Build COMPETENCY
across your TEAM

SYNERGETICS
GET IT RIGHT

Microsoft Partner
Gold Cloud Platform
Silver Learning

Programming Fundamentals

Google Cloud Platform

Windows Azure

amazon web services

APACHE kafka
A distributed streaming platform

Apache SOFTWARE FOUNDATION

APACHE Spark

Java

hadoop

Programming Fundamentals

Module 1: Pseudo code/Algorithms/Flow Charts
Module 2:Variables
Module 3:Decision Constructs
Module 4:Loop

APACHE Spark

APACHE kafka®
A distributed streaming platform

hadoop

cassandra

MESOS

APACHE HBASE

Module1

Pseudo code/Algorithms/Flow Charts

Flowcharts and pseudocode

Flowcharts and pseudocode are used to:

- ✓ Plan the logical steps for solving a programming problem.

What is Pseudocode?

- Pseudocode is a natural language description of a program.
- Allows the developer to focus on the logic of the code not details of syntax.
- It describe the entire logic of the task so that implementation becomes easier.
- The pseudocode acts as a blueprint for the source code.

Why Pseudocode ?

- Increase the Quality of program – Easy way for Analyst to ensure the code matches with design specifications.

- Thus it helps to ensure requirements are met and that program code meets good software development practice.

- Less Cost activity. Since Catching Logical errors is less tedious than catching them in development process.

- **There is no one defined way of writing pseudo code but a pseudo-code should possess three elements: clarity, precision and concise.**

Pseudo code can be broken down into five components:

- Variables
- Assignment
- Input/output
- Selection
- Repetition

Pseudo code can be broken down into five components:

- **A variable -** has a name, a data type, and a value. There is a location in memory associated with each variable.

- **Assignment** - is the physical act of placing a value into a variable. Assignment can be shown using set = 5; set = num + set;

- **Input / Output-** both deal with an outside source can be a user or another program) receiving or giving information.

- **Selection** is our conditional statements – If else constructs

- **Repetition** is a construct that allows instructions to be executed multiple time – Loop constructs

Example 2: Get 5 numbers and output average Enter 5 numbers

```
Prompt & get the score for number1
Prompt & get the score for number2
Prompt & get the score for number3
Prompt & get the score for number4
Prompt & get the score for number5

    average = number1 + number2 + number3 + number4 + number5) / 5

Output average
```

Example 1 : Write a PseudoCode for Finding Whether a number is odd or even:

PROMPT is a Pseudocode keyword used to take inputs from the keyboard

STORE in a variable

```
BEGIN
    PROMPT "Enter the number“
    STORE IN num
    IF num MOD 2 == 0) THEN
        DISPLAY "Even Number“
     ELSE
        DISPLAY "Odd Number”
    ENDIF
END
```

Example 3: Program to find the sum of first n natural numbers.

```
BEGIN
    DECLARE num,count,sum AS INTEGER
    PROMPT "Enter the value of n" AND STORE IN num
        FOR COUNT = 1 TO num
            sum+=count;
        END FOR
    PRINT "Sum is " + sum
END
```

Algorithm

- An algorithm is a set of instructions for solving a problem.
- It is a basic technique of how to do a specific task.
- It takes input, processes it according to a set of instructions, and generates output.
- An algorithm must provide correct output for every possible input condition.
- An algorithm must have a definite end point so that when the input has been processed and the desired output achieved, the process stops.

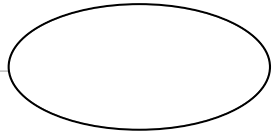
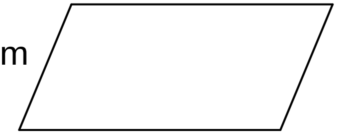
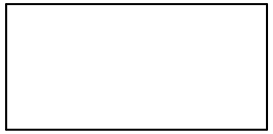
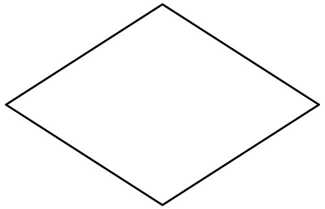
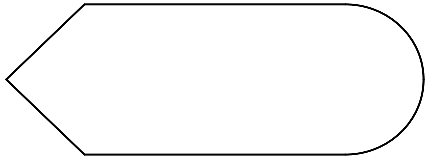
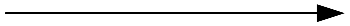
Example: Algorithm to calculate the difference between two numbers

```
START

    READ A
    READ B

            DIFF = A – B

    PRINT DIFF

STOP
```

