# Session 3
# Operators and Control Flow Statements

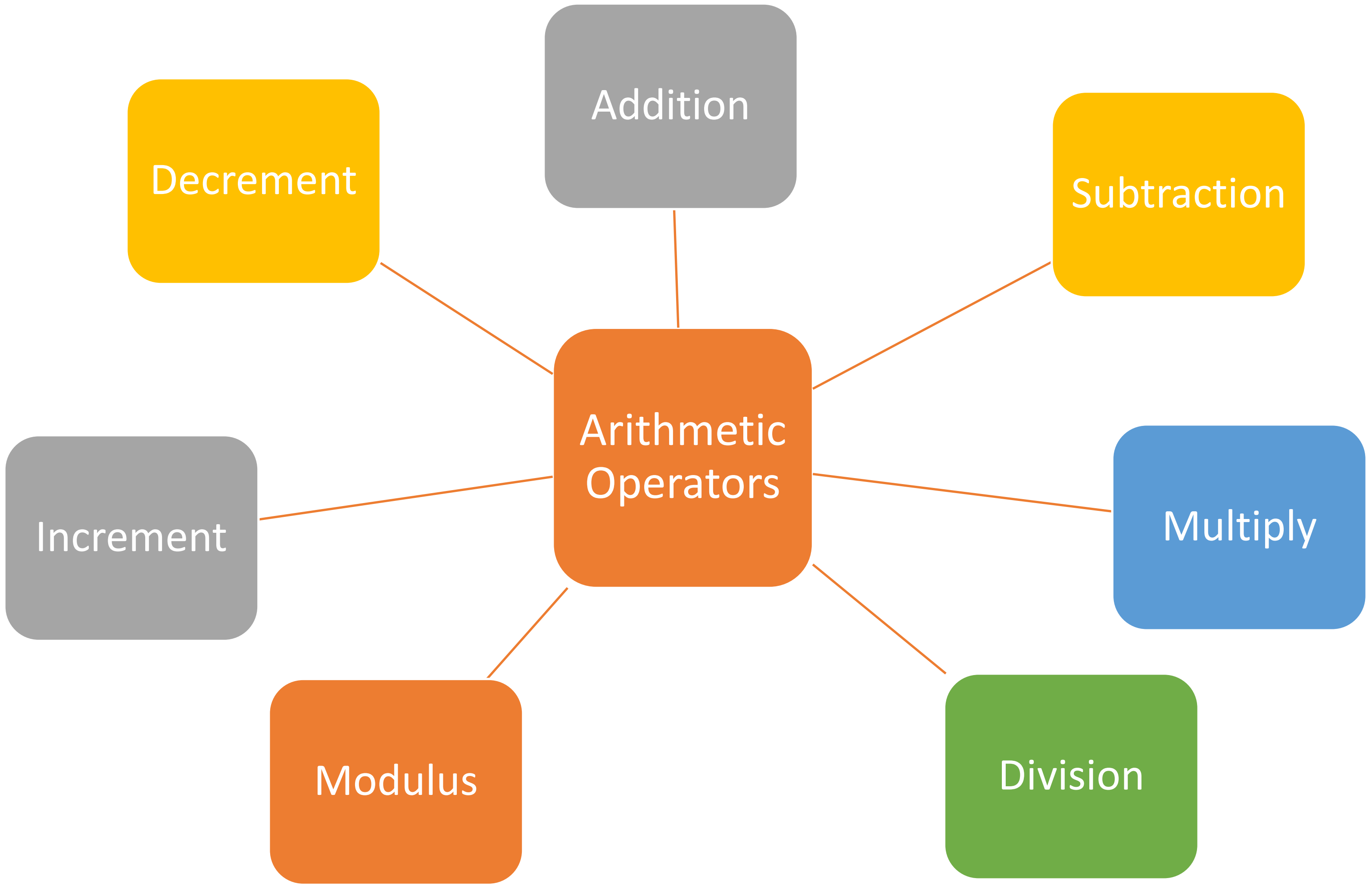
## Session Overview

- Define operators
- Explain the various operators
- Outline the usage of operators
- Define control flow statements in Dart
- Explain the various control flow statements in Dart
- Define decision-making statements in Dart
- Explain the various decision-making statements in Dart
- Define looping statements in Dart
- Explain the various looping statements in Dart
- Illustrate jump statements in Dart
- List the various jump statements in Dart

