ScriptPorter Unilang

Sample Programs

***Notice:** We recommend not copy-pasting code directly into the compiler, as it may cause formatting or execution errors for some programs. If you run into issues, please enter the code manually.

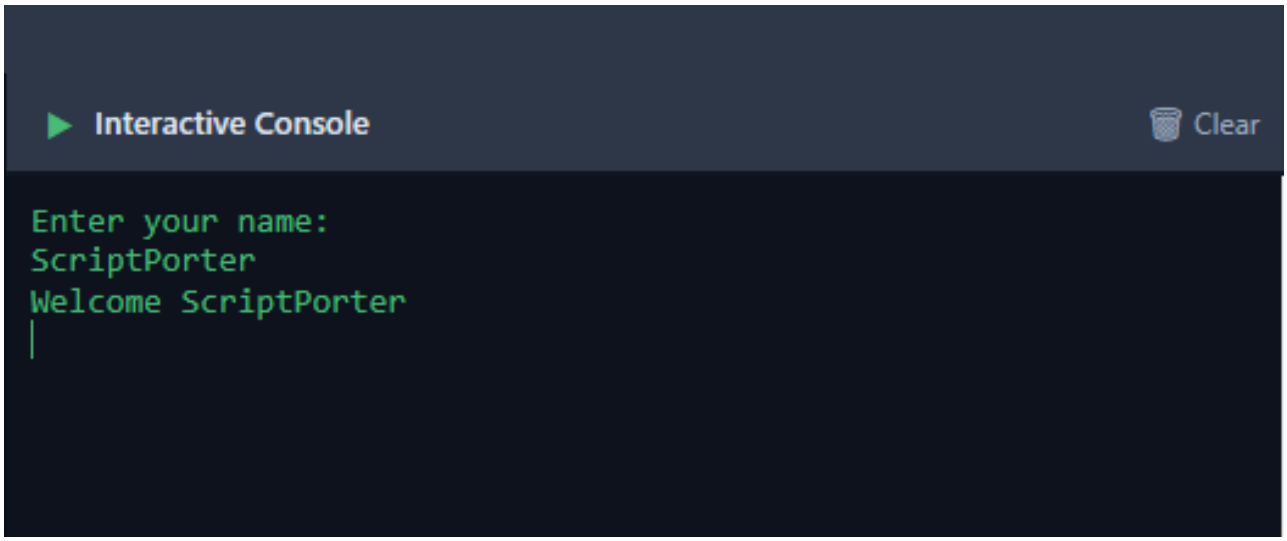
1. Simple I/O

program

```
//output operator
->"Enter your name:"
//reading user input
name<-

->"Welcome",name
```

output



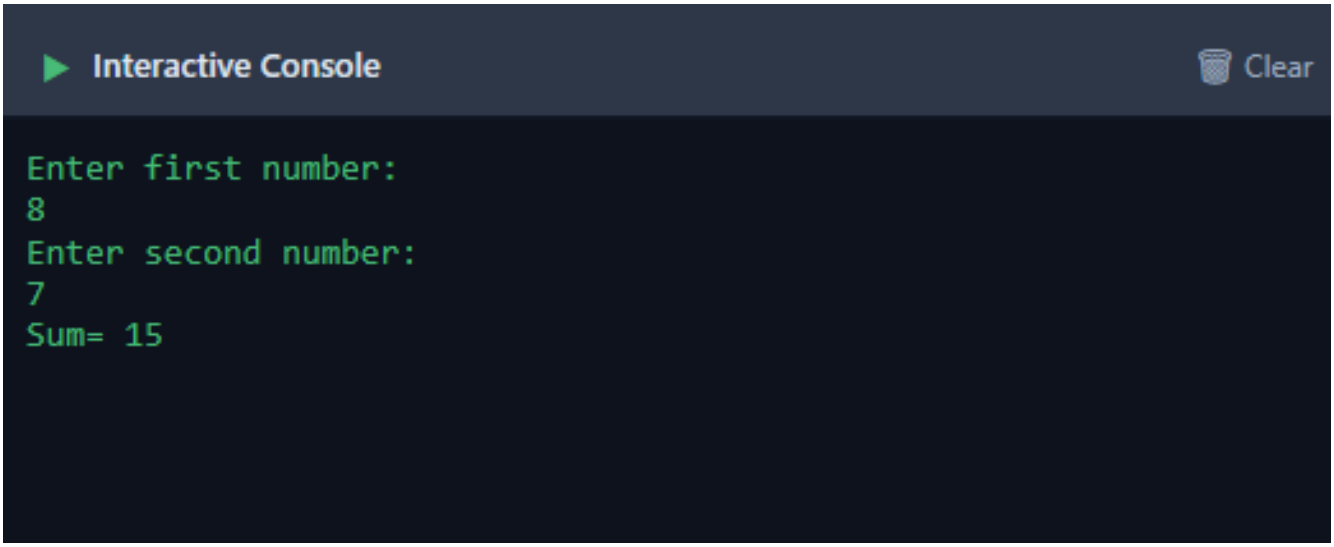
2. Sum of two numbers

program

```
->"Enter first number:"
//expects an integer value from user (0)
a<-(0)
->"Enter second number:"
b<-(0)

//outputs sum of the numbers
->"Sum=",a+b
```

