**SECTION 1: Error-Driven Learning Assignment: Loop Errors**

**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?  
  
ANS: The loop condition is i < 10, which means the loop should terminate when i reaches or exceeds 10. However, the update statement is i--, which decreases i on each iteration. Since i never increases, it will always remain less than 10, causing the loop to run forever.  
The loop control variable be adjusted either by increasing i (i++) for an ascending loop or change the condition (i > 0) for a descending loop.  
  
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?  
  
ANS: count = 0 is an assignment, not a condition. The loop never runs because count is set to 0.  
Use == (while (count == 0)) for comparison or change the condition to count > 0 for proper execution.  
  
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 `do while` loop?  
  
ANS: The condition while (num > 0); allows the loop to run infinitely as num keeps increasing.  
If you want the loop to run only once, use while (num < 0); to ensure it stops immediately.  
  
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?  
  
ANS: The condition i <= 10 includes 10, causing an off-by-one error if the expected range is 1 to 9.  
Change the condition to i < 10 to ensure the loop stops at 9.  
  
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?  
  
ANS: The loop increments i (i++), which prevents it from decreasing and reaching the stopping condition (i >= 0).  
Use i-- instead of i++ to count downward correctly.  
  
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?  
  
ANS: Without {}, only the first statement is part of the loop, and "Done" prints once after the loop.  
 Use {} to include both statements in the loop body.   
  
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?   
  
ANS: count is declared but not initialized before being used in the loop condition.  
Initialize count (e.g., int count = 0;) before the loop.  
  
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?**

**ANS: The original loop decrements num, causing it to exit after printing only 1.  
Change num-- to num++ and modify the condition to while (num <= 5).  
  
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?   
  
ANS: The loop is not infinite, but it skips numbers because i increments by 2.  
Use i++ if sequential numbers are required.  
  
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?**

**ANS: error: incompatible types: int cannot be converted to boolean**

**while (num = 10) {**

**^  
while (num = 10) is an assignment, not a condition.  
Change to while (num == 10) or while (num > 0).  
  
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?**

**ANS: i += 2 skips numbers, printing only 0, 2, 4.  
Use i++ instead of i += 2 to print all numbers sequentially.  
  
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**

**ANS: x was declared inside the loop, making it inaccessible outside.  
Declare x before the loop so it remains accessible after the 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.  
  
ANS: Code Execution:**

**Outer Loop (i) runs from 1 to 3**

**When i = 1**

**Inner Loop (j) runs from 1 to 2**

**j = 1 Prints "1 1 "**

**j = 2 Prints "1 2 "**

**Moves to a new line**

**When i = 2**

**Inner Loop (j) runs from 1 to 2**

**j = 1 Prints "2 1 "**

**j = 2 Prints "2 2 "**

**Moves to a new line**

**When i = 3**

**Inner Loop (j) runs from 1 to 2**

**j = 1 Prints "3 1 "**

**j = 2 Prints "3 2 "**

**Moves to a new line**

**Final Output:**

**1 1 1 2**

**2 1 2 2**

**3 1 3 2   
  
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  
  
ANS:**

| **Iteration** | **i** | **total += i** | **if (i == 3) continue** | **total -= 1** | **Final total** |
| --- | --- | --- | --- | --- | --- |
| **1st** | **5** | **total = 0 + 5 = 5** | **No skip** | **total = 5 - 1 = 4** | **4** |
| **2nd** | **4** | **total = 4 + 4 = 8** | **No skip** | **total = 8 - 1 = 7** | **7** |
| **3rd** | **3** | **total = 7 + 3 = 10** | **Skip remaining part (continue)** | **total remains 10** | **10** |
| **4th** | **2** | **total = 10 + 2 = 12** | **No skip** | **total = 12 - 1 = 11** | **11** |
| **5th** | **1** | **total = 11 + 1 = 12** | **No skip** | **total = 12 - 1 = 11** | **11** |

**Final Output: 11  
  
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.  
  
