Task 1:

Problem Statement

Create a Scala application to find the GCD of two numbers.

Task: GCD of two numbers

Before going to create a SCALA application, we will just see the overview of GCD formula.

Greatest Common Divisor (GCD) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.

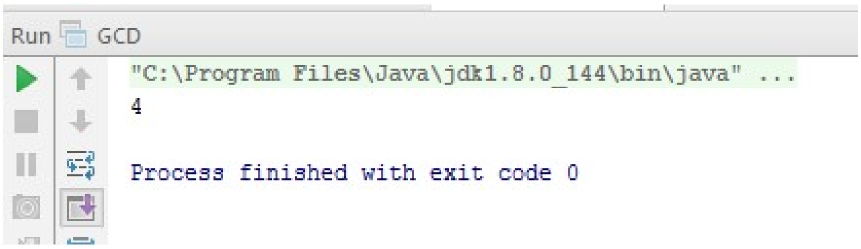
For example, the **gcd** of **8 and 12 is 4.**

Scala Application using IntelliJ

In the below scala code, we are going to find the gcd of the two numbers 12 and 8.

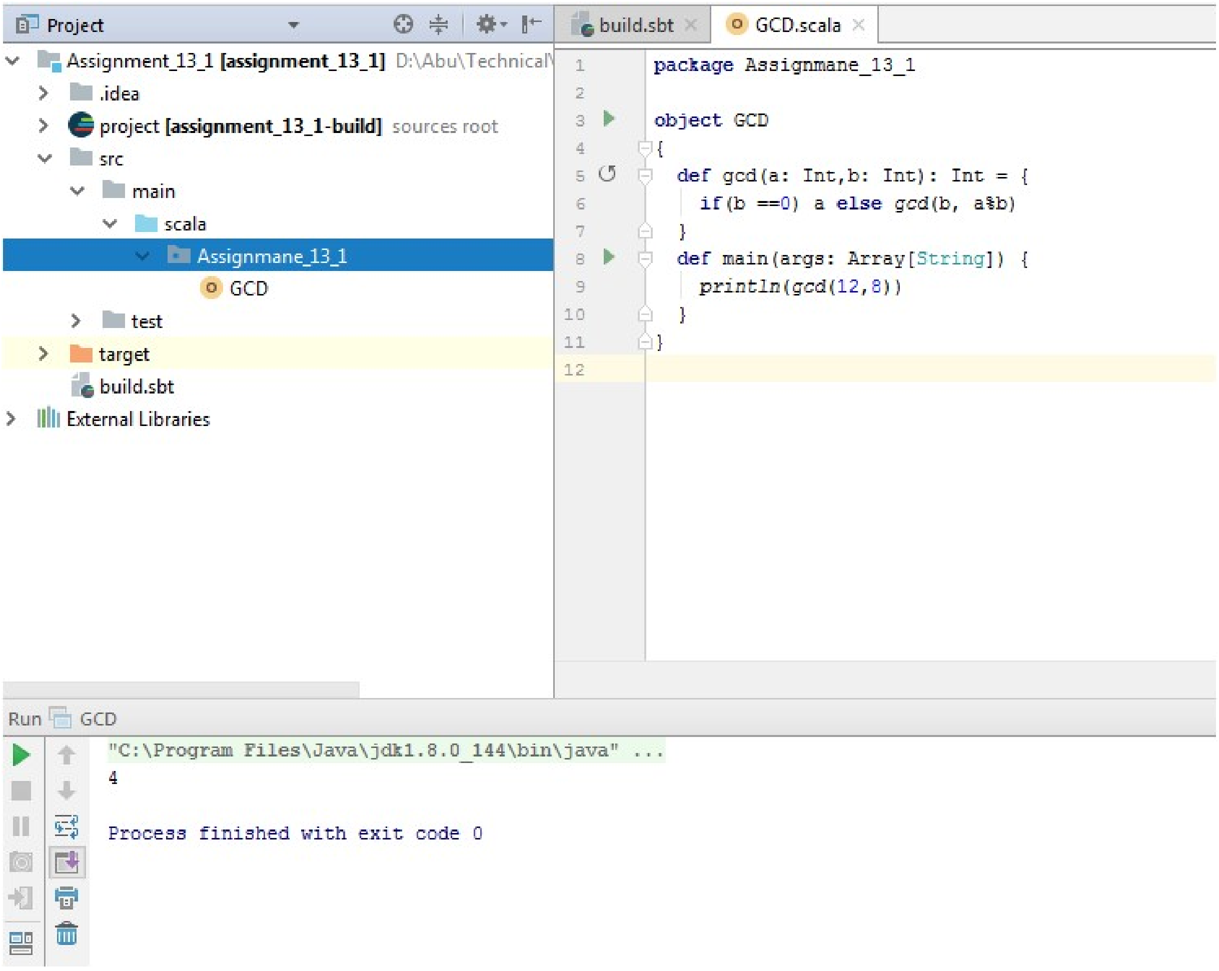
```
package Assignmane_13_1                    //package which we created

object GCD//a new object GCD is created
{
defgcd(a: Int,b: Int): Int = {   // declaring a function gcd
if(b ==0) a else gcd(b, a%b)      2 integer variables a,b
  }
defmain(args: Array[String])      //Our main function takes in a named
{                                 parameterargs which is an Array of String.
println(gcd(12,8))                //  print the result
  }
}
```