output



3. Positive, negative or zero program

```
->"Enter a number:"
x<-(0)

//if, else-if, else statements
?(x<0)
{
   ->"Negative Number"
}

??(x==0)
{
  ->"Zero"
}

$()
{
  ->"Positive Number"
}
```

output


Interactive Console
Clear
Enter a number:
7
Positive Number

4. Odd or even

program

```
->"Enter a number:"
x<-(0)

//if, else statements
?(x%2==0)
{
 ->"Even number"
}

$()
{
 ->"Odd number"
}
```

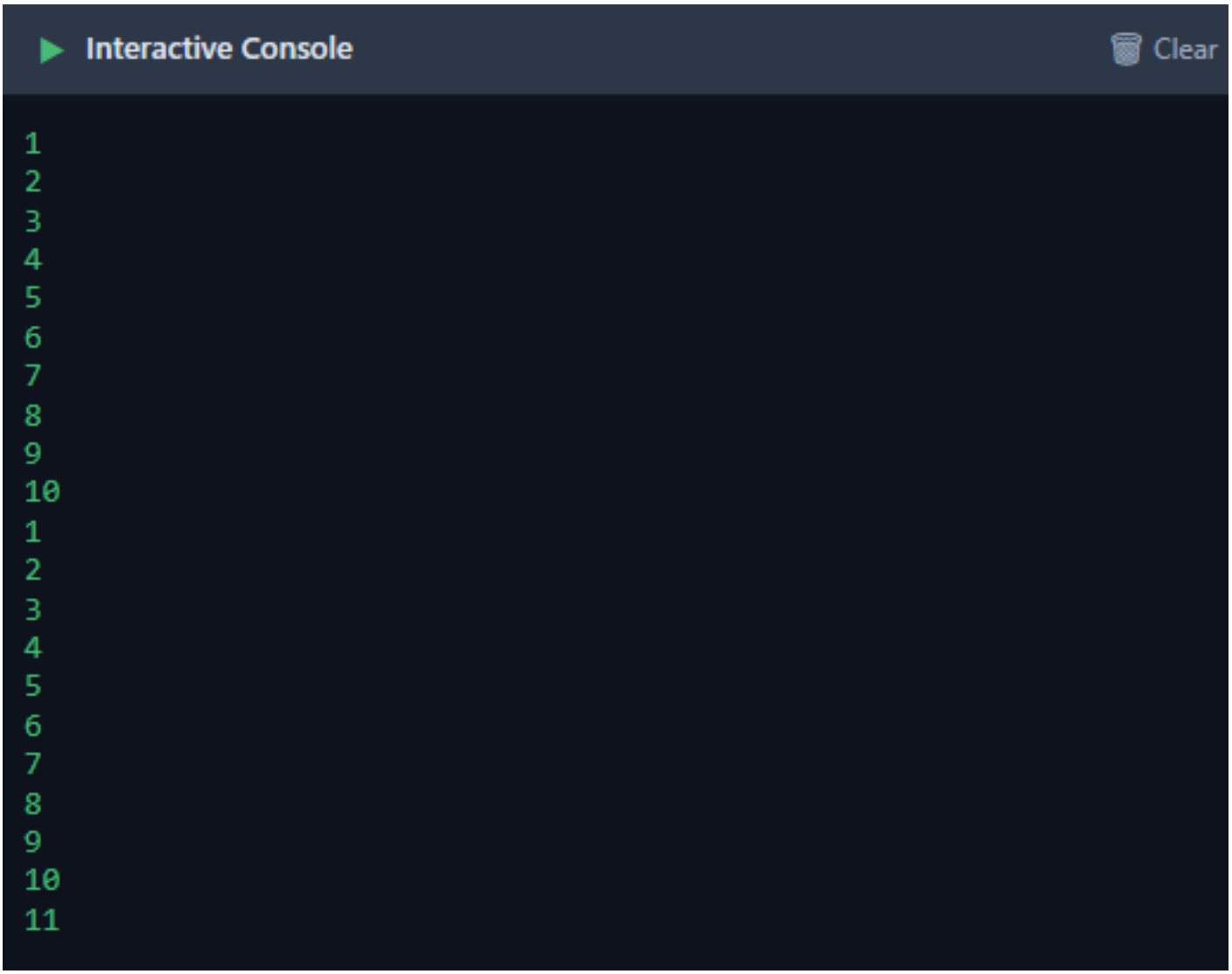
output



5. Printing 1-10 using different loops program

```
i=1
//while loop
(i<=10)...
{
   ->i
   i=i+1
}
//for loop
(j=1,11,1)...
{
   ->j
}
//do-while loop (it will print 11 because the value of i is 10)
...(i<=10)...
{
   ->i
   i=i+1
}
```

