Basic syntax in groovy

Groovy Script to print the basic script.

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

Output:

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

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

```
1 package com.app
2
3 class Demo {
4     static void main(args) {
5         println("Welcome to Javatpoint tutorial on Groovy... ")
6     }
7 }
8
```

Output:

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

```
1 package com.app
2
3 class Demo {
4     static void main(args) {
5         println 'Welcome to Javatpoint tutorial on Groovy...'
6     }
7 }
8
```

Output:

```
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Jun 8, 2019, 6:07:45 PM)
Welcome to Javatpoint tutorial on Groovy...
```

Script:

```
1 package com.app
2
3 class Demo {
4     static void main(args) {
5         println 'Welcome to Javatpoint tutorial on Groovy...'
6     }
7 }
8
```

Output:

```
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...
```

Arithmetic operator

Example 1:

```
1  package com.app
2
3  class GroovyOperatorsExample1 {
4      static void main(args) {
5          int a = 10
6          int b = 5
7          int c
8
9          c = a + b
10         println "Addition = " + c
11
12         c = a - b
13         println "Subtraction = " + c
14
15         c = a * b
16         println "Multiplication = " + c
17
18         c = a / b
19         println "Division = " + c
20
21         c = a % b
22         println "Remainder  = " + c
23
24         c = a ** b
25         println "Power = " + c
26     }
27 }
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample1 {
groovy>     static void main(args) {
groovy>         int a = 10
groovy>         int b = 5
groovy>         int c
groovy>
groovy>         c = a + b
groovy>         println "Addition = " + c
groovy>
groovy>         c = a - b
groovy>         println "Subtraction = " + c
groovy>
groovy>         c = a * b
groovy>         println "Multiplication = " + c
groovy>
groovy>         c = a / b
groovy>         println "Division = " + c
groovy>
groovy>         c = a % b
groovy>         println "Remainder  = " + c
groovy>
groovy>         c = a ** b
groovy>         println "Power = " + c
groovy>     }
groovy> }

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

Example 2:

```
1  package com.app
2
3  class GroovyOperatorsExample2 {
4      static void main(args) {
5
6          double a = 10.3   // Use double for floating-point numbers
7          int b = 5
8          int c
9
10         c = a.plus(b)      // plus method for addition
11         println "plus = " + c
12
13         c = a.minus(b)     // minus method for subtraction
14         println "minus = " + c
15
16         // Use regular division for floating-point numbers
17         def result = a / b  // This will perform floating-point divis:
18         println "Division = " + result
19
20         // You can cast to int if you need integer division result
21         c = (a / b) as int  // Cast result to int for integer divisio
22         println "Integer Division = " + c
23
24         c = a.power(b)      // power method for exponentiation
25         println "Power = " + c
26     }
27 }
28
29
```

Output:

```
        at com.app.GroovyOperatorsExample2.main(ConsoleScript6:16)
groovy> package com.app
groovy> class GroovyOperatorsExample2 {
groovy>     static void main(args) {
groovy>         double a = 10.3   // Use double for floating-point numbers
groovy>         int b = 5
groovy>         int c
groovy>
groovy>         c = a.plus(b)      // plus method for addition
groovy>         println "plus = " + c
groovy>
groovy>         c = a.minus(b)     // minus method for subtraction
groovy>         println "minus = " + c
groovy>
groovy>         // Use regular division for floating-point numbers
groovy>         def result = a / b  // This will perform floating-point division
groovy>         println "Division = " + result
groovy>
groovy>         // You can cast to int if you need integer division result
groovy>         c = (a / b) as int  // Cast result to int for integer division
groovy>         println "Integer Division = " + c
groovy>
groovy>         c = a.power(b)      // power method for exponentiation
groovy>         println "Power = " + c
groovy>     }
groovy> }

plus = 15
minus = 5
Division = 2.06
Integer Division = 2
Power = 115927
Execution complete. Result was null. Elapsed time: 46ms.                28:1
```

Unary operators

Example 3:

```
1  package com.app
2
3  class GroovyOperatorsExample3 {
4      static void main(args) {
5          int a = 10
6          int c
7
8          c = +a  // Unary plus: this doesn't change the value of a
9          println "Unary plus = " + c
10
11         c = -a  // Unary minus: negates the value of a
12         println "Unary minus = " + c
13     }
14 }
15
16
17
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample3 {
groovy>     static void main(args) {
groovy>         int a = 10
groovy>         int c
groovy>
groovy>         c = +a  // Unary plus: this doesn't change the value of a
groovy>         println "Unary plus = " + c
groovy>
groovy>         c = -a  // Unary minus: negates the value of a
groovy>         println "Unary minus = " + c
groovy>     }
groovy> }

Unary plus = 10
Unary minus = -10
Execution complete. Result was null. Elapsed time: 26ms.                    17:1
```

Example 4:

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2
3  class GroovyOperatorsExample4 {
4      static void main(args) {
5          int a = 10
6          int c
7
8          // Post-increment: Returns the value, then increments
9          c = a++
10         println "Post Increment = " + c
11         println "Value of a after Post Increment = " + a
12
13         // Pre-increment: Increments the value, then returns it
14         c = ++a
15         println "Pre Increment = " + c
16         println "Value of a after Pre Increment = " + a
17
18         int b = 10
19
20         // Post-decrement: Returns the value, then decrements
21         c = b--
22         println "Post Decrement = " + c
23         println "Value of b after Post Decrement = " + b
24
25         // Pre-decrement: Decrements the value, then returns it
26         c = --b
27         println "Pre Decrement = " + c
28         println "Value of b after Pre Decrement = " + b
29     }
30 }
31
32
```

Output:

```
groovy>
groovy>         // Post-increment: Returns the value, then increments
groovy>         c = a++
groovy>         println "Post Increment = " + c
groovy>         println "Value of a after Post Increment = " + a
groovy>
groovy>         // Pre-increment: Increments the value, then returns it
groovy>         c = ++a
groovy>         println "Pre Increment = " + c
groovy>         println "Value of a after Pre Increment = " + a
groovy>
groovy>         int b = 10
groovy>
groovy>         // Post-decrement: Returns the value, then decrements
groovy>         c = b--
groovy>         println "Post Decrement = " + c
groovy>         println "Value of b after Post Decrement = " + b
groovy>
groovy>         // Pre-decrement: Decrements the value, then returns it
groovy>         c = --b
groovy>         println "Pre Decrement = " + c
groovy>         println "Value of b 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 b after Post Decrement = 9
Pre Decrement = 8
Value of b after Pre Decrement = 8
```

Execution complete. Result was null. Elapsed time: 47ms. 31:1

Assignment arithmetic operators

Example 5:

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2
3  class GroovyOperatorsExample5 {
4      static void main(args) {
5          int a = 10
6
7          a += 3   // a = a + 3
8          println "a += 3 ------> " + a
9
10         a -= 3   // a = a - 3
11         println "a -= 3 ------> " + a
12
13         a *= 3   // a = a * 3
14         println "a *= 3 ------> " + a
15
16         a /= 3   // a = a / 3
17         println "a /= 3 ------> " + a
18
19         a %= 3   // a = a % 3 (remainder when divided by 3)
20         println "a %= 3 ------> " + a
21
22         a **= 3  // a = a raised to the power of 3
23         println "a **= 3 ------> " + a
24     }
25 }
26
27
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample5 {
groovy>     static void main(args) {
groovy>         int a = 10
groovy>
groovy>         a += 3   // a = a + 3
groovy>         println "a += 3 ------> " + a
groovy>
groovy>         a -= 3   // a = a - 3
groovy>         println "a -= 3 ------> " + a
groovy>
groovy>         a *= 3   // a = a * 3
groovy>         println "a *= 3 ------> " + a
groovy>
groovy>         a /= 3   // a = a / 3
groovy>         println "a /= 3 ------> " + a
groovy>
groovy>         a %= 3   // a = a % 3 (remainder when divided by 3)
groovy>         println "a %= 3 ------> " + a
groovy>
groovy>         a **= 3  // a = a raised to the power of 3
groovy>         println "a **= 3 ------> " + a
groovy>     }
groovy> }

