GROOVY ASSIGNMENT

1. Write a groovy program to print a line using round brackets to console.

GroovyConsole

File Edit View History Script Help

```
1 package com.app
2 class Demo {
3     static void main(args) {
4         println ("Welcome to Javatpoint tutorial on Groovy...")
5     }
6 }
7
```

```
groovy> package com.app
groovy> class Demo {
groovy>     static void main(args) {
groovy>         println ("Welcome to Javatpoint tutorial on Groovy...")
groovy>     }
groovy> }

Welcome to Javatpoint tutorial on Groovy...
```

2. Write a groovy program to print a line without using round brackets.

GroovyConsole

File Edit View History Script Help

```
1 package com.app
2 class Demo {
3     static void main(args) {
4         println "Welcome to Javatpoint tutorial on Groovy..."
5     }
6 }
7
```

```
groovy> package com.app
groovy> class Demo {
groovy>     static void main(args) {
groovy>         println "Welcome to Javatpoint tutorial on Groovy..."
groovy>     }
groovy> }

Welcome to Javatpoint tutorial on Groovy...
```

3. In groovy, double quotes as well as single quotes can be used in a string.

GroovyConsole

File Edit View History Script Help

```
1 package com.app
2 class Demo {
3     static void main(args) {
4         println 'Welcome to Javatpoint tutorial on Groovy...'
5     }
6 }
7
```

```
groovy> package com.app
groovy> class Demo {
groovy>     static void main(args) {
groovy>         println 'Welcome to Javatpoint tutorial on Groovy...'
groovy>     }
groovy> }

Welcome to Javatpoint tutorial on Groovy...
```

4. In Groovy, we can have a single line comment as well as a multi-line comment just like in java.

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class Demo {
3  //this is a single line comment
4  /*
5  *
6  *this is a
7  *multi-line comment
8  */
9  static void main(args) {
10 println "Welcome to Javatpoint tutorial on Groovy..."
11     }
12 }
13
```

```
groovy> package com.app
groovy> class Demo {
groovy> //this is a single line comment
groovy> /*
groovy> *
groovy> *this is a
groovy> *multi-line comment
groovy> */
groovy> static void main(args) {
groovy> println "Welcome to Javatpoint tutorial on Groovy..."
groovy>     }
groovy> }

Welcome to Javatpoint tutorial on Groovy...
```

5. In Groovy, it is not necessary to have a class or the main function

GroovyConsole

File Edit View History Script Help

```
1 package com.app
2 println "Welcome to Javatpoint tutorial on Groovy..."
3
4
```

```
groovy> package com.app
groovy> println "Welcome to Javatpoint tutorial on Groovy..."

Welcome to Javatpoint tutorial on Groovy...
```

6. Write a Groovy program that performs basic arithmetic operations on two integers and prints the results.

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample1 {
3      static void main(args) {
4          int a=10
5          int b=5
6          int c
7          c=a+b
8          println "Addition =" +c
9          c=a-b
10         println "Subtraction =" +c
11         c=a*b
12         println "Multiplication =" +c
13         c=a/b
14         println "Division =" +c
15         c=a%b
16         println "Remainder =" +c
17         c=a**b
18         println "Power =" +c
19     }
20 }
21
```

```
groovy>     }
groovy> }

Addition =15
Subtraction =5
Multiplication =50
Division =2
Remainder =0
Power =100000
```

7. Groovy program that takes two numbers as user input and performs basic arithmetic operations.

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample1 {
3  static void main(args) {
4              int a=10.3
5              int b=5
6              int c
7              c=a.plus(b)
8              println "plus =" +c
9              c=a.minus(b)
10             println "minus =" +c
11             c=a.intdiv(b)
12             println "intdiv =" +c
13             c=a.power(b)
14             println "Power =" +c
15         }
16 }
17
```

```
groovy>             c=a.plus(b)
groovy>             println "plus =" +c
groovy>             c=a.minus(b)
groovy>             println "minus =" +c
groovy>             c=a.intdiv(b)
groovy>             println "intdiv =" +c
groovy>             c=a.power(b)
groovy>             println "Power =" +c
groovy>         }
groovy> }

plus =15
minus =5
intdiv =2
Power =100000
```

8. Groovy program that takes user input and checks whether a number is even or odd

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample1 {
3  static void main(args) {
4              int a=10
5              int c
6              c=+a
7              println "Unary plus =" +c
8              c=-a
9              println "Unary minus =" +c
10         }
11 }
12
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample1 {
groovy> static void main(args) {
groovy>         int a=10
groovy>         int c
groovy>         c=+a
groovy>         println "Unary plus =" +c
groovy>         c=-a
groovy>         println "Unary minus =" +c
groovy>     }
groovy> }

Unary plus =10
Unary minus =-10
```