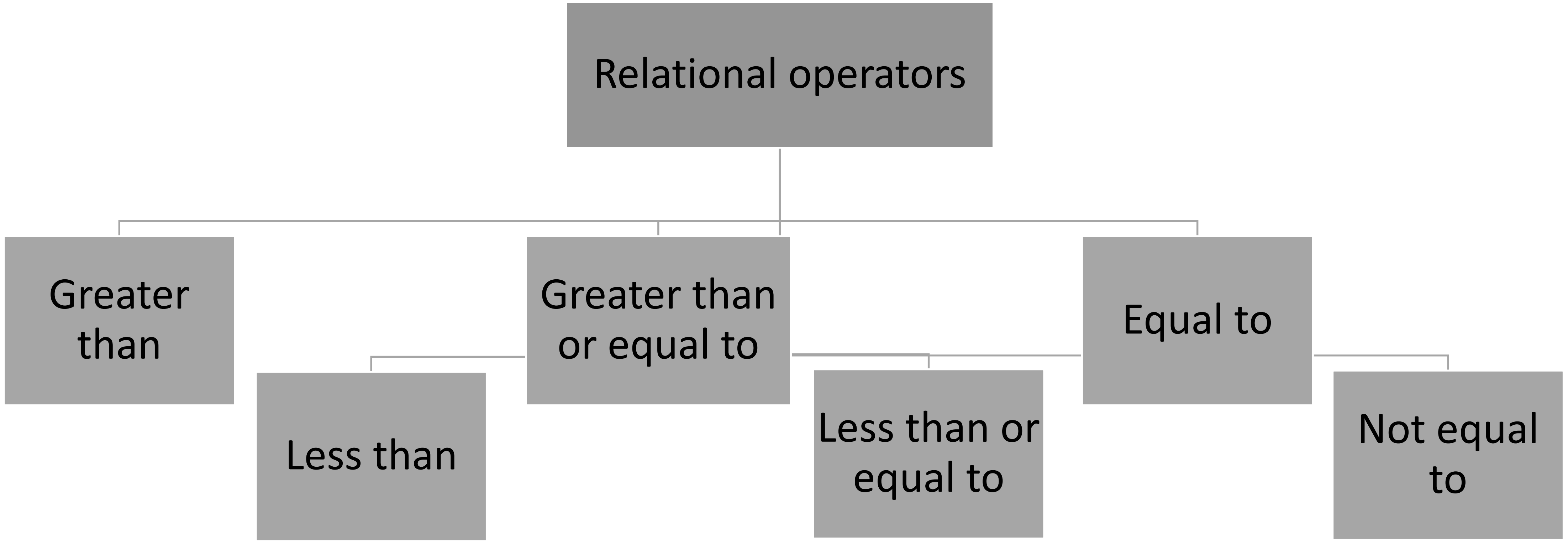
## Different Types of Operators

- Arithmetic operators
- Relational operators
- Assignment operators
- Logical operators
- Conditional operators

## Arithmetic Operators



# Relational Operators



# Assignment Operators

| | |
|---|---|
| = | Assignment operator |
| += | Add and assign |
| -= | Subtract and assign |
| *= | Multiply and assign |
| /= | Divide and assign |
| ~/= | Divide and assign (integer) |
| %= | Mod and assign |
| <<= | Left shift and assign |
| >>= | Right shift and assign |
| &= | Bitwise AND assign |
| ^= | Bitwise exclusive OR assign |
| \|= | Bitwise inclusive OR assign |

## Conditional Operators

Condition ? Expression1 : Expression2

Expression1 ?? Expression2

**Code Snippet 1**:

```
void main()
{
var n1= 10;
var n2 = 15;
var n3 = null;
var result = n1 >n2 ? n1 : n2;
print(result);
var result1 = n3 ?? n2;
print(result1);
}
```

## Logical Operators

AND → &&

OR → ||

Not → !