Flowchart:

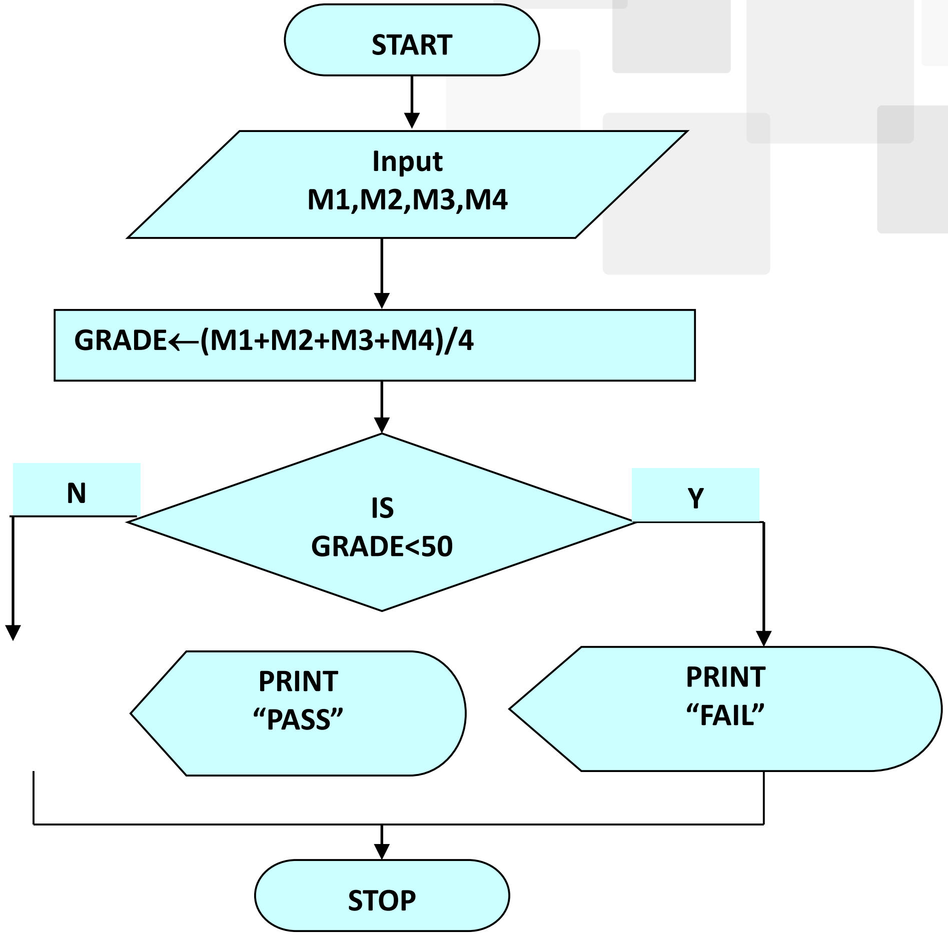
- A flow chart, is a graphical representation of a process or system that details the sequencing of steps required to create output.
    - shows logic of an algorithm
    - emphasizes individual steps and their interconnections
    - e.g. control flow from one action to the next
- Different symbols are used to draw each type of flowchart.

Flowchart Symbols

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the program

Example 1:Calculate Grade

```
Step 1:   Input M1,M2,M3,M4

Step 2:   GRADE ← (M1+M2+M3+M4)/4

Step 3:   if (GRADE <50) then
                    Print “FAIL”
          else
                    Print “PASS”
          endif
```



Module3

Module 2: Variables

Variables

- **What is Variable?**

- A piece of your computer's memory that is given a name and type, and can store a value.

- We use variables to store the results of a computation and use those results later in our program.

- Variables are a bit like the 6 preset stations on your car stereo, except we can, essentially, have as many of them as we want, and we give them names, not numbers.