9. Groovy program that demonstrates post-increment, pre-increment, post-decrement, and pre-decrement operations

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample1 {
3  static void main(args) {
4          int a=10
5          int c
6          c=a++
7          println "Post increment =" +c
8          println "Value of a after Post increment =" +a
9          c=++a
10         println "Pre Increment =" +c
11         println "Value of a after Pre increment =" +a
12         int b=10
13         c=b--
14         println "Post decrement =" +c
15         println "Value of a after Post decrement =" +b
16         c=--b
17         println "Pre decrement =" +c
18         println "Value of a after Pre decrement =" +b
19     }
20 }
21
```

```
groovy>         println "Pre decrement =" +c
groovy>         println "Value of a after Pre decrement =" +b
groovy>     }
groovy> }

Post increment =10
Value of a after Post increment =11
Pre Increment =12
Value of a after Pre increment =12
Post decrement =10
Value of a after Post decrement =9
Pre decrement =8
Value of a after Pre decrement =8
```

10. Groovy Program: Compound Assignment Operators

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample1 {
3  static void main(args) {
4          int a=10
5          a+=3
6          println "a+3----> " +a
7          a-=3
8          println "a-=3----> " +a
9          a*=3
10         println "a*=3----> " +a
11         a/=3
12         println "a/=3----> " +a
13         a%=3
14         println "a%=3----> " +a
15         a**=3
16         println "a**=3----> " +a
17     }
18 }
```

```
a+3----> 13
a-=3----> 10
a*=3----> 30
a/=3----> 10
a%=3----> 1
a**=3----> 1
```

11. Groovy program that demonstrates the use of relational operators

GroovyConsole

File Edit View History Script Help

```
package com.app
class GroovyOperatorsExample1 {
static void main(args) {
        int a=10
        int b=12
        boolean c
        println "a=10"
        println "b=12"
        c=a==b
        println "Relational Operator equals [c=a==b]----> " +c
        c=a!=b
        println "Relational Operator different [c=a==b]----> " +c
        c=a<b
        println "Relational Operator less than [c=a<b]----> " +c
        c=a<=b
        println "Relational Operator less than equal to [c=a<=b]----> " +c
        c=a>b
        println "Relational Operator greater than [c=a>b]----> " +c
        c=a>=b
        println "Relational Operator greater than [c=a>=b]----> " +c

    }
}
```

```
a=10
b=12
Relational Operator equals [c=a==b]----> false
Relational Operator different [c=a==b]----> true
Relational Operator less than [c=a<b]----> true
Relational Operator less than equal to [c=a<=b]----> true
Relational Operator greater than [c=a>b]----> false
Relational Operator greater than [c=a>=b]----> false
```

12. Groovy program demonstrating the use of logical operators

GroovyConsole

File Edit View History Script Help

```
package com.app
class GroovyOperatorsExample {
static void main(args) {
        boolean c
        c=true && true
        println "Logical AND operator =" +c
        c=true || false
        println "Logical OR operator =" +c
        c=!false
        println "Logical NOT operator =" +c
    }
}
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample {
groovy> static void main(args) {
groovy>         boolean c
groovy>         c=true && true
groovy>         println "Logical AND operator =" +c
groovy>         c=true || false
groovy>         println "Logical OR operator =" +c
groovy>         c=!false
groovy>         println "Logical NOT operator =" +c
groovy>     }
groovy> }

Logical AND operator =true
Logical OR operator =true
Logical NOT operator =true
```

13. Groovy Program: Logical Operators with Multiple Conditions

GroovyConsole

File Edit View History Script Help

```
package com.app
class GroovyOperatorsExample {
static void main(args) {
        boolean c
        c=(!false && false)
        print c
    }
}
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample {
groovy> static void main(args) {
groovy>         boolean c
groovy>         c=(!false && false)
groovy>         print c
groovy>     }
groovy> }

false
```