**Code Snippet 2**:

```
void main()
{
  int a=5;
  int b=7;

  bool c = a > 10 && b < 10;
  print(c);

  bool d = a > 10 || b < 10;
  print(d);

  bool e = !(a>10);
  print(e);
}
```

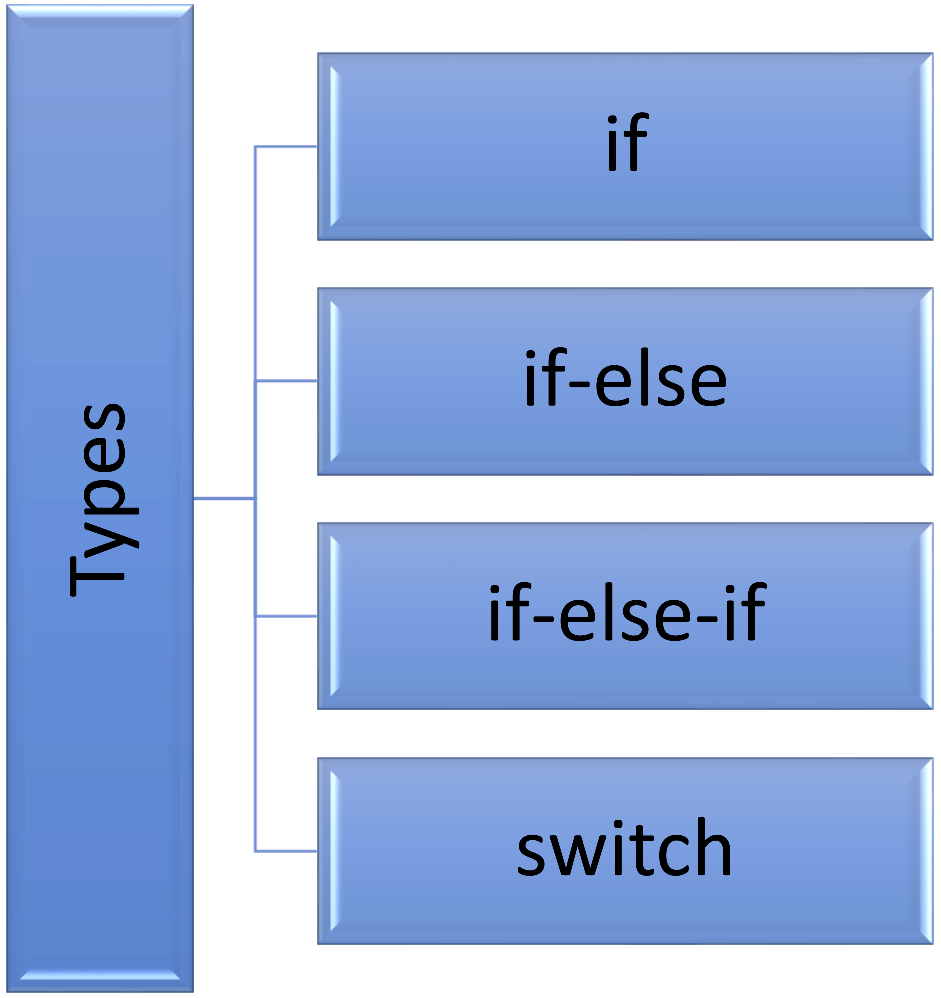
# Control Flow Statements

These statements change the flow of control as and when required.

- Decision-making statements
- Looping statements
- Jump statements

## Decision-making Statements [1-3]

These statements decide which block of statements has to be executed based on given conditions.



## Decision-making Statements [2-3]

**Code Snippet 3**:

```
int num=5;
if (num>0) {
      print('Number is positive');
}
```

**Code Snippet 4**:

```
int num=0;
if (num>0) {
      print('Number is positive');
}
else {
      print('Number is not positive');
}
```