Required Output



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
4

Process finished with exit code 0
```

Screen Shot



```scala
package Assignmane_13_1

object GCD
{
  def gcd(a: Int,b: Int): Int = {
    if(b ==0) a else gcd(b, a%b)
  }
  def main(args: Array[String]) {
    println(gcd(12,8))
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
4

Process finished with exit code 0
```

Task 2:

A Fibonacci series (starting from 1) written in order without any spaces in between, thusproducing a sequence of digits.

Write a Scala application to find the nth digit in the sequence.

- Write the function using standard for loop
- Write the function using recursion

Before going in to the tasks, we will just see an over view that what is he Fibonacci number,

The Fibonacci sequence is the series of numbers,
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it.

The 2 is found by adding the two numbers before it (1+1)

The 3 is found by adding the two numbers before it (1+2),

And the 5 is (2+3),

And so on!

Example: the next number in the sequence above is 21+34 = 55

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
xn =	0	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987

Formula,

xn = xn-1 + xn-2

Example,

The 8th term is the 7th term plus the 6th term: X8 = X7+X6

From the above table,

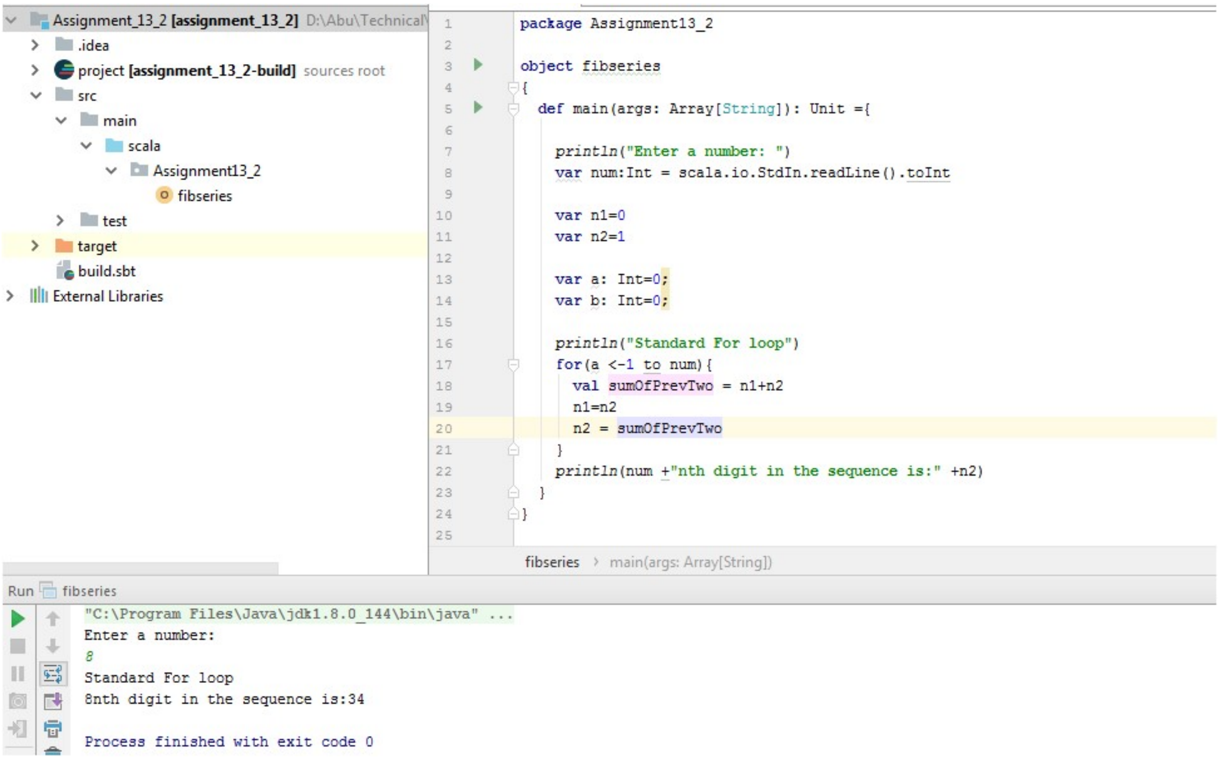
The 8th term is 21, hence the 7th term 21+the 6th term 13 = 34.

Task 1: write function using standard for loop

Scala code

