Syllabus Content:

9.2. Algorithms

- show understanding that an algorithm is a solution to a problem expressed as a sequence of defined steps
- use suitable identifier names for the representation of data used by a problem
    - summarise identifier names using an identifier table
- show understanding that many algorithms are expressed using the four basic constructs of assignment, sequence, selection and repetition
- show understanding that simple algorithms consist of input, process, output at various stages
- document a simple algorithm using:
    - **Structured English**
    - **pseudocode** (on the examination paper, any given pseudocode will be presented using the Courier New font)
    - **program flowchart**
- derive pseudocode or a program flowchart from a structured English description of a problem
- derive pseudocode from a given program flowchart or vice versa
- use the process of stepwise refinement to express an algorithm to a level of detail from which the task may be programmed
- decompose a problem into sub-tasks leading to the concept of a program module (procedure/ function)
- show an appreciation of why logic statements are used to define parts of an algorithm solution
- use logic statements to define parts of an algorithm solution

10.1 Data types

- select appropriate data types for a problem solution
- use in practical programming the data types that are common to procedural high-level languages: integer, real, char, string, Boolean, date (pseudocode will use the following data types: **INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE, ARRAY, FILE)**
- show understanding of how character and string data are represented by software including the ASCII and Unicode character sets

10.2 Arrays

- use the technical terms associated with arrays including upper and lower bound
- select a suitable data structure (1D or 2D array) to use for a given task
- use pseudocode for 1D and 2D arrays (pseudocode will use square brackets to contain the array subscript, for example a 1D array as A[1:n] and a 2D array as C[1:m, 1:n])
- write program code using 1D and 2D arrays
- write algorithms/program code to process array data including:

Syllabus Content:

11.1 Programming basics

- Implement and write pseudocode from a given design presented as either a program flowchart or structured English

Notes and guidance

- Write pseudocode statements for implement and write a program from a given design presented as either a program flowchart or pseudocode
- write pseudocode for:
  - the declaration of variables and constants
  - the assignment of values to variables and constants
  - expressions involving any of the arithmetic or logical operators
  - input from the keyboard and output to the console given pseudocode will use the following structures:
  - DECLARE <identifier> : <data type> // declaration
  - CONSTANT <identifier> = <value>
  - <identifier> ← <value> or <expression> // assignment
  - INPUT <identifier>
  - OUTPUT <string> , OUTPUT <identifier(s)>

11.2 Constructs

- Write pseudocode to write:

Selection

- use an 'IF' structure including the 'ELSE' clause and nested IF statements
- use a 'CASE' structure

Iteration

- use a 'count-controlled' loop:
  - FOR <identifier> ← <value1> TO <value2> <statement(s)> ENDFOR
  - alternatively: FOR <identifier> ← <value1> TO <value2> STEP <value3> <statement(s)> ENDFOR
- use a 'post-condition' loop:
  - REPEAT <statement(s)> UNTIL <condition>
- use a 'pre-condition' loop
  - WHILE <condition> <statement(s)> ENDWHILE • justify why one loop structure may be better suited to a problem than the others

9.2 Algorithms:

An algorithm is a sequence of steps done to perform some task.

- The essential aim of an algorithm is to get a specific output,
- An algorithm involves with several continuous steps,
- The output comes after the algorithm finished the whole process.

So basically, all algorithms perform logically while following the steps to get an output for a given input.

Types of Algorithms:

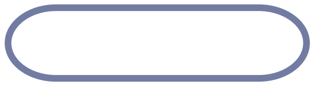
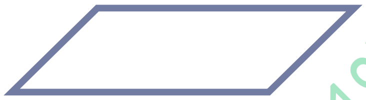
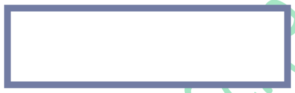
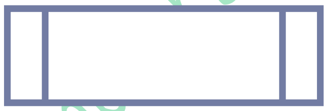
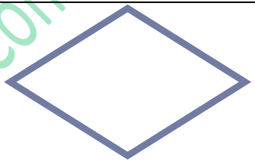
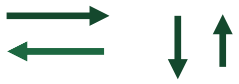
- Structured English
- Flowcharts
- Pseudo codes
- Program Code

STRUCTURED ENGLISH:

Structured English provides a more formal way of documenting the stages of the algorithm. Structured English is a subset of English language that consists of command statements used to describe an algorithm.

FLOWCHARTS:

F**low chart** is a graphical representation of a program.
Flowcharts use different symbols containing information about steps or a sequence of events.

Symbol	Name	Usage
	Terminator	To start and stop the program
	**INPUT** or **OUTPUT**	To INPUT or OUTPUT data
	Process	To show a process
	**PROCEDURE** or **FUNCTION**	To Represent a Pre Defined Function/Procedure/Subroutine
	Decision Symbol	A Condition statement with Yes/No/True/False decision
	Data flow lines	Represent the flow of data from one component to next.

PSEUDOCODE: Pseudo code is **an outline or a rough draft of a program**, written as a series of instruction.
Pseudo code uses **keywords** commonly found in ***high-level programming languages***, **without being bound** to the **syntax** of any particular language.
It describes an algorithm's steps like program statements.

Variable:

Variable is a named memory location with **DataType** where value can be stored. The content of a variable can change at runtime.

```
                              DECLARE variable : Datatype
e.g DECLATE name : STRING
```

The variable are always declared with **meaningful name** and appropriate data type.
Variable name cannot start with a number, and it cannot contain a space in between two letters.

```
e.g DECLATE First Name : STRING (WRONG Declaration)
     DECLATE FirstName : STRING (Correct Declaration)

DECLATE 1number : REAL (WRONG Declaration)
DECLATE number1 : REAL (CORRECT Declaration)
```

Constants:

Just like variables, constants are "dataholders". They can be used to store data that is needed at runtime.
In contrast to variable, the content of a constant can't change at runtime, it has a constant value.
Before the program can be executed (or compiled) the value for a constant must be known.

```
e.g   CONSTANT gravity = 9.8
or    CONSTANT gravity ← 9.8
```

Arithmetic

Use the arithmetic operators.
`(+) addition), (-) subtraction) , (*) multiplication, (/) division`

Assignment

Assignment is the process of writing a value into a variable (a named memory location).
For example, **Count ← 1** can be read as 'Count is assigned the value 1', 'Count is made equal to 1' or 'Count becomes 1'.

Initialization:

If an algorithm needs to read the value of a variable *before* it assigns input data or a calculated value to the variable, the algorithm should assign an appropriate initial value to the variable, known as Initialization.

```
DECLATE marks : INTEGER   //Declaration
marks = 0                 //initialization
```

Input

We indicate input by words such as **INPUT**, **READ** **or** **ENTER**, followed by the name of a variable to which we wish to assign the input value.

Output:

We indicate output by words such as **OUTPUT**, **WRITE** **or** **PRINT**, followed by a comma-separated list of expressions.

Totaling

To keep a running total, we can use a variable such as Total or Sum to hold the running total and assignment statements such as:

Total ← Total + Number (Adds Number to Total)

Counting

It is sometimes necessary to count how many times something happens.To count up or increment by 1, we can use statements such as:

Count ← Count + 1

INCREMENT Count by 1

Structured statements

In the sequence structure the processing steps are carried out one after the other. The instructions are carried out in sequence, unless a selection or loop is encountered.

Mathematical Operators in Pseudocodes and Programming languages

Pseudocode	Operator (VB)	Operator (Python)	Mathematical operator
+	+	+	Addition
-	-	-	Subtraction
*	*	*	Multiplication
/	/	/	Division
=	=	==	Equal
<>	<>	!=	Not equal
MOD	Mod	%	Modulus
^	^	**	Exponent 2^3 = 2^3 or 2**3

Logical Operators in Pseudocodes and Programming languages

Pseudocode	Operator (VB)	Operator (Python)	Comparison
>	>	>	Greater than
<	<	<	Less than
>=	>=	>=	Greater than equal to
<=	<=	<=	Less than equal to
=	=	==	Equals to
<>	<>	!=	Not equal
( )	( )	( )	Group in Brackets
^	^	**	Exponent
OR	OR	OR	Or
NOT	NOT	NOT	Not
AND	AND	AND	And

10.1 Data types

The following table shows the Visual Basic data types, their supporting common language runtime types, their nominal storage allocation, and their value ranges.

Basic Data Types:
A variable can store one type of data. The most used data types are:

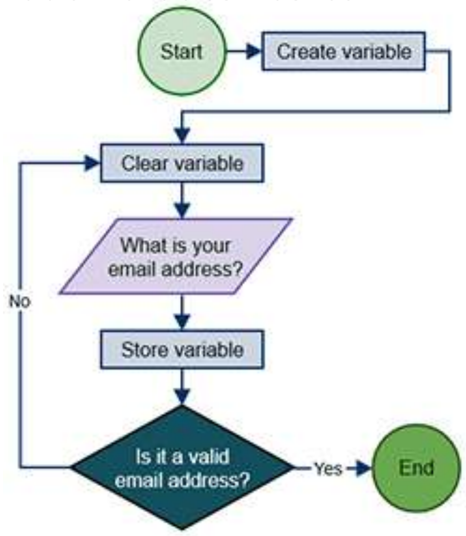
Pseudo code	Operator (VB)	Operator (Python)	DATA TYPE Formats
INTEGER	Integer	int	Integer (Whole numbers)
REAL	Decimal	float	Decimal numbers
CHAR	Char	Not used in Python	Single character e.g "F" for female or "M" for male
BOOLEAN	Boolean	bool	Boolean e.g True or False
STRING	String	str	Text
DATE	Date	class datetime	Date

11.1 Declaration of Variables and Constant:

The process of creating a variable is called declaring a variable. Variables must be created or declared where users enter their data.

Pseudo code

```
BEGIN
DECLARE variable : Datatype
Variable ⟵ 0  //initialization
OUTPUT ("What is your Email address")
INPUT variable value\
IF valid email address?
     Then ...
END IF
```



Each declaration needs 4 things:

Pseudo code

- **DECLARE** keyword
- Variable name
- : keyword
- Variable data type

VB code example:

- **DIM** keyword
- Variable name
- **AS** keyword
- Variable data type

```
DECLARE variable : Datatype
```

Declaring Multiple Variables:

Pseudocodes	VB Code Console Mode
`DECLARE index : Integer`	`Dim index As Integer`
`DECLARE grade : Integer`	`Dim grade As Integer`
`DECLARE counter : Integer`	`Dim counter As Integer`

PYTHON:

```
Index, grade, counter = 0
```

In Python you have to initialize variable with a value

The three declarations above can be rewritten as one declaration if same data type is used:

```
DECLARE index, grade, counter : Integer
```

VB Code Console Mode

```
Dim index, grade, counter As Integer
```

Constants

Creating Constants in Pseudocode is just writing costant name and value with it. In contrast to variable, the content of a constant can't change at runtime, it has a constant value.

