seq 1 2 20)
do
  echo "$i";
done
```

- Infinite loops

  ```
  # infinite for-loop

  for (( ; ; ))
  do
     echo "infinite for loop";
  done

  # infinite while loop

  while :
  do
    echo "i";
  done
  ```

- **Command line arguments**

```
echo "$@";      # getting all command line arguments
echo "@#";      # getting the number of command line argum

# iterating through each command line argument
for i in "$@":
do
    echo "$i";
done
```

- **Array**

```
arr=(1 2 3)

echo "${arr[0]}";         # getting the first element
echo "${arr[@]}";         # getting all the array elements
echo "${#arr[@]}";        # getting length of the array
```

```
for i in "${arr[@]}"      # iterating through each element
do
  echo "$i";
done

arr+=(4)                  # appending array elements
```

- **Function**

```
# function declaration
function add() {

  # getting parameters
  sum=$(($1 + $2))
  echo "$sum"
}

# calling the function
add 1 2
```