COMPUTER PROGRAMMING

LECTURE NOTES

CP115

Topic 6: Use of Selection Control Structure

Learning Outcomes:

(a) Identify types of selection structure (single, dual and multiple)

(2 hour)

LECTURE NOTES

Types of Selection

- *if [Single selection]*
- *if-else [Dual selection]*
- *if-elif-else [Multiple selection]*

Single Selection

`if` statement

- It is used to make a decision whether a certain process is to be executed or not.
- Syntax for **`if`**

```
if condition:
    statement 1
```
	```
	if condition:
	    statement 1
	    statement 2
	    statement 3
	```
		Notes:
		Every statements must be indented

Single Selection

Example 1:

Write a code snippet that displays appropriate messages based on a student's mark. If the mark is greater than or equal to 85, the message 'Excellent!' should be displayed.

`mark = int(input("Enter mark:"))`
`if mark >=85:`
` print(" Excellent!")`
	Notes:
	If the condition is true, then statement 1 is executed, otherwise statement will not be executed

`mark = int(input("Enter mark:"))`
`if mark >= 85:`
`    print("Excellent")`
`    print("\nYou PASS the exam")`
	If the condition is true, then statement 1 and statement 2 is executed (both statements must be indented), otherwise statement will not be executed

Single Selection

Example 2:
Write a program segment to determine the status of a number whether it is a positive number.

Segment code:
(code below)

```
# Prompt the user to enter a number
number = float(input("Enter a number: "))

# Check if the number is positive
if number > 0:
    print("The number is positive.")
```

Dual Selection

`if-else` statement

- It allows program execution to choose between two actions based on the value of the condition.
- Syntax for **`if-else`**

```
if condition:
    statement 1;
else:
    statement 2;
```
		Notes:
		All statement must be indented

Dual Selection

Example 3:

Create a program segment to compare between two numbers entered by user and identify the larger one.

Segment code:

```
# Prompt the user to enter the first number
num1 = float(input("Enter the first number: "))
# Prompt the user to enter the second number
num2 = float(input("Enter the second number: "))
# Compare the two numbers
if num1 > num2:
    print("The first number is larger than the second number”)
else:
    print("The second number is larger than the first number”)
```

Dual Selection

Example 4 :

Write a program segment to display the message 'PASS' if the mark is more than 40, or 'FAIL' otherwise, based on the entered mark.

Problem Analysis

Input : mark
Process : Determine message based on mark
Output : “PASS” or “FAIL”

Segment code:

```
# Prompt the user to enter the mark
mark = float(input("Enter the mark: "))

# Check if the mark is more than 40
if mark > 40:
    print("PASS")
else:
    print("FAIL")
```

Dual Selection

Example 5 Create a program that will determine whether a number is odd or even.

Input	Condition	Output
3	$3\%2 == 0 \rightarrow$ false	3 is an odd number

```
# Prompt the user to enter a number
number = int(input("Enter a number: "))

# Check if the number is even or odd
if number % 2 == 0:
    print("The number is even.")
else:
    print("The number is odd.")
```

Selection Control Structures

In python, selection control structure is implemented using the if keyword. There are three (3) types of selection statements:

1. Single selection: `if`
2. Dual selection: `if-else`
3. Multiple selection: `if-elif-else`

COMPUTER PROGRAMMING

LECTURE NOTES

Types of Selection

- *if [Single selection]*
- *if-else [Dual selection]*
- *if-elif-else [Multiple selection]*

LECTURE NOTES

Control Structure Selection

Multiple Selection

Concept

- Multiple selection control structure checks many conditions to choose between many actions.
- Condition will be checked one by one.
- When a condition is true, performs the action and stops checking the rest.

Multiple Selection

if-elif-else statement

- It is used to make a decision whether a certain process is to be executed or not.
- Syntax for **if-elif-else**

```
if condition1:
    #Code to execute if condition1 is True
elif condition2:
    #Code to execute if condition1 is False and
     condition2 is True
elif condition3:
    #Code to execute if condition1 and condition2 are
     False, and condition3 is True
else:
    #Code to execute if all conditions are False
```

How It Works:

1. The program starts by evaluating `condition1` (the `if` condition):
   - If **True**, the corresponding code block runs, and stop checking all other conditions.
2. If `condition1` is **False**, the program checks `condition2`:
   - If **True**, the corresponding block runs, and the rest of the conditions are skipped.
3. This process continues for all `elif` conditions.
4. If none of the conditions are **True**, the program executes the `else` block statement(s).
5. If none of the conditions are **True** and there's no `else`, the program does nothing.

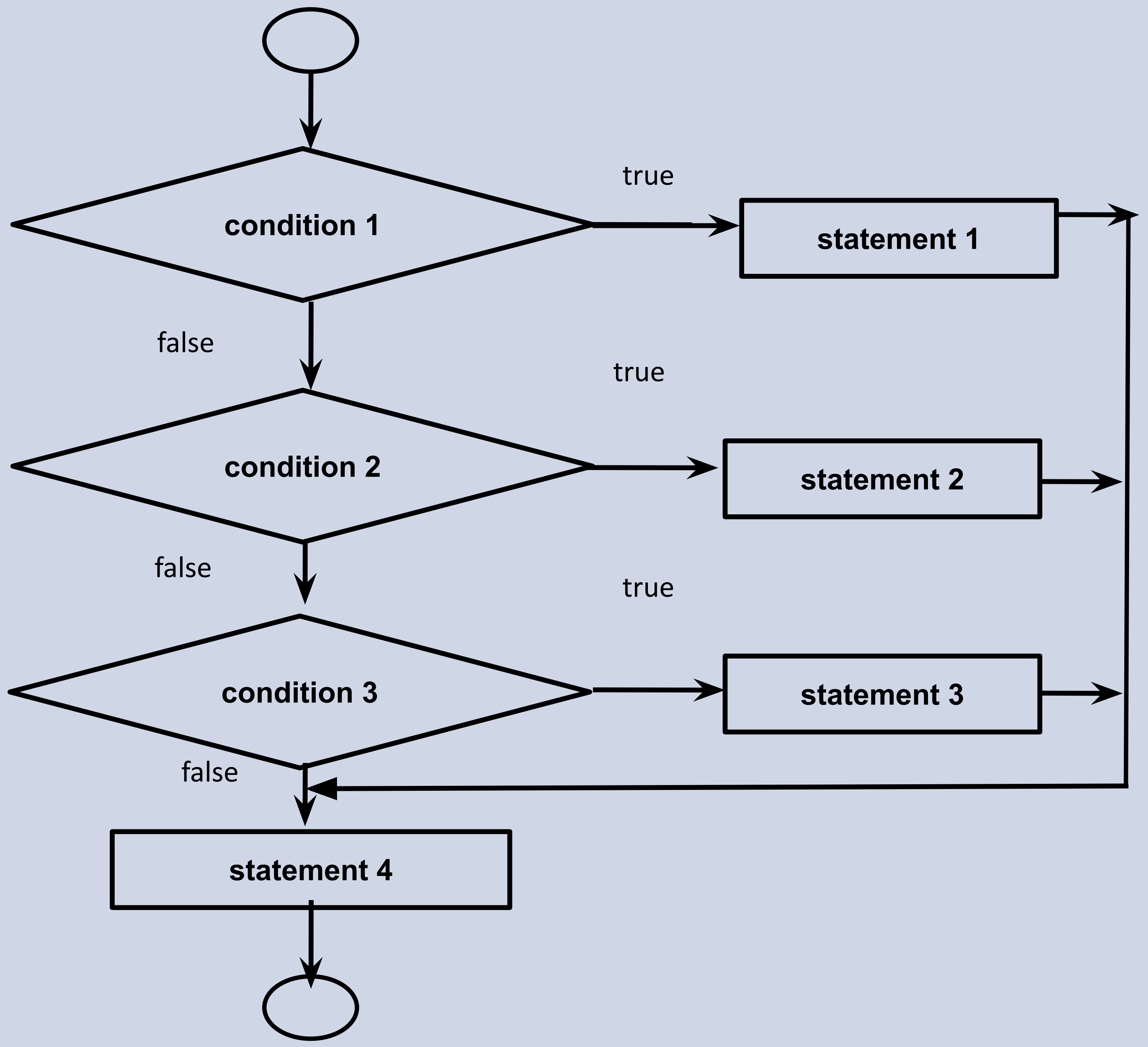
Note:
Every statements must be indented.

COMPUTER PROGRAMMING

LECTURE NOTES

General Format of Pseudocode & Flowchart

Pseudocode	Flowchart

```
if (condition 1)
    statement
else if (condition 2)
        statement 2
else if (condition 3)
        statement 3
else
        statement 4
end if
```
	 condition 1
	true
	statement 1
	false
	condition 2
	true
	statement 2
	false
	condition 3
	true
	statement 3
	false
	statement 4

Note:

- The diamond in a multiple alternative selection structure has several flow lines leading out of the symbols.
- Each flow line represents a possible path and must be marked to indicate the values necessary for the path to be chosen.
- Condition will be checked one by one
- When a condition is true, performs the action and stops checking the rest.