Variable DataTypes

- Numeric
- Boolean
- Character
- Date

Rules for naming a variable:

- Variable name may consists of number, characters, $ sign and _ score but variable can't start with number.
- Variables can never used special characters like +, -, &, * #, ., etc.
- Space is not allowed in variable names
- Reserve words can not used as variable names like print, get, read, display if etc.
- Variable name must be unique in its scope.

Constants

- Constants are similar to variables except that they hold a fixed value. They are also called “READ” only variables.

- MAX_LENGTH = 420;
- PI = 3.1428;

- By convention upper case letters are used for defining constants.

Module3

Module 3:Decision Constructs

Decision or Selection

- Drive car or take BEST bus?
- Party or study? •
- Fly or drive?
- What is the common idea for all these activities?

- **Selection**

```
• if condition statement
• if condition statement1 else statement2
• if condition
     statement1; …
     else statement2
•  …
```

Decision Making

- To facilitate conditional control flow in Java there are relational and logical operators to form a conditional expression that returns either true or false.

if-else

```
if i < 10

print  “i is less than 10”

else
   print “i is greater than or equal to 10”
```

Nested IF

Nesting of ifstatements is very helpful when you have something to do by following more than one decision.

```
boolean isSat = true;     int whichSat = 2;    boolean isHoliday = false;

if isSat

    if      whichSat == 2

        if isHoliday == false

            print "It is meeting today."
else

            print “ No meeting today."
```

If-Else Ladder

```
ch = 'o';
 if ch == 'a’  or  ch == 'A'
      print ch + " is vowel.“
else if ch == 'e' || ch == ‘E’
      print ch + " is vowel.“
else if ch == 'i' || ch == ‘I’)
      print ch + " is vowel.“
else if ch == 'o' || ch == 'O’)
      print ch + " is vowel.“
else if ch == 'u' || ch == ‘U’)
     print ch + " is vowel.“
else
     print ch + " is a consonant."
```

Module4

Loops

How does the loop works?

- The test expression inside parenthesis is a boolean expression.
  - If the test expression is evaluated to true,
  - statements inside the while loop are executed.
  - then, the test expression is evaluated again.
- This process goes on until the test expression is evaluated to false.
- If the test expression is evaluated to false,
- while loop is terminated.

Repetitions:

- `while` **Loops**
- `Repeat` **Loops**
- `for` **Loops**
- `break` **and** `continue`

While Loop

while loop-continuation-condition

// the syntax for the while loop

// loop-body;