output



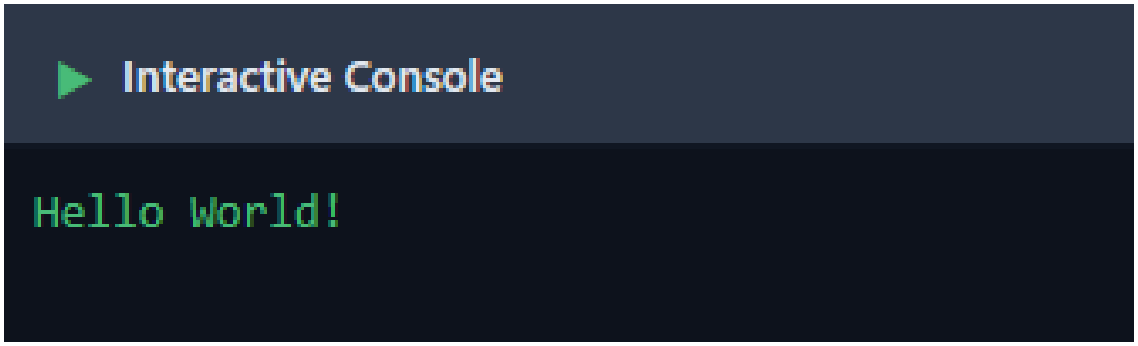
6. Function without arguments

program

```
//Creating a function named 'Greet'
Greet<-{

                ->"Hello World!"
 }
//Calling the function 'Greet'
(Greet)
```

output



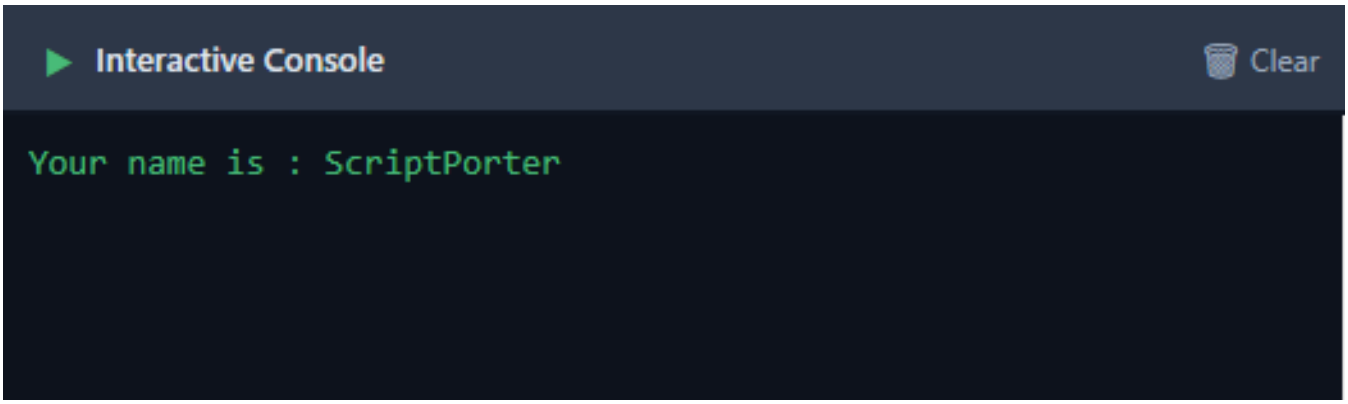
7. Function with arguments

program

```
//Creating a function named 'PrintName'
PrintName<-
 {
     ->"Your name is :", x
 }<-x

//Calling function 'PrintName' and passing argument
(PrintName:"ScriptPorter")
```

output



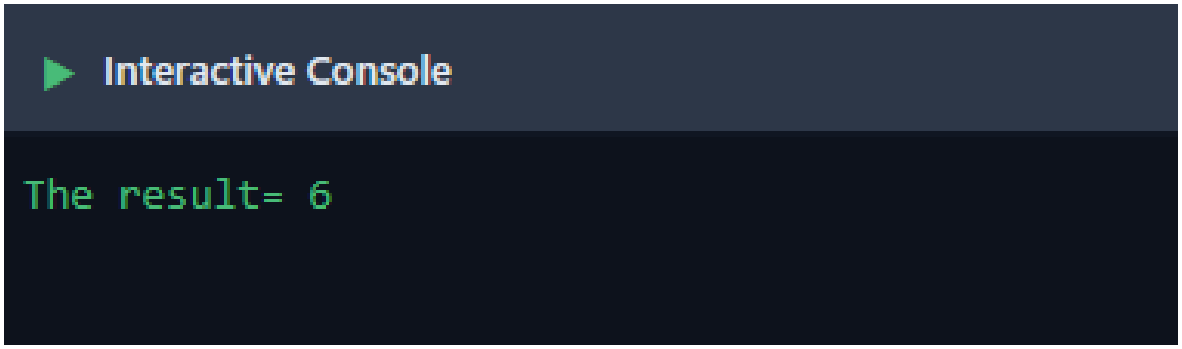
8. Function with return value

program

```
//Creates a function named 'Subtract'
Subtract<-{
                ret x-y
 }<-x,y

//Calls the function and stores return value in ‘res’
res=(Subtract:10,4)
->"The result=",res
```

