**Javascript Assignment -**

**Generators**

1. What is the difference between a generator and a function?

Generator:

(1)  **Generator functions** provide a powerful alternative: they allow you to define an iterative algorithm by writing a single function whose execution is not continuous.

(2) Generator functions are written using the [function\*](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*) syntax.

(3) yeild: It gives control over function body, which lines to be executed and which to be paused. Also used as alternative of ‘return’ keyword.

(4) execution control is over next(), whenever next() function gets called, new value will be printed, else further lines of code get paused.

(4) Ex: function\* numberGen() {

    console.log("Inside Genrator");

    yield 1;

    yield 2;

    }

let num = numberGen();

console.log(num.next());

**Output:**

PS C:\Users\ADMIN\Documents\c\JS\_CODE> node assign\_18.js

Inside Genrator

{ value: 1, done: false }

Function: (1) functions are the block of code/instruction which we can use repeatedly.

(2) we can return the value from function or can directly print into function as well.

(3) At a time whole function to be executed, don’t have legacy to execute some lines of code.

(4) Ex: function sum(a,b) {

    return a+b;

}

let store = sum(3,6);

console.log(store);

OUTPUT: 9

1. What is the syntax of a generator?

Function\* fun\_name() {}

1. Are function generators iterable in JavaScript?

Yes, function generators are iterable in JavaScript.

Ex: function\* itr(arr) {

    let i = 0;

    while(true) {

        i++;

        yield arr[i];

    }

}

let num1 = itr([5,7,3,8]);

console.log(num1.next());

{ value: 7, done: false }

4. Create a generator for multiplying?

Output -



function\* mul(num) {

    yield num \* num;

}

console.log(mul(2).next());

function\* add(num) {

    yield num+num;

}

console.log(add(3).next());

1. Print an infinite series of natural numbers using a generator.

function\* itr() {

    let i = 0;

    while(true) {

        i++;

        //console.log(i);

        yield i;

    }

}

let num = itr();

while(true){

console.log(num.next().value);}

Output:

1

2

3

.

.

.

1. Create a generator that can throw an exception.

function\* gen() {

    while (true) {

      try {

        yield 42;

      } catch (e) {

        console.log('Error caught!');

      }

    }

  }

  const g = gen();

  g.next();

  // { value: 42, done: false }

  g.throw(new Error('Something went wrong'));

Output:

PS C:\Users\ADMIN\Documents\c\JS\_CODE> node tryyy.js

Error caught!