```
package Assignment13_2

object fibseries
{
defmain(args: Array[String]): Unit ={

println("Enter a number: ")
varnum:Int = scala.io.StdIn.readLine().toInt

varn1=0
varn2=1

vara: Int=0;
varb: Int=0;
```

Screen Shot:

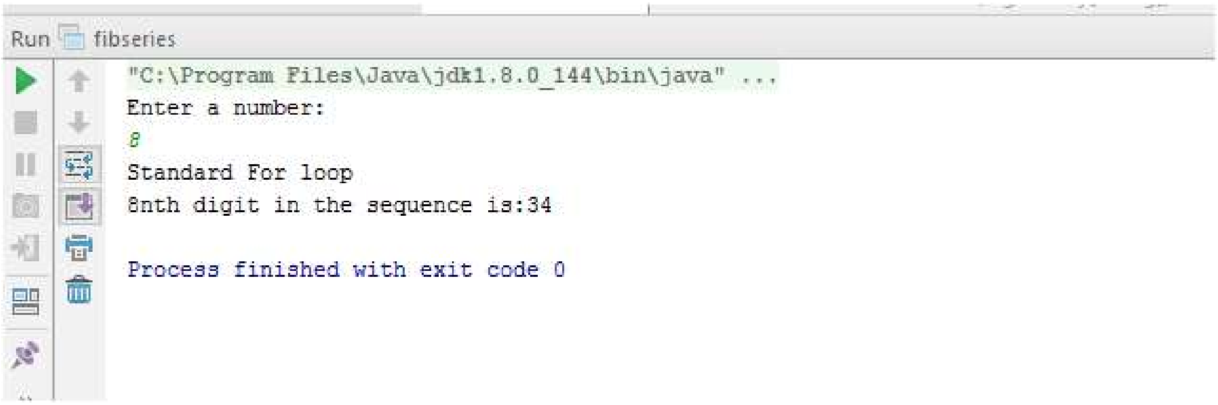

Assignment_13_2 [assignment_13_2] D:\Abu\Technical\
.idea
project [assignment_13_2-build] sources root
src
main
scala
Assignment13_2
fibseries
test
target
build.sbt
External Libraries
fibseries > main(args: Array[String])
Run fibseries