output



9. Switch

progam

```
-> "Enter a number (1, 2, or 3):"
temp <-(0)

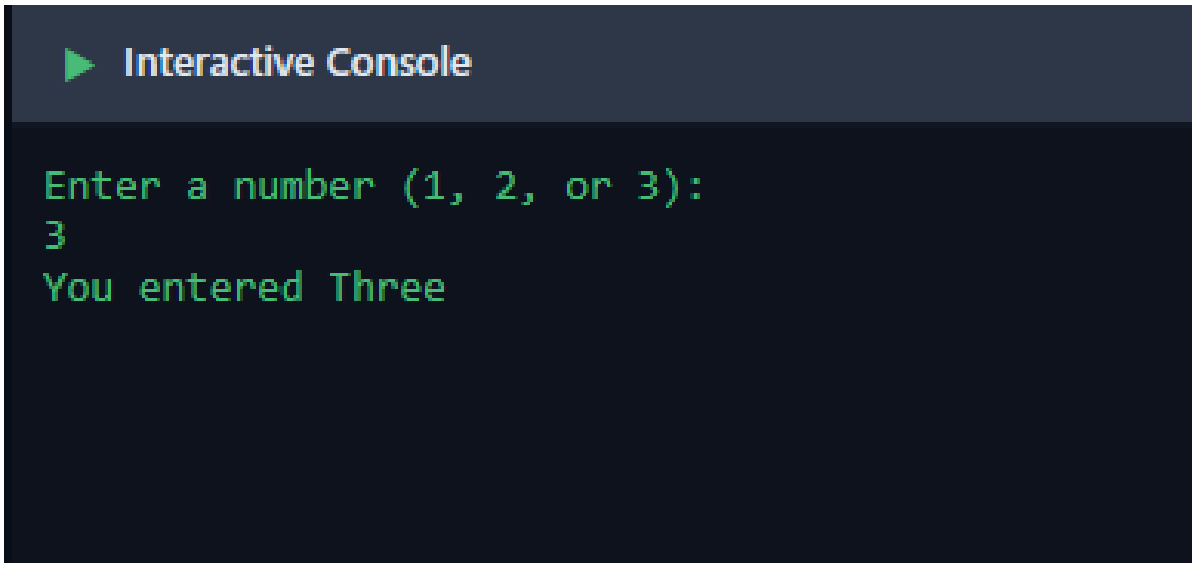
//You can use numbers, characters and strings
//For using strings and characters use " "
...(temp)
{
  1:
     -> "You entered One"

  2:
     -> "You entered Two"

  3:
     -> "You entered Three"

  def:
     ->"Invalid option!"
}
```

Output


Interactive Console
Enter a number (1, 2, or 3):
3
You entered Three

10. String operations

program

```
//Using string functions
//(len) - returns length of string
//(up)     - returns string after converting to uppercase
//(low) - returns string after converting to lowercase

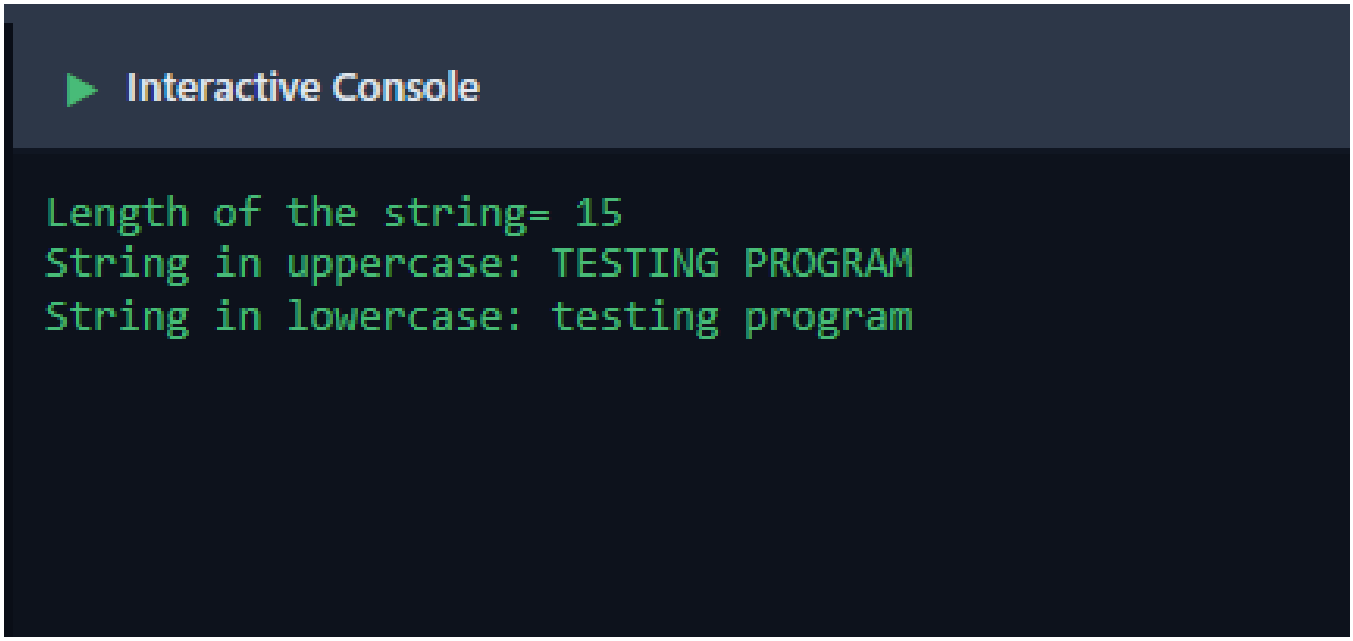
str="testing program"

l=(len:str)

->"Length of the string=",l

str2=(up:str)
str3=(low:str)
->"String in uppercase:",str2
->"String in lowercase:",str3
```