LECTURE NOTES

Example 1: Mr. Tan teaches Science in Matrix High School. He wants a program that displays a letter grade with the marks as he enters the marks. The valid letter grade and their corresponding message are shown below.

GRADE	**MARKS**
A	mark is greater than or equal to 80
B	mark is greater than or equal to 60
C	mark is greater than or equal to 50
Failed	mark is less than 50

Step 1: Problem Analysis

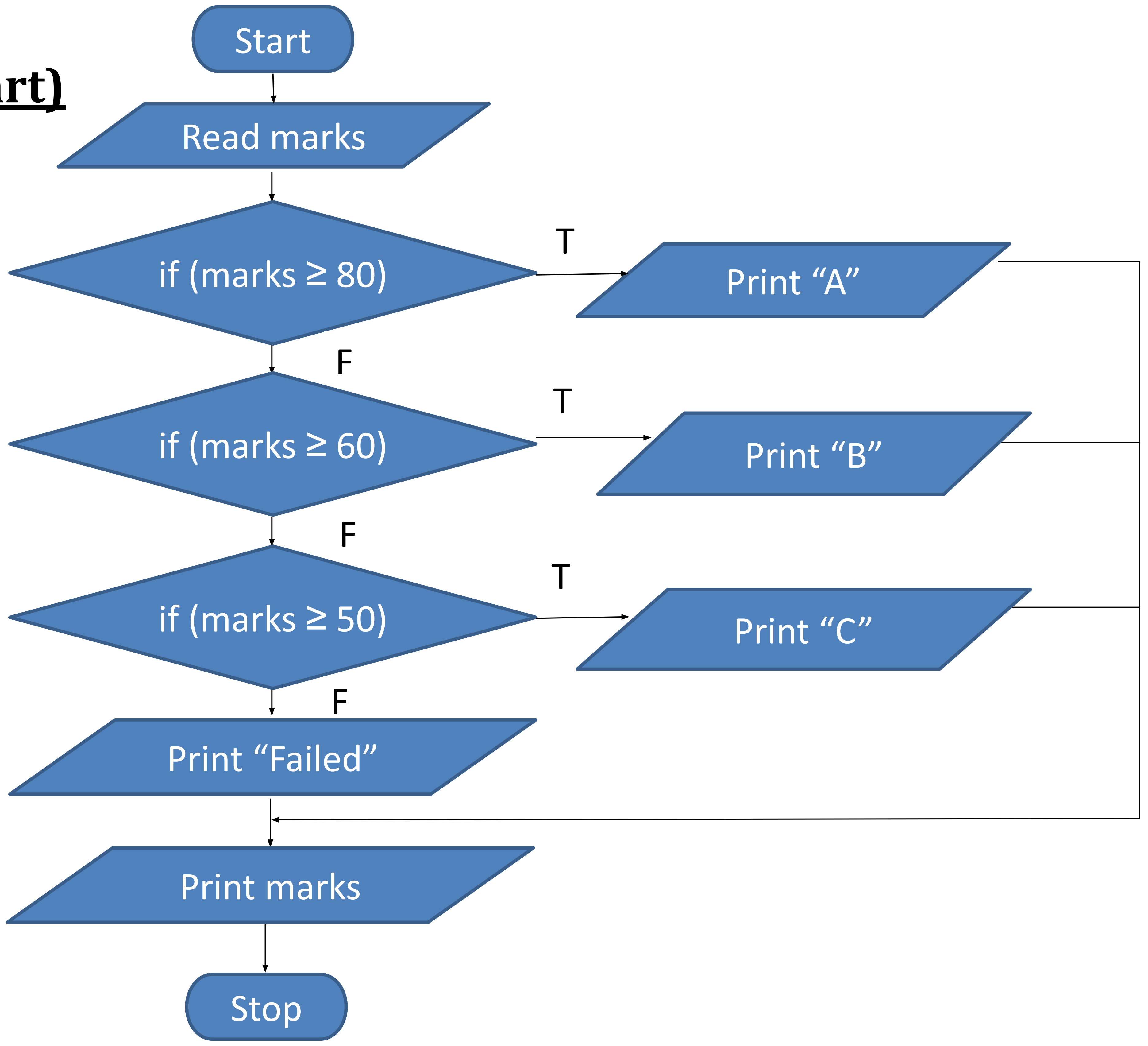
Input	**Process**	**Output**
marks	Determine the grades and based on the marks.	"A" or "B" or "C" or "Failed", marks

Step 2: Design a Solution (Pseudocode)

```
Start
    Read marks
    if (marks ≥ 80)
            Print ‘A
    else if (marks ≥ 60)
            Print ‘B
    else if (marks ≥ 50)
            Print ‘C’
    else
            Print ‘Failed’
    end if
    Print marks
Stop
```

LECTURE NOTES

Step 2: Design a Solution (Flowchart)



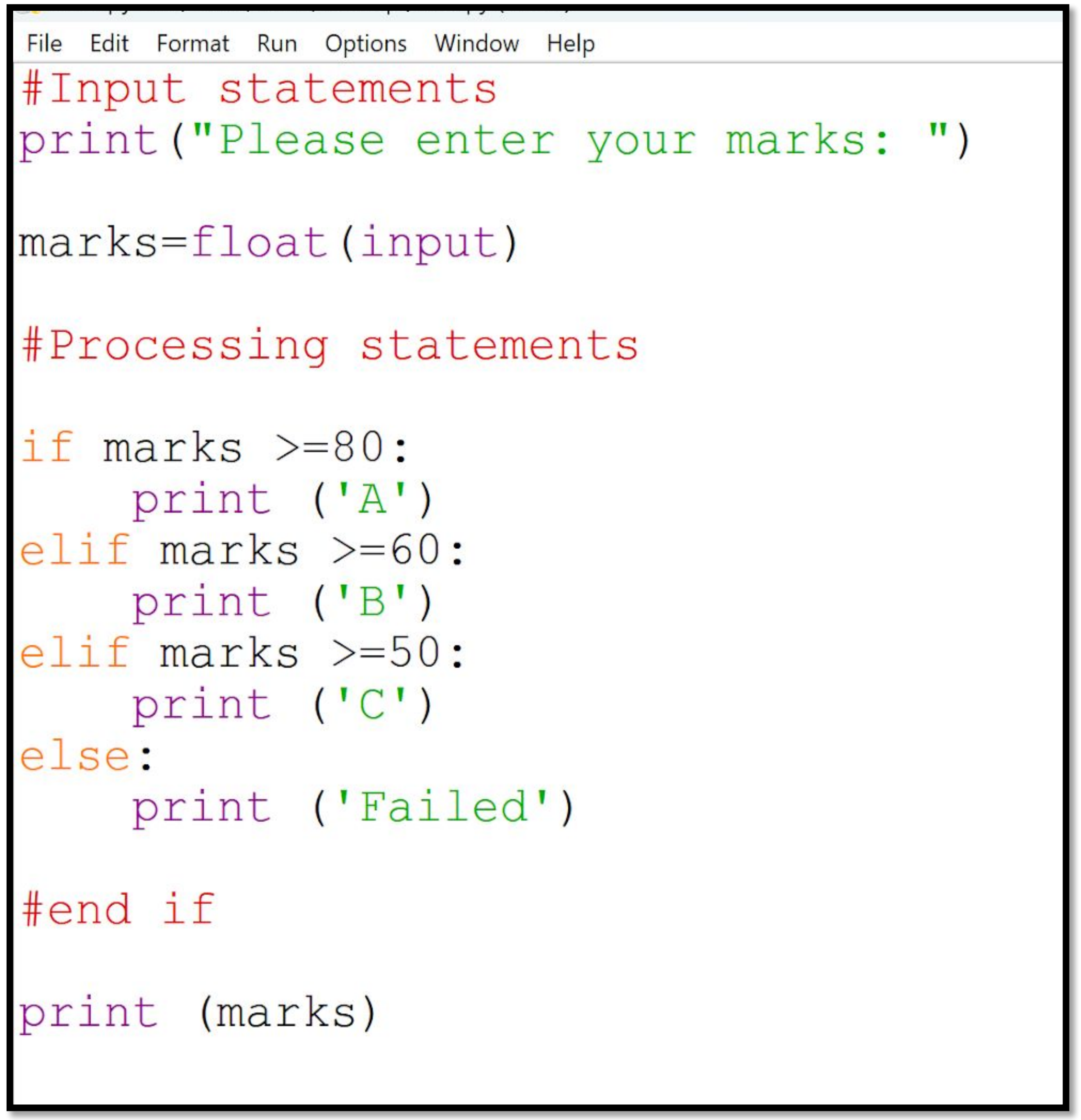
LECTURE NOTES

<u>Step 3: Implementation</u> (Pseudocode Programming)