```scala
package Assignment13_2

object fibseries
{
  def main(args: Array[String]): Unit ={

    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt

    var n1=0
    var n2=1

    var a: Int=0;
    var b: Int=0;

    println("Standard For loop")
    for(a <-1 to num){
      val sumOfPrevTwo = n1+n2
      n1=n2
      n2 = sumOfPrevTwo
    }
    println(num +"nth digit in the sequence is:" +n2)
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
8
Standard For loop
8nth digit in the sequence is:34

Process finished with exit code 0
```

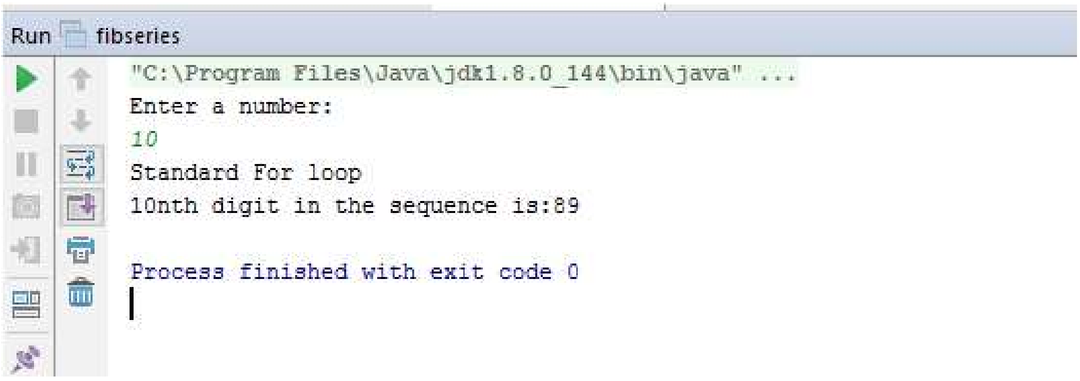
Output

When we provide number 8 as input, the 8th digit in the Fibonacci sequence is 34.



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
8
Standard For loop
8nth digit in the sequence is:34

Process finished with exit code 0
```

If we give the input as 10, the 10th digit of Fibonacci sequence is 89



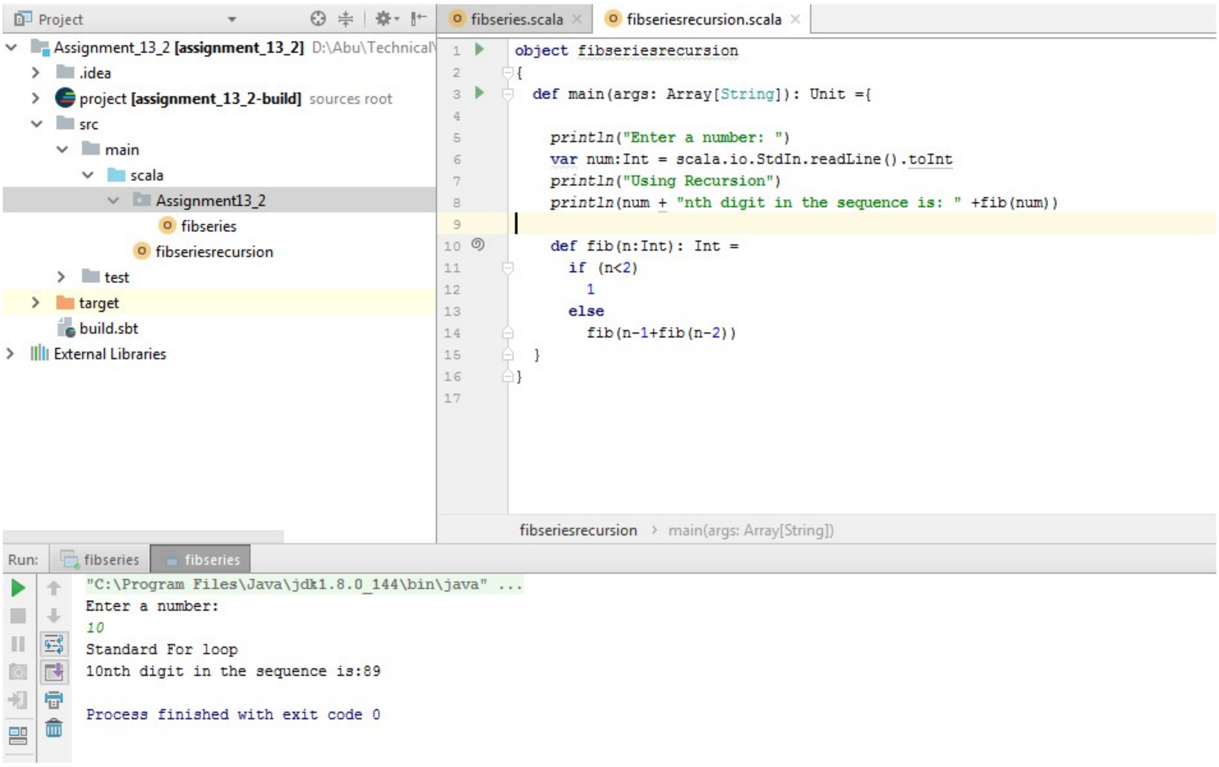
```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
10
Standard For loop
10nth digit in the sequence is:89

Process finished with exit code 0
```