Statements

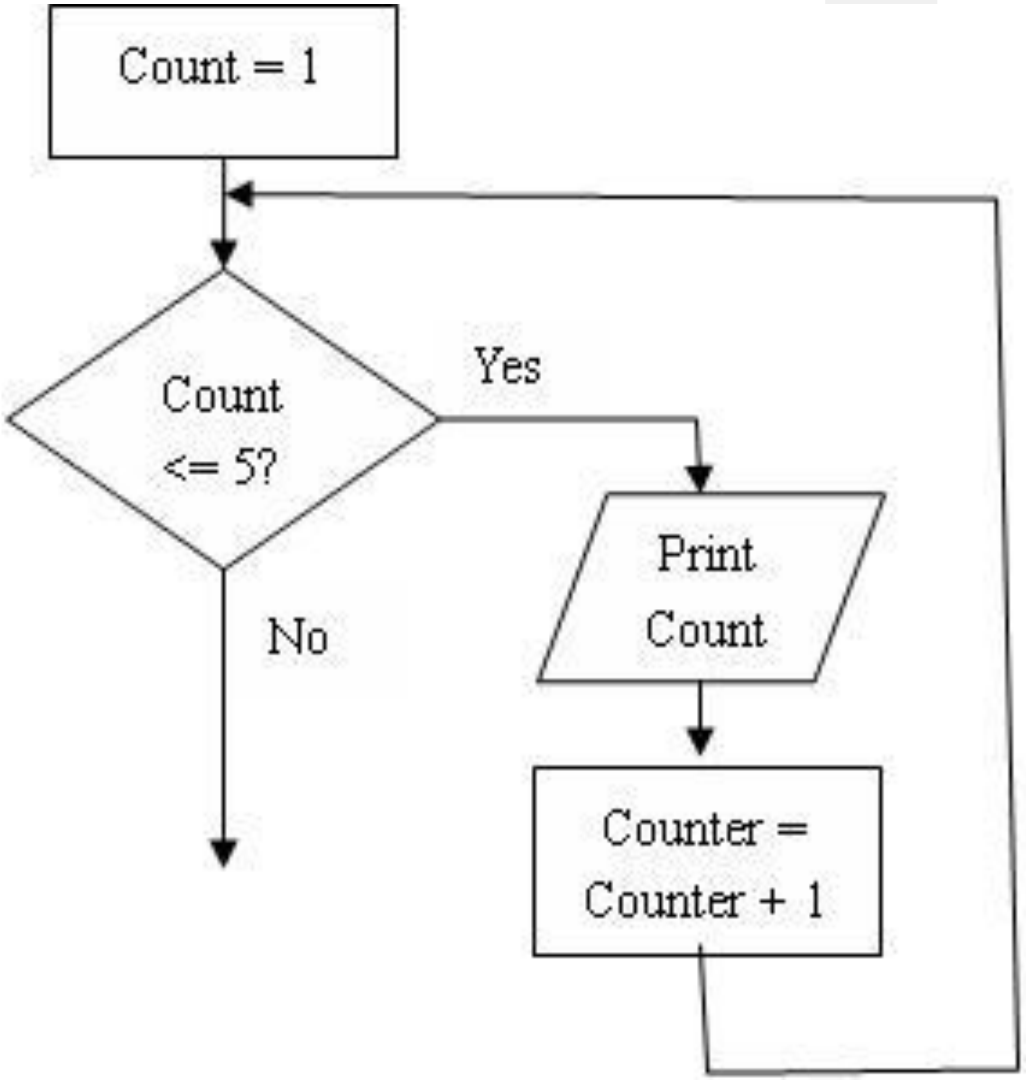
- If the `condition` is true, the `statement` is executed
- Then the condition is evaluated again, and if it is still true, the statement is executed again
- The statement is executed repeatedly until the condition becomes false

Example:

```
int count = 0;

while count < 100
print "Welcome to Java!"
  count++;
```

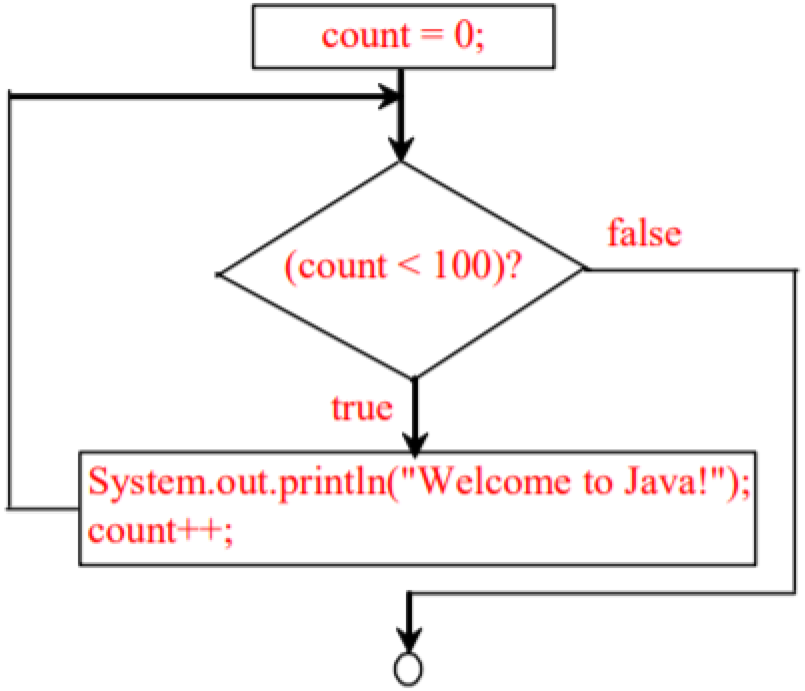
- The variable count is initially **0**. The loop checks whether count < 100 is true.
- If so, it executes the loop body to print the message "Welcome to Java!" and increments count by 1.
- It repeatedly executes the loop body until count < 100 becomes false.
- When count < 100 is false, the loop terminates and the next statement after the loop statement is executed.