output


Interactive Console
Length of the string= 15
String in uppercase: TESTING PROGRAM
String in lowercase: testing program

11. Array operations

program

```
arr1=[]
->"Enter no.of elements:"
n<-(0)
->"Enter elements:"

(i=0,n,1)...
{
    x<-(0)
    (add:arr1,i,x) //Adding elements(x) at index(i)
}

->"The array is:",arr1

s=(len:arr1)   //size of the array
->"Length of the array=",s

->"Enter index of element to be removed:"
rm<-(0)
(remove:arr1,rm)    //remove element
```

```
->"Array after removing elements:",arr1

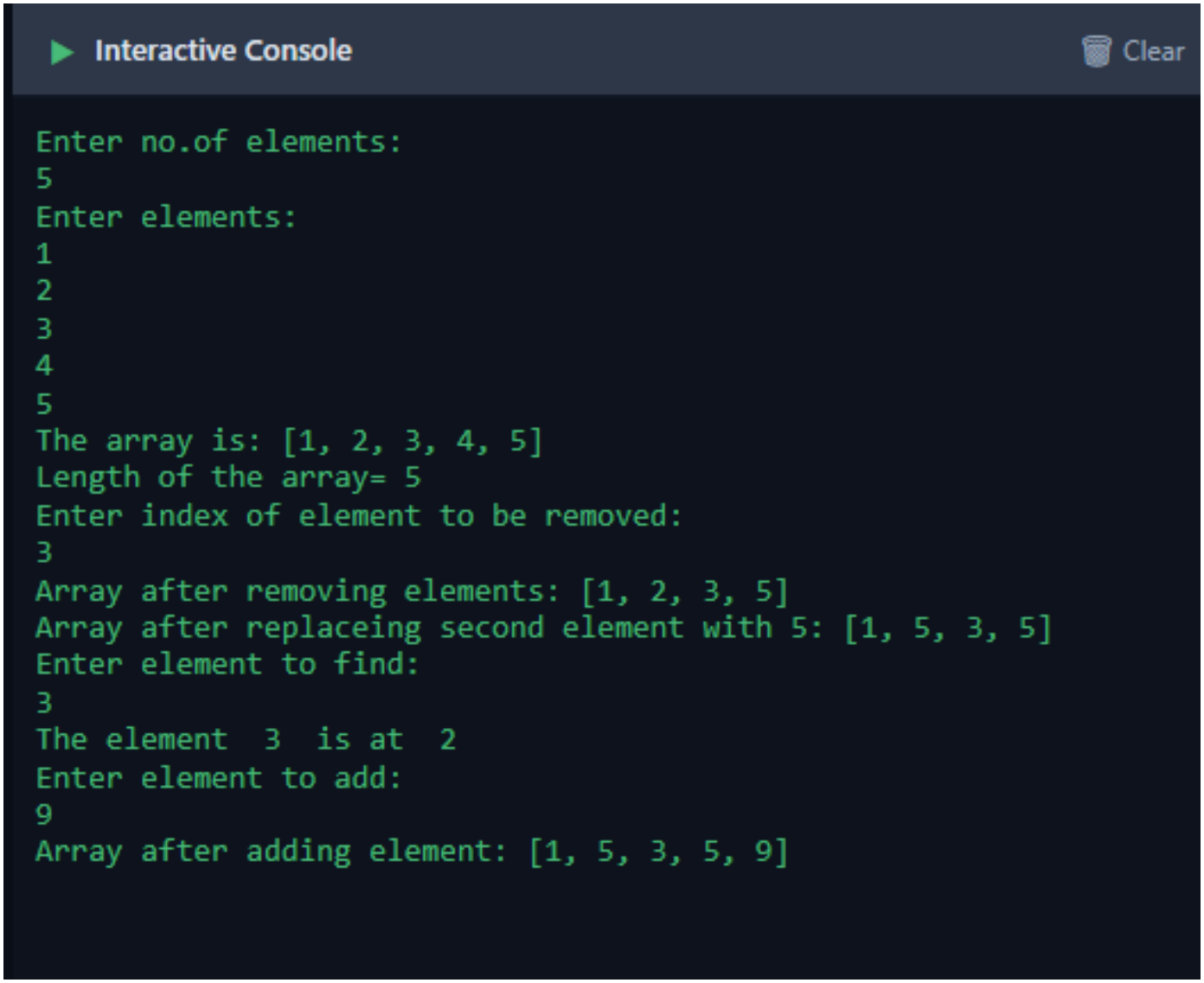
(replace:arr1,1,5)      //replacing element
->"Array after replaceing second element
with 5:",arr1

->"Enter element to find:"
el<-(0)

loc=(where:arr1,el) //finding index of element
->"The element ",el," is at ",loc

->"Enter element to add:"
num<-(0)
(add:arr1,num)
->"Array after adding element:",arr1
```

