Snippet 1:

```
public class InfiniteForLoop {
public static void main(String[] args) {
for (int i = 0; i < 10; i--) {
System.out.println(i);
}
}
}
// Error to investigate: Why does this loop run infinitely? How should the loop control variable be adjusted?
```

Error:

Loop Will Run Infinite times.

Corrected Code: (Solution)

```
public class InfiniteForLoop {
public static void main(String[] args) {
for (int i = 0; i < 10; i++) {
System.out.println(i);
}
}
}
// Error to investigate: Why does this loop run infinitely? How should the loop control variable be adjusted?
```

Answer:

It will run infinitly because we are decrementing the for loop and values are going in _ve side so it will never meet terminating condition.

Snippet 2:

```
public class IncorrectWhileCondition {
public static void main(String[] args) {
int count = 5;
while (count = 0) {
System.out.println(count);
count--;
}
}
}
// Error to investigate: Why does the loop not execute as expected? What is the issue with the condition in the
`while` loop?
```

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment3>javac Main.java
Main.java:4: error: incompatible types: int cannot be converted to boolean
while (count = 0) {
             ^
1 error
```

Corrected Code: (Solution)

```
public class IncorrectWhileCondition {
public static void main(String[] args) {
int count = 5;
while (count == 0) {
System.out.println(count);
count--;
}
}
}
```

Answer:

In this snippet it was expected that we shoul pass a assignment operator in while condition however it was considering as an assignment operator in while condition, so by correcting it with comparison operator error resolved.

Snippet 3:

```
public class DoWhileIncorrectCondition {
public static void main(String[] args) {
int num = 0;
do {
System.out.println(num);
num++;
} while (num > 0);
}
}
// Error to investigate: Why does the loop only execute once? What is wrong with the loop condition in the `dowhile`
loop?
```

Error:

No Error

Corrected Code: (Solution)

```
public class DoWhileIncorrectCondition {
public static void main(String[] args) {
int num = 0;
do {
System.out.println(num);
num++;
} while (num >= 0);
}
}
```

Answer:

In this code num was initialized to 0 and in while condition it was **num>0,** condition was not getting fulfilled so it returned 0, as do while loop was used the body will be executed first and then condition will be checked, so after 1st iteration condition became false and control got out of the loop.

Snippet 4:

```
public class OffByOneErrorForLoop {
public static void main(String[] args) {
   for (int i = 1; i <= 10; i++)
      {
       System.out.println(i);
      }
// Expected: 10 iterations with numbers 1 to 10
// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9
}
}
```

// Error to investigate: What is the issue with the loop boundaries? How should the loop be adjusted to meet the
expected output?

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment2>javac Main.java

E:\CDAC\Logic Building\Assignments\Assignment2>java Main
Error: Main method not found in class Main, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

Corrected Code: (Solution)

```
public class OffByOneErrorForLoop {
public static void main(String[] args) {
   for (int i = 1; i <= 10; i++)
      {
       System.out.println(i);
      }
// Expected: 10 iterations with numbers 1 to 10
// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9
}
}
```

Answer:

To met the expectation we have to replave the <= operator to < so that loop will iterate till n-1.

Snippet 5:

```
public class WrongInitializationForLoop {
public static void main(String[] args) {
for (int i = 10; i >= 0; i++) {
System.out.println(i);
}
}
}
// Error to investigate: Why does this loop not print numbers in the expected order? What is the problem with the
initialization and update statements in the `for` loop?
```

Error:

InFinite Loop

Corrected Code: (Solution)

```
public class WrongInitializationForLoop {
public static void main(String[] args) {
for (int i = 10; i >= 0; i--) {
System.out.println(i);
}
}
}
```

Answer:

In above code there is infinite time loop is executing, to print values in order 0 to 10 we need to change the condition to **for(int i = 0; i<=10; i++)** and to print the values from 10 to 0 we need to **decrement i**.

Snippet 6:

```
public class MisplacedForLoopBody {
public static void main(String[] args) {
for (int i = 0; i < 5; i++)
System.out.println(i);
System.out.println("Done");
}
}
// Error to investigate: Why does "Done" print only once, outside the loop?
How should the loop body be enclosed to
include all statements within the loop?
```

Error/Output:

1

2

3

4

Done

Corrected Code: (Solution)