Pseudocode:
```
CONSTANT <identifier> = <Value>
CONSTANT  Pi ⟵ 3.1415
```
or
```
CONSTANT   Pi = 3 .14
```

VB Console mode:
```
Const pi As Decimal = 3.1415
Dim Pi As Decimal = 3.1415
```

PYTHON:
```
ConstPi =3.1415
```

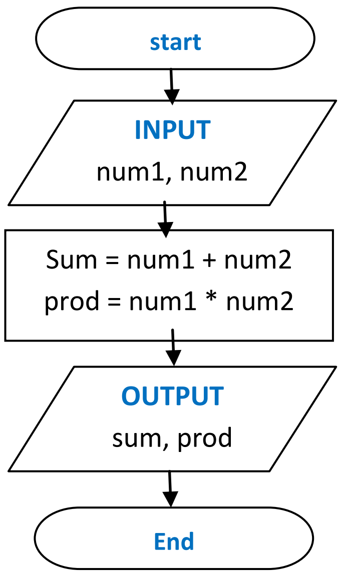
Type of Programs:
- Sequence
- Selection
- Repetitions/Loops

Sequence

Statements are followed in sequence so the order of the statements in a program is important. Assignment statements rely on the variables used in the expression on the right-hand side of the statement all having been given values. Input statements often provide values for assignment statements. Output statements often use the results from assignment statements.

PSEUDOCODE
```
BEGIN
DECLARE num1, num2 : Integer
DECLARE sum, product :Integer
OUTPUT ("Enter number 1")
INPUT number1
OUTPUT ("Enter number 2")
INPUT number2
Sum ⟵ number1 + number2
product ⟵ number1 * number2
OUTPUT ("the sum is", sum)
OUTPUT ("the product is", product)
END
```

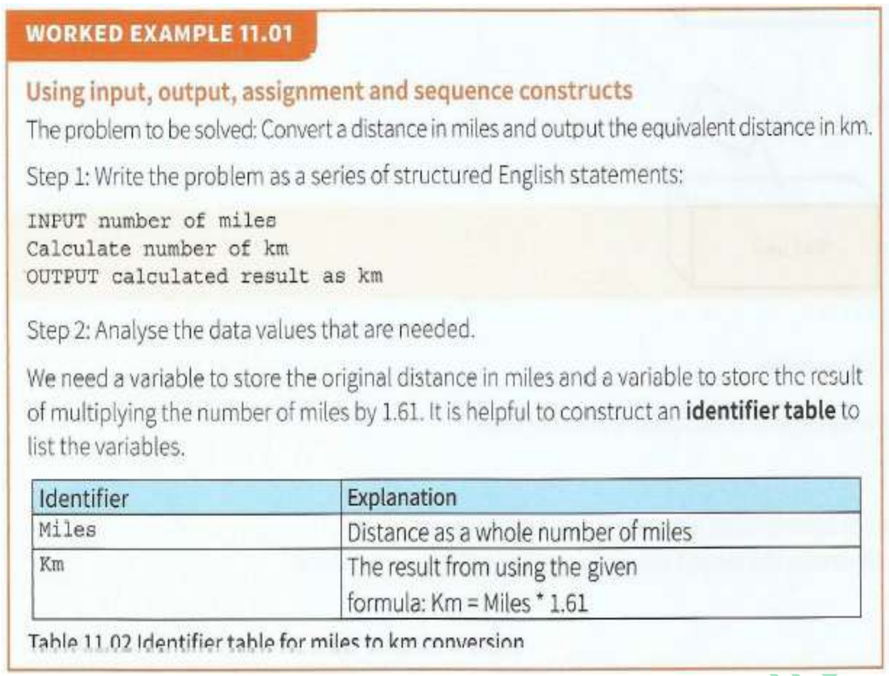
Flow Chart



VB code
```
Sub main()
    Dim Num1 As Integer
    Dim Num2 As Integer
    Dim Sum As Integer
    Dim Procduct As Integer
    Console.Writeline("Enter number1")
    Num1 = Console.Readline()
    Console.Writeline("Enter number2")
    Num2 = Console.Readline()
    Sum = Num1+Num2
    Product = Num1*Num2
    Console.Writeline("Sum is" & sum)
    Console.Writeline("Product is" & Product)
End Sub
```

PYTHON
```
num1=int(input("enter number1"))
num2=int(input("enter number2"))
total = num1+num2
prod = num1*num2
print("Total is", total)
print("Product is", prod)
```

STRUCTURED ENGLISH



WORKED EXAMPLE 11.01

Using input, output, assignment and sequence constructs

The problem to be solved: Convert a distance in miles and output the equivalent distance in km.

Step 1: Write the problem as a series of structured English statements:

```
INPUT number of miles
Calculate number of km
OUTPUT calculated result as km
```

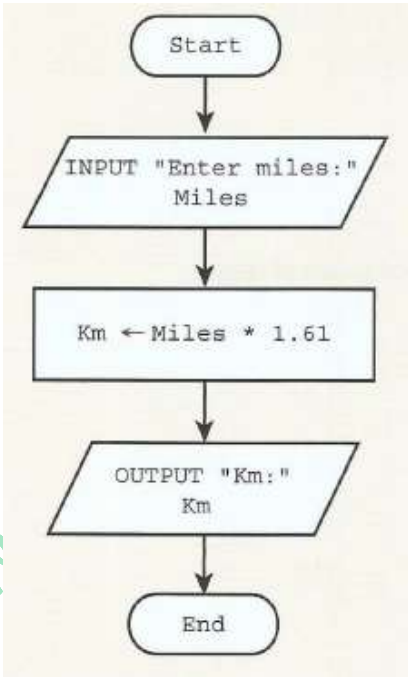
Step 2: Analyse the data values that are needed.

We need a variable to store the original distance in miles and a variable to store the result of multiplying the number of miles by 1.61. It is helpful to construct an **identifier table** to list the variables.

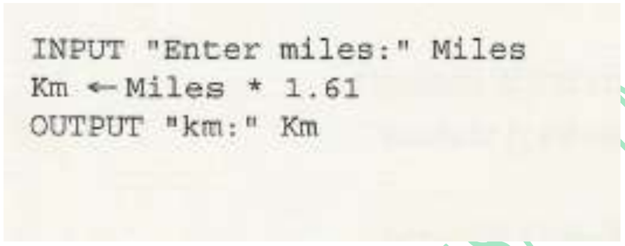
Identifier	Explanation
Miles	Distance as a whole number of miles
Km	The result from using the given formula: Km = Miles * 1.61

Table 11.02 Identifier table for miles to km conversion

FLOWCHART

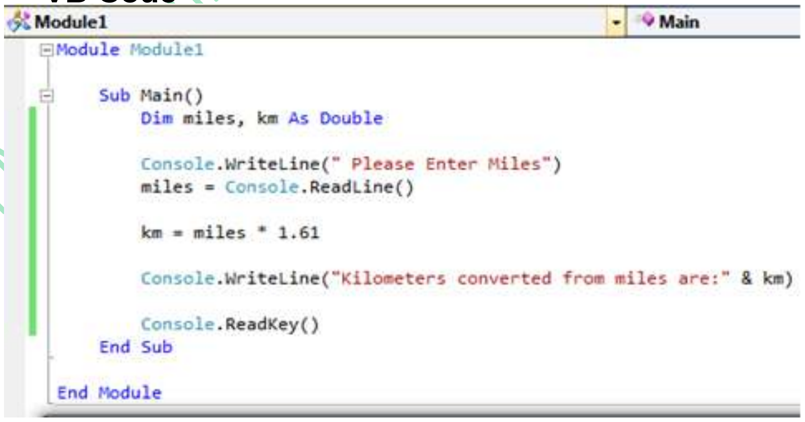


Pseudocode



```
INPUT "Enter miles:" Miles
Km ← Miles * 1.61
OUTPUT "km:" Km
```

VB Code



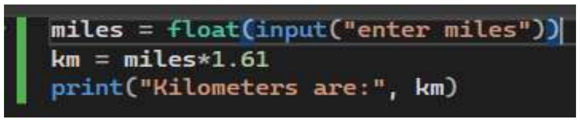
```
Module Module1

    Sub Main()
        Dim miles, km As Double

        Console.WriteLine(" Please Enter Miles")
        miles = Console.ReadLine()

        km = miles * 1.61

        Console.WriteLine("Kilometers converted from miles are:" & km)

        Console.ReadKey()
    End Sub

End Module
```

Pseudocode:

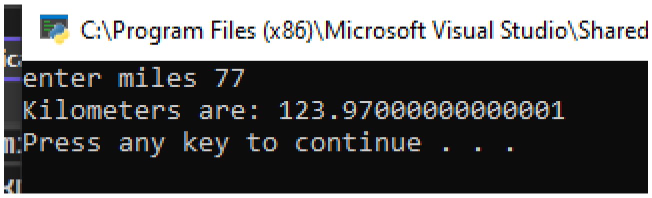
```
BEGIN
DECLARE miles,km : REAL
OUTPUT (“Enter miles”)
INPUT miles
km ⟵ miles * 1.61
OUTPUT(“Km are : ” & km)
END
```

PYTHON:

```
miles = float(input("enter miles"))
km = miles*1.61
print("Kilometers are:", km)
```



```
miles = float(input("enter miles"))
km = miles*1.61
print("Kilometers are:", km)
```



C:\Program Files (x86)\Microsoft Visual Studio\Shared

```
enter miles 77
Kilometers are: 123.97000000000001
Press any key to continue . . .
```

11.2 Structured statements for selection (conditional statements)

These statements are used to select alternative routes through an algorithm; selection's logical expressions often involve comparisons, which can operate on text strings as well as numbers

- IF…THEN…ELSE…ENDIF
- CASE…OF…OTHERWISE…ENDCASE

IF…THEN…ELSE…ENDIF

For an IF condition the THEN path is followed if the condition is true and the ELSE path is followed if the condition is false.
There may or may not be an ELSE path. The end of the statement is shown by ENDIF.