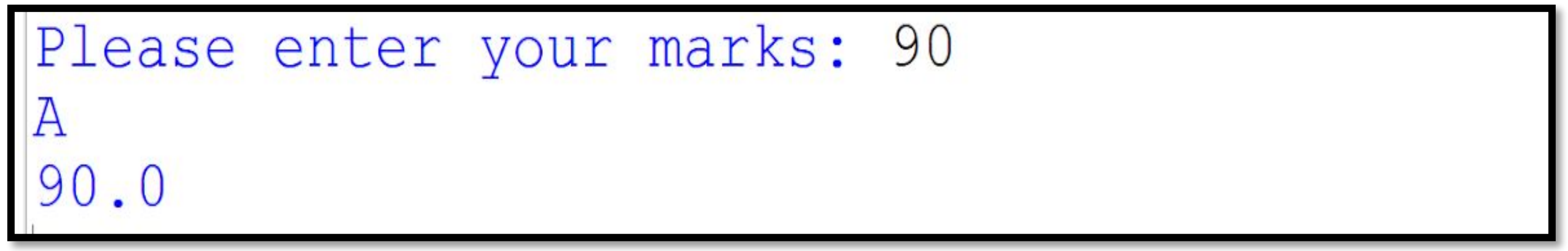
Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
Read marks	`marks= float (input)`
if (marks ≥ 80)	`if marks >= 80:`
Print ‘A’	`  print (‘A’)`
else if (marks ≥ 60)	`elif marks >= 60:`
Print ‘B’	`  print (‘B’)`
else if (marks ≥ 50)	`elif marks >= 50:`
Print ‘C’	`  print (‘C’)`
else	`else:`
Print ‘Failed’	`  print (‘Failed’)`
end if	`#end if`
Print marks	`print (marks)`
Stop	

LECTURE NOTES

Step 4: Testing

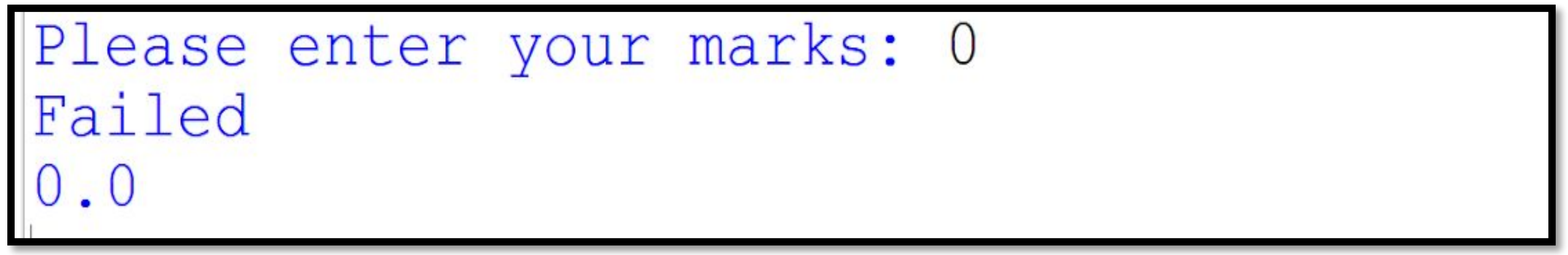


```
#Input statements
print("Please enter your marks: ")

marks=float(input)

#Processing statements

if marks >=80:
    print ('A')
elif marks >=60:
    print ('B')
elif marks >=50:
    print ('C')
else:
    print ('Failed')

#end if

print (marks)
```



```
Please enter your marks: 90
A
90.0
```

Assume user enters marks- 90



```
Please enter your marks: 0
Failed
0.0
```

Assume user enters marks- 0

LECTURE NOTES

Example 2: By inserting length, users choose one from two listed calculation programs either to calculate perimeter of hexagon or area of square. Print the output and message "Wrong Choice" if users choose other than those programs mentioned.

CHOICE	CALCULATION
1	Calculate the perimeter of hexagon
2	Calculate the area of square

Step 1: Problem Analysis

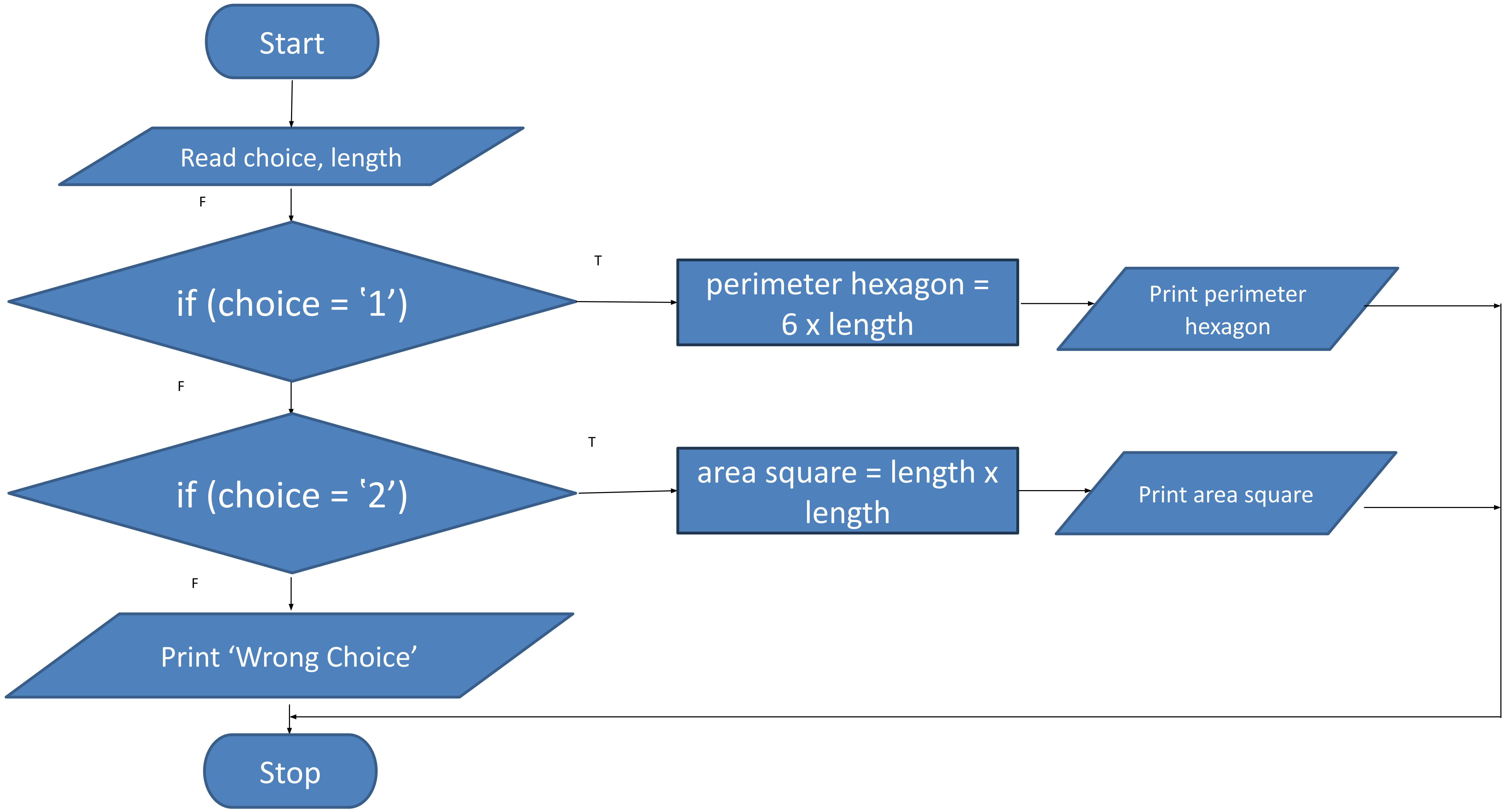
Input	Process	Output
choice, length	Determine and calculate perimeter hexagon or area square or "Wrong Choice" *based on* the choice and length.	perimeter hexagon or area square or "Wrong Choice"

LECTURE NOTES

Step 2: Design a Solution (Pseudocode)

```
Start
    Read choice, length
    if (choice = '1')
        perimeter hexagon = 6 x length
        Print perimeter hexagon
    else if (choice ='2')
        area square = length x length
        Print area square
    else
        Print 'Wrong Choice'
    end if
Stop
```

Step 2: Design a Solution (Flowchart)



LECTURE NOTES

Step 3: Implementation (Pseudocode Programming)

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
Read choice, length	`choice = input("Please enter your choice either 1 or 2 : ");`
	`length = float(input("Please enter the length : "));`
if (choice = **'1'**)	`if choice == '1' :`
perimeter hexagon = 6 x length	`    perimeterhexagon = 6 * length`
Print perimeter hexagon	`    print("The perimeter hexagon is : ",perimeterhexagon)`
else if (choice =**'2'**)	`elif choice == '2' :`
area square = length x length	`    areasquare = length * length`
Print area square	`    print("The area of square is : ",areasquare)`
else	`else:`
Print 'Wrong Choice'	`print("Wrong Choice");`
end if	`#end if`
Stop	