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

Execution complete. Result was null. Elapsed time: 24ms. 27:1

Relational operators

Example 6:

```
1  package com.app
2
3  class GroovyOperatorsExample6 {
4      static void main(args) {
5          int a = 10
6          int b = 12
7          boolean c
8
9          println "a = 10"
10         println "b = 12"
11
12         c = a == b  // Equals operator
13         println "Relational Operator equals [c = a == b] ----> " + c
14
15         c = a != b  // Not equals operator
16         println "Relational Operator different [c = a != b] ----> " + c
17
18         c = a < b   // Less than operator
19         println "Relational Operator less than [c = a < b] ----> " + c
20
21         c = a <= b  // Less than or equal to operator
22         println "Relational Operator less than or equal to [c = a <= b] ----> " + c
23
24         c = a > b   // Greater than operator
25         println "Relational Operator greater than [c = a > b] ----> " + c
26
27         c = a >= b  // Greater than or equal to operator
28         println "Relational Operator greater than or equal to [c = a >= b] ----> " + c
29     }
30 }
31
32
```

Output:

```
groovy>         boolean c
groovy>
groovy>         println "a = 10"
groovy>         println "b = 12"
groovy>
groovy>         c = a == b  // Equals operator
groovy>         println "Relational Operator equals [c = a == b] ----> " + c
groovy>
groovy>         c = a != b  // Not equals operator
groovy>         println "Relational Operator different [c = a != b] ----> " + c
groovy>
groovy>         c = a < b   // Less than operator
groovy>         println "Relational Operator less than [c = a < b] ----> " + c
groovy>
groovy>         c = a <= b  // Less than or equal to operator
groovy>         println "Relational Operator less than or equal to [c = a <= b] ----> " + c
groovy>
groovy>         c = a > b   // Greater than operator
groovy>         println "Relational Operator greater than [c = a > b] ----> " + c
groovy>
groovy>         c = a >= b  // Greater than or equal to operator
groovy>         println "Relational Operator greater than or equal to [c = a >= b] ----> " + c
groovy>     }
groovy> }

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 or equal to [c = a <= b] ----> true
Relational Operator greater than [c = a > b] ----> false
Relational Operator greater than or equal to [c = a >= b] ----> false
```

Logical operators

Example 7:

```
1  package com.app
2
3  class GroovyOperatorsExample7 {
4      static void main(args) {
5          boolean c
6
7          c = true && true   // Logical AND: both must be true
8          println "Logical AND operator = " + c
9
10         c = true || false  // Logical OR: at least one must be true
11         println "Logical OR operator = " + c
12
13         c = !false         // Logical NOT: negates the value
14         println "Logical NOT operator = " + c
15     }
16 }
17
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample7 {
groovy>     static void main(args) {
groovy>         boolean c
groovy>
groovy>         c = true && true   // Logical AND: both must be true
groovy>         println "Logical AND operator = " + c
groovy>
groovy>         c = true || false  // Logical OR: at least one must be true
groovy>         println "Logical OR operator = " + c
groovy>
groovy>         c = !false         // Logical NOT: negates the value
groovy>         println "Logical NOT operator = " + c
groovy>     }
groovy> }

Logical AND operator = true
Logical OR operator = true
Logical NOT operator = true
Execution complete. Result was null. Elapsed time: 20ms.                    17:1
```

Example 8:

```
package com.app

class GroovyOperatorsExample8 {
    static void main(args) {
        boolean c
        c = (!false && false)  // NOT false AND false
        println c
    }
}
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample8 {
groovy>     static void main(args) {
groovy>         boolean c
groovy>         c = (!false && false)  // NOT false AND false
groovy>         println c
groovy>     }
groovy> }

false
```

Execution complete. Result was null. Elapsed time: 19ms.

Example 9:

```
package com.app

class GroovyOperatorsExample1 {
    static void main(args) {
        boolean c
        c = true || true && false  // OR and AND operators
        println c
    }
}
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample1 {
groovy>     static void main(args) {
groovy>         boolean c
groovy>         c = true || true && false  // OR and AND operators
groovy>         println c
groovy>     }
groovy> }

true
```

Execution complete. Result was null. Elapsed time: 14ms.