While Loop:



Example : Print Welcome to Java! 100 times

```
int count =0;
while count<100

     Print  "Welcome to Java!"
     count ++;
```



Repeat untill loop

```
Repeat

        // Loop body;
        Statements
untill continue-condition
```

The `statement` is executed once initially, and then the `condition` is evaluated

The statement is executed repeatedly until the condition becomes false

```
int count = 0;
repeat
count++;
  print count
untill count < 5
```

for loop

```
for condition
                    statement;
```

- The initialization is executed once before the loop begins
- The statement is executed until the condition becomes false

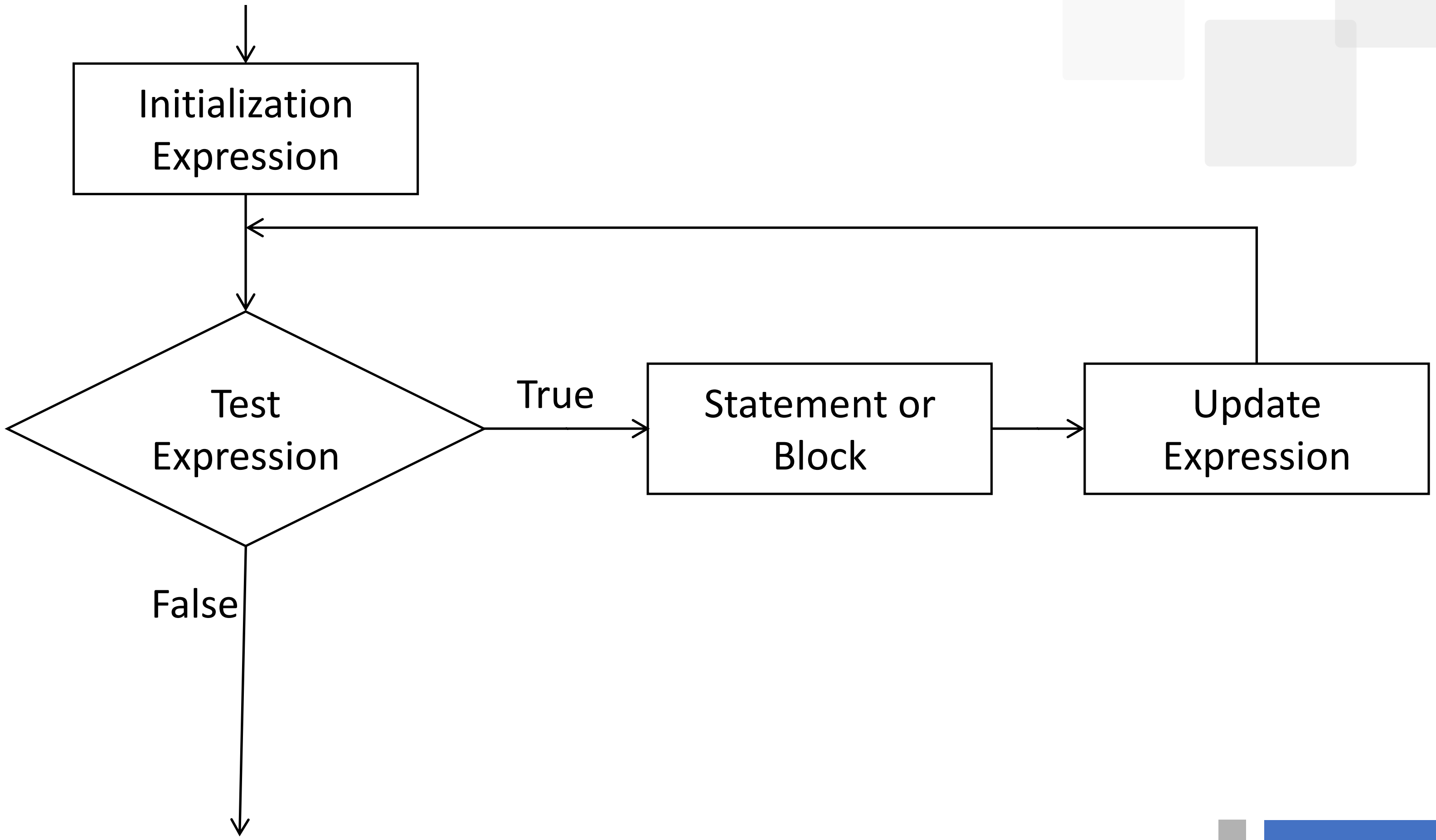
For Loop:

```
for count = 1 to 5
        println count
```