Task2 - Write the function using recursion

Scala code

```
object fibseriesrecursion
{
defmain(args: Array[String]): Unit ={

println("Enter a number: ")
varnum:Int = scala.io.StdIn.readLine().toInt
println("Using Recursion")
println(num + "nth digit in the sequence is: " +fib(num))

deffib(n:Int): Int =
if (n<2)
1
else
fib(n-1+fib(n-2))
   }
}
```

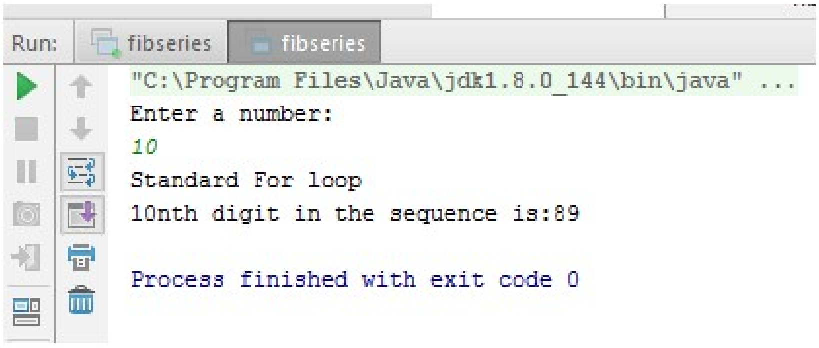
Screen shot:



```
object fibseriesrecursion
{
  def main(args: Array[String]): Unit ={

    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt
    println("Using Recursion")
    println(num + "nth digit in the sequence is: " +fib(num))

    def fib(n:Int): Int =
      if (n<2)
        1
      else
        fib(n-1+fib(n-2))
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
10
Standard For loop
10nth digit in the sequence is:89

Process finished with exit code 0
```

Output



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
10
Standard For loop
10nth digit in the sequence is:89