14. Groovy Program: Logical Operators with Precedence

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample {
3  static void main(args) {
4          boolean c
5          c=true||true && false
6          print c
7      }
8  }
9
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample {
groovy> static void main(args) {
groovy>         boolean c
groovy>         c=true||true && false
groovy>         print c
groovy>     }
groovy> }

true
```

15. Groovy program that demonstrates bitwise operators (AND, OR, XOR, NOT) with binary numbers .

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2  class GroovyOperatorsExample {
3  static void main(args) {
4          int a=0b00101111
5          println "a=0b00101111-----> " +a
6          int b=0b000010101
7          println "b=0b000010101----> " +b
8          println "(a&a)----> " +(a&a)
9          println "(a&b)----> " +(a&b)
10         println "(a|a)----> " +(a|a)
11         println "(a|a)----> " +(a|b)
12         int c=0b111111111
13         println "c=0b111111111"
14         println "((a^a)&c)----> " +((a^a)&c)
15         println "((a^b)&c)----> " +((a^b)&c)
16         println "((~a)&c)----> " +((~a)&c)
17     }
18 }
```

```
a=0b00101111-----> 47
b=0b000010101----> 21
(a&a)----> 47
(a&b)----> 5
(a|a)----> 47
(a|a)----> 63
c=0b111111111
((a^a)&c)----> 0
((a^b)&c)----> 58
((~a)&c)----> 208
```

16. Groovy program that demonstrates converting integers to binary strings and vice versa.

```
package com.app
class GroovyOperatorsExample {
static void main(args) {
        int a=23
        int b=43
        println "Converting Integer to Binary a=23----> " +Integer.toBinaryString(a)
        println "Converting Integer to Binary a=43----> " +Integer.toBinaryString(b)
        println "Converting binary to integer 10111----> a= " +Integer.parseInt("10111",2)
        println "Converting binary to integer 101011-----> b= " +Integer.parseInt("10111",2)
    }
}
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample {
groovy> static void main(args) {
groovy>         int a=23
groovy>         int b=43
groovy>         println "Converting Integer to Binary a=23----> " +Integer.toBinaryString(a)
groovy>         println "Converting Integer to Binary a=43----> " +Integer.toBinaryString(b)
groovy>         println "Converting binary to integer 10111----> a= " +Integer.parseInt("10111",2)
groovy>         println "Converting binary to integer 101011-----> b= " +Integer.parseInt("10111",2)
groovy>     }
groovy> }

Converting Integer to Binary a=23----> 10111
Converting Integer to Binary a=43----> 101011
Converting binary to integer 10111----> a= 23
Converting binary to integer 101011-----> b= 23
```

17. Groovy program that demonstrates the use of the logical NOT operator (!) with different types of values.

```
package com.app

class GroovyOperatorsExample12 {
    static void main(String[] args) {
        println "(!true) ----> " + (!true)
        println "(!'javatpoint') ----> " + (!'javatpoint')
        println "!Null ----> " + (!'')
    }
}
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample12 {
groovy>     static void main(String[] args) {
groovy>         println "(!true) ----> " + (!true)
groovy>         println "(!'javatpoint') ----> " + (!'javatpoint')
groovy>         println "!Null ----> " + (!'')
groovy>     }
groovy> }

(!true) ----> false
(!'javatpoint') ----> false
!Null ----> true
```

18. Groovy program that demonstrates the use of the ternary operator for checking if a string is non-null and has a length greater than zero.

```
package com.app
class GroovyOperatorsExample13 {
    static void main(String[] args) {
        String Answer
        String s = 'javatpoint'
        Answer = (s != null && s.length() > 0) ? 'Found' : 'Not found'
        println Answer
    }
}
```

```
groovy> package com.app
groovy> class GroovyOperatorsExample13 {
groovy>     static void main(String[] args) {
groovy>         String Answer
groovy>         String s = 'javatpoint'
groovy>         Answer = (s != null && s.length() > 0) ? 'Found' : 'Not
found'
groovy>         println Answer
groovy>     }
groovy> }

Found
```

19. Groovy program that demonstrates two forms of conditional expressions (ternary operator and Elvis operator)

GroovyConsole

File Edit View History Script Help

```
package com.app
class GroovyOperatorsExample1 {
    static void main(String[] args) {
        String s = 'javatpoint'
        // Using ternary operator and printing directly
        def Answer = s ? 'Found' : 'Not Found'
        println Answer
        // Using Elvis operator and printing directly
        Answer = s ?: 'Found'
        println Answer
    }
}
```

```
javatpoint
groovy> package com.app
groovy> class GroovyOperatorsExample1 {
groovy>     static void main(String[] args) {
groovy>         String s = 'javatpoint'
groovy>         // Using ternary operator and printing directly
groovy>         def Answer = s ? 'Found' : 'Not Found'
groovy>         println Answer
groovy>         // Using Elvis operator and printing directly
groovy>         Answer = s ?: 'Found'
groovy>         println Answer
groovy>     }
groovy> }

Found
javatpoint
```