COMPUTER PROGRAMMING

LECTURE NOTES

Step 4: Testing

```
#Read input
choice = input("Please enter your choice either 1 or 2 : ");

#Read input
length = float(input("Please enter the length : "));

#Process
if  choice == '1' :
    perimeterhexagon = 6 * length
    print("The perimeter hexagon is : ",perimeterhexagon)

elif choice == '2' :
    areasquare = length * length
    print("The area of square is : ",areasquare)

else:
    print("Wrong Choice");

#end if
```

```
Please enter your choice either 1 or 2 : 1
Please enter the length : 3
The perimeter hexagon is :  18.0
```

Assume user enters choice- 1 and length- 3

```
Please enter your choice either 1 or 2 : 2
Please enter the length : 5
The area of square is :  25.0
```

Assume user enters choice- 2 and length- 5

Indah Jaya Shopping Complex charges RM4.00 per hour to park for the first 3 hours. For the next two hours, the charge is RM3.00 per hour, and RM2.50 per hour for the next two hours. A flat rate of RM25.00 is charged if the car is parked for more than 7 hours.
Calculate the total parking fees

LECTURE NOTES

Topic 6: Use of Selection Control Structure

Learning Outcomes:

(c) Identify the use of relational and logical operators in selection control structures (single, dual and multiple)

Combining Relational and Logical Operators

Example 1: Calculate Body Mass Index (BMI) when user enters weight (in kg) and height (in cm). From the BMI's result, the system will display a message as in the table below.

BMI result	Weight Category
Under 18.5	display "Underweight"
18.5 - 24.9	display "Healthy Weight"
25 - 29.9	display "Overweight"
30 or over	"Obese"

Step 1: Problem Analysis

Input	Process	Output
weight, height	Calculate BMI and determine the weight category based on the weight and height.	BMI , “Underweight” or “Healthy Weight” or “Overweight” or “Obese”

Step 2: Design a Solution (Pseudocode)

```
Start
    Read weight, height
    BMI = weight/ (height x height)
    Print BMI
    if (BMI ≥ 0 & BMI < 18.5)
        Print "Underweight"
    else if (BMI ≥ 18.5 & BMI < 25)
        Print "Healthy Weight"
    else if (BMI ≥ 25 & BMI < 30)
        Print "Overweight "
    else
        Print "Obese"
    end if
Stop
```

Combining Relational and Logical Operators

- **≥ (greater than or equal to)**: Use when checking if one value is at least as large as another.
  `BMI ≥ 0` (checks if BMI is at least 0)
- `and`: Used to combine multiple conditions, ensuring that **both conditions** are true.
  `BMI ≥ 0 and BMI < 18.5` (ensures the BMI is **within a specific range**)
- **< (less than)**: Use when checking if one value is strictly smaller than another.
  `BMI < 18.5` (checks if BMI is less than 18.5)

Assume BMI: 17.5

if (BMI ≥ 0 & BMI < 18.5)

17.5 ≥ 0 and 17.5 < 18.5

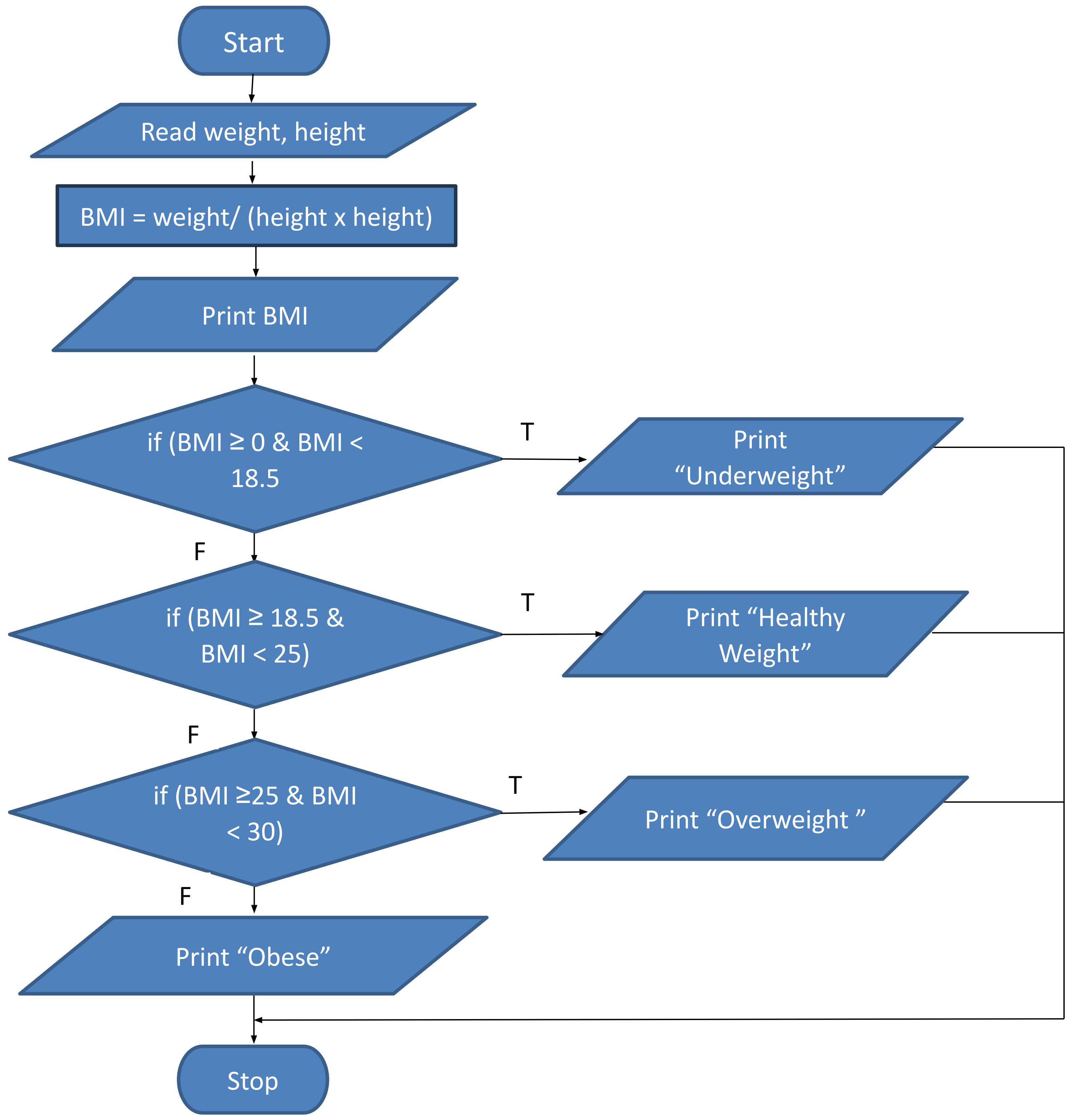
True and True

= True

- In this case, since **condition1** is **True**, the statement Print "Underweight" is executed and no other conditions are checked.
- If **condition1** is **False**, the program checks **condition2**. If **True**, the corresponding block runs, and the rest of the conditions are skipped.
- This process continues for all **else if** conditions.
- If none of the conditions are **True**, the program executes the **else** block statement(s).

LECTURE NOTES

Step 2: Design a Solution (Flowchart)



