```
// Function Generator Example1
function* display() {
  console.log("CVR college");
  console.log("CSE Department");
  console.log("III Year CSE");
}
const obj = display();
console.log(obj);
console.log(obj.next());

// Function Generator Example2
function* display() {
  console.log("CVR college");
  yield;
  console.log("CSE Department");
  yield;
  console.log("III Year CSE");
}
const obj = display();
console.log(obj.next());
console.log(obj.next());
console.log(obj.next());

// Function Generator Example3
function* generatorfunc() {
  yield 1;
  yield 2;
  yield 3;
}
const genfun = generatorfunc();
console.log(genfun);
console.log(genfun.next());
console.log(genfun.next());
console.log(genfun.next());
console.log(genfun.next());

// Function Generator Example4
function* simpleGenerator() {
  console.log("Before 1");
  yield 1;
  console.log("After 1");
  console.log("Before 2");
  yield 2;
  console.log("After 2");
  console.log("Before 3");
  yield 3;
  console.log("After 3");
  console.log("Exit");
}
let genObj = simpleGenerator();
console.log(genObj.next());
console.log(genObj.next());
console.log(genObj.next());
console.log(genObj.next());

// Use cases
// 1. Multiple generators
function* simpleGenerator() {
  yield 1;
  yield 2;
  yield 3;
}
```

```
let genObj1 = simpleGenerator();
let genObj2 = simpleGenerator();
console.log(genObj1.next());
console.log(genObj1.next());
console.log(genObj2.next());
console.log(genObj2.next());
console.log(genObj1.next());
console.log(genObj2.next());
console.log(genObj1.next());
console.log(genObj2.next());

// 2. Infinite loop
function* genId() {
  let id = 1;
  while (true) {
    yield id;
    id++;
  }
}
const fungen = genId();
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
console.log(fungen.next());
const fungen2 = genId();
console.log(fungen2.next());
```

```
// 3. Iterating the elements of an array
function* generatorfn(array) {
  for (let i = 0; i < array.length; i++) {
    yield array[i];
  }
}
const genobj = generatorfn([1, 3, 5, 7, 9]);
console.log(genobj.next());
console.log(genobj.next());
console.log(genobj.next());
console.log(genobj.next());
console.log(genobj.next());
console.log(genobj.next());

// 4. return can be used to exit out of the generator.
function* display() {
  yield 10;
  yield "How are you";
  return 20;
}
const obj = display();
console.log(obj.next());
console.log(obj.next());
console.log(obj.next());
console.log(obj.next());

// 5. using for..of loop
function* genfun() {
  yield 1;
  yield 2;
  yield 3;
  yield 4;
  yield 5;
  return 6;
}

for (a of genfun()) {
  console.log(a);
}
```