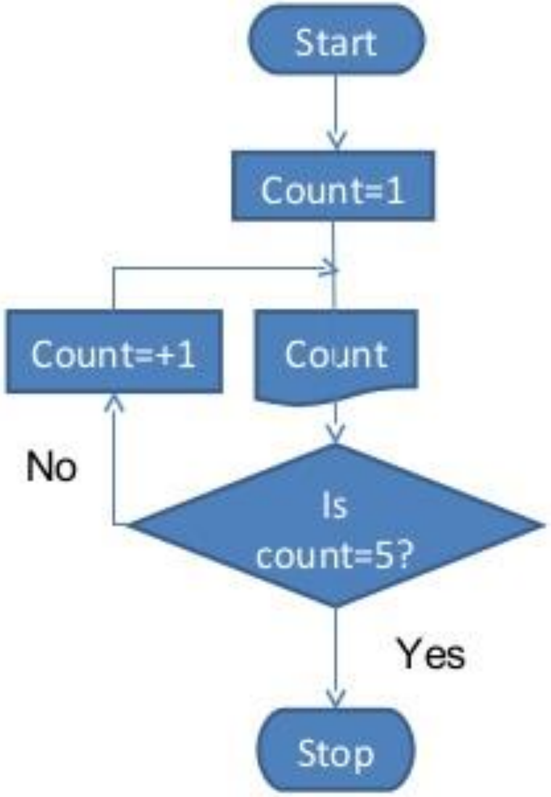
- Like a `while` loop, the **condition** of a `for` loop is tested prior to executing the loop body

- Therefore, the body of a `for` loop will execute zero or more times

for Loop Flowchart



Flowchart Iteration



FOR LOOP

Range

```
Start
for Count = 1 to 5 by 1 do
   Print Count
end for
end
```

WHILE LOOP

Range

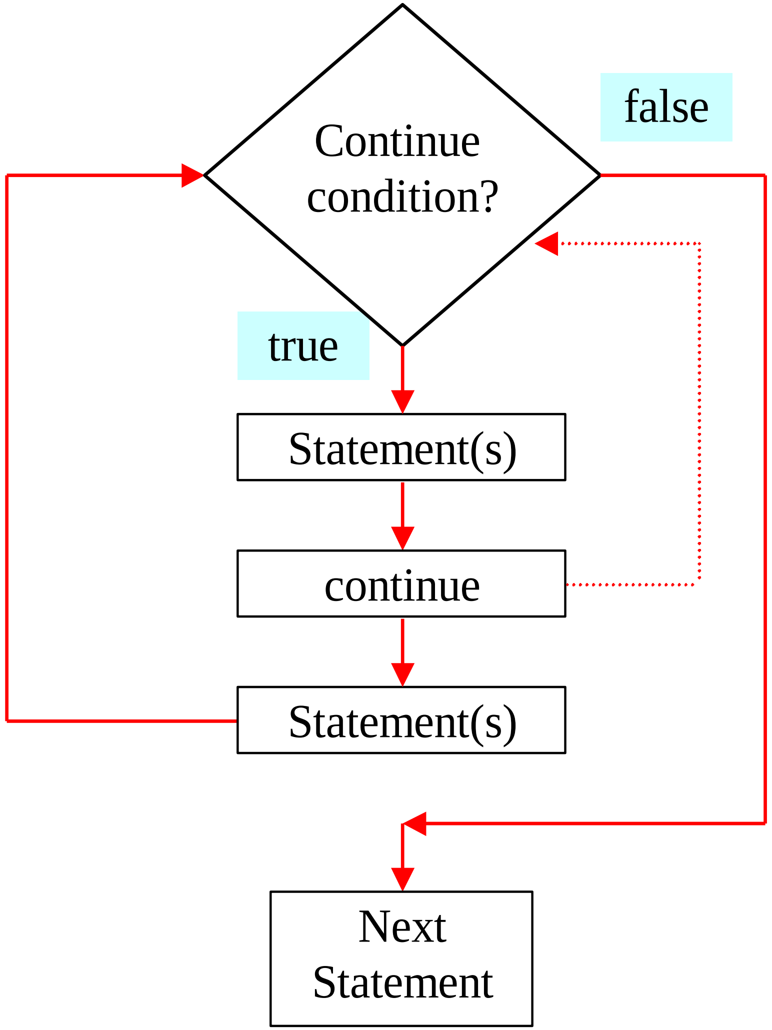
```
Start
Count = 1
while Count  <= 5 do
     Print Count
      Count=Count+1
end while
end
```