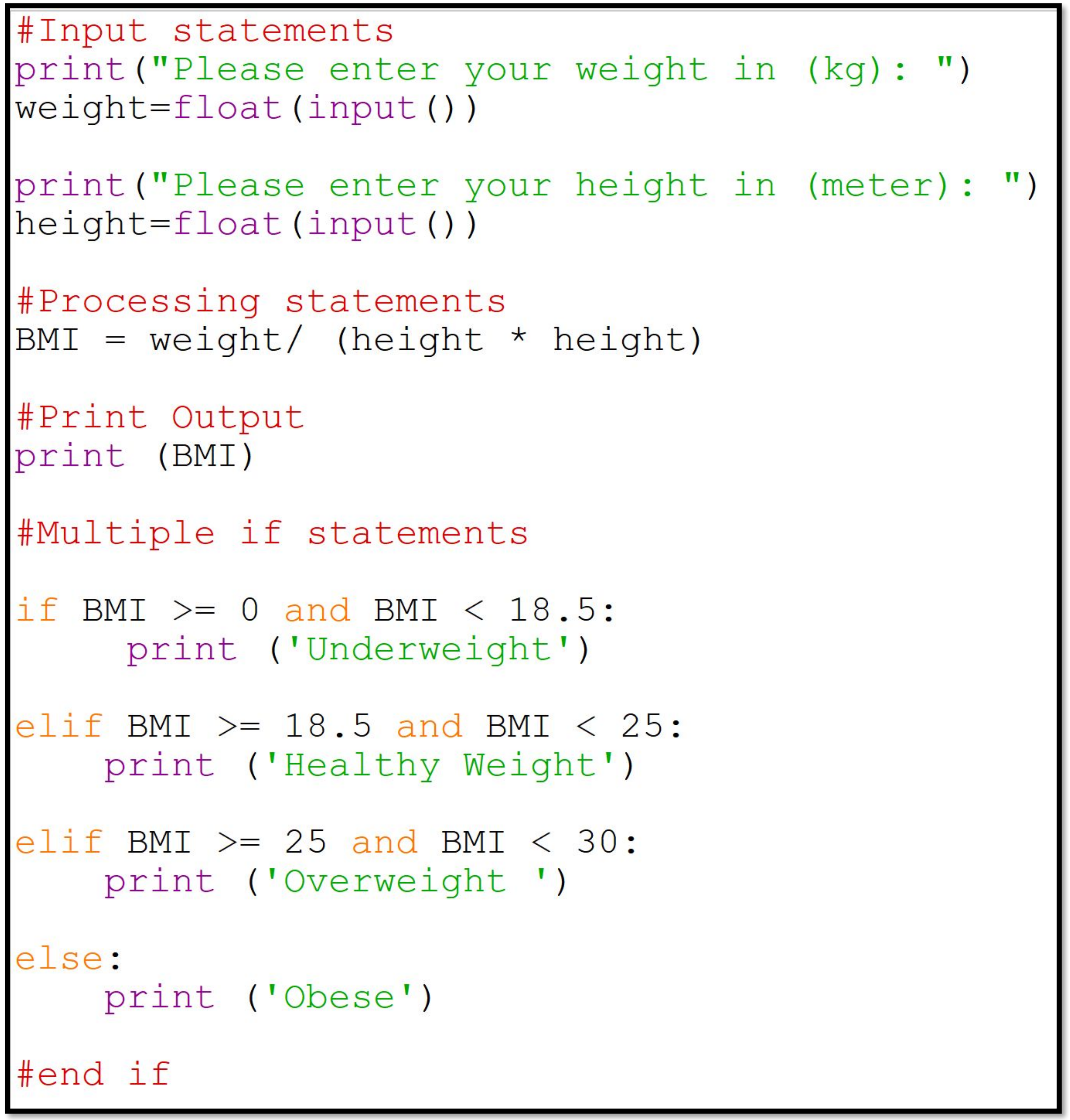
LECTURE NOTES

Step 3: Implementation (Pseudocode Programming)

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
Read weight, height	`weight=float(input())`
	`height=float(input())`
BMI = weight/ (height x height)	`BMI = weight/ (height * height)`
Print BMI	`print (BMI)`
if (BMI ≥ 0 & BMI < 18.5)	`if BMI >= 0 and BMI < 18.5:`
Print “Underweight”	`  print ('Underweight')`
else if (BMI ≥ 18.5 & BMI < 25)	`elif BMI >= 18.5 and BMI < 25:`
Print “Healthy Weight”	`  print ('Healthy Weight')`
else if (BMI ≥ 25 & BMI < 30)	`elif BMI >= 25 and BMI < 30:`
Print Overweight	`  print ('Overweight')`
else	`else:`
Print “Obese”	`  print ('Obese')`
end if	`#end if`
Stop	

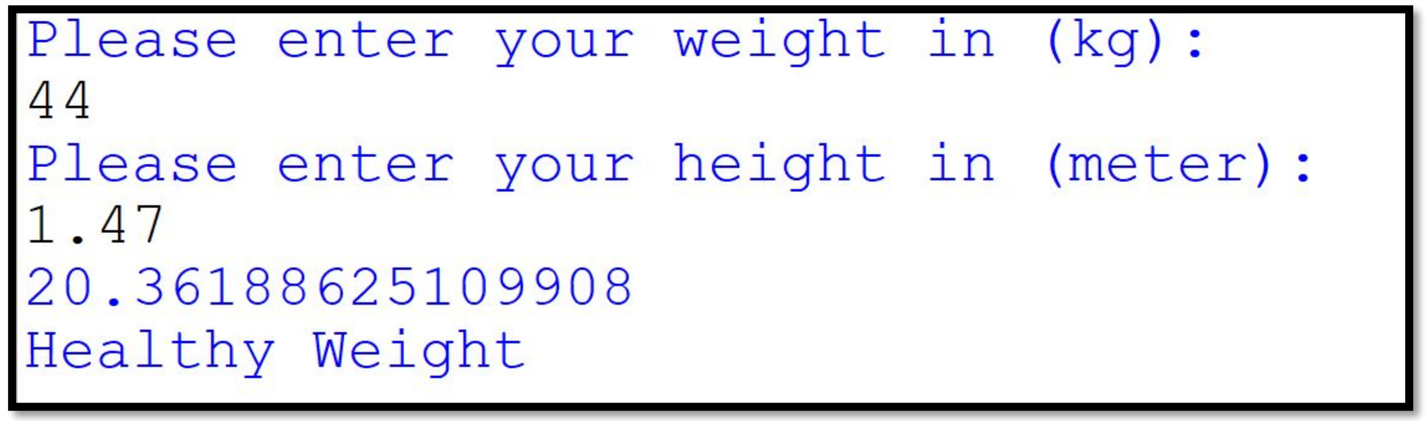
COMPUTER PROGRAMMING

LECTURE NOTES



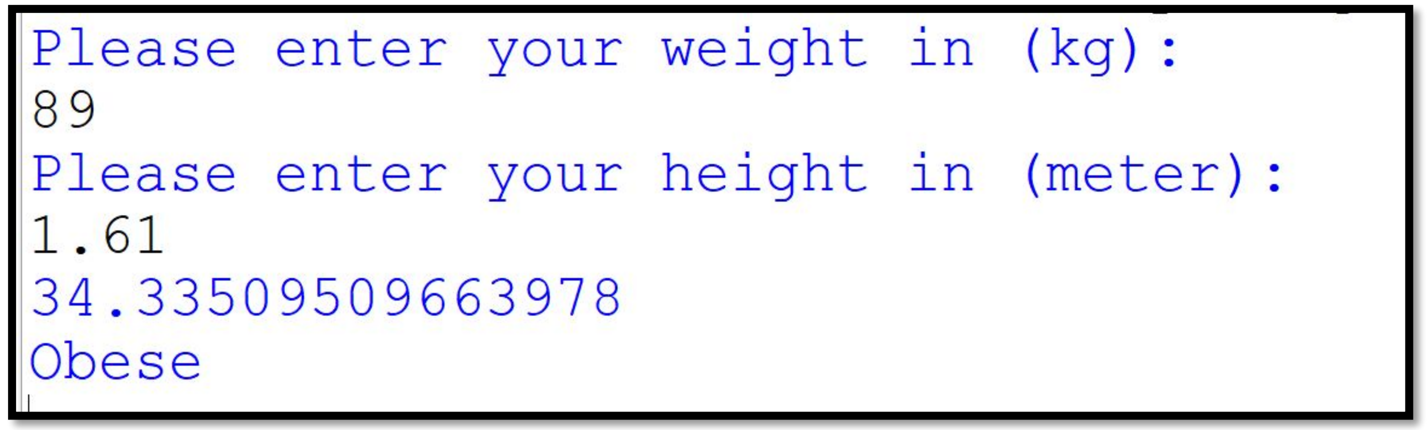
```
#Input statements
print("Please enter your weight in (kg): ")
weight=float(input())

print("Please enter your height in (meter): ")
height=float(input())

#Processing statements
BMI = weight/ (height * height)

#Print Output
print (BMI)

#Multiple if statements

if BMI >= 0 and BMI < 18.5:
     print ('Underweight')

elif BMI >= 18.5 and BMI < 25:
    print ('Healthy Weight')

elif BMI >= 25 and BMI < 30:
    print ('Overweight ')

else:
    print ('Obese')

#end if
```

Step 4: Testing



```
Please enter your weight in (kg):
44
Please enter your height in (meter):
1.47
20.36188625109908
Healthy Weight
```

Assume user enters weight- 44 and height 1.47



```
Please enter your weight in (kg):
89
Please enter your height in (meter):
1.61
34.33509509663978
Obese
```

Assume user enters weight- 89 and height 1.61

LECTURE NOTES

Combining Relational and Logical Operators

Example 2: The colors red, blue and yellow are known as the primary colors because they cannot be mixed from other colors, When you mix two primary colors, you get a secondary color, as shown below.

- Red + Blue = Purple
- Red + Yellow = Orange
- Blue + Yellow = Green

Design a program that prompts the user to enter the names of two primary colors to mix. If the user enters any other colours besides “Red”, “Blue” or “Yellow”, the program should display an error message. Otherwise, the program should display the name of the secondary color.

Step 1: Problem Analysis

Input	Process	Output
colour 1, colour 2	Determine the secondary colour based on colour 1 and colour 2.	"Purple" or "Orange" or "Green" or "Wrong Input"

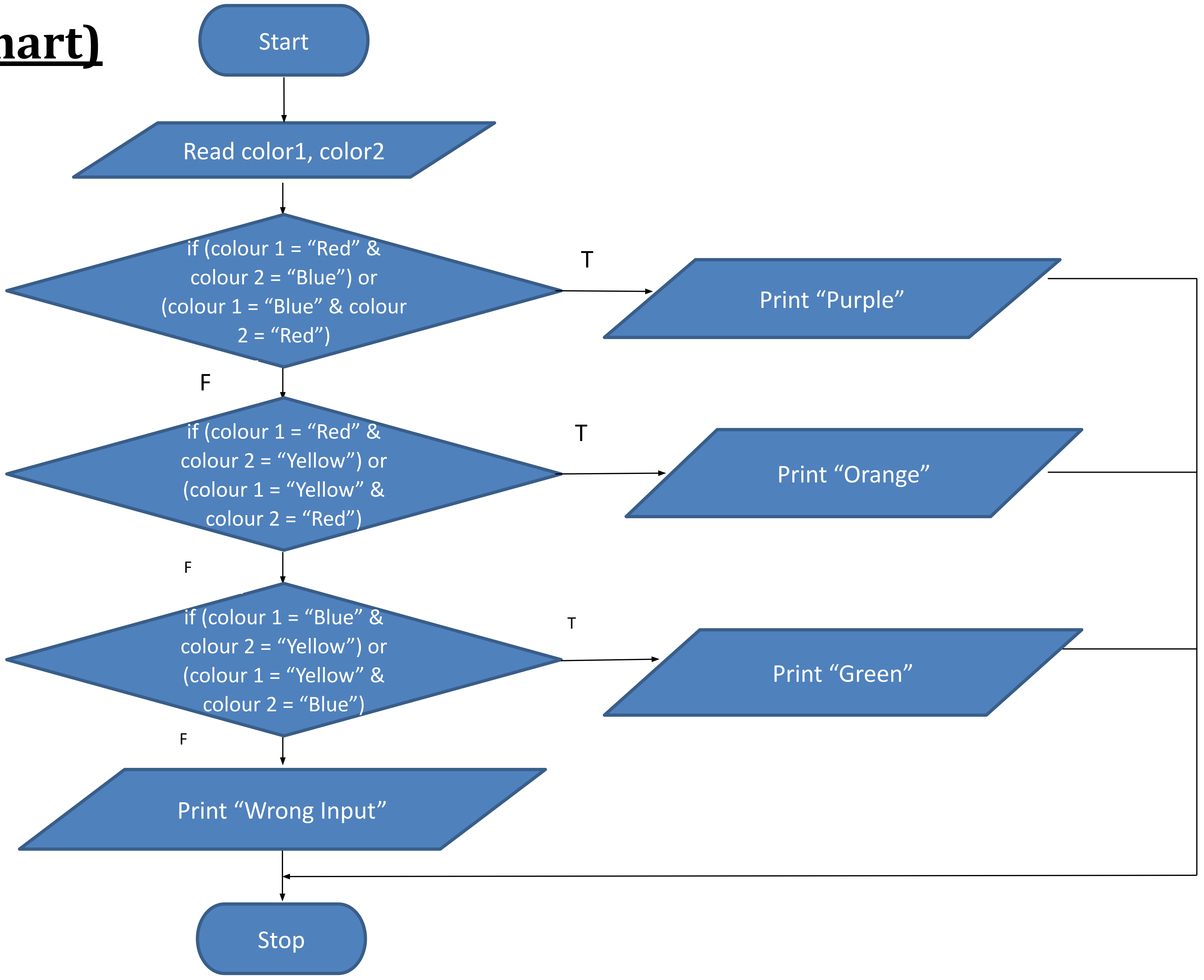
LECTURE NOTES

Step 2: Design a Solution (Pseudocode)

```
Start
    Read colour 1, colour 2
        if (colour 1 = “Red” & colour 2 = “Blue”) or (colour 1 = “Blue” & colour 2 = “Red”)
            Print “Purple”
        else if (colour 1 = “Red” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Red”)
            Print “Orange”
        else if (colour 1 = “Blue” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Blue”)
            Print “Green”
        else
            Print “Wrong Input”
        end if
Stop
```

LECTURE NOTES

Step 2: Design a Solution (Flowchart)


Start
Read color1, color2
if (colour 1 = “Red” & colour 2 = “Blue”) or (colour 1 = “Blue” & colour 2 = “Red”)
T
Print “Purple”
F
if (colour 1 = “Red” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Red”)
T
Print “Orange”
F
if (colour 1 = “Blue” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Blue”)
T
Print “Green”
F
Print “Wrong Input”
Stop

Step 3: Implementation (Pseudocode Programming)

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
Read colour 1, colour 2	`colour1 = input("Please enter your choice of the first primary color : ");`
	`colour2 = input("Please enter your choice of the second primary color : ");`
if (colour 1 = “Red” & colour 2 = “Blue”) or (colour 1 = “Blue” & colour 2 = “Red”)	`if colour1=="Red" and colour2=="Blue" or colour1=="Blue" and colour2=="Red":`
Print “Purple”	`    print("Purple");`
else if (colour 1 = “Red” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Red”)	`elif colour1=="Red" and colour2=="Yellow" or colour1=="Yellow" and colour2=="Red":`
Print “Orange”	`    print("Orange");`
else if (colour 1 = “Blue” & colour 2 = “Yellow”) or (colour 1 = “Yellow” & colour 2 = “Blue”)	`elif colour1=="Blue" and colour2=="Yellow" or colour1=="Yellow" and colour2=="Blue":`
Print “Green”	`    print("Green");`
else	`else:`
Print “Wrong Input”	`    print("Wrong Input");`
end if	`#end if`
Stop	`Stop`

COMPUTER PROGRAMMING

LECTURE NOTES

```
#Read input
colour1 = input("Please enter your choice of the first primary color : ");

#Read input
colour2 = input("Please enter your choice of the second primary color : ");

#Process

if colour1=="Red" and colour2=="Blue" or colour1=="Blue" and colour2=="Red":
    print("Purple");

elif colour1=="Red" and colour2=="Yellow" or colour1=="Yellow" and colour2=="Red":
    print("Orange");

elif colour1=="Blue" and colour2=="Yellow" or colour1=="Yellow" and colour2=="Blue":
    print("Green");

else:
    print("Wrong Input");

#end if
```

Step 4: Testing

Assume user enters first primary color- Yellow and second primary color- Red

```
Please enter your choice of the first primary color : Yellow
Please enter your choice of the second primary color : Red
Orange
```

Assume user enters first primary color- Blue and second primary color- Black

```
Please enter your choice of the first primary color : Blue
Please enter your choice of the second primary color : Black
Wrong Input
```

Combining Relational and Logical Operators

Example 3: The Danville city manager wants a program that determines voter eligibility and displays one of three messages. The messages and the criteria for displaying each message are as follows.

MESSAGE	CRITERIA
You are too young to vote	Person is younger than 18 years old
You can vote	Person is at least 18 years old and registered to vote
You must register before you can vote	Person is at least 18 years old but is not registered to vote

Step 1: Problem Analysis

Input	Process	Output
age, registration status	Determine the voter's eligibility based on their age and their registration status. OR Determine whether the voter is too young to vote or can vote or must register before voting based on the age and registration status.	"You are too young to vote" or "You can vote" or "You must register before you can vote"