Process finished with exit code 0
```

Task 3:

Find square root of number using Babylonian method.

1. 1 Start with an arbitrary positive start value x (the closer to theRoot, the better).
2. Initialize y = 1.
3. Do following until desired approximation is achieved.
   a) Get the next approximation for root using average of x and y
   b) Set y = n/x

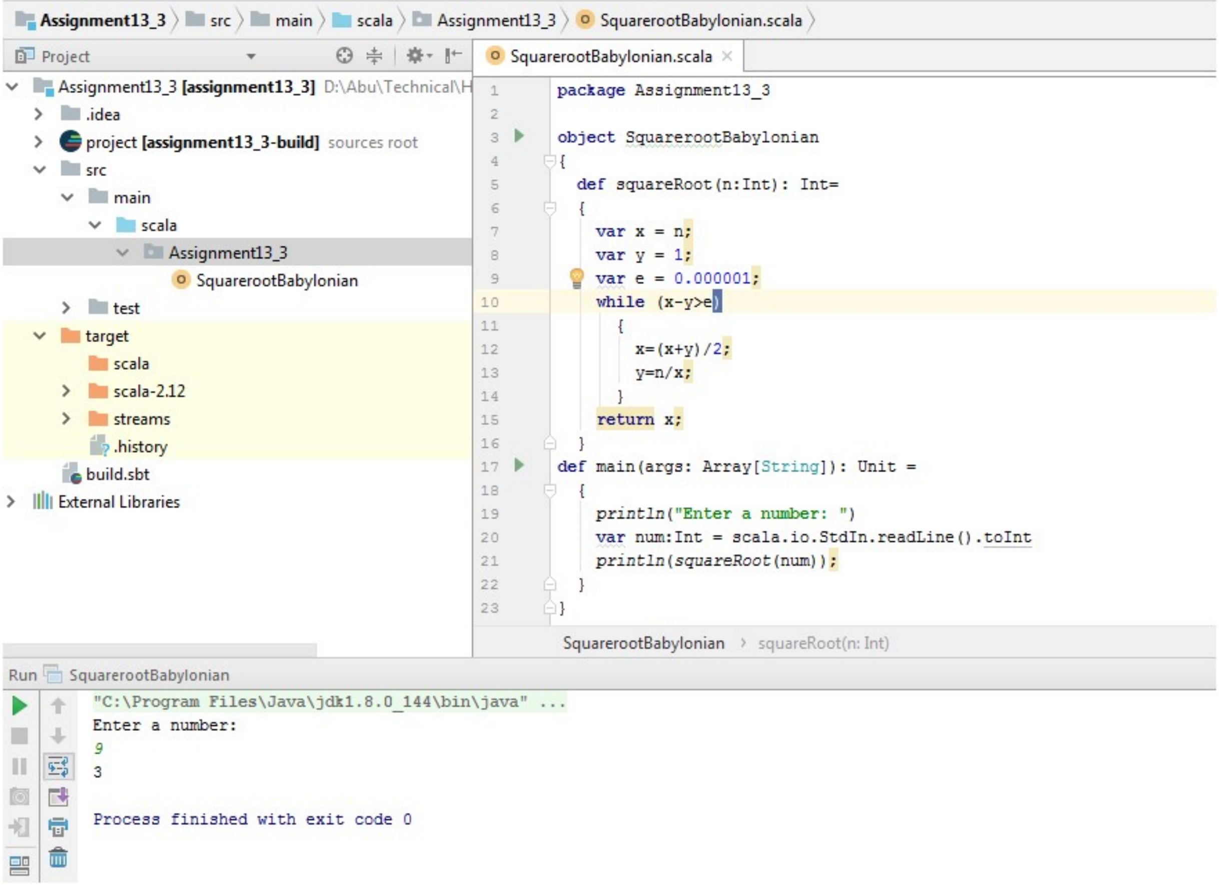
The Babylonian method for finding square roots involves dividing and averaging, over and over, to obtain a more accurate solution with each repeat of the process. Step 2: Divide your original number by your guess. Step 3: Find the average of these numbers. Step 4: Use this average as your next guess.

Task – Find square root if a number using Babylonian Method

Scala code

```
package Assignment13_3

object SquarerootBabylonian
{
defsquareRoot(n:Int): Int=
  {
varx = n;
vary = 1;
vare = 0.000001;
while (x-y>e)
      {
        x=(x+y)/2;
        y=n/x;
      }
return x;
  }
defmain(args: Array[String]): Unit =
  {
println("Enter a number: ")
varnum:Int = scala.io.StdIn.readLine().toInt
println(squareRoot(num));
  }
}
```

Screen Shot



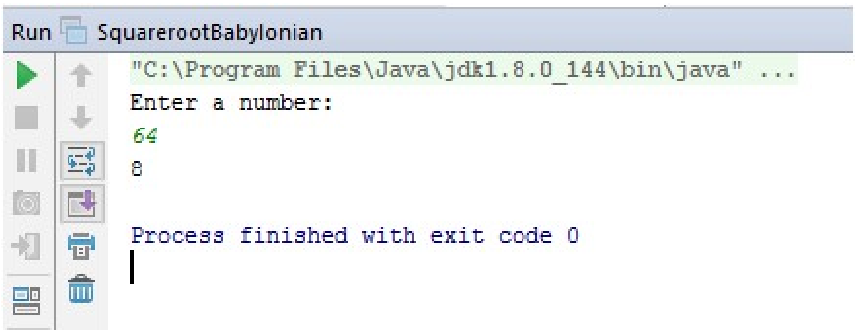
Assignment13_3 > src > main > scala > Assignment13_3 > SquarerootBabylonian.scala
Project
Assignment13_3 [assignment13_3] D:\Abu\Technical\H
.idea
project [assignment13_3-build] sources root
src
main
scala
Assignment13_3
SquarerootBabylonian
test
target
scala
scala-2.12
streams
.history
build.sbt
External Libraries
SquarerootBabylonian.scala
SquarerootBabylonian > squareRoot(n: Int)
Run SquarerootBabylonian