```
public class MisplacedForLoopBody {
public static void main(String[] args) {
for (int i = 0; i < 5; i++){
System.out.println(i);
System.out.println("Done");
}
}
}
```

Answer:

Actually, if we use for loop without {} curly braces then it considers only one line after loop in iteration so in this case it printed from 0 to 4 and “Done” got considered outside of the loop.

Snippet 7:

```
public class UninitializedWhileLoop {
public static void main(String[] args) {
int count;
while (count < 10) {
System.out.println(count);
count++;
}
}
}
// Error to investigate: Why does this code produce a compilation error? What needs to be done to initialize the loop
variable properly?
```

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment3>javac Main.java
Main.java:4: error: variable count might not have been initializ
ed
while (count < 10) {
       ^
1 error
```

Corrected Code: (Solution)

```
public class UninitializedWhileLoop {
public static void main(String[] args) {
int count = 0;
while (count < 10) {
System.out.println(count);
count++;
}
}
```

```
}
}
```

Answer:

initialize the count variable.

Snippet 8:

```
public class OffByOneDoWhileLoop {
public static void main(String[] args) {
int num = 1;
do {
System.out.println(num);
num--;
} while (num > 0);
}
}
// Error to investigate: Why does this loop print unexpected numbers? What
adjustments are needed to print the
numbers from 1 to 5?
```

Error:

Corrected Code: (Solution)

```
public class OffByOneDoWhileLoop {
public static void main(String[] args) {
int num = 1;
do {
System.out.println(num);
num++;   //num--
} while (num <= 5);num >0
}
}
```

Answer:

Snippet 9:

```
public class InfiniteForLoopUpdate {
public static void main(String[] args) {
for (int i = 0; i < 5; i += 2) {
System.out.println(i);
}
}
}
// Error to investigate: Why does the loop print unexpected results or run
infinitely? How should the loop update
expression be corrected?
```

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment3>java Main
0
2
4
```

Corrected Code: (Solution)

No Error in Code it is not printing unexpected numbers or running infinitely

Answer:

String[] args is needed to pass a Command line argument to a main method.

Snippet 10:

```
public class IncorrectWhileLoopControl {
public static void main(String[] args) {
int num = 10;
while (num = 10) {
```

```
System.out.println(num);
num--;
}
}
}
```

// Error to investigate: Why does the loop execute indefinitely? What is wrong with the loop condition?

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment3>javac Main.java
Main.java:4: error: incompatible types: int cannot be converted
to boolean
while (num = 10) {
           ^
1 error
```

Corrected Code: (Solution)

```
public class IncorrectWhileLoopControl {
public static void main(String[] args) {
int num = 10;
while (num == 10) {
System.out.println(num);
num--;
}
}
}
```

Answer:

In this case instead of assignment operator comparison operator was expected.

Snippet 11:

```
public class IncorrectLoopUpdate {
public static void main(String[] args) {
int i = 0;
while (i < 5) {
System.out.println(i);
i += 2; // Error: This may cause unexpected results in output
}
}
}
// Error to investigate: What will be the output of this loop? How should the
loop variable be updated to achieve the
desired result?
```

Error:

```
E:\CDAC\Logic Building\Assignments\Assignment3>java Main
0
2
4
```

Corrected Code: (Solution)

Answer:

It is giving correct output,, because I in incrementing with 2 numbers.

Snippet 12:

```
public class LoopVariableScope {
public static void main(String[] args) {
for (int i = 0; i < 5; i++) {
int x = i * 2;
}
System.out.println(x); // Error: 'x' is not accessible here
}
```

```
}
// Error to investigate: Why does the variable 'x' cause a compilation error?
How does scope
```

Error:

Corrected Code: (Solution)

```
public class LoopVariableScope {
public static void main(String[] args) {
int x;
for (int i = 0; i < 5; i++) {
x = i * 2;
}
System.out.println(x); // defined x outside loop
}
}
```

Answer:

As x was declared in for loop it is going to be local for loop only, we can't access that outside the scope, so after defining it outside loop it will be accessible in loop as well as outside loop.