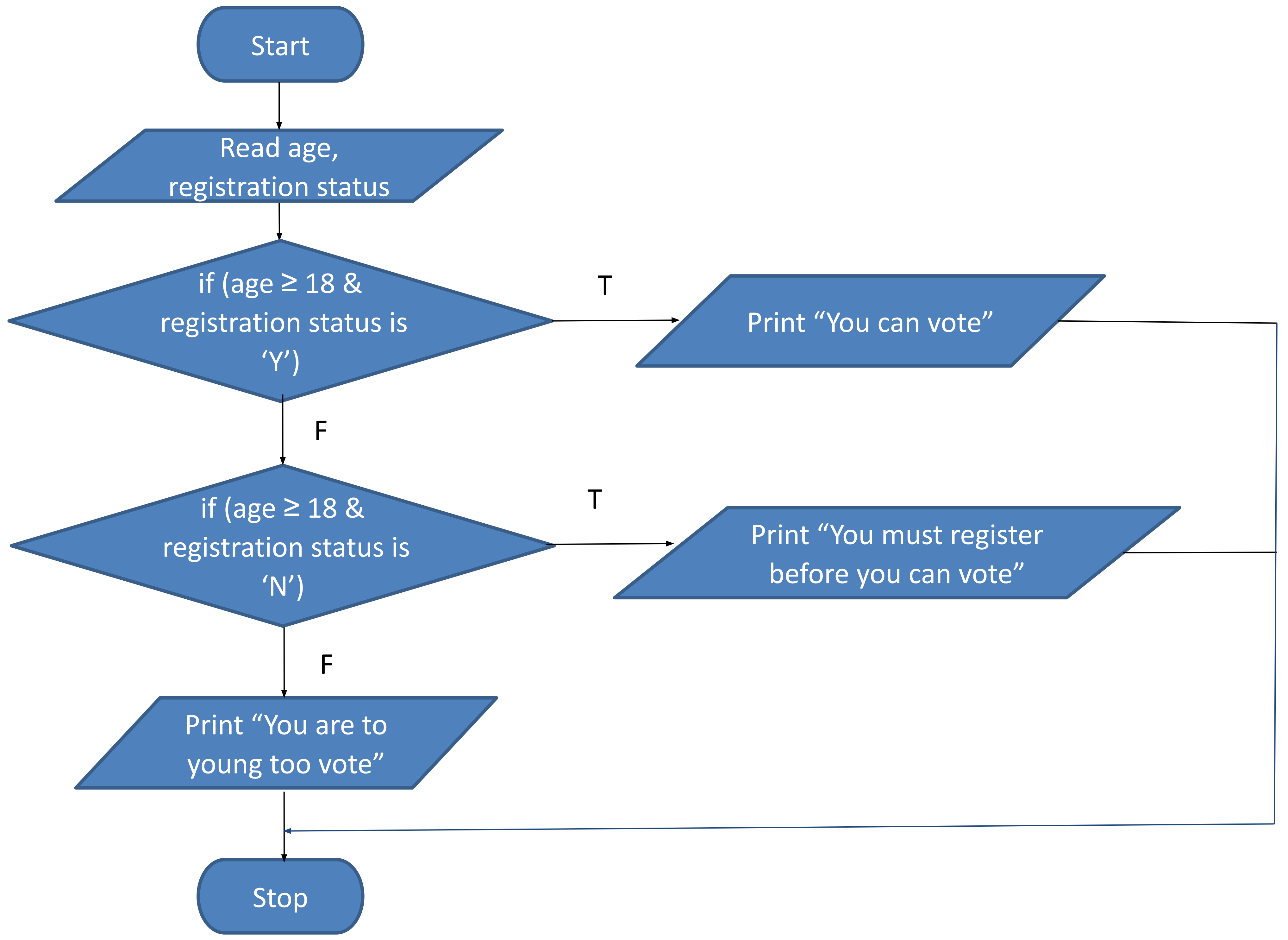
LECTURE NOTES

Step 2: Design a Solution (Pseudocode)

```
Start
    Read age, registration status
    if (age ≥ 18 & registration status = 'Y')
        Display "You can vote"
    else if (age ≥ 18 & registration status = 'N')
        Display "You must register before  you can vote"
    else
        Display "You are too young to vote"
    endif
Stop
```

LECTURE NOTES

Step 2: Design a Solution (Flowchart)


Start
Read age, registration status
if (age ≥ 18 & registration status is 'Y')
T
Print "You can vote"
F
if (age ≥ 18 & registration status is 'N')
T
Print "You must register before you can vote"
F
Print "You are to young too vote"
Stop

<u>Step 3: Implementation</u> (Pseudocode □ Programming)

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	

Read age, registration status	`age = int(input("Enter your age: "))`
	`status = input("Enter registration status (Y=Yes or N=No): ")`

if (age ≥ 18 & registration status = ‘Y’)	`if (age >= 18 and status == 'Y'):`
Display “You can vote”	`print ("You can vote")`
else if (age ≥ 18 & registration status = ‘N’)	`elif (age >= 18 and status == 'N'):`
Display “You must register before you can vote”	`print ("You must register before you can vote")`
else	`else:`
Display “You are too young to vote”	`print ("You are too young to vote")`
endif	`#endif`
Stop	

LECTURE NOTES

```
age = int(input("Enter your age:  "))
status = input("Enter registration status (Y=Yes or N=No):  ")

if (age >= 18 and status == 'Y'):
    print ("You can vote")
elif (age >= 18 and status == 'N'):
    print ("You must register before you can vote")
else:
    print ("You are too young to vote")
#endif
```

Step 4: Testing

```
Enter your age: 19
Enter registration status (Y=Yes or N=No): N
You must register before you can vote
```

Assume user enters age- 19 and registration status- N

```
Enter your age: 15
You are too young to vote
```

Assume user enters age- 15

Combining Relational and Logical Operators

Example 4: The ABC Company pays each sales person an 8% bonus on his or her sales. However, sales people having a sales code of X receive an additional RM 150 bonus when their sales are greater than or equal to RM 10,000; otherwise, they receive an additional RM 125 bonus.

LECTURE NOTES

Step 3: Implementation (Pseudocode Programming)

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	

Read code, sales	`code = input("Enter sales code: ")`
	`sales = float(input("Enter sales: "))`

if (code is ‘X’ or code is ‘x’ & (sales ≥ 10000))	`if (code == 'X' or code == 'x') and sales >= 10000:`
bonus = sales x 0.08	`    bonus = sales * 0.08`
bonus = bonus + 150	`    bonus = bonus + 150`
else if (code is ‘X’ or code is ‘x’ & (sales < 10000))	`elif (code == 'X' or code == 'x') and sales <= 10000:`
bonus = sales x 0.08	`    bonus = sales * 0.08`
bonus = bonus + 125	`    bonus = bonus + 125`
endif	`#endif`
Print ("Bonus is: ", bonus)	`print ("Bonus is: ", bonus)`
Stop	

COMPUTER PROGRAMMING

LECTURE NOTES

```
code = input("Enter sales code: \n")
sales = float(input("Enter sales: "))

if (code == 'X' or code == 'x') and sales >= 10000:
    bonus = sales * 0.08
    bonus = bonus + 150
elif (code == 'X' or code == 'x') and sales <= 10000:
    bonus = sales * 0.08
    bonus = bonus + 125
#endif
print ("Bonus is: ",bonus)
```

Step 4: Testing

```
Enter sales code: x
Enter sales: 10000
Bonus is:  950.0
```

Assume user enters code- x and sales- 10000

```
Enter sales code: X
Enter sales: 1000
Bonus is:  205.0
```

Assume user enters code- X and sales- 1000

LECTURE NOTES

Topic 6: Use of Selection Control Structure

Learning Outcomes:

(c) Identify the use of relational and logical operators in selection control structures (single, dual and multiple)

Equ
al to

COMPUTER PROGRAMMING

LECTURE NOTES

What Are Relational Operators?

- Relational operators are commonly used to form a condition in `if` statement for selection, and `while` or `for` statements for looping.
- Relational operators compares two values and return a Boolean result (**`True`** or **`False`**) .

Description	Relational Operators	Example
Equal to	==	5 == 5 ⟶ True
Not equal to	!=	8 != 8 ⟶ False
Less than	<	3 > 7 ⟶ False
Greater than	>	3 < 7 ⟶ True
Less than or equal to	<=	5 >= 5 ⟶ True
Greater than or equal to	>=	4 <= 6 ⟶ True

Relational Operator VS Assignment Operator

Relational operator

==

To compare two values to determine whether they are equal

Example:

```
if floor == 13:
```

# Test whether `floor` equals `13`

VS

Assignment operator

=

To assign a value to a variable

Example:

```
floor = 13
```

# Assign `13` to `floor`

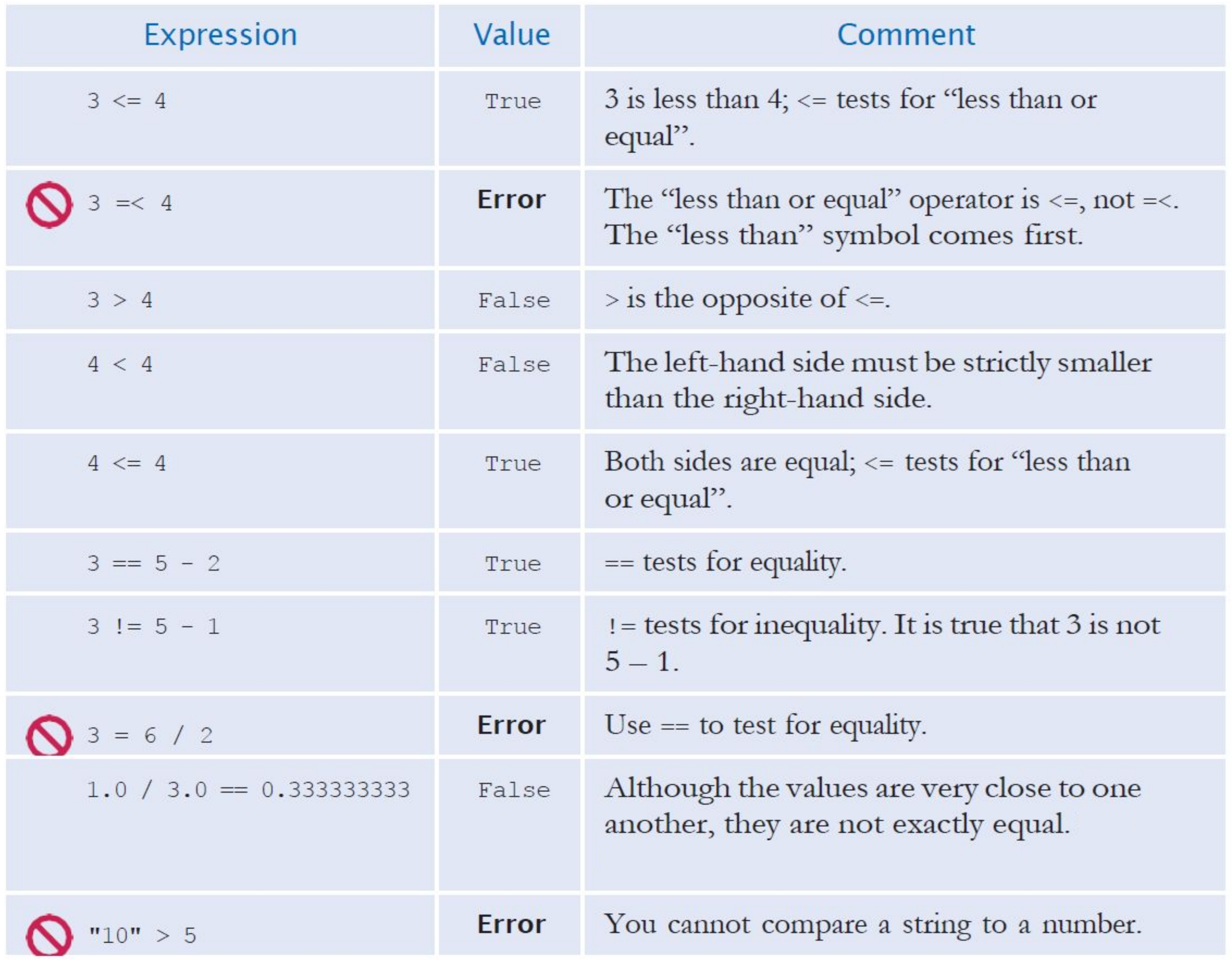
- You must remember to use **==** inside tests and to use **=** outside tests.