# Decision-making Statements [3-3]

**Code Snippet 5:**

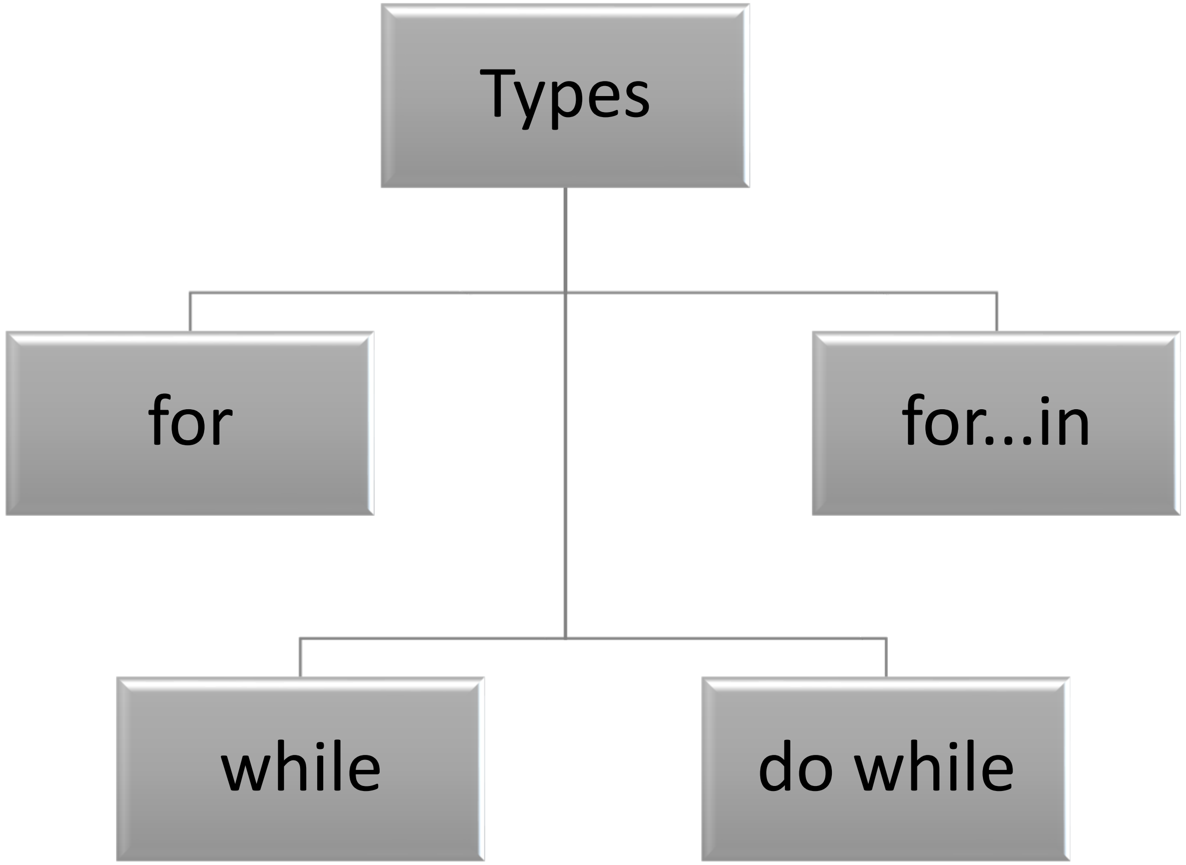
```
int num=-5;
if (num>0) {
        print('Number is positive');
}
else if(num == 0) {
        print('Number is zero');
}
else {
        print('Number is negative');
}
```

**Code Snippet 6:**

```
var grade='B';
switch(grade) {
case 'A': { print('Excellent'); }
break;
case 'B': { print('Good'); }
break;
case 'C': { print('Fair'); }
break;
case 'D': { print('Poor'); }
break;
default:  { print('Excellent'); }
break;
        }
}
```

## Looping Statements [1-3]

These statements are used for executing a set of statements multiple times.