```
package Assignment13_3

object SquarerootBabylonian
{
  def squareRoot(n:Int): Int=
  {
    var x = n;
    var y = 1;
    var e = 0.000001;
    while (x-y>e)
      {
        x=(x+y)/2;
        y=n/x;
      }
    return x;
  }
def main(args: Array[String]): Unit =
  {
    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt
    println(squareRoot(num));
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
9
3

Process finished with exit code 0
```

Output

If we enter a number 64, the square root of that value is 8



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
64
8

Process finished with exit code 0
```

Find square root of number using Babylonian method.

1. 1 Start with an arbitrary positive start value x (the closer to theRoot, the better).
2. Initialize y = 1.
3. Do following until desired approximation is achieved.
   a) Get the next approximation for root using average of x and y
   b) Set y = n/x

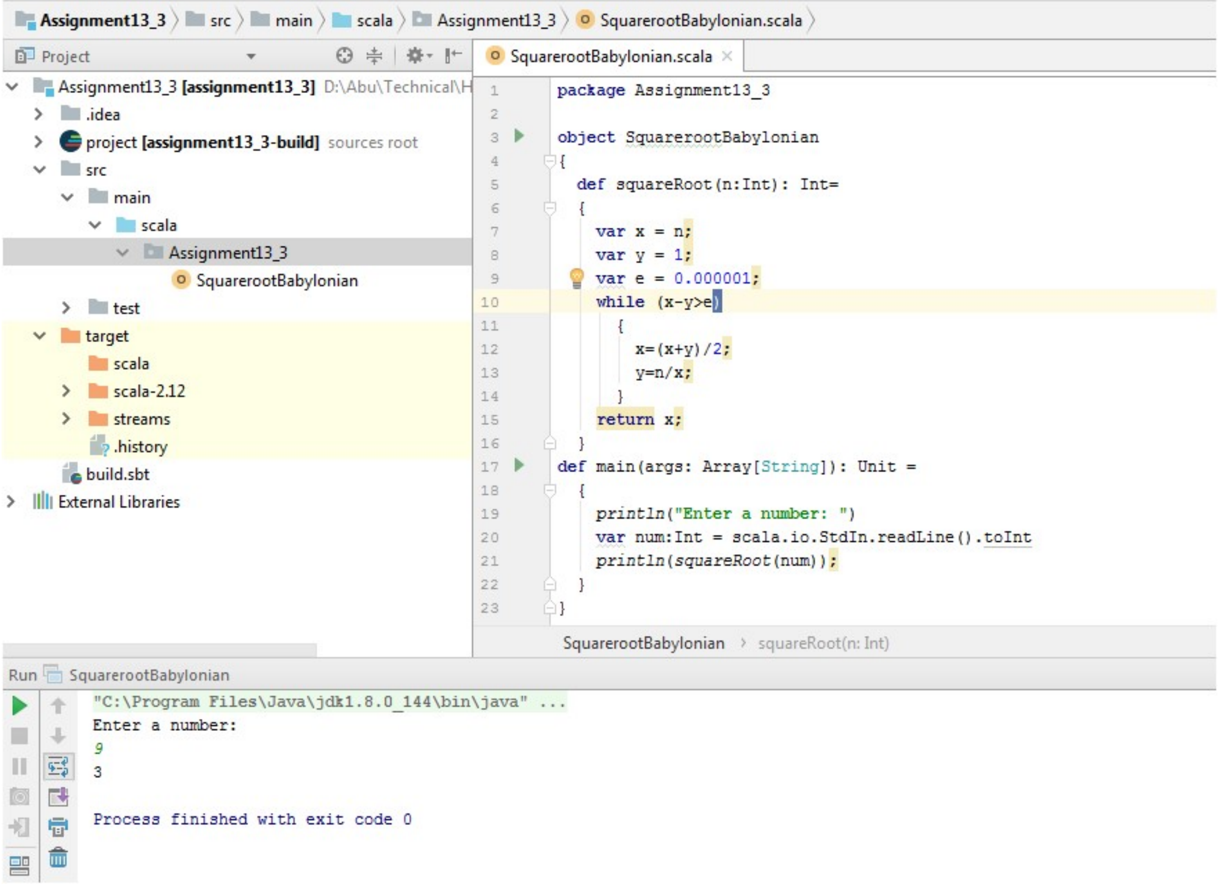
The Babylonian method for finding square roots involves dividing and averaging, over and over, to obtain a more accurate solution with each repeat of the process. Step 2: Divide your original number by your guess. Step 3: Find the average of these numbers. Step 4: Use this average as your next guess.