REPEAT LOOP

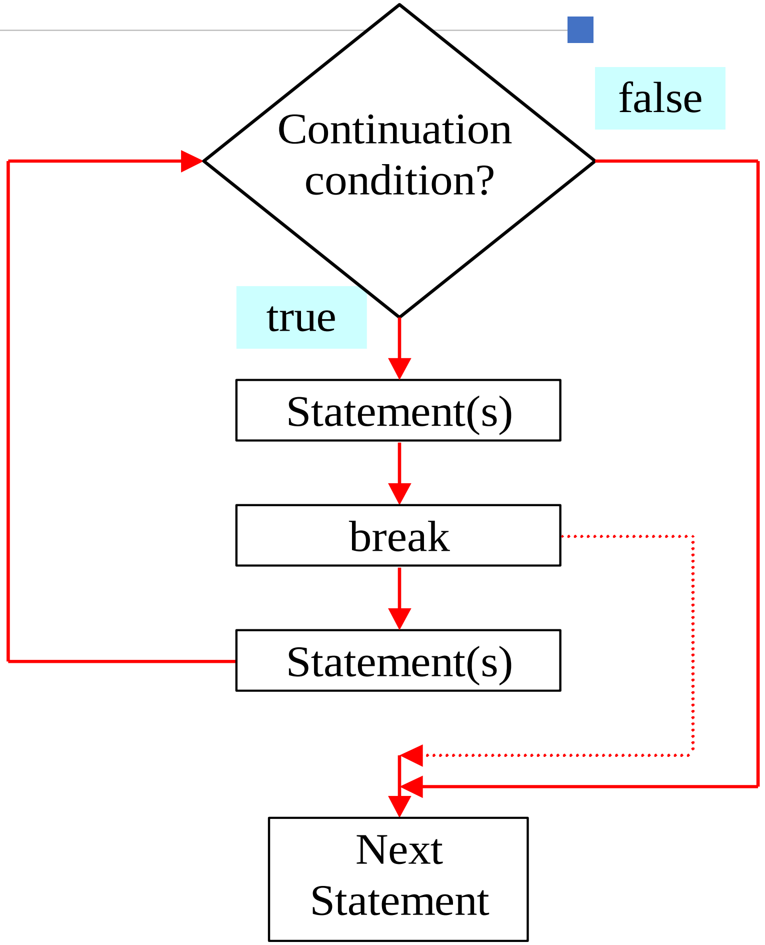
Range

```
Start
Count = 1
repeat
     Print Count
      Count=Count+1
until count > 5
end
```