A condition can be set up in different ways:

```
IF ((Height > 1) OR (Weight > 20) OR (Age > 5)) AND (Age < 70)
      THEN
            PRINT  ("You can ride")
      ELSE
            PRINT ("Too small, too young or too old")
ENDIF
```

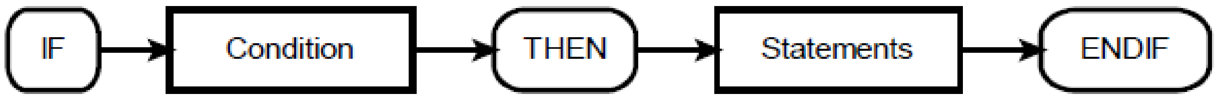
CASE … OF … OTHERWISE … ENDCASE

For a CASE condition the value of the variable decides the path to be taken. Several values are usually specified. OTHERWISE is the path taken for all other values. The end of the statement is shown by ENDCASE.

The algorithm below specifies what happens if the value of Choice is 1, 2, 3 or 4.

```
CASE Choice  OF
      1:  Answer ← Num1 + Num2
      2:  Answer ← Num1 - Num2
      3:  Answer ← Num1 * Num2
      4:  Answer ← Num1 / Num2
      OTHERWISE   PRINT  ("Please enter a valid choice")
ENDCASE
```

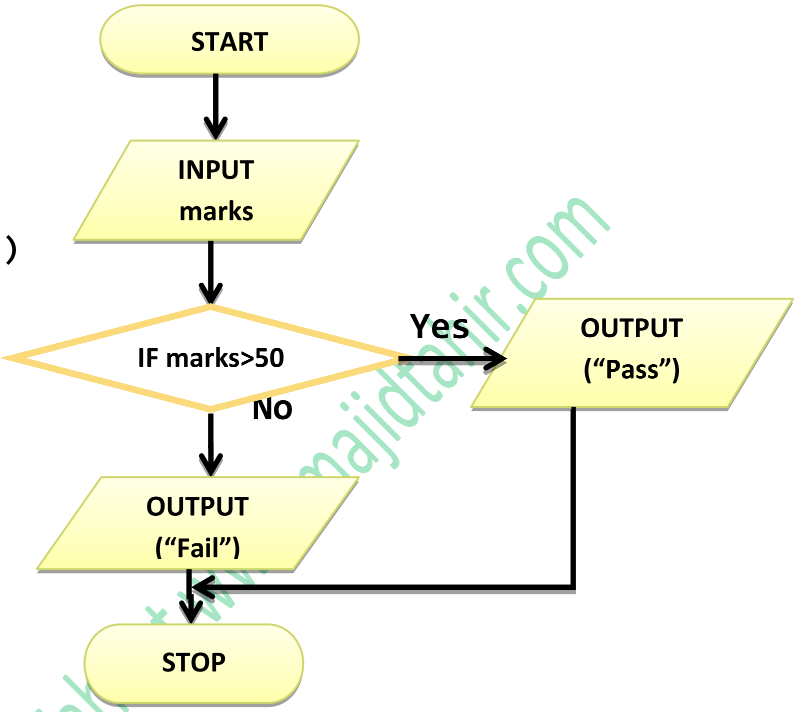
The IF THEN ELSE statement



PSEUDOCODE

```
BEGIN
DECLARE marks : Integer
PRINT ("Enter your grade")
INPUT marks
   IF marks > 50
      THEN
            PRINT ("You have passed")
      ELSE
            PRINT (“You’ve failed”)
    END IF
END
```

FLOWCHART:



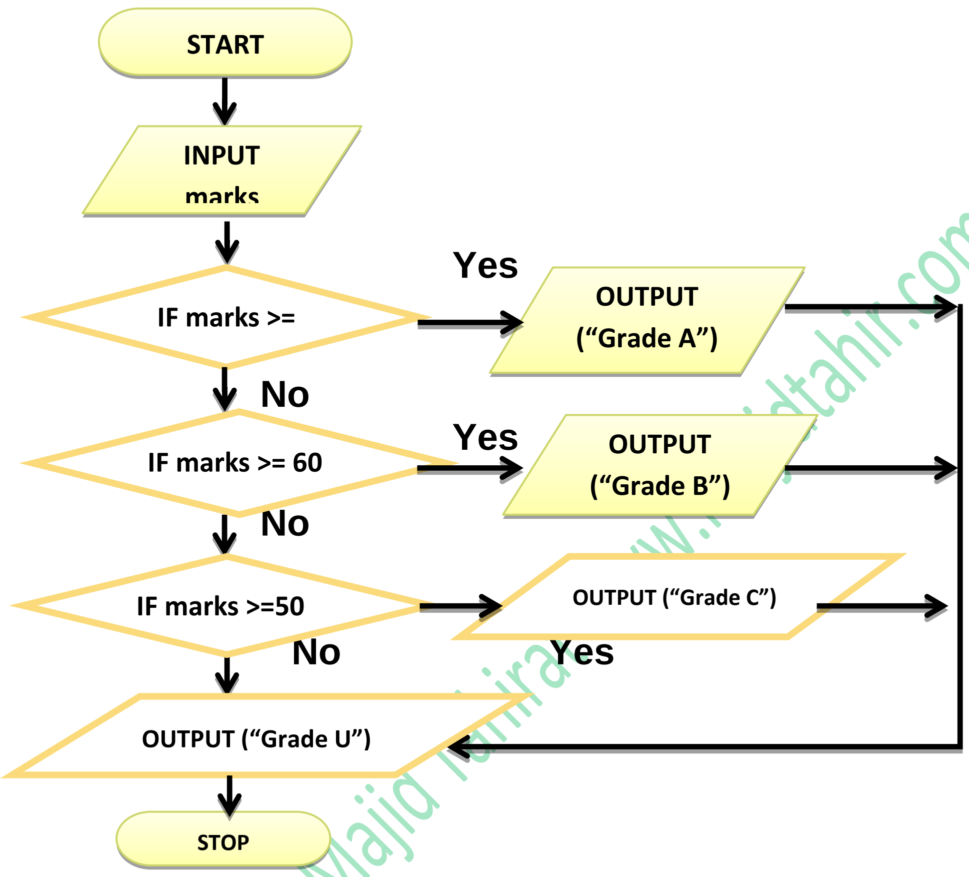
PYTHON Code:

```python
marks = int(input(" Enter your marks "))
if marks>=50:
     print("Pass")
else: print("Fail")
```

VB Code

```vb
Sub main()
   Dim marks As Integer
   Console.Writeline(“Enter marks”)
   Marks = Console.Readline()
         If marks >= 50 Then
            Console.Writeline(“pass”)
         Else
            Console.Writeline(“fail”)
         End If
End Sub
```

FLOWCHART:



IF THEN, ELSE-IF statements

```
BEGIN
DECLARE marks : INTEGER
PRINT ("Enter marks")
INPUT marks
 IF marks >= 80
  THEN PRINT ("Grade A")
  ELSE IF marks >= 70
       THEN PRINT ("Grade B")
       ELSE IF marks >= 60
              THEN PRINT ("Grade C")
              ELSE PRINT ("Grade U")
            END IF
       END IF
  END IF

END
```

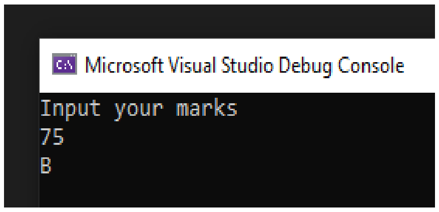
VB code example

```
Sub main()
   Dim marks As Integer
   Console.Writeline("Enter marks")
   Marks = Console.Readline()
        If marks >= 80 Then
           Console.Writeline(" A ")
        Elseif marks >= 70 Then
           Console.Writeline(" B ")
        Elseif marks >= 60 Then
           Console.Writeline(" C ")
        Else
           Console.Writeline(" U ")
        End If
End Sub
```

Python code

```
marks = int(input(" Enter your marks "))
if marks>=80:
      print("Grade A")
elif marks>=70:
      print("Grade B")
elif marks>=60:
      print("Grade C")
else:
      print("Grade U")
int("Grade U")
```

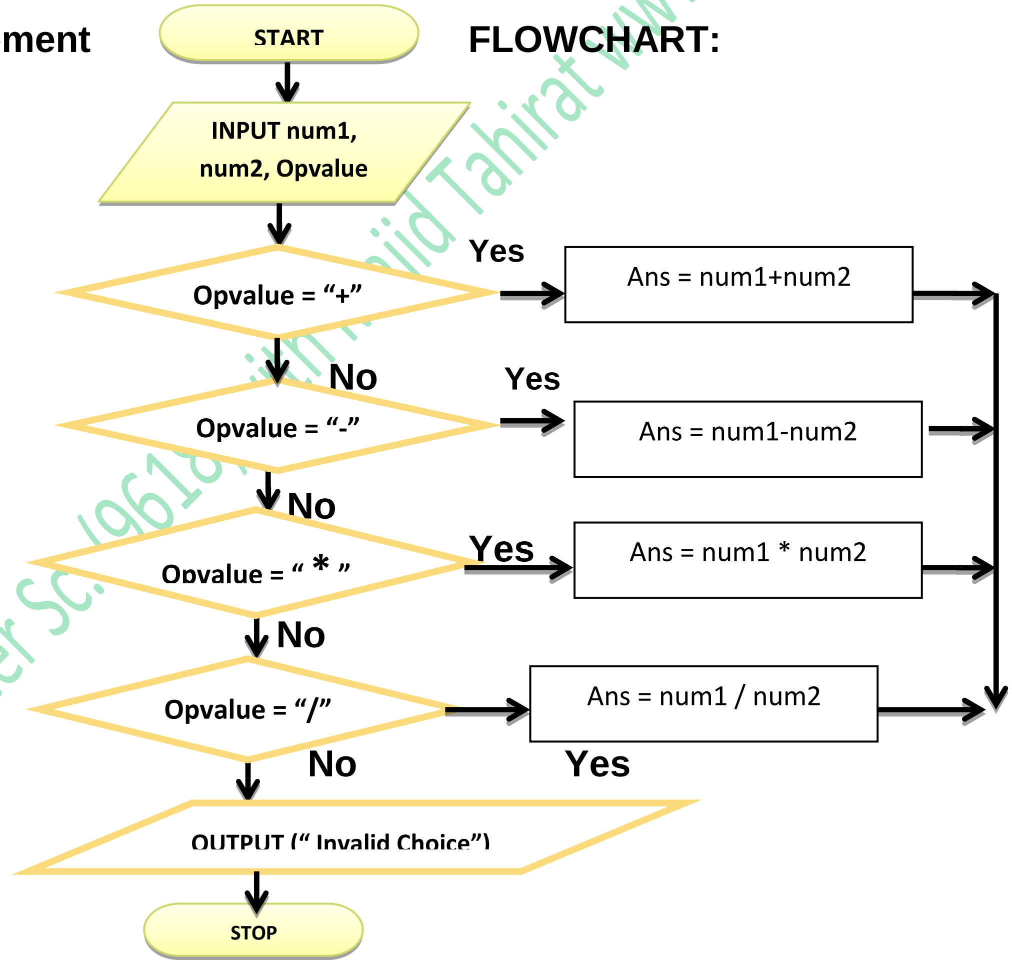
OUTPUT



```
Microsoft Visual Studio Debug Console
Input your marks
75
B
```

The IF statement is useful, but can get clumsy if you want to consider "multi-way selections

CASE Statement FLOWCHART:



CASE OF OTHERWISE… Pseudocode

```
BEGIN
DECLARE num1, num2, Ans : INTEGER
DECLARE Opvalue : CHAR
INPUT num1, num2
OUTPUT (“Enter Operator value + add, - sub, * multiply, / divison”)
INPUT Opvalue
CASE OF OpValue
      "+" : Answer ¨ Number1 + Number2
      "-" : Answer ¨ Number1 - Number2
      "*" : Answer ¨ Number1 * Number2
      "/" : Answer ¨ Number1 / Number2
OTHERWISE OUTPUT ("Please enter a valid choice")
ENDCASE
OUTPUT (“Answer is :”,Ans)
END
```

PYTHON

```
num1=int(input("Enter number 1 "))
num2=int(input("Enter number 2 "))
OpValue = ("Enter Opvalue, + is add), - is subract), * is Multiply, / is divide")
if OpValue == "+":
      Answer = num1 + num2
elif OpValue == "-":
      Answer = num1 + num2
elif OpValue == "*":
      Answer = num1 + num2
elif OpValue == "/":
      Answer = num1 + num2
else: print("invalid operator")
print("Answer is : ", Answer)
```

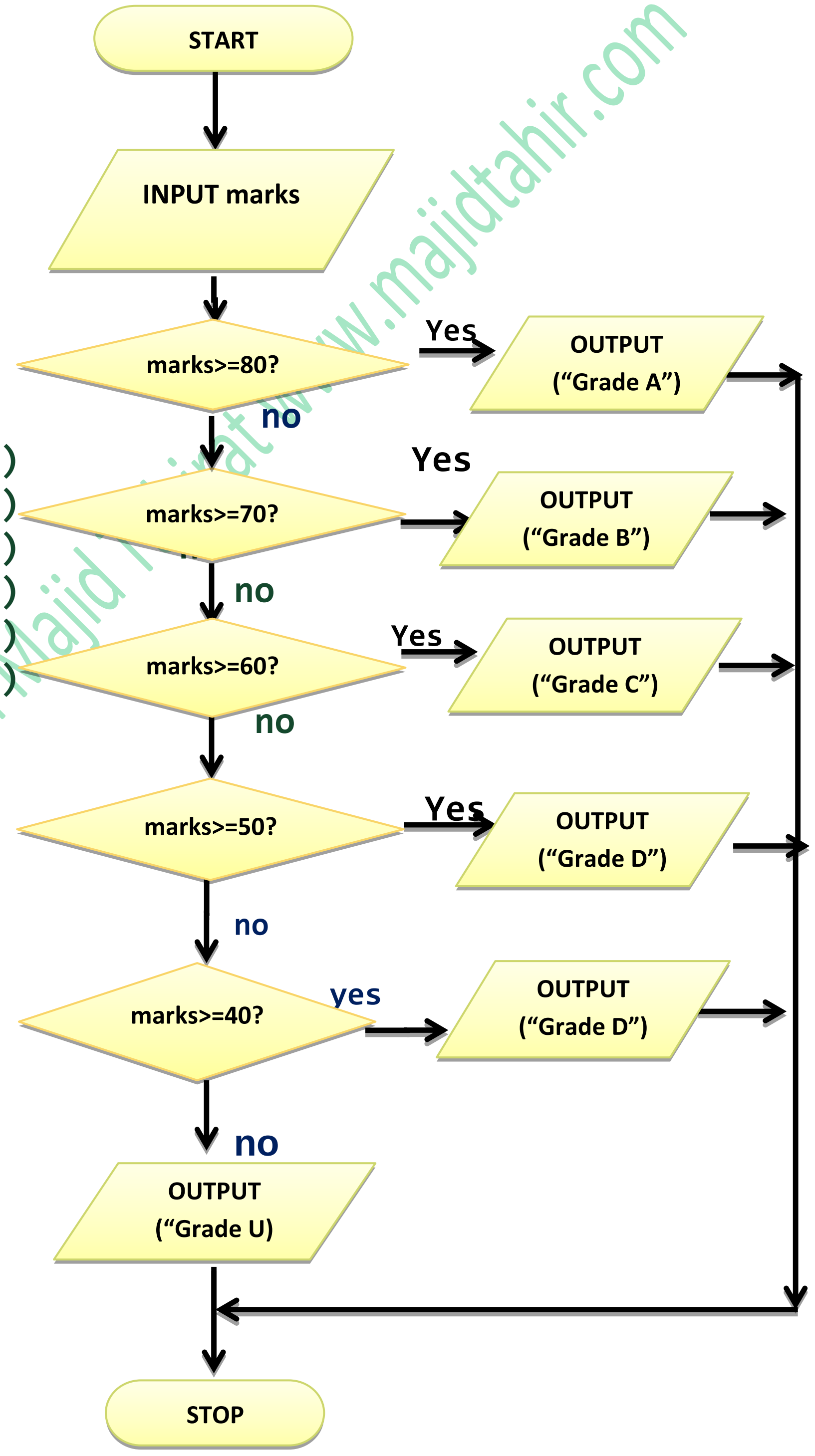
Visual Basic (Console mode) .

```
            Dim Num1, Num2, Answer As Integer
            Dim Opvalue As Char
            Console.Writeline(“INPUT num1 and num2”)
            Num1 = Console.Readline()
            Num2 = Console.Readline()
            Console.Writeline(“Enter Opvalue”)
            Console.Writeline (+ add, - sub, * multiply, / divison”)
            Opvalue = Console.Readline()
            Select CASE OpValue
                  CASE "+"
                        Answer = Number1 + Number2
                  CASE "-"
                        Answer = Number1 - Number2
                  CASE "*"
                        Answer ¨ Number1 * Number2
                  CASE "/"
                        Answer ¨ Number1 / Number2
                  CASE Else
                        Console.Writeline ("input valid choice")
      End Select
  Console.Writeline (“Answer is :” & Answer)
```

CASE OF OTHERWISE…

CASE OF OTHERWISE can also work with the Range of Value, Please see PSEUDOCODE below:

FLOWCHART

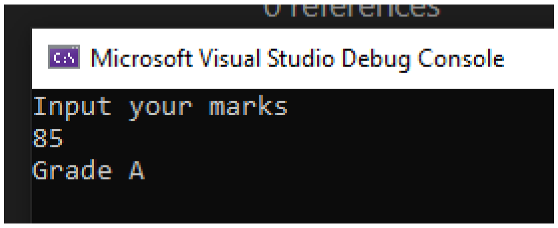


Pseudo code

```
BEGIN
DECLARE marks : Integer

PRINT ("Enter your marks")
INPUT marks
CASE OF marks
 80 TO 90 :PRINT("Grade A")
 70 TO 79 :PRINT("Grade B")
 60 TO 69 :PRINT("Grade C")
 50 TO 59 :PRINT("Grade D")
 40 TO 49 :PRINT("Grade E")
 OTHERWISE PRINT("Grade U")
 END CASE
END
```

Program Code in Visual Basic Console Mode:

```
Sub Main()
    Dim marks As Integer
    Console.WriteLine("Input your marks")
    marks = Console.ReadLine()
    Select Case marks
        Case >= 80
            Console.WriteLine("Grade A")
        Case >= 60
            Console.WriteLine("Grade B")
        Case >= 50
            Console.WriteLine("Grade C")
        Case Else
            Console.WriteLine("Grade U")
    End Select
End Sub
```

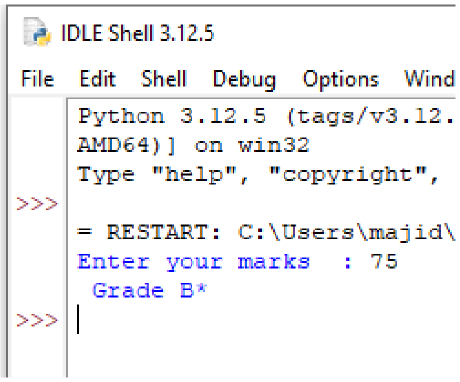


```
Input your marks
85
Grade A
```

Python does't use CASE Statements so Elif is used:

PYTHON Code

```
marks = int(input(" Enter your marks "))
 if marks>=80:
      print("Grade A")
elif marks>=60:
      print("Grade B")
elif marks>=50:
      print("Grade C")
else: print("Grade U")
```



```
    Python 3.12.5 (tags/v3.12.
    AMD64)] on win32
    Type "help", "copyright",
>>>
    = RESTART: C:\Users\majid\
    Enter your marks  : 75
     Grade B*
>>> |
```

LOOPS (Structured statements for iteration (repetition)

Many problems involve repeating one or more statements, so it is useful to have structured statements for controlling these iterations or repetitions.
Exit conditions consist of logical expressions whose truth can be tested, such as Count = 10 or Score < 0.

At a particular time, a logical expression is either **True** or **False.**
There are three type of Loops

- FOR…TO…NEXT (Count controlled Loop)
- WHILE…DO…ENDWHILE (Pre- Condition Loop)
- REPEAT…UNTIL (Post Condition Loop)

FOR … NEXT LOOP

This is to be used when loop is to be repeated a known fixed number of times.
The counter is automatically increased each time the loop is performed.

```
FOR count = 1 to 10
    INPUT number
        total = total + number
NEXT count
```

WHILE … Do LOOP

This loop is used when we don’t know how many times the loop is to be performed. The Loop is ended when a certain condition is true.
This condition is checked before starting the loop.

```
While COUNT < 10 DO
    Input NUMBER
        TOTAL = TOTAL + NUMBER
        COUNT = COUNT + 1
Endwhile
    Output TOTAL
```

REPEAT … UNTIL LOOP

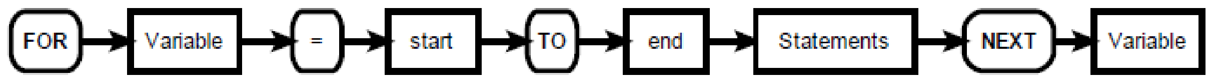
REPEAT UNTIL Loop is used when we do not know how many times loop will be performed.
The Loop is ended when a certain conation is true.
The Condition is checked at the end of the Loop and so a REPEAT Loop always has to be performed at least once.

```
REPEAT
    Input NUMBER
        TOTAL = TOTAL + NUMBER
        COUNT = COUNT + 1
Until COUNT = 10
Output Total
```

FOR Loop PSEUDOCODE

The fore loop repeats statements a set number of time.

It uses a variable to count how many time it goes round the loop and stops when it reaches its limit.

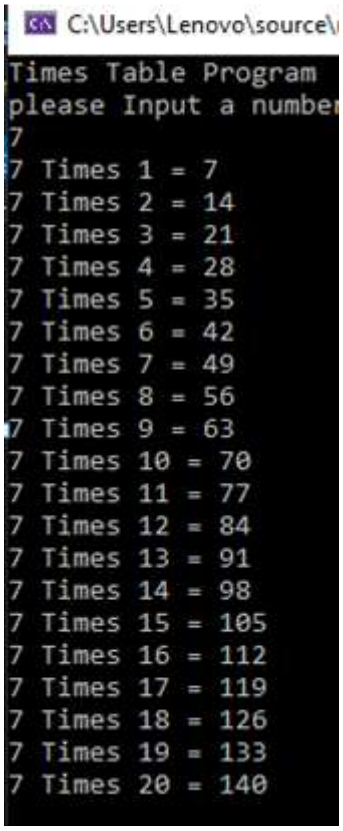


```
BEGIN
DECLARE count, number : Integer
OUTPUT (“Input a number for its times table")
INPUT number
     FOR count = 1 To 20
          PRINT (number , “times" , count , “ = ” number * Count”)
     NEXT  count
```

VB code example FOR LOOP:

```
Sub Main(args As String())
  Console.WriteLine("Times Table Program")
  Dim count, num As Integer
  Console.WriteLine("please Input a number for its TimesTable")
  num = Console.ReadLine()
   For count = 1 To 20
     Console.WriteLine(num & " Times " & count & " = " & num * count)
   Next
End Sub
```

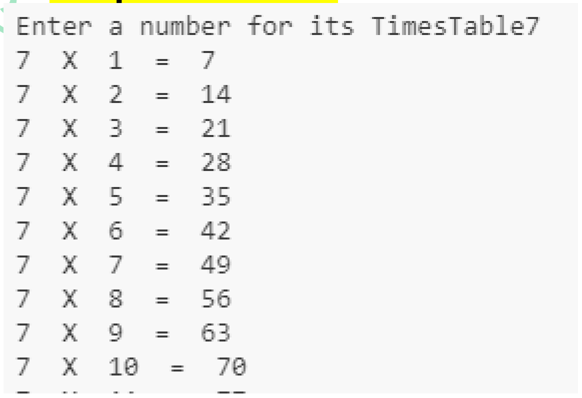
Output VB


```
C:\Users\Lenovo\source\
Times Table Program
please Input a numbe
7
7 Times 1 = 7
7 Times 2 = 14
7 Times 3 = 21
7 Times 4 = 28
7 Times 5 = 35
7 Times 6 = 42
7 Times 7 = 49
7 Times 8 = 56
7 Times 9 = 63
7 Times 10 = 70
7 Times 11 = 77
7 Times 12 = 84
7 Times 13 = 91
7 Times 14 = 98
7 Times 15 = 105
7 Times 16 = 112
7 Times 17 = 119
7 Times 18 = 126
7 Times 19 = 133
7 Times 20 = 140
```

PYTHON Code FOR LOOP

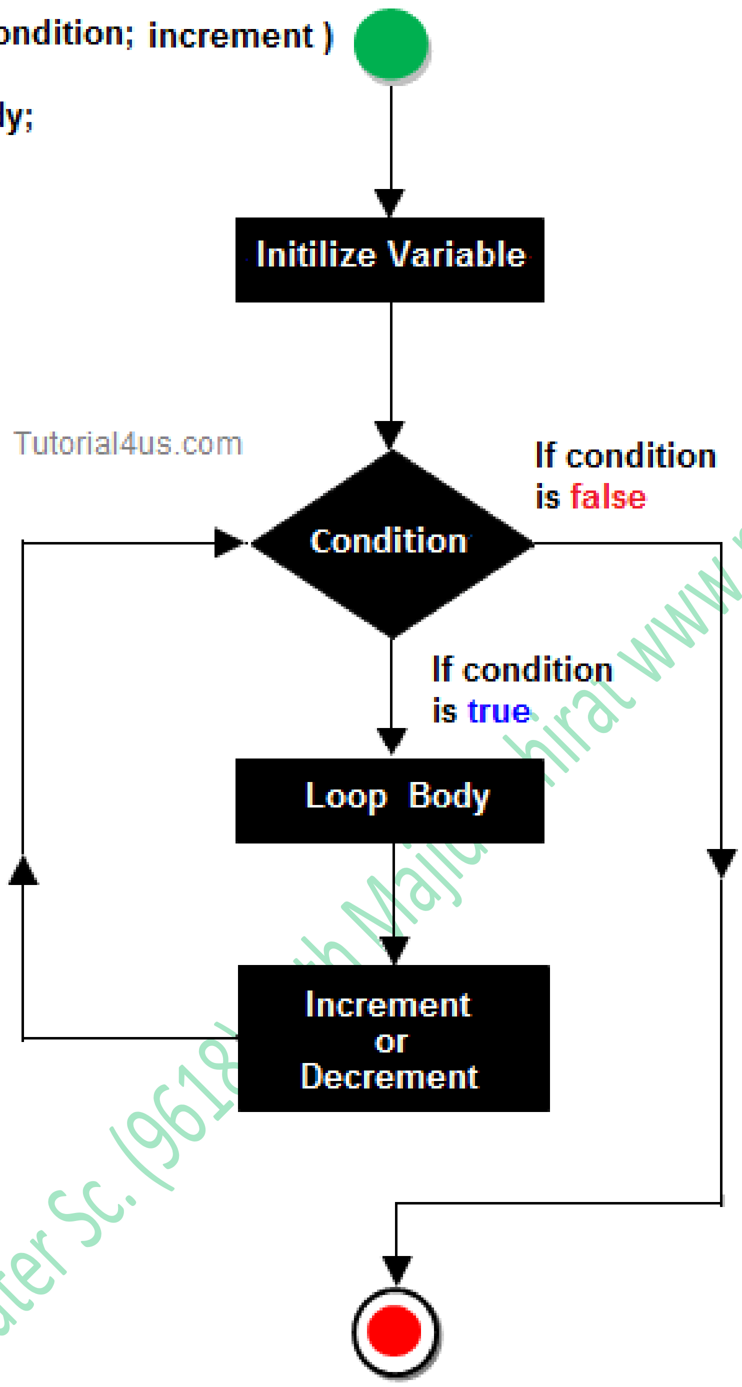
```
print(" Times Table Program ")
num = int(input("Enter a number for its TimesTable"))
for count in range(1,10):
     print(num, " X ", count, " = ", num*count)
```

Output PYTHON


```
Enter a number for its TimesTable7
7  X  1  =  7
7  X  2  =  14
7  X  3  =  21
7  X  4  =  28
7  X  5  =  35
7  X  6  =  42
7  X  7  =  49
7  X  8  =  56
7  X  9  =  63
7  X  10  =  70
```

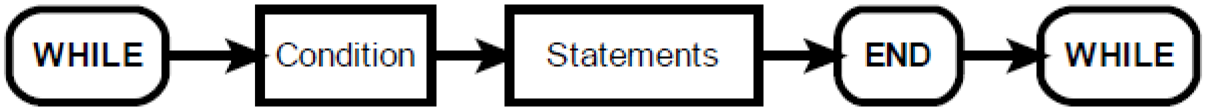
FLOWCHART FOR LOOP

```
for( init; condition; increment )
{
  loop body;
}
```



WHILE DO ENDWHILE loop

The wile loop is known as a **test before loop**. The condition is tested before entering the loop, but tested each time it goes round the loop. The number of times the statements within the loop are executed varies. The test before loop goes round 0 or more times.
This method is useful when processing files and using "read ahead" data



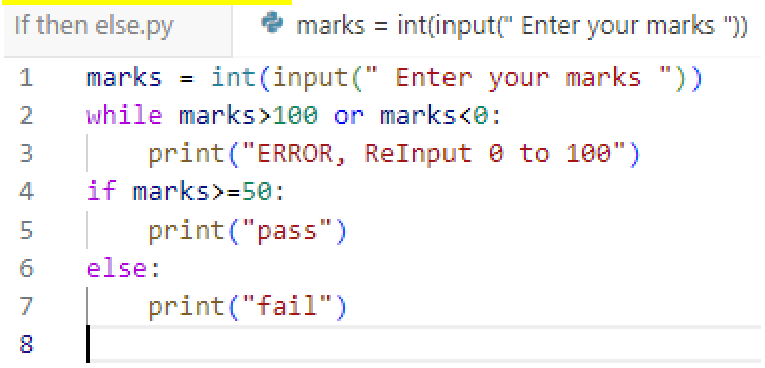
VB Code example

```
Sub main()
Dim marks As Integer
Console.Writeline("Enter marks")
marks = Console.Readline()
While marks > 100 OR marks < 0
   Console.Writeline("REINPUT 0 to 100")
   marks = Console.Readline()
End While
     if marks >= 50 Then
       Console.Writeline(" Pass ")
     Else
       Console.Writeline(" Fail ")
   End If
End Sub
```

PSEUDOCODE

```
BEGIN
DECLARE marks : REAL
INPUT marks
     WHILE marks > 100 OR < 0
        PRINT ("ERROR, RE-Input ")
        INPUT marks
     END WHILE
IF marks>=50
     THEN
          OUTPUT ("Pass")
     ELSE
          OUTPUT ("Fail")
     END IF
END
```

PYTHON Code .

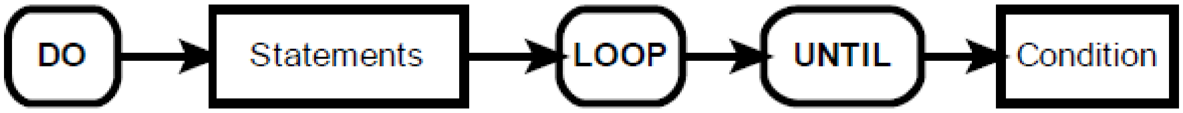


If then else.py   marks = int(input(" Enter your marks "))

```
marks = int(input(" Enter your marks "))
while marks>100 or marks<0:
    print("ERROR, ReInput 0 to 100")
if marks>=50:
    print("pass")
else:
    print("fail")
```

REPEAT UNTIL loop

The repeat loop is similar to the while loop, but it tests the condition after the statements have been executed once. This means that this test after loop goes round 1 or more times.



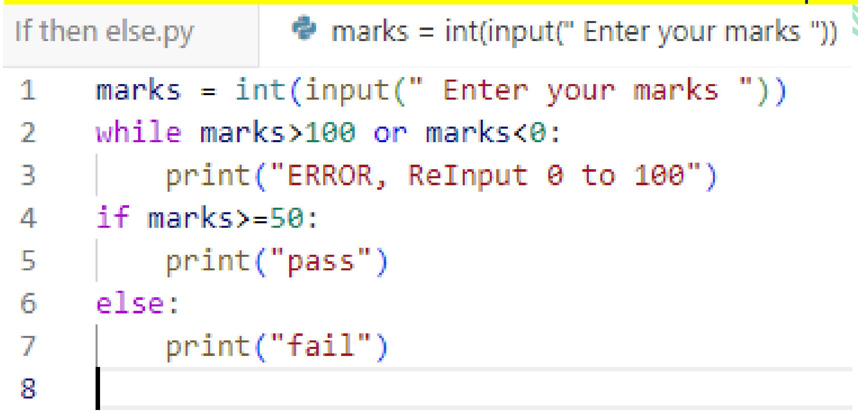
VB Code

```
Sub main()
Dim marks As Integer
Do
Console.Writeline("Enter marks 0 to100")
Marks = Console.Readline()
Loop Until marks>=100 AND <=100
     if marks >= 50 Then
       Console.Writeline(" Pass ")
     Else
       Console.Writeline(" Fail ")
    End If
End Sub
```

PSEUDOCODE

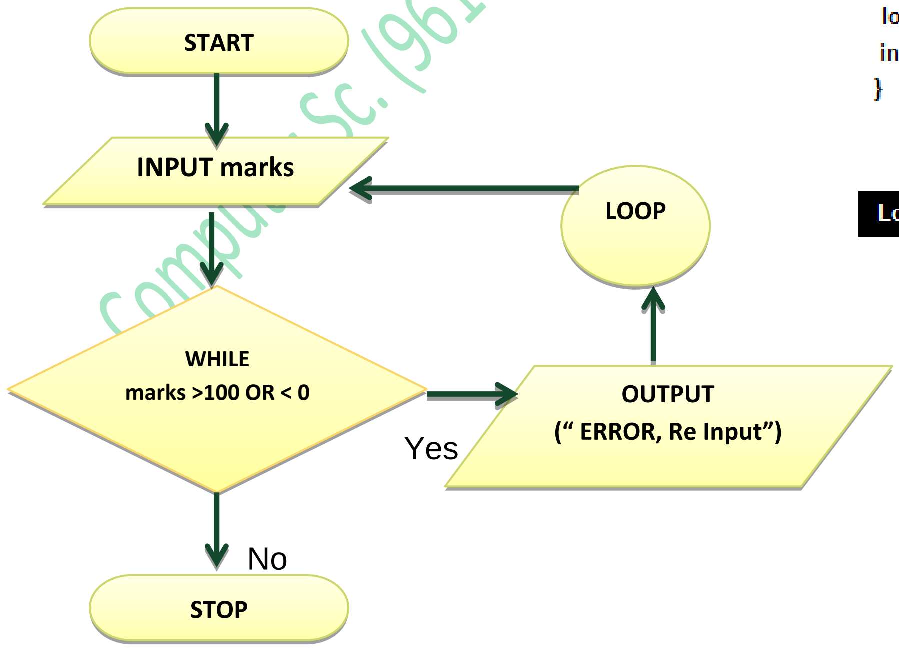
```
BEGIN
DECLARE name: STRING
 REPEAT
    PRINT ("Enter marks 0 to 100")
    INPUT marks
 UNTIL marks>=0 AND marks <=100

 IF marks>=50
   THEN
        OUTPUT("Pass")
   ELSE
        OUTPUT("Fail")
  END IF
END
```

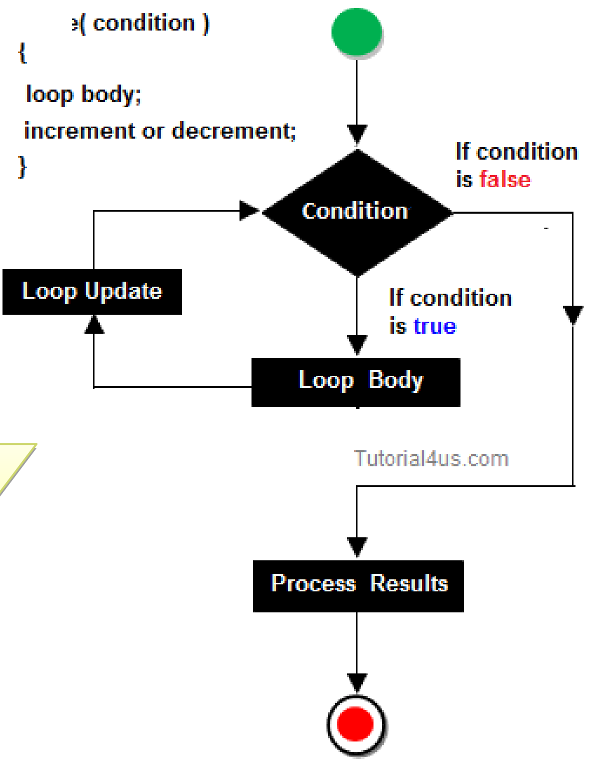
PYTHON Does not have REPEAT LOOP so WHILE Loop is used.


If then else.py   marks = int(input(" Enter your marks "))

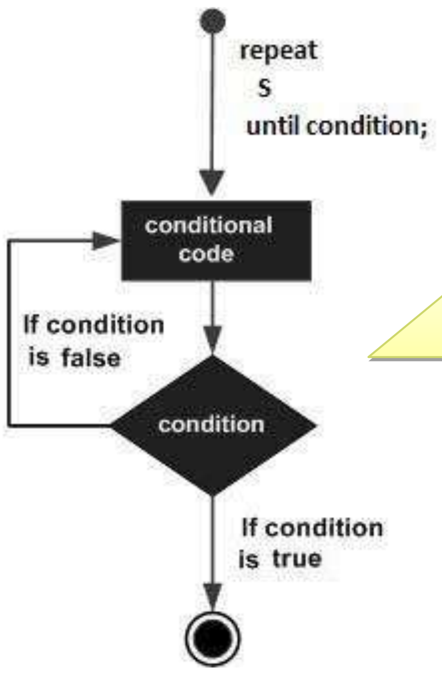
```
marks = int(input(" Enter your marks "))
while marks>100 or marks<0:
    print("ERROR, ReInput 0 to 100")
if marks>=50:
    print("pass")
else:
    print("fail")

```

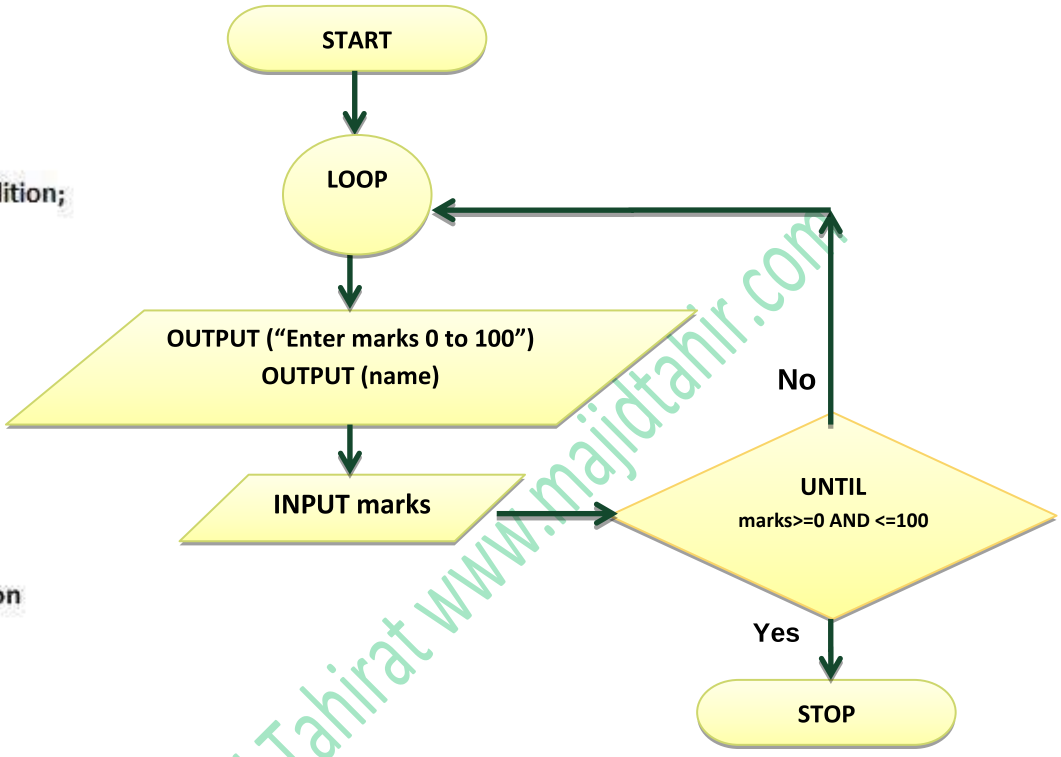
FLOWCHART…WHILE-ENDWHILE




```
( condition )
{
 loop body;
 increment or decrement;
}
```

FLOWCHART…REPEAT-UNTIL





10.2 Array Data Type

An array is a special variable that has one name, but can store multiple values. Each value is stored in an element pointed to by an index.
The first element in the array has index value 0 or 1depending on programming language syntax. In PSEUDOCODES we use 1 as first index.

One Dimensional Arrays

A one dimensional array can be thought as a list. An array with 10 elements, called names, can store 10 names and could be visualized as this: **Lower bound of ARRAY can start from 1 or 0**

Index	Name
1	Fred
2	James
3	Tom
4	Robert
5	Jonah
6	Chris
7	John
8	Matthew
9	Mikey
10	Jack

index	Element
0	Fred
1	James
2	Tom
3	Robert
4	Jonah
5	Chris
6	Jon
7	Matthew
8	Mikey
9	Jack

Arrays (One-dimensional arrays)

In order to use a one-dimensional array in a computer program, you need to consider:

- What the array is going to be used for, so it can be given a meaningful name
- How many items are going to be stored, so the size of the array can be determined.
- What sort of data is to be stored, so that the array can be the appropriate data type.

DECLARATION on Blank Array with 10 slots can be done like this:

Pseudocode:

```
DECLARE names[1:10] ARRAY OF STRING
```

VB code example:

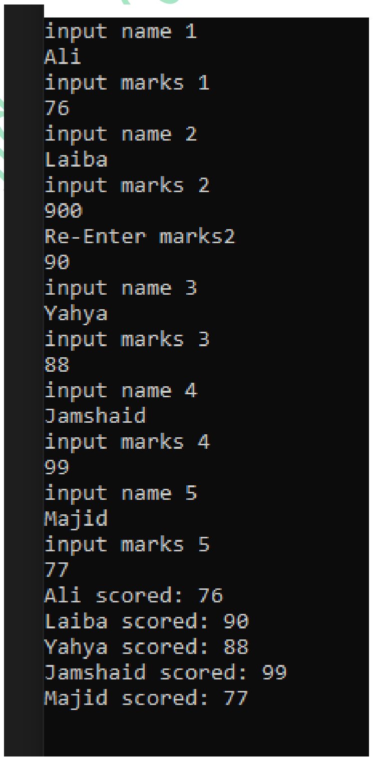
```
Dim names(9) As String
```

PYTHON Code

```
name[]
```

Entering Values in One-Dimension Array

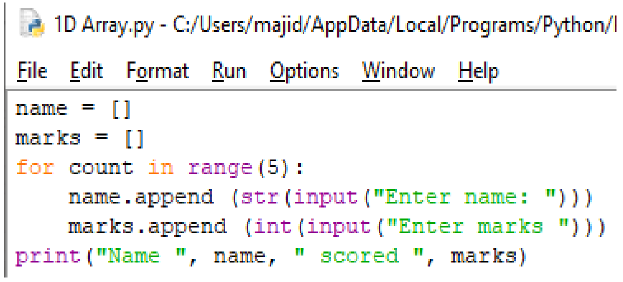
```
BEGIN
DECLARE count : Integer
DECLARE name [1:5] ARRAY OF STRING  // for5 elements in ARRAY
DECLARE marks [1:5] ARRAY OF Integer
   FOR count = 1 to 5 // for inputting 5 names and grades
      PRINT ("Enter Name" , count)
      INPUT name [count]
      PRINT ("Enter grade for " , name(count))
      INPUT marks [count]
   NEXT count
  FOR count 1 to 5 // for displaying 5 names and grades
      PRINT (name (count) , "has scored" , marks(count))
  NEXT count
END
```



```
input name 1
Ali
input marks 1
76
input name 2
Laiba
input marks 2
900
Re-Enter marks2
90
input name 3
Yahya
input marks 3
88
input name 4
Jamshaid
input marks 4
99
input name 5
Majid
input marks 5
77
Ali scored: 76
Laiba scored: 90
Yahya scored: 88
Jamshaid scored: 99
Majid scored: 77
```

PYTHON Code:

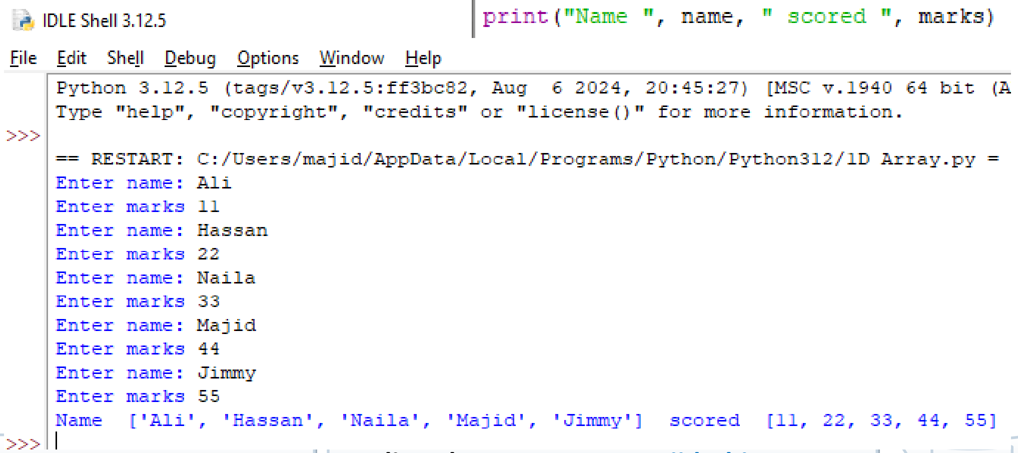
```
name = []
marks = []
for count in range(5):
    name.append (str(input("Enter name: ")))
    marks.append (int(input("Enter marks ")))
print("Name ", name, " scored ", marks)
```



1D Array.py - C:/Users/majid/AppData/Local/Programs/Python/I

```
name = []
marks = []
for count in range(5):
    name.append (str(input("Enter name: ")))
    marks.append (int(input("Enter marks ")))
print("Name ", name, " scored ", marks)
```

OUTPUT screen



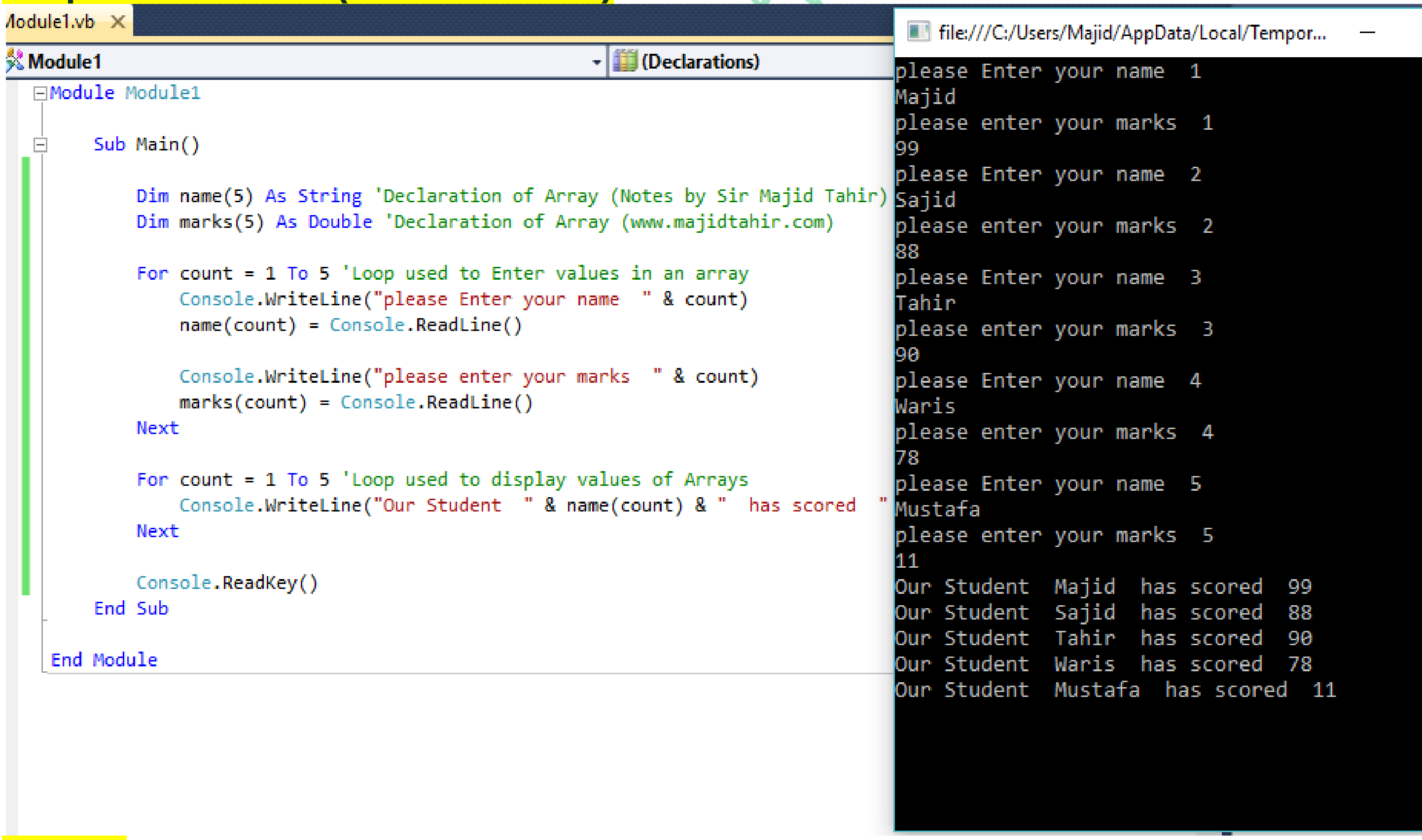
IDLE Shell 3.12.5

```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug  6 2024, 20:45:27) [MSC v.1940 64 bit (A
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/majid/AppData/Local/Programs/Python/Python312/1D Array.py =
Enter name: Ali
Enter marks 11
Enter name: Hassan
Enter marks 22
Enter name: Naila
Enter marks 33
Enter name: Majid
Enter marks 44
Enter name: Jimmy
Enter marks 55
Name  ['Ali', 'Hassan', 'Naila', 'Majid', 'Jimmy']  scored  [11, 22, 33, 44, 55]
>>>
```

VB Code in Console Mode

```
Dim name(5) As String
Dim marks(5) As Integer
     For count = 1 To 5
          Console.WriteLine("input name " & count)
          name(count) = Console.ReadLine()
          Console.WriteLine("input marks " & count)
          marks(count) = Console.ReadLine()
            While marks(count) > 100 Or marks(count) < 0
                 Console.WriteLine("Re-Enter marks" & count)
                  marks(count) = Console.ReadLine()
            End While
    Next

     For count = 1 To 5
          Console.WriteLine(name(count) & " scored: " & marks(count))
    Next
```

Output of VB code (Console mode)



```
Module Module1

    Sub Main()

        Dim name(5) As String 'Declaration of Array (Notes by Sir Majid Tahir)
        Dim marks(5) As Double 'Declaration of Array (www.majidtahir.com)

        For count = 1 To 5 'Loop used to Enter values in an array
            Console.WriteLine("please Enter your name  " & count)
            name(count) = Console.ReadLine()

            Console.WriteLine("please enter your marks  " & count)
            marks(count) = Console.ReadLine()
        Next

        For count = 1 To 5 'Loop used to display values of Arrays
            Console.WriteLine("Our Student  " & name(count) & "  has scored  "
        Next

        Console.ReadKey()
    End Sub

End Module
```

```
please Enter your name  1
Majid
please enter your marks  1
99
please Enter your name  2
Sajid
please enter your marks  2
88
please Enter your name  3
Tahir
please enter your marks  3
90
please Enter your name  4
Waris
please enter your marks  4
78
please Enter your name  5
Mustafa
please enter your marks  5
11
Our Student  Majid  has scored  99
Our Student  Sajid  has scored  88
Our Student  Tahir  has scored  90
Our Student  Waris  has scored  78
Our Student  Mustafa  has scored  11
```

Python One-dimensional array with values in it.

```
num = [1, 2, 3]
num.append(4)
num.extend([5, 6])
print(num)  # will print this in output [1, 2, 3, 4, 5, 6]
```

Another example of One-Dimensional Array

```
Module Module1
    Sub Main()
        Dim count As Integer
        Dim name(4) As String
        Dim marks(4) As Integer
        Dim gender(4) As String
            For count = 0 To 4
            Console.WriteLine("please enter your name" & count)
            name(count) = Console.ReadLine()
            Console.WriteLine("please enter your gender" & count)
            gender(count) = Console.ReadLine()
            Console.WriteLine("please enter your marks" & count)
            marks(count) = Console.ReadLine()
            Next count
        For count = 0 To 4
            Console.WriteLine("your name is : " & name(count))
            Console.WriteLine("your gender is : " & gender(count))
            Console.WriteLine("your marks are : " & marks(count))
        Next count
        Console.ReadKey()
    End Sub
End Module
```

Multi-Dimensional Arrays or Two dimensional Arrays (2D Array):

A multi-dimensional array can be thought of as a table, each element has a row and column index. Following example declares a two-dimensional array called `table with 3 rows and 4 colums` and would be declared in **PseudoCode** as follows:

```
DECLARE table[1:3, 1:4] : ARRAY OF INTEGER
```

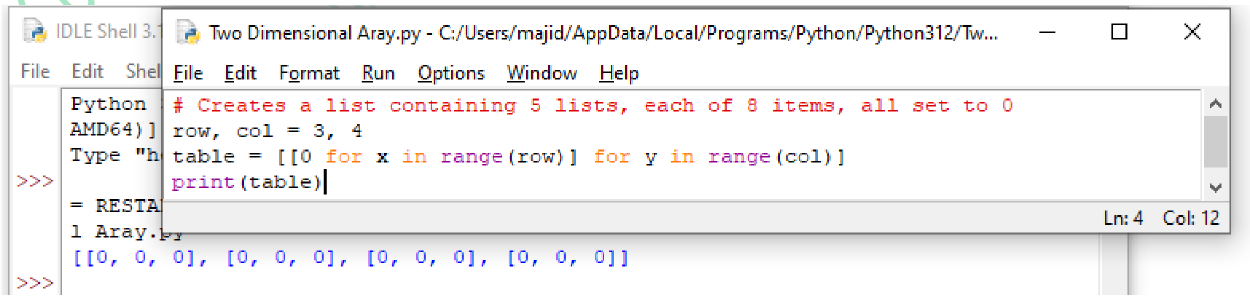
Visual Basic(Console mode)

```
Dim table(3, 4) : As Integer
```

Python Code

```
row, col = 3, 4
table = [[0 for x in range(row)] for y in range(col)]
```

Creates a list containing 5 lists, each of 8 items, all set to 0



```
# Creates a list containing 5 lists, each of 8 items, all set to 0
row, col = 3, 4
table = [[0 for x in range(row)] for y in range(col)]
print(table)
```

```
Python
AMD64)]
Type "h
>>>
= RESTA
l Aray.py
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
```

PSEUDOCODE Example of Two-Dimension Array

BEGIN

11	12	13	14
21	22	23	24
31	32	33	34

```
DECLARE table[1:3, 1:4] : ARRAY OF INTEGER
  DECLARE row, col : INTEGER
    FOR row = 1 TO 3  //NESTED FOR LOOP for values in 2D ARRAY
       FOR col = 1 TO 4
        PRINT("Please Input Value in Row:",row, "column : ", col)
        INPUT table(row, col)
       NEXT
    NEXT
   FOR row = 1 To 3
      FOR col = 1 To 4
       PRINT ("Row = " & row & "column = " & col & “has Value”)
       PRINT (table(row, col))
       NEXT
   NEXT
END
```

VB Code Example of Two-Dimension Array

```
Sub Main()
  Dim table(2, 3) As Integer
    For row = 0 To 2
       For col = 0 To 3
        Console.WriteLine("Please Input Value in Row: " & row & "col:" & col)
        table(row, col) = Console.ReadLine()
       Next
    Next
 Console.Clear()

   For row = 0 To 2
      For col = 0 To 3
       Console.WriteLine("Row = " & row & "column = " & col & “has Value”)
       Console.WriteLine(table(row, col))
       Next
   Next
Console.ReadKey()
End Sub
```

Multi-Dimensional Arrays:

A multi-dimensional array can be thought of as a table, each element has a row and column index.
Following example declares a two-dimensional array called matrix and would be declared b

```
    Dim matrix(2,3) As Integer
```

Usually we refer to the first dimension as being the rows, and the second dimension as being the columns.

index	0	1	2	3
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L

The following statements would generate the following

```
    Console.WriteLine(matrix(0, 0))
```

Would display A

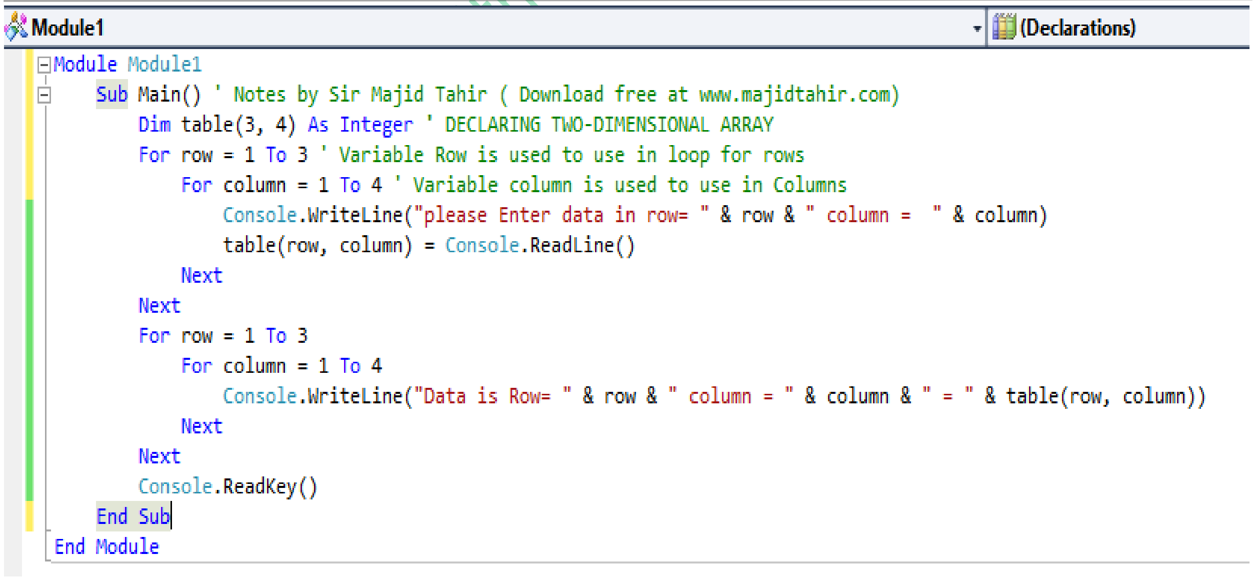
```
    Console.WriteLine(matrix(2, 1))
```

Would display J

```
    Console.WriteLine("first row, first column   : " & matrix(2, 3))
```

Would display `first row, first column   :  L`

VB Code for 2-D Array is:



```
Module Module1
    Sub Main() ' Notes by Sir Majid Tahir ( Download free at www.majidtahir.com)
        Dim table(3, 4) As Integer ' DECLARING TWO-DIMENSIONAL ARRAY
        For row = 1 To 3 ' Variable Row is used to use in loop for rows
            For column = 1 To 4 ' Variable column is used to use in Columns
                Console.WriteLine("please Enter data in row= " & row & " column =  " & column)
                table(row, column) = Console.ReadLine()
            Next
        Next
        For row = 1 To 3
            For column = 1 To 4
                Console.WriteLine("Data is Row= " & row & " column = " & column & " = " & table(row, column))
            Next
        Next
        Console.ReadKey()
    End Sub
End Module
```

Refrences:

- **Computer Science by David Watson & Helen Williams**
- **Visual Basic Console Cook Book**
- **Computer Science AS and A level by Sylvia Langfield and Dave Duddell**
- **https://www.sitesbay.com/javascript/javascript-looping-statement**
- **http://wiki.jikexueyuan.com/project/lua/if-else-if-statement.html**