output


Interactive Console
Clear
Enter no.of elements:
5
Enter elements:
1
2
3
4
5
The array is: [1, 2, 3, 4, 5]
Length of the array= 5
Enter index of element to be removed:
3
Array after removing elements: [1, 2, 3, 5]
Array after replaceing second element with 5: [1, 5, 3, 5]
Enter element to find:
3
The element 3 is at 2
Enter element to add:
9
Array after adding element: [1, 5, 3, 5, 9]

12. Program using class

program

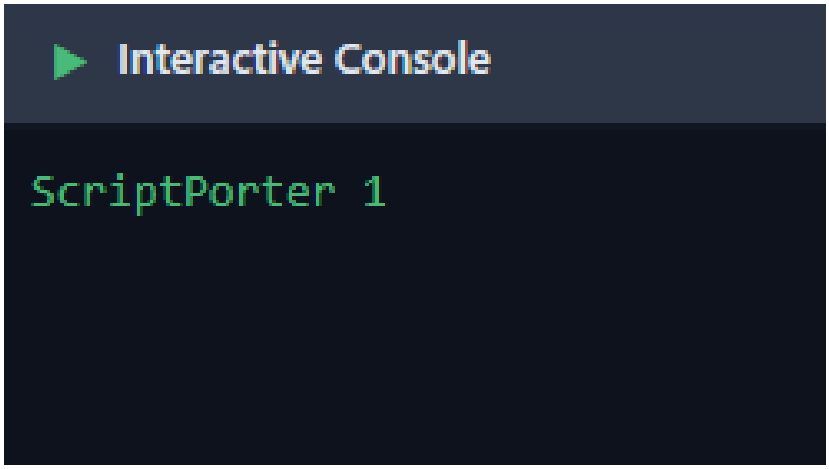
```
//Creating class named 'Person'
Person<-[
        //variables
        Name="NA"
        Age=10

        //functions
        Display<-{

                ->this.Name,this.Age
            }
        ]

x=Person()      //creating class object
x.Name="ScriptPorter"  //changing class properties
x.Age=1
x.(Display)     //calling class function
```