SECTION 2: Guess the Output

Snippet 1:

```
public class NestedLoopOutput {
public static void main(String[] args) {
for (int i = 1; i <= 3; i++) {
for (int j = 1; j <= 2; j++) {
System.out.print(i + " " + j + " ");
}
System.out.println();
}
}
}
// Guess the output of this nested loop.
```

DryRun:

i = 1 j = 1 1 1

j= 2 1 2

i= 2 j = 1 2 1

j = 2 2 2

I = 3 j = 1 3 1

J =2 3 2

Observations:

Snippet 2:

```
public class DecrementingLoop {
public static void main(String[] args) {
int total = 0;
for (int i = 5; i > 0; i--) {
total += i;
if (i == 3) continue;
total -= 1;
}
System.out.println(total);
}
}
// Guess the output of this loop.
```

DryRun:

I = 5 total = 5

Total = 4

I = 4 total = 8

Total = 7

I = 3 total = 7+3 = 10

Skip (i==3) // so no total -=1

Total = 10

I =2 total = 10+2 = 12

Total = 11

I = 1 total = 11+1 = 12

Total = 11

Observations:

Snippet 3:

```
public class WhileLoopBreak {
public static void main(String[] args) {
int count = 0;
while (count < 5) {
System.out.print(count + " ");
count++;
if (count == 3) break;
}
System.out.println(count);
}
}
// Guess the output of this while loop.
```

DryRun:

Count = 0

Output = 0

Count++ //1

Count = 1

Output = 0 1

Count++ //2

Count = 2

Output = 0 1 2

Count++ //3

If count==3

Break

OutPut = 3

Observations:

Snippet 4:

```
public class DoWhileLoop {
public static void main(String[] args) {
int i = 1;
do {
System.out.print(i + " ");
i++;
} while (i < 5);
System.out.println(i);
}
}
// Guess the output of this do-while loop.
```

DryRun:

I = 1

Output= 1

I++ //

Condition check

i=2

Output= 1 2

I++

Condition check

I= 3

Output = 1 2 3

I++

Condition check

I = 4

Output = 1 2 3 4

I++ //i =5

Condition check (false)

SOP(i) // 5

Output = 1 2 3 4 5

Observations:

Snippet 5:

```
public class ConditionalLoopOutput {
public static void main(String[] args) {
int num = 1;
        for (int i = 1; i <= 4; i++)
         {
        if (i % 2 == 0) {
        num += i;
        }
        else {
        num -= i;
        }
}
```

```
System.out.println(num);
}}
```

DryRun: num = 1

I = 1 i%2 !=0 false num = num-I True // num = 1-1 = 0

I = 2 i%2 ==0 True num += i; // num = 0+2= || num = 2
I = 3 i%2 ==0 false num = num-I True // num = 2-3 = -1
I = 4 i%2 ==0 True num += i; // num = -1+4= || num = 3

Observations:

Snippet 6:

```
public class IncrementDecrement {
public static void main(String[] args) {
int x = 5;
int y = ++x - x-- + --x + x++;
System.out.println(y);
}
}
// Guess the output of this code snippet.
```

DryRun: x = 5

Y = 6- 6 + 4+ 4

Y = 8

Observations:

Increment decrement operator have more precedence then arithmetic operator so first operation will pe performed on unary operator and then binary operator, from left to right associativity.

Snippet 7:

```
public class NestedIncrement {
public static void main(String[] args) {
int a = 10;
int b = 5;
int result = ++a * b-- - --a + b++;
System.out.println(result);
}
}
// Guess the output of this code snippet.
```

DryRun: a = 10, b = 5

Result = 11*5 – 10 +4 = 55-10 = 45 +4 = 49

Observations:

Snippet 8:

```
public class LoopIncrement {
public static void main(String[] args) {
int count = 0;
for (int i = 0; i < 4; i++) {
count += i++ - ++i;
}
System.out.println(count);
}
}
// Guess the output of this code snippet.
```

DryRun:

Count = 0

i = 0

Count = count + i++ - ++i

0 + 0 - 2

Count = 2

i = 3

Count = count + i++ -++i

2 + 3- 5

Count = -4

i = 6

Condition = false = exit

Output: Count = -4

Observations: