

ST. JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -
AUTONOMOUS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALGORITHMIC THINKING WITH PYTHON

Prof. Sarju S

15 October 2024

Module 2

Module 2



- ▶ **ALGORITHM AND PSEUDOCODE REPRESENTATION:-** Meaning and Definition of Pseudocode, Reasons for using pseudocode, The main constructs of pseudocode - Sequencing, selection (if-else structure, case structure) and repetition (for, while, repeat-until loops), Sample problems
- ▶ **FLOWCHARTS :-** Symbols used in creating a Flowchart - start and end, arithmetic calculations, input/output operation, decision (selection), module name (call), for loop (Hexagon), flow-lines, on-page connector, off-page connector.

Algorithms and pseudocodes

Algorithms and pseudocodes

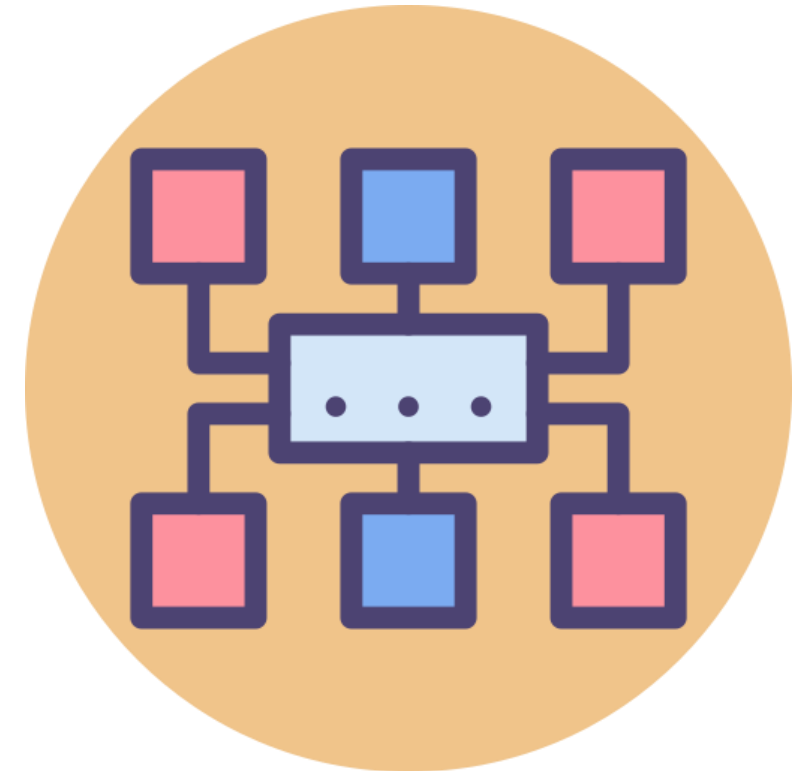


▶ Algorithm

- ▶ A step-by-step procedure for solving a problem.
- ▶ Uses pure English phrases or sentences.

▶ Pseudocode

- ▶ A high-level representation of an algorithm.
- ▶ Uses a mix of natural language and programming-like syntax.
- ▶ More structured than algorithms.



Pseudocodes



- Confused between algorithms and pseudocodes? Let us take an example. We will now write an algorithm and a pseudocode to evaluate an expression, say $d = a + b * c$.

EVALUATE-ALGO

- 1 Start
- 2 Read the values of a , b and c .
- 3 Find the product of b and c .
- 4 Store the product in a temporary variable $temp$.
- 5 Find the sum of a and $temp$.
- 6 Store the sum in d .
- 7 Print the value of d .
- 8 Stop.

Algorithm

EVALUATE-PSEUDO

- 1 Start
- 2 READ(a, b, c)
- 3 $d = a + b * c$
- 4 PRINT(d)
- 5 Stop

Pseudocode

Why pseudocodes?



Wondering why pseudocodes are important? Here are a few motivating reasons:

- ▶ **Ease of understanding:** Since the pseudocode is programming language independent, novice developers can also understand it very easily.
- ▶ **Focus on logic:** A pseudocode allows you to focus on the algorithm's logic without bothering about the syntax of a specific programming language.
- ▶ **More legible:** Combining programming constructs with English phrases makes pseudocode more legible and conveys the logic precisely.
- ▶ **Consistent:** As the constructs used in pseudocode are standardized, it is useful in sharing ideas among developers from various domains.
- ▶ **Easy translation to a program:** Using programming constructs makes mapping the pseudocode to a program straightforward.
- ▶ **Identification of flaws:** A pseudocode helps identify flaws in the solution logic before implementation.

Difference between Algorithm and Pseudocode



Criteria	Algorithm	Pseudocode
Level of Detail	More abstract and language-agnostic.	More structured and resembles a programming language.
Readability	Often less readable due to abstraction.	More readable as it mimics coding structures.
Execution	Can be directly implemented in a programming language.	Not directly executable. It requires multiple transitions to be used in the programming language.
Purpose	Defines the logic and steps of a solution.	An intermediary step for planning before coding.

Constructs of a pseudocode



Constructs of a pseudocode

- ▶ A good pseudocode should follow the structured programming approach.
- ▶ Structured coding aims to improve the readability of pseudocode by ensuring that the execution sequence follows the order in which the code is written.
- ▶ The main constructs are
 - ▶ Sequencing
 - ▶ Selection
 - ▶ Repetition (loop)

Constructs of a pseudocode - Sequence

- ▶ This is the most elementary construct where the instructions of the algorithm are executed in the order listed.
- ▶ It is the logical equivalent of a straight line.
- ▶ Consider the code below.

S1	▶ The statement S1 is executed first, which is then
S2	followed by statement S2, so on and so forth, Sn until all
S3	the instructions are executed.
·	
·	▶ No instruction is skipped and every instruction is
Sn	executed only once.



Constructs of a pseudocode - Sequence

- ▶ Example: Write a pseudocode to find the area of a square

1. Start
2. Read the side length of the square
3. area: `area = side_length * side_length`
4. Print the area
5. End

- ▶ All the statements in the given pseudocode will execute in sequential order, without skipping any line.



Constructs of a pseudocode - Decision or Selection

- ▶ A selection structure consists of a test condition together with one or more blocks of statements.
- ▶ The result of the test determines which of these blocks is executed.
- ▶ There are mainly two types of selection structures
- ▶ **if** structure
 - ▶ Simple **if** structure
 - ▶ **If...else** structure
 - ▶ **if ..else if.. else** structure
- ▶ Case Structure

Constructs of a pseudocode - Decision or Selection



- ▶ Simple If
- ▶ The general form of this structure is

```
if (condition)  
    TRUE__INSTRUCTIONS  
endif
```

```
CHECKPOSITIVE(x)  
1  if ( $x > 0$ )  
2      PRINT(x, “ is positive”)  
3  endif
```

The pseudocode CheckPositive(*x*) checks if an input value *x* is positive.

Constructs of a pseudocode - Decision or Selection

- ▶ if .. else structure
- ▶ The general form of this structure is

```
if (condition)  
    TRUE_INSTRUCTIONS  
else  
    FALSE_INSTRUCTIONS  
endif
```

- ▶ This structure contains two blocks of statements.
- ▶ If the test condition is met, the first block (denoted by true_instructions) is executed and the algorithm skips over the second block (denoted by false_instructions).
- ▶ If the test condition is not met, the first block is skipped and only the second block is executed.

```
PERSONTYPE(age)  
1  if (age >= 18)  
2      PRINT("You are a major")  
3  else  
4      PRINT("You are a minor")  
5  endif
```

The pseudocode PersonType(*age*) checks if a person is a major or not.

Constructs of a pseudocode - Decision or Selection

- ▶ if ..else if .. else structure
- ▶ When a selection is to be made out of a set of more than two possibilities, you need to use the if else if else structure, whose general form is given below:
 - ▶ Here, if condition1 is met, TRUE_INSTRUCTIONS1 will be executed.
 - ▶ Else condition2 is checked. If it evaluates to True, TRUE_INSTRUCTIONS2 will be selected.
 - ▶ There is no limit to the number of **else if** statements, but in the end, there has to be an **else** statement.
 - ▶ The conditions are tested one by one starting from the top, proceeding downwards.
 - ▶ Once a condition is evaluated to be True, the corresponding block is executed, and the rest of the structure is skipped.
 - ▶ **If none of the conditions are met, the final else part is executed.**

```
if (condition1)  
    TRUE_INSTRUCTIONS1  
else if (condition2)  
    TRUE_INSTRUCTIONS2  
else  
    FALSE_INSTRUCTIONS  
endif
```




Constructs of a pseudocode - Decision or Selection

- ▶ if ..else if .. else structure
- ▶ Example

```
COMPAREVARS( $x, y$ )  
1  if ( $x > y$ )  
2      PRINT( $x$ , "is greater than",  $y$ )  
3  else if ( $x < y$ )  
4      PRINT( $y$ , "is greater than",  $x$ )  
5  else  
6      PRINT("The two values are equal")  
7  endif
```

The pseudocode CompareVars(x, y) compares two variables x and y and prints the relation between them.

Constructs of a pseudocode - Case Structure

- ▶ The case structure is a refined alternative to if else if else structure.
- ▶ The pseudocode representation of the case structure is given below.

```
caseof (expression)  
case 1 value1:  
    BLOCK1  
case 2 value2:  
    BLOCK2  
    ⋮  
default :  
    DEFAULT_BLOCK  
endcase
```

- ▶ First, the value of an expression (which could also be a single variable) is compared to a predefined value1.
- ▶ If a match is found, the first block of code, referred to as block1, is executed.
- ▶ Typically, each block ends with a break statement, which exits the case structure.

Constructs of a pseudocode - Case Structure

- ▶ The case structure is a refined alternative to if else if else structure.
- ▶ The pseudocode representation of the case structure is given below.

```
caseof (expression)  
  case 1 value1:  
    BLOCK1  
  case 2 value2:  
    BLOCK2  
    ⋮  
  default :  
    DEFAULT__BLOCK  
endcase
```

- ▶ If there is no match with value₁, the expression or variable is then compared with value₂.
- ▶ If this comparison results in a match, block₂ is executed, and the case structure is exited with the corresponding break statement.

Constructs of a pseudocode - Case Structure

- ▶ The case structure is a refined alternative to if else if else structure.
- ▶ The pseudocode representation of the case structure is given below.

```
caseof (expression)  
  case 1 value1:  
    BLOCK1  
  case 2 value2:  
    BLOCK2  
    ⋮  
  default :  
    DEFAULT_BLOCK  
endcase
```

- ▶ If a break statement is omitted from a block, the program will continue executing the subsequent blocks, regardless of whether the subsequent cases match, until a break is encountered or the end of the case structure is reached.

Constructs of a pseudocode - Case Structure

- ▶ The case structure is a refined alternative to if else if else structure.
- ▶ The pseudocode representation of the case structure is given below.

```
caseof (expression)  
case 1 value1:  
    BLOCK1  
case 2 value2:  
    BLOCK2  
    ⋮  
default :  
    DEFAULT_BLOCK  
endcase
```

- ▶ This comparison process continues for each case until either a match is found, or all cases are exhausted.
- ▶ If no matches are found for any of the cases, the default_block will be executed.
- ▶ This block handles cases where the expression doesn't match any predefined values.

Constructs of a pseudocode - Case Structure



► Example

PRINTDIRECTION(*dir*)

```
1  caseof (dir)
2      case 'N':
3          PRINT("North")
4          break
5      case 'S':
6          PRINT("South")
7          break
8      case 'E':
9          PRINT("East")
10         break
11     case 'W':
12         PRINT("West")
13         break
14     default :
15         PRINT("Invalid direction code")
16 endcase
```

The pseudocode PRINTDIRECTION(*dir*) prints the direction name based on the value of a character called *dir*.

Exercises

Exercises



- **Problem Statement:** Write a pseudocode that checks if a number is greater than 100. If it is, print "Number is greater than 100."

```
BEGIN
    PRINT "Enter a number:"
    INPUT number
    IF number > 100 THEN
        PRINT "Number is greater than 100."
    END IF
END
```


Exercises



- **Problem Statement:** Write a pseudocode to check if a person is eligible for a driving license. The minimum age requirement is 16 years. If the person is 16 or older, print "Eligible for a driving license." Otherwise, print "Not eligible for a driving license."

```
BEGIN
    PRINT "Enter your age:"
    INPUT age
    IF age >= 16 THEN
        PRINT "Eligible for a driving license."
    ELSE
        PRINT "Not eligible for a driving license."
    END IF
END
```

Exercises



- **Problem Statement:** Write a pseudocode to check whether a number is divisible by 5. If it is, print "Divisible by 5." Otherwise, print "Not divisible by 5."

```
BEGIN
    PRINT "Enter a number:"
    INPUT number
    IF number MOD 5 = 0 THEN
        PRINT "Divisible by 5."
    ELSE
        PRINT "Not divisible by 5."
    END IF
END
```

Exercises



- ▶ **Problem Statement:** A customer visits an online store that offers a discount based on the total purchase amount:
 - ▶ If the total is greater than ₹500, apply a 20% discount.
 - ▶ If the total is between ₹ 200 and ₹ 500 (inclusive), apply a 10% discount.
 - ▶ If the total is less than ₹ 200, no discount is applied.

Write a pseudocode that calculates the final amount after applying the discount.

Exercises



```
BEGIN
  PRINT "Enter the total purchase amount:"
  INPUT total_amount

  IF total_amount > 500 THEN
    discount = total_amount * 0.20
    final_amount = total_amount - discount
    PRINT "20% discount applied. Final amount is:", final_amount
  ELSE IF total_amount >= 200 AND total_amount <= 500 THEN
    discount = total_amount * 0.10
    final_amount = total_amount - discount
    PRINT "10% discount applied. Final amount is:", final_amount
  ELSE
    PRINT "No discount applied. Final amount is:", total_amount
  END IF
END
```

Exercises



- ▶ **Problem Statement:** Create a pseudocode for a simple traffic light system. The system operates in the following way:
 - ▶ If the light is green, cars can go.
 - ▶ If the light is yellow, cars should slow down.
 - ▶ If the light is red, cars must stop.

Write the pseudocode that determines what message to display based on the light color input.

Exercises



```
BEGIN
    PRINT "Enter the traffic light color
(green, yellow, red):"
    INPUT light_color

    IF light_color = "green" THEN
        PRINT "Go!"
    ELSE IF light_color = "yellow" THEN
        PRINT "Slow down!"
    ELSE IF light_color = "red" THEN
        PRINT "Stop!"
    ELSE
        PRINT "Invalid color!"
    END IF
END
```

Exercises



- ▶ **Problem Statement:** Write a pseudocode that checks the strength of a user's password. The rules for checking the strength are:
 - ▶ If the password length is less than 6 characters, it is considered "Weak."
 - ▶ If the password contains at least 6 characters but less than 10 characters, it is "Moderate."
 - ▶ If the password contains 10 or more characters, it is "Strong."

Exercises



```
BEGIN
    PRINT "Enter your password:"
    INPUT password

    password_length = LENGTH(password)

    IF password_length < 6 THEN
        PRINT "Password strength: Weak"
    ELSE IF password_length >= 6 AND password_length < 10 THEN
        PRINT "Password strength: Moderate"
    ELSE IF password_length >= 10 THEN
        PRINT "Password strength: Strong"
    END IF
END
```


Exercises



- ▶ Create a pseudocode program that accepts a numerical score (0-100) as input and assigns a letter grade (A, B, C, D, F) based on that score using a case structure. For example, if the score is between 90 and 100, the grade should be "A". Ensure the program handles scores outside the 0-100 range by displaying an error message

Exercises



```
START
  Declare score as Integer
  Input score

  CASEOF (score)
    CASE score >= 90 AND score <= 100:
      PRINT "Grade A"
      BREAK
    CASE score >= 80 AND score < 90:
      PRINT "Grade B"
      BREAK
    CASE score >= 70 AND score < 80:
      PRINT "Grade C"
      BREAK
    CASE score >= 60 AND score < 70:
      PRINT "Grade D"
      BREAK
    CASE score < 60:
      PRINT "Grade F"
      BREAK
    DEFAULT:
      PRINT "Invalid score entered."
  END CASE
END
```

Exercises



- ▶ Create a pseudocode program that takes the number of a month (1 for January, 2 for February, etc.) as input and uses a case structure to print how many days are in that month. For simplicity, assume February always has 28 days. Group the months with the same number of days into one case in the case structure where possible. If an invalid month number is input (e.g., 13), display an error message

Exercises



```
START
  Declare month as Integer
  Input month

  CASEOF (month)
    CASE 1, 3, 5, 7, 8, 10, 12:
      PRINT "31 days"
      BREAK
    CASE 4, 6, 9, 11:
      PRINT "30 days"
      BREAK
    CASE 2:
      PRINT "28 days"
      BREAK
    DEFAULT:
      PRINT "Invalid month entered."
  END CASE
END
```



Constructs of a pseudocode - Repetition or loop

- ▶ When a certain block of instructions is to be repeatedly executed, we use the repetition or loop construct.
- ▶ Each execution of the block is called an **iteration** or a **pass**.
- ▶ **Definite iteration:** If the number of iterations (how many times the block is to be executed) is known in advance.
- ▶ **Indefinite or conditional iteration:** If the number of iterations is not known in advance
- ▶ The block that is repeatedly executed is called the **loop body**.

Constructs of a pseudocode - Repetition or loop



- ▶ There are three types of loop constructs as discussed below
 - ▶ while loop
 - ▶ repeat-until loop
 - ▶ for loop



Constructs of a pseudocode – while loop

- ▶ A while loop is generally used to implement indefinite iteration. The general form of the while loop is as follows:

```
while (condition)  
    TRUE_INSTRUCTIONS  
endwhile
```

- ▶ Here, the loop body (TRUE_INSTRUCTIONS) is executed repeatedly as long as condition evaluates to True.
- ▶ When the condition is evaluated as False, the loop body is bypassed



Constructs of a pseudocode – repeat-until loop

- ▶ The second type of loop structure is the **repeat-until** structure.
- ▶ This type of loop is also used for **indefinite iteration**.
- ▶ Here the set of instructions constituting the loop body is **repeated** as long as **condition evaluates to False**.
- ▶ When the condition evaluates to **True**, the **loop is exited**

```
repeat
    FALSE_INSTRUCTIONS
until (condition)
```


Constructs of a pseudocode – repeat-until loop



- ▶ There are two major differences between while and repeat-until loop constructs:
 - ▶ In the **while loop**, the pseudocode **continues** to execute as long as the resultant of the condition is **True**; in the **repeat-until** loop, the looping process **stops** when the resultant of the condition becomes **True**.
 - ▶ In the **while** loop, the condition is **tested at the beginning**; in the **repeat until** loop, the condition is **tested at the end**.
- ▶ For this reason, the **while** loop is known as an **entry controlled** loop and the **repeat-until** loop is known as an **exit controlled** loop.

Constructs of a pseudocode – for loop

- ▶ The for loop implements **definite iteration**.
- ▶ for loop constructs use a variable (call it the *loop variable*) as a counter that starts counting from a specific value called *begin* and updates the loop variable after each iteration.
- ▶ The loop body repeats execution until the loop variable value reaches *end*.

```
for var = begin to end  
    LOOP_INSTRUCTIONS  
endfor
```

- ▶ The condition $\text{var} \leq \text{end}$ is tested first.
- ▶ If the condition is True, the loop body is executed.
- ▶ After the first iteration, the loop variable is incremented (increased by 1).
- ▶ The condition $\text{var} \leq \text{end}$ is tested again with the updated value of *var*.
- ▶ If the condition remains True, the loop body is executed again. The loop continues: updating the loop variable after each iteration and testing the condition. When the loop variable becomes greater than *end*, the condition evaluates to False. At this point, the loop execution stops.

Constructs of a pseudocode – for loop



```
for var = begin downto end  
    LOOP__INSTRUCTIONS  
endfor
```

- ▶ In the second for loop variant, whose pseudocode syntax is given
- ▶ The loop variable is decremented (decreased by 1) after every iteration
- ▶ The condition being tested is $\text{var} \geq \text{end}$.
- ▶ Here, *begin* should be greater than or equal to *end*, and the loop exits when this condition is violated.

Constructs of a pseudocode – for loop



```
for var = begin to end by step  
    LOOP__INSTRUCTIONS  
endfor
```

- ▶ It is also possible to update the loop variable by an amount other than 1 after every iteration.
- ▶ The value by which the loop variable is increased or decreased is known as step.
- ▶ In the pseudocode shown below, the step value is specified using the keyword **by**.

Constructs of a pseudocode – for loop



Loop construct	Description	Values taken by <i>var</i>
for <i>var</i> = 1 to 10	<i>var</i> gets incremented by 1 till it reaches 10	1, 2, \dots 9, 10
for <i>var</i> = 10 downto 1	<i>var</i> gets decremented by 1 till it reaches 1	10, 9, \dots 2, 1
for <i>var</i> = 2 to 20 by 2	<i>var</i> gets increased by 2 till it reaches 20	2, 4, \dots 18, 20
for <i>var</i> = 20 downto 2 by 2	<i>var</i> gets decreased by 2 till it reaches 2	20, 18, \dots 4, 2

lists some examples of **for** loops. In these examples, *var* is the loop variable.

Solved problems - Algorithms



- Problem: To find simple interest

SIMPLEINTEREST

```
1  Start
2  READ(principal, rate, years)
3   $SI = (principal * rate * years) / 100$ 
4  PRINT(SI)
5  Stop.
```

Solved problems - Algorithms



- Problem: To determine the larger of two numbers

LargerTwo

```
1 Start
2 Read(number1, number2)
3 if (number1 > number2)
4   large = number1
5 else
6   large = number2
7 endif
8 Print(large)
9 Stop.
```

Solved problems - Algorithms



- Problem: To determine the smallest of three numbers.

```
SmallestThree
1 Start
2 Read(number1, number2, number3)
3 if (number1 < number2)
4 small = number1
5 else
6 small = number2
7 endif
8 if (number3 < small )
9 small = number3
10 endif
11 Print(small )
12 Stop.
```


Solved problems - Algorithms



- Problem: To determine the entry-ticket fare in a zoo based on age.

Age	Fare
< 10	7
≥ 10 and < 60	10
≥ 60	5

```
TicketFare
1 Start
2 Read(age)
3 if (age < 10)
4   fare = 7
5 else if (age < 60)
6   fare = 10
7 else
8   fare = 5
9 endif
10 Print(fare)
11 Stop
```

Solved problems - Algorithms



- Problem: To print the colour based on a code value as follows:

Grade	Message
<i>R</i>	Red
<i>G</i>	Green
<i>B</i>	Blue
Any other value	Wrong code

```
PrintColour
1 Start
2 Read(code)
3 caseof (code)
4 case 'R':
5   Print("Red")
6 break
7 case 'G':
8   Print("Green")
9 break
10 case 'B':
11   Print("Blue")
12 break
13 default :
14   Print("Wrong code")
15 endcase
16 Stop
```

Solved problems - Algorithms



- Problem: To print the numbers from 1 to 50 in descending order.

```
PrintDown
1 Start
2 for count = 50 downto 1
3   Print(count)
4 endfor
5 Stop
```

Solved problems - Algorithms



- Problem: To find the factorial of a number.

Factorial

1 Start

2 Read(*number*)

3 *fact* = 1

4 **for** *var* = *number* **downto** 1

5 *fact* = *fact* * *var*

6 **endfor**

7 Print(*fact*)

8 Stop

Solved problems - Algorithms



- Problem: To determine the largest of n numbers.

LARGEN

```
1  Start
2  READ( $n, num$ )
3   $large = num$ 
4  for  $count = 1$  to  $n - 1$ 
5      READ( $num$ )
6      if ( $num > large$ )
7           $large = num$ 
8      endif
9  endfor
10 PRINT( $large$ )
11 Stop
```

Solved problems - Algorithms



- Problem: To determine the average age of students in a class. The user will stop giving the input by giving the age as 0.

AVERAGEAGEV1

```
1  Start
2  sum = 0
3  count = 0
4  READ(age)
5  while (age!=0)
6      sum = sum + age
7      count = count + 1
8      READ(age)
9  endwhile
10 average = sum/count
11 PRINT(average)
12 Stop
```

Solved problems - Algorithms



- Problem: To determine the average age of students in a class. The user will stop giving the input by giving the age as 0. use repeat-until loop

AVERAGEAGEV2

```
1  Start
2  sum = 0
3  count = 0
4  READ(age)
5  repeat
6      sum = sum + age
7      count = count + 1
8      READ(age)
9  until (age == 0)
10 average = sum / count
11 PRINT(average)
12 Stop
```

Solved problems - Algorithms



- Problem: To find the average height of boys and average height of girls in a class of n students

AVERAGEHEIGHT

```
1  Start
2  READ( $n$ )
3   $btotal = 0$ 
4   $bcount = 0$ 
5   $gtotal = 0$ 
6   $gcount = 0$ 
7  for  $var = 1$  to  $n$ 
8      READ( $gender, height$ )
9      if ( $gender == 'M'$ )
10           $btotal = btotal + height$ 
```

```
11           $bcount = bcount + 1$ 
12      else
13           $gtotal = gtotal + height$ 
14           $gcount = gcount + 1$ 
15      endif
16 endfor
17  $bavg = btotal / bcount$ 
18  $gavg = gtotal / gcount$ 
19 PRINT( $bavg, gavg$ )
20 Stop
```


References



- ▶ <https://www.scaler.com/topics/how-to-write-pseudo-code/>
- ▶ Algorithmic Thinking with Python – Ajeesh Ramanujan, Narasimhan T



ST. JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -
AUTONOMOUS

Thank You



Prof. Sarju S

Department of Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Palai (Autonomous)
sarju.s@sjcetpalai.ac.in