output



13. Class with inheritance

program

```
//Creating parent class 'Vehicle'
vehicle<-[
                Wheels=0
                Speed=0

                display<-{

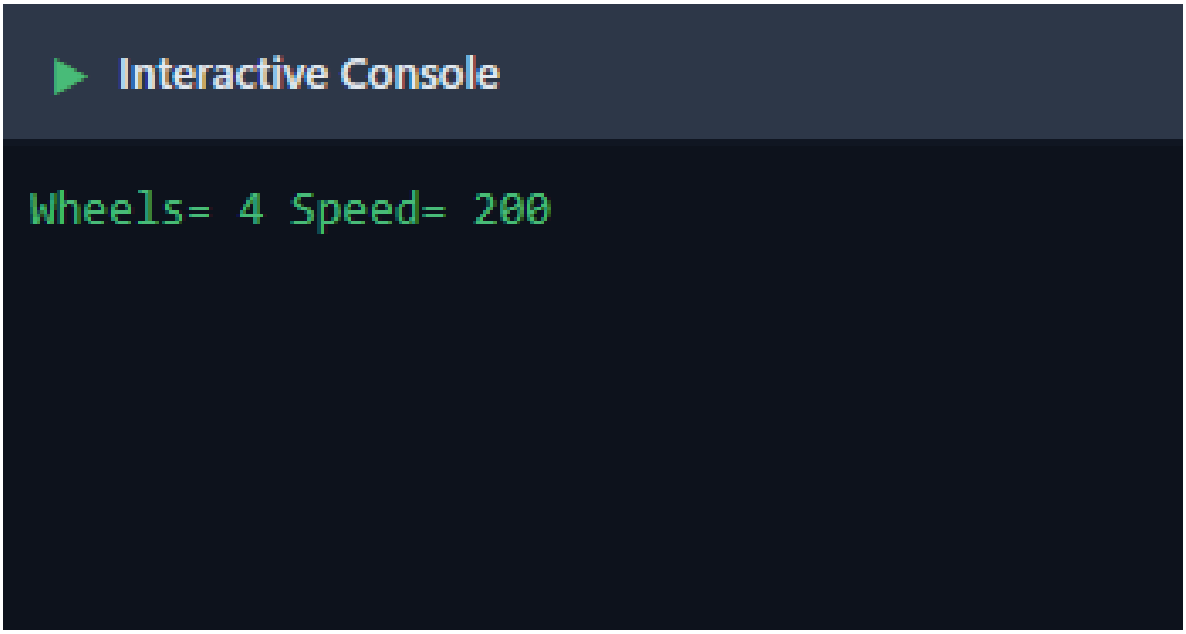
                    ->"Wheels=",this.Wheels,
                    "Speed=",this.Speed
                 }

            ]

//Creating child class of 'Vehicle'
car@vehicle<-[
                    Wheels=4
                    Speed=200
                ]
```

```
//creating object of child class ‘car’
c1=car()
c1.(display)
```