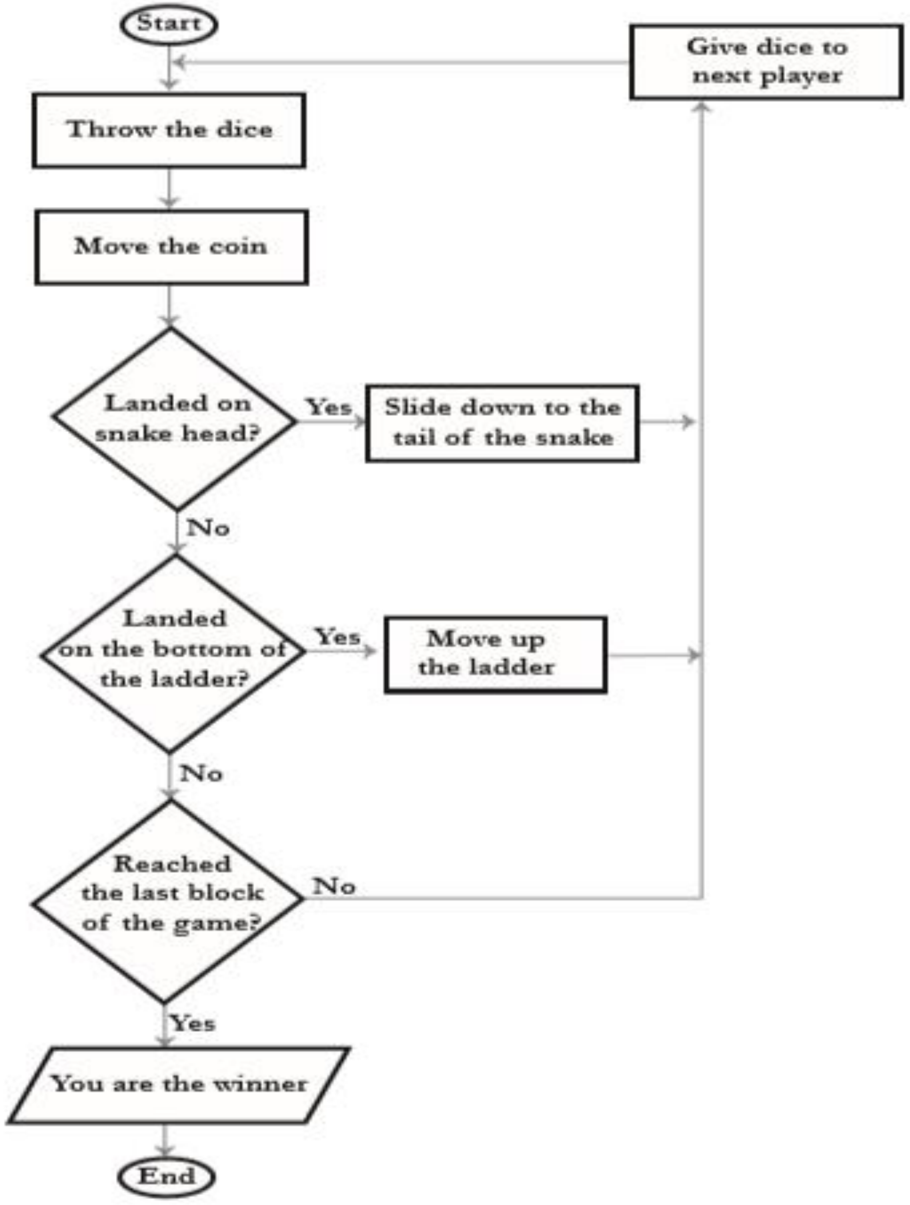
The `continue` Keyword



The `break` Keyword



Flowchart - Snakes and Ladder game



Start

Throw the dice

Move the coin

Landed on snake head?

Yes

Slide down to the tail of the snake

No

Landed on the bottom of the ladder?

Yes

Move up the ladder

No

Reached the last block of the game?

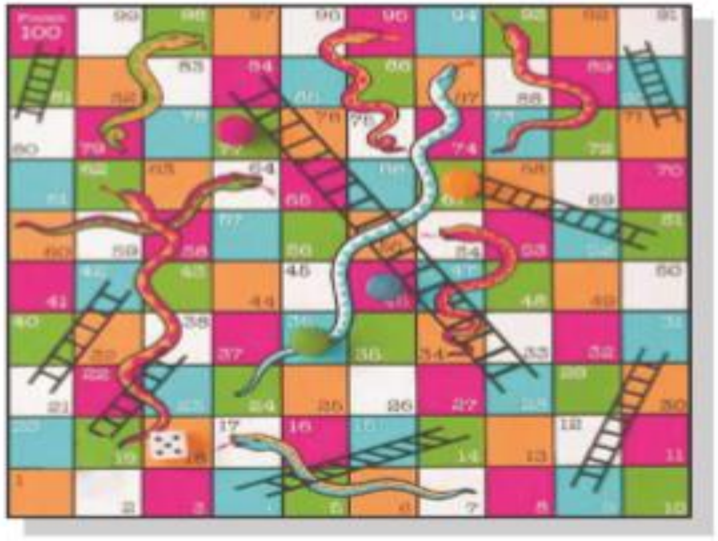
No

Give dice to next player

Yes

You are the winner

End



Thank You