## Looping Statements [2-3]

**Code Snippet 7:**

```
int num=1;
for(num; num<=10;num++){
      print(num);
}
```

**Code Snippet 8:**

```
var list = [10,20,30,40,50];
for(var i in list) {
      print(i);
}
```

## Looping Statements [3-3]

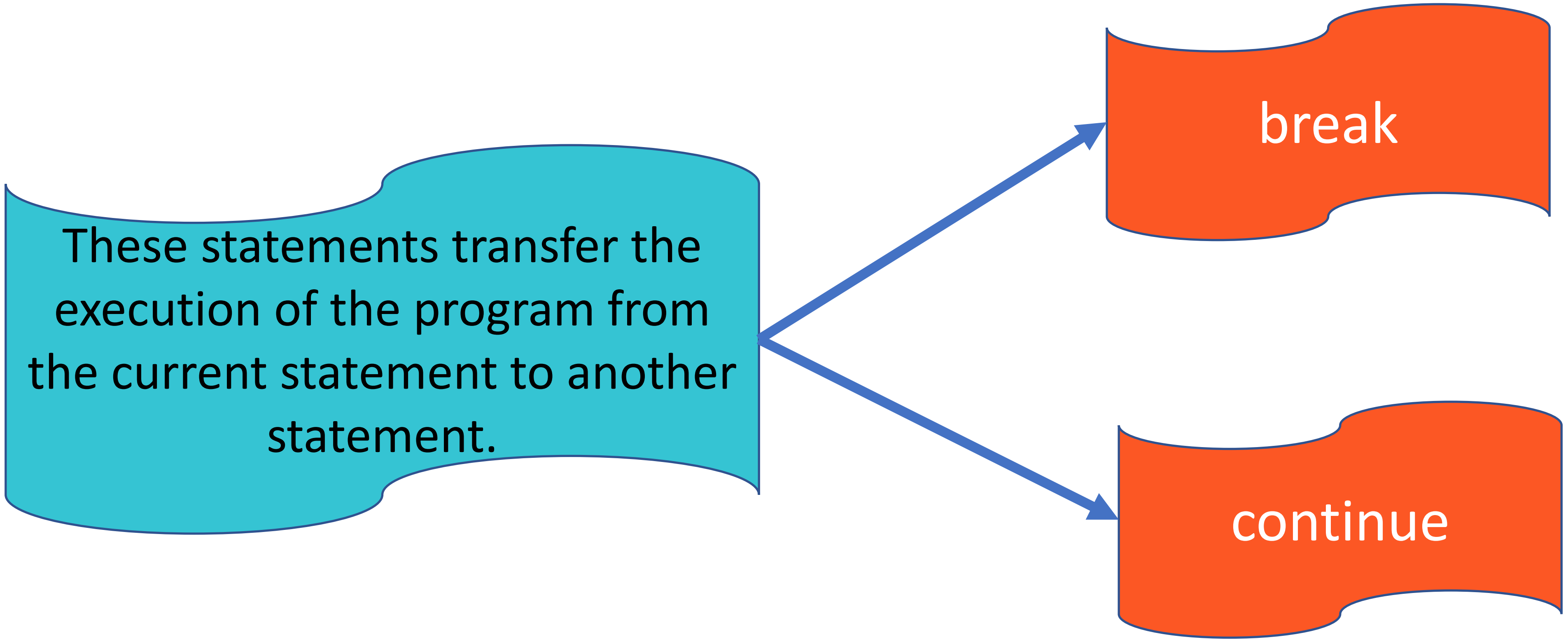
**Code Snippet 9**:

```
var a = 1;

var num = 5;

while(a<num) {

      print(a);

      a=a+1;
}
```

**Code Snippet 10**:

```
var n = 10;

do {

print(n);

n--;

} while(n>=0);
```

## Jump Statements [1-2]


These statements transfer the execution of the program from the current statement to another statement.
break
continue

## Jump Statements [2-2]

**Code Snippet 11**:

```
var count = 0;
print('Dart break statement');
while(count<=10) {
      count = count + 1;
      if(count == 5) {
         break;
       }
      print('Inside loop ${count}');
}
print('Out of while loop');
```

**Code Snippet 12**:

```
var num = 0;
print('Dart continue statement');
while( num < 10) {
      num = num + 1;
      if( num == 5) {
            print('5 is skipped');
            continue;
      }
      print('Number is ${num}');
}
print('Out of while loop');
```

## Summary

- Depending on the type of operator, actions can be performed on operands. For example, in an arithmetic operator, an addition operator (+) would add operands whereas a subtraction operator (-) would subtract one operand from the other.
- A control statement allows smooth flow of the program. In Dart, statements inside the code are generally executed sequentially, from top to bottom, within the order that they appear.
- One may not want to execute the code sequentially each time. Instead, one might want to skip a certain set of instructions or execute a code repeatedly.
- Depending on the scenario, a code can be written with a control flow statement by using decision-making statements, looping statements, or jump statements.
- All three statements or either of them can be used depending upon the scenario and also depending on the logic being built.