Bitwise operators

Example 10:

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2
3  class GroovyOperatorsExample10 {
4      static void main(args) {
5          int a = 0b00101111   // Binary representation
6          println "a = 0b00101111 ----> " + a
7
8          int b = 0b000010101  // Binary representation
9          println "b = 0b000010101 ----> " + b
10
11         println "(a & a) ----> " + (a & a)  // Bitwise AND (AND with itself)
12         println "(a & b) ----> " + (a & b)  // Bitwise AND between a and b
13
14         println "(a | a) ----> " + (a | a)  // Bitwise OR (OR with itself)
15         println "(a | b) ----> " + (a | b)  // Bitwise OR between a and b
16
17         int c = 0b11111111  // Another binary value
18         println "c = 0b11111111"
19
20         println "((a ^ a) & c) ----> " + ((a ^ a) & c)  // Bitwise XOR with AND
21         println "((a ^ b) & c) ----> " + ((a ^ b) & c)  // Bitwise XOR between a and b with AND
22
23         println "((~a) & c) ----> " + ((~a) & c)  // Bitwise NOT a with AND
24     }
25 }
26
```

Output:

```
groovy>     static void main(args) {
groovy>         int a = 0b00101111   // Binary representation
groovy>         println "a = 0b00101111 ----> " + a
groovy>
groovy>         int b = 0b000010101  // Binary representation
groovy>         println "b = 0b000010101 ----> " + b
groovy>
groovy>         println "(a & a) ----> " + (a & a)  // Bitwise AND (AND with itself)
groovy>         println "(a & b) ----> " + (a & b)  // Bitwise AND between a and b
groovy>
groovy>         println "(a | a) ----> " + (a | a)  // Bitwise OR (OR with itself)
groovy>         println "(a | b) ----> " + (a | b)  // Bitwise OR between a and b
groovy>
groovy>         int c = 0b11111111  // Another binary value
groovy>         println "c = 0b11111111"
groovy>
groovy>         println "((a ^ a) & c) ----> " + ((a ^ a) & c)  // Bitwise XOR with AND
groovy>         println "((a ^ b) & c) ----> " + ((a ^ b) & c)  // Bitwise XOR between a and b with AND
groovy>
groovy>         println "((~a) & c) ----> " + ((~a) & c)  // Bitwise NOT a with AND
groovy>     }
groovy> }

a = 0b00101111 ----> 47
b = 0b000010101 ----> 21
(a & a) ----> 47
(a & b) ----> 5
(a | a) ----> 47
(a | b) ----> 63
c = 0b11111111
((a ^ a) & c) ----> 0
((a ^ b) & c) ----> 58
((~a) & c) ----> 208
```

Execution complete. Result was null. Elapsed time: 43ms. 26:1

Example 11:

GroovyConsole

File Edit View History Script Help

```
package com.app

class GroovyOperatorsExample11 {
    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 b = 43 ----> " + Integer.toBinaryString(b)

        println "Converting binary to integer 10111 ----> a = " + Integer.parseInt("10111", 2)
        println "Converting binary to integer 101011 ----> b = " + Integer.parseInt("101011", 2)
    }
}
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample11 {
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 b = 43 ----> " + Integer.toBinaryString(b)
groovy>
groovy>         println "Converting binary to integer 10111 ----> a = " + Integer.parseInt("10111", 2)
groovy>         println "Converting binary to integer 101011 ----> b = " + Integer.parseInt("101011", 2)
groovy>     }
groovy> }

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

Execution complete. Result was null. Elapsed time: 25ms. 15:1

Conditional operators

Example 12:

```
package com.app

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

Output:

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

(!true) ----> false
(!'javatpoint') ----> false
!Null ----> true
Execution complete. Result was null. Elapsed time: 18ms.
```

Example 13:

```
package com.app

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

Output:

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

Found
Execution complete. Result was null. Elapsed time: 29ms.
```

Example 14:

GroovyConsole

File Edit View History Script Help

```
1  package com.app
2
3  class GroovyOperatorsExample1 {
4      static void main(args) {
5          String Answer
6          String s = 'javatpoint'
7
8          // Using ternary operator
9          println Answer = s ? 'Found' : 'Not Found'
10
11         // Using Groovy's safe navigation operator
12         println Answer = s ?: 'Found'
13     }
14 }
```

Output:

```
groovy> package com.app
groovy> class GroovyOperatorsExample1 {
groovy>     static void main(args) {
groovy>         String s = 'javatpoint'
groovy>         // Using ternary operator
groovy>         String Answer = s ? 'Found' : 'Not Found'
groovy>         println Answer  // Prints "Found"
groovy>         // Using Groovy's safe navigation operator
groovy>         Answer = s ?: 'Found'
groovy>         println Answer  // Prints "javatpoint"
groovy>     }
groovy> }

Found
javatpoint
Execution complete. Result was null. Elapsed time: 20ms.
```