COMPUTER PROGRAMMING

LECTURE NOTES

CP115

Relational Operator Examples



Expression	Value	Comment
`3 <= 4`	`True`	3 is less than 4; <= tests for "less than or equal".
🚫 `3 =< 4`	**Error**	The "less than or equal" operator is <=, not =<. The "less than" symbol comes first.
`3 > 4`	`False`	> is the opposite of <=.
`4 < 4`	`False`	The left-hand side must be strictly smaller than the right-hand side.
`4 <= 4`	`True`	Both sides are equal; <= tests for "less than or equal".
`3 == 5 - 2`	`True`	== tests for equality.
`3 != 5 - 1`	`True`	!= tests for inequality. It is true that 3 is not 5 – 1.
🚫 `3 = 6 / 2`	**Error**	Use == to test for equality.
`1.0 / 3.0 == 0.333333333`	`False`	Although the values are very close to one another, they are not exactly equal.
🚫 `"10" > 5`	**Error**	You cannot compare a string to a number.

What Are Logical Operators?

- Logical operators sometimes are referred to as Boolean Operators.
- Logical operators combine multiple conditions and return a Boolean result (`True` or `False`).

Description	Logical Operators	Example
True if both are true	and	(5 > 3) and (3 < 7) → True
True if at least one is true	or	(5 > 3) or (3 > 7) → True
Inverts the Boolean value	not	not (5 > 3) → False

COMPUTER PROGRAMMING

LECTURE NOTES

Precedence of Operators

HIGHEST

LOWEST

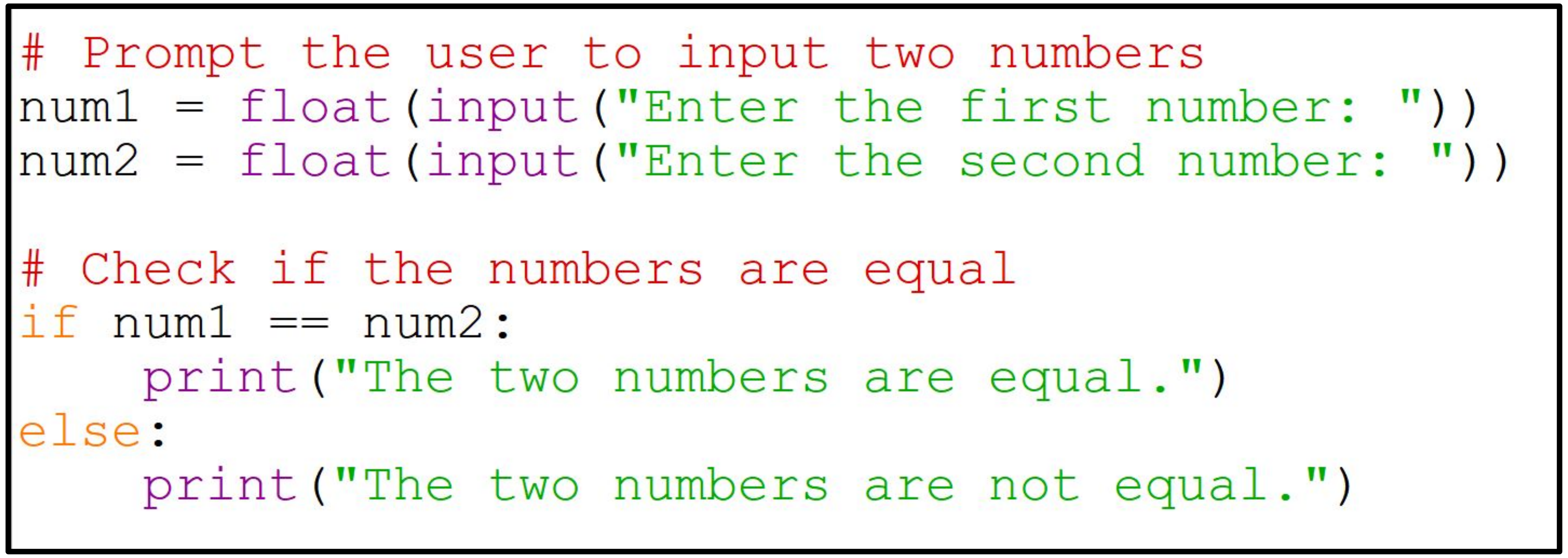
Precedence Level	Operator	Explanation
1 (highest)	( )	Parentheses
2	**	Exponentiation
3	-a, +a	Negative, positive argument
4	* , / , // , %, @	Multiplication, division, floor division, modulus, at
5	+, -	Addition, subtraction
6	< , <=, >, >=, ==, !=	Less than, less than or equal, greater, greater or equal, equal, not equal
7	not	Boolean Not
8	and	Boolean And
9	or	Boolean Or

Example 1: The relational operator (==) usage.

== operator checks if two values are **equal**.

Write a program to determine if two input numbers are equal. If both numbers are the same, print “The numbers are equal”. Otherwise, print “The numbers are not equal”.

Syntax:



```python
# Prompt the user to input two numbers
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Check if the numbers are equal
if num1 == num2:
    print("The two numbers are equal.")
else:
    print("The two numbers are not equal.")
```

Example 2: The relational operator (> , <) usage.

> operator checks if one value is **greater than** another.
< operator checks if one value is **less than** another.

Write a program that allows user to input a number. The program should determine whether the number is positive, negative, or zero, and print the appropriate message.

Syntax:

```
# Prompt the user to input a number
num = float(input("Enter a number: "))

# Determine if the number is positive, negative, or zero
if num > 0:
    print("The number is positive.")
elif num < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

LECTURE NOTES

Example 3: The relational operator (! =) usage.

!= operator checks if two values are **not equal**.

Write a program that asks user to input their age. The program should check if the age is not equal to 18 and print an appropriate message.

Syntax:

```
# Prompt the user to input their age
age = int(input("Enter your age: "))

# Check if the age is not equal to 18
if age != 18:
    print("You are not 18 years old.")
```

Example 4: The logical operator (`not`) usage.

not operator inverts the Boolean value

Write a program that asks user to input a string. The program should print True if the string is not empty, and False if the string is empty. Use the `not` operator in your solution.

Syntax:

```
# Prompt the user to input a string
user_input = input("Enter a string: ")

# Check if the string is not empty using the 'not' operator
if not user_input:
    print("False")  # Empty string
else:
    print("True")   # Non-empty string
```

Explanation:

- The `not` operator is used to check if the string is empty. In Python, an empty string is considered `False`, and any non-empty string is considered `True`.
- If `user_input` is empty, `not user_input` will evaluate to `True`, and the program will print `False` (indicating the string is empty).
- If `user_input` is not empty, `not user_input` will evaluate to `False`, and the program will print `True` (indicating the string is not empty).

LECTURE NOTES

Example 5: The logical operator (`not`) usage.

not operator inverts the Boolean value

Write a program to determine if a number is not equal to 10. If it is not equal to 10, print "The number is not 10". Otherwise, print "The number is 10". Use the `not` operator in your solution.

Syntax:

```
# Input: Get a number from the user
number = int(input("Enter a number: "))

# Check if the number is not equal to 10 using 'not'
if not number == 10:
    print("The number is not 10")
else:
    print("The number is 10")
```

Explanation:

Condition with `not`

- The expression `not number == 10` inverts the condition.
- If `number == 10` is **`True`**, **`not`** makes it **`False`**.
- If `number == 10` is **`False`**, **`not`** makes it **`True`**.