output



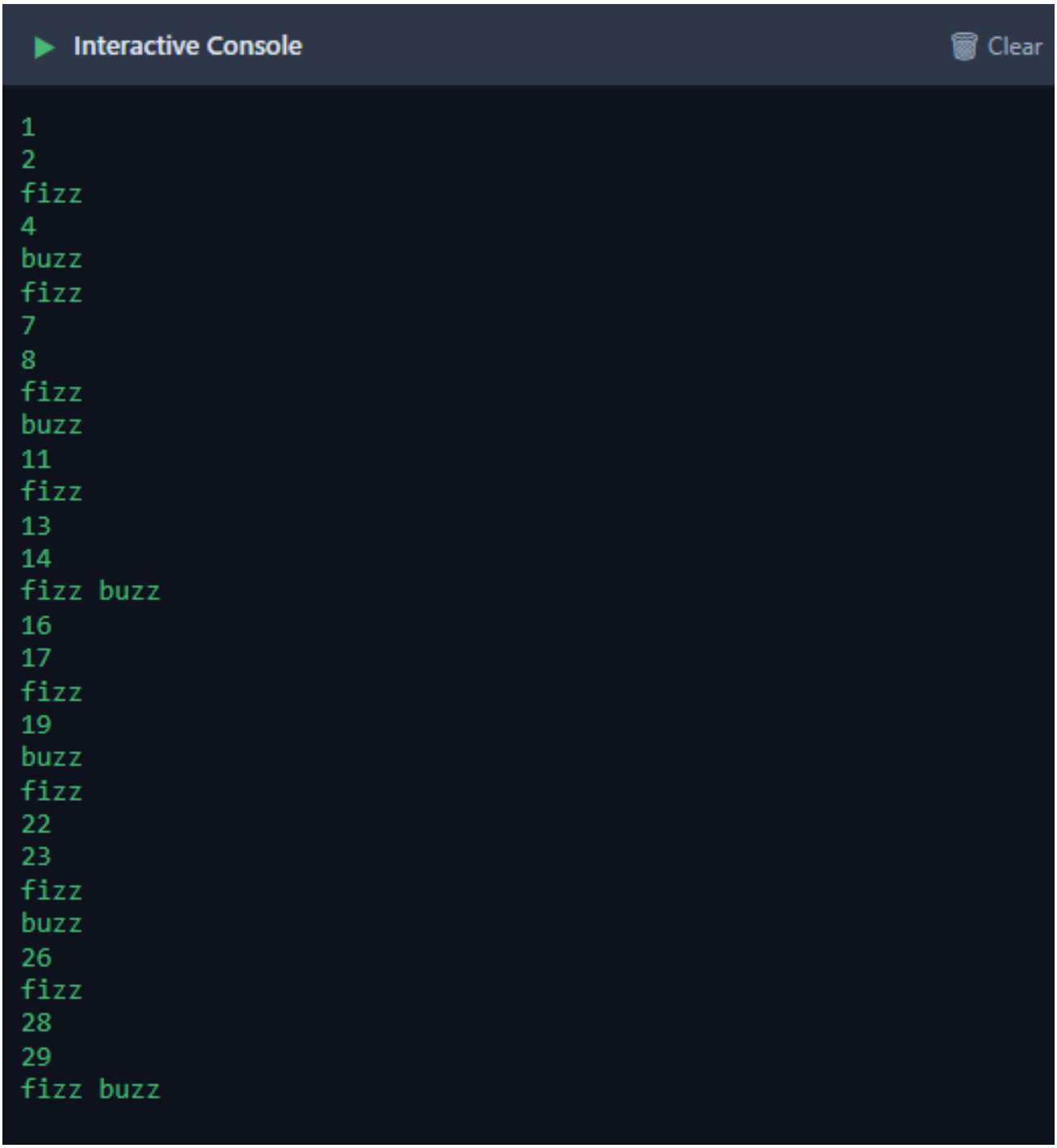
14. Fizz buzz game

program

```
//'for' loop 1-30
(i=1,31,1)...
 {
        ?(i%3==0 && i%5==0)
        {
            ->"fizz buzz"
         }
        ??(i%3==0)
         {
            ->"fizz"
        }
        ??(i%5==0)
        {
            ->"buzz"
        }
        $()
         {
            ->i
```

```
        }
}
```

output


Interactive Console
Clear
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizz buzz
16
17
fizz
19
buzz
fizz
22
23
fizz
buzz
26
fizz
28
29
fizz buzz

15. Odd-Even array program

```
//Program for printing even and odd numbers in range 1-10

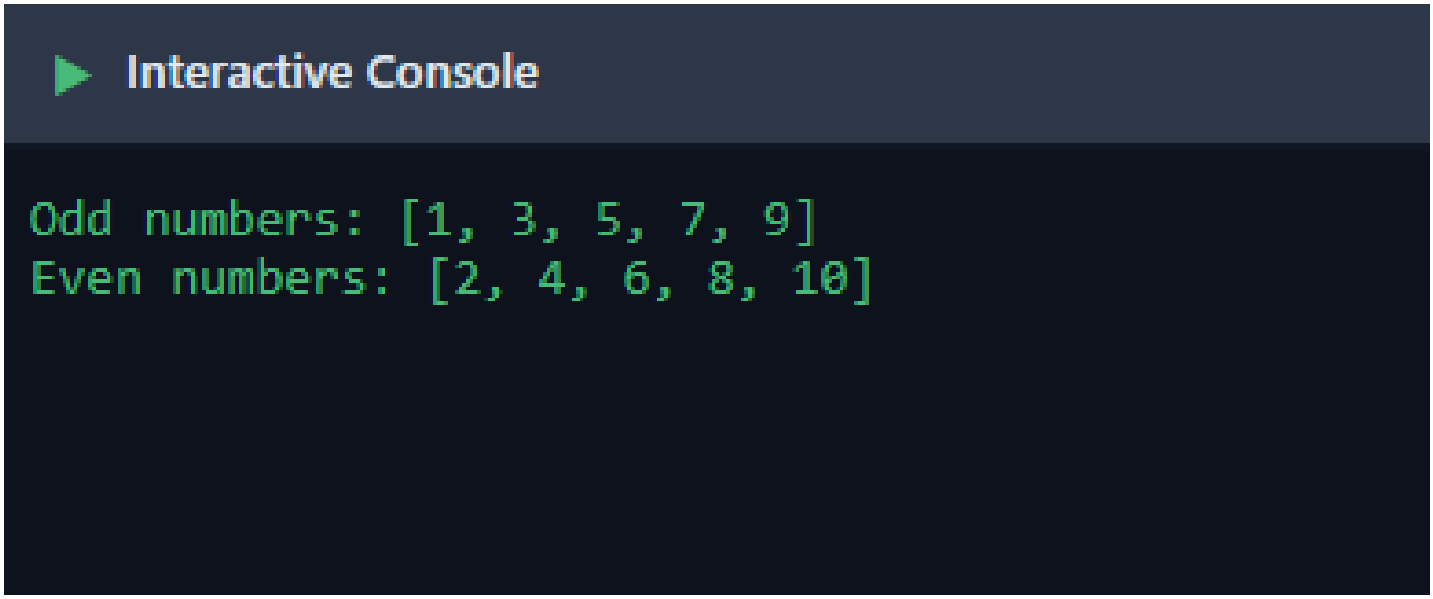
//creating arrays for storing odd and even numbers

odd=[]
even=[]


(i=1,11,1)...
{
    ?(i%2==0)
    {
        (add:even,i)    //adding even numbers to array
    }
    $()
    {
        (add:odd,i)    //adding odd numbers to array
    }
}
```

->"Odd numbers:",odd
->"Even numbers:",even

output



16. Calculation with functions

program

```
//Creating function 'sum' without return value
sum<-{
        ->"Sum=",x+y
    }<-x,y


//Creating function 'diff' without return value
diff<-{
        ->"Difference=",x-y
    }<-x,y

//Creating function 'pro' with return value
pro<-{
        ret x*y
    }<-x,y

//Creating function 'quo' with return value
quo<-{
        ret x/y
    }<-x,y
```

```
//Calling functions
(sum:10,5)
(diff:10,5)
->"Product=",(pro:10,5)
->"Quotient=",(quo:10,5)
```

output

