Topic 7: Use of Repetition Control Structure

Learning Outcomes:

(a) Identify types of repetition (**counter-controlled** and sentinel-controlled)

LECTURE NOTES

A series of statements in a loop that are repeated until a certain condition is met.

*Repetition structure also called **looping** control structure

Repetition

Counter-Controlled

(when we know how many times loop body will be executed)

Sentinel-Controlled

(when we don’t know exactly how many times loop body will be executed

Definition

Counter-Controlled Repetition

- **Counter-controlled repetition**, also known as a **definite loop**, is a type of loop that executes a specific number of times.
- It uses a counter variable to control the number of iterations/repetitions.

*** A counter is a numeric variable used for counting something.**

Concept

The counter-controlled has 3 important parts:

1. ***Initialization*** (Name the counter & Set the initial value of the counter variable).
   * any names can be given to the loop counter.
   * initializing means to assign a beginning value to the counter .
   * Counters are usually initialized to the number 0 or 1; however, they can be initialized to any number; depending on the value required by the algorithm.

2. ***Condition*** (Test the counter variable to determine whether the loop should continue or stop).

3. ***Update*** The Counter (Increment/Decrement)
   *means adding a number to the value stored in the counter. The number can be either positive or negative, integer or non integer.

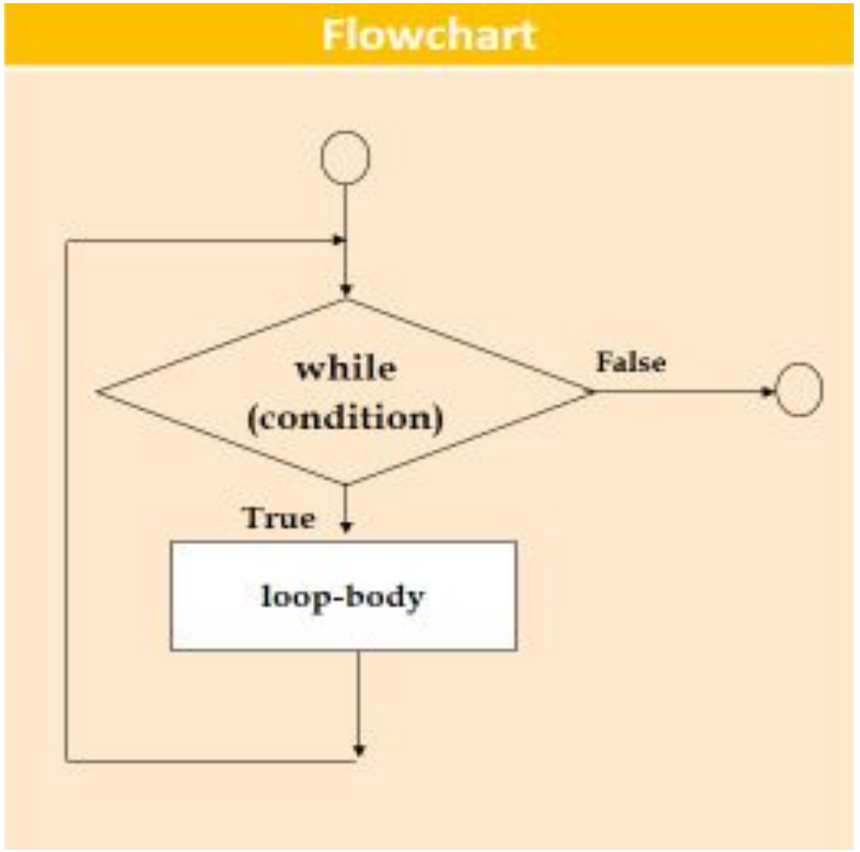
General Format of Pseudocode & Flowchart of a Counter-Controlled Repetition

Pseudocode