Task – Find square root if a number using Babylonian Method

Scala code

```
package Assignment13_3

object SquarerootBabylonian
{
defsquareRoot(n:Int): Int=
  {
varx = n;
vary = 1;
vare = 0.000001;
while (x-y>e)
      {
        x=(x+y)/2;
        y=n/x;
      }
return x;
  }
defmain(args: Array[String]): Unit =
  {
println("Enter a number: ")
varnum:Int = scala.io.StdIn.readLine().toInt
println(squareRoot(num));
  }
}
```

Screen Shot

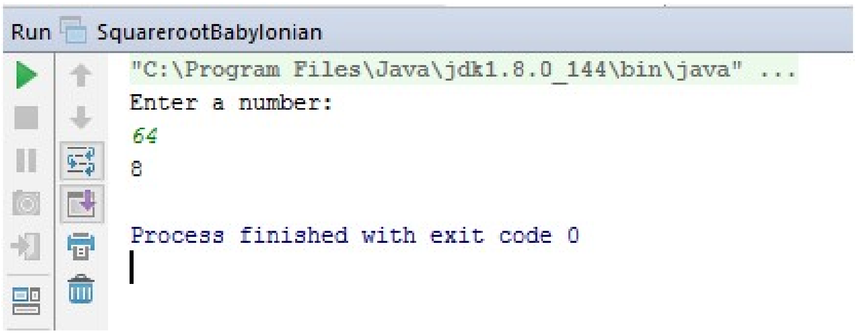

Assignment13_3 > src > main > scala > Assignment13_3 > SquarerootBabylonian.scala
Project
Assignment13_3 [assignment13_3] D:\Abu\Technical\H
.idea
project [assignment13_3-build] sources root
src
main
scala
Assignment13_3
SquarerootBabylonian
test
target
scala
scala-2.12
streams
.history
build.sbt
External Libraries
SquarerootBabylonian.scala
SquarerootBabylonian > squareRoot(n: Int)
Run SquarerootBabylonian

```
package Assignment13_3

object SquarerootBabylonian
{
  def squareRoot(n:Int): Int=
  {
    var x = n;
    var y = 1;
    var e = 0.000001;
    while (x-y>e)
      {
        x=(x+y)/2;
        y=n/x;
      }
    return x;
  }
def main(args: Array[String]): Unit =
  {
    println("Enter a number: ")
    var num:Int = scala.io.StdIn.readLine().toInt
    println(squareRoot(num));
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
9
3

Process finished with exit code 0
```

Output

If we enter a number 64, the square root of that value is 8



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
Enter a number:
64
8

Process finished with exit code 0
```