ANS : Loop Iterations:**

| **Iteration** | **count before print** | **Printed** | **count++** | **if (count == 3) break** | **Final count** |
| --- | --- | --- | --- | --- | --- |
| **1st** | **0** | **0** | **count = 1** | **No break** | **1** |
| **2nd** | **1** | **1** | **count = 2** | **No break** | **2** |
| **3rd** | **2** | **2** | **count = 3** | **Break** | **3 (loop exits)** |

**Final Output: 0 1 2 3  
  
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.  
  
ANS: Loop Iterations:**

| **Iteration** | **i before print** | **Printed** | **i++ (incremented)** | **Condition check i < 5** |
| --- | --- | --- | --- | --- |
| **1st** | **1** | **1** | **i = 2** | **True (continue loop)** |
| **2nd** | **2** | **2** | **i = 3** | **True (continue loop)** |
| **3rd** | **3** | **3** | **i = 4** | **True (continue loop)** |
| **4th** | **4** | **4** | **i = 5** | **False (exit loop)** |

**The loop terminates when i = 5, and System.out.println(i); executes.  
Final Output:**

**1 2 3 4 5  
  
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);

}

}

**// Guess the output of this loop.   
  
ANS: Dry Run:**

**Initial values: num = 1**

**Loop Iterations:**

| **Iteration** | **i** | **Condition i % 2 == 0 (Even?)** | **Operation** | **Updated num** |
| --- | --- | --- | --- | --- |
| **1st** | **1** | **No (Odd)** | **num -= 1** | **1 - 1 = 0** |
| **2nd** | **2** | **Yes (Even)** | **num += 2** | **0 + 2 = 2** |
| **3rd** | **3** | **No (Odd)** | **num -= 3** | **2 - 3 = -1** |
| **4th** | **4** | **Yes (Even)** | **num += 4** | **-1 + 4 = 3** |

**Final Output:  
3**

**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.   
  
ANS: Summary of x Values at Each Step:**

| **Step** | **Operation** | **x Value** | **Expression Evaluation** |
| --- | --- | --- | --- |
| **Start** | **x = 5** | **5** | **-** |
| **++x** | **Pre-increment** | **6** | **6 - x-- + --x + x++** |
| **x--** | **Post-decrement** | **6 → 5** | **6 - 6 + --x + x++** |
| **--x** | **Pre-decrement** | **4** | **6 - 6 + 4 + x++** |
| **x++** | **Post-increment** | **4 → 5** | **6 - 6 + 4 + 4** |

**Final Output:   
8  
  
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.  
  
  
  
  
  
  
  
ANS: Summary of Variable Changes:**

| **Step** | **Operation** | **a Value** | **b Value** | **Expression Evaluation** |
| --- | --- | --- | --- | --- |
| **Start** | **a = 10, b = 5** | **10** | **5** | **-** |
| **++a** | **Pre-increment** | **11** | **5** | **11 \* b-- - --a + b++** |
| **b--** | **Post-decrement** | **11** | **5 → 4** | **11 \* 5 - --a + b++** |
| **11 \* 5** | **Multiplication** | **11** | **4** | **55 - --a + b++** |
| **--a** | **Pre-decrement** | **10** | **4** | **55 – 10 + b++** |
| **b++** | **Post-increment** | **10** | **4 → 5** | **55 – 10 + 4** |

**Final Output:  
49**

**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.  
  
ANS: Summary of Variable Changes**

| **Iteration** | **i (Before)** | **i++ (Used)** | **i (After i++)** | **++i (Used)** | **i (After ++i)** | **Expression** | **count** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **0** | **1** | **2** | **2** | **0 - 2 = -2** | **-2** |
| **2** | **3** | **3** | **4** | **5** | **5** | **3 - 5 = -2** | **-4** |
| **3** | **5** | **-** | **-** | **-** | **-** | **-** | **-** |

**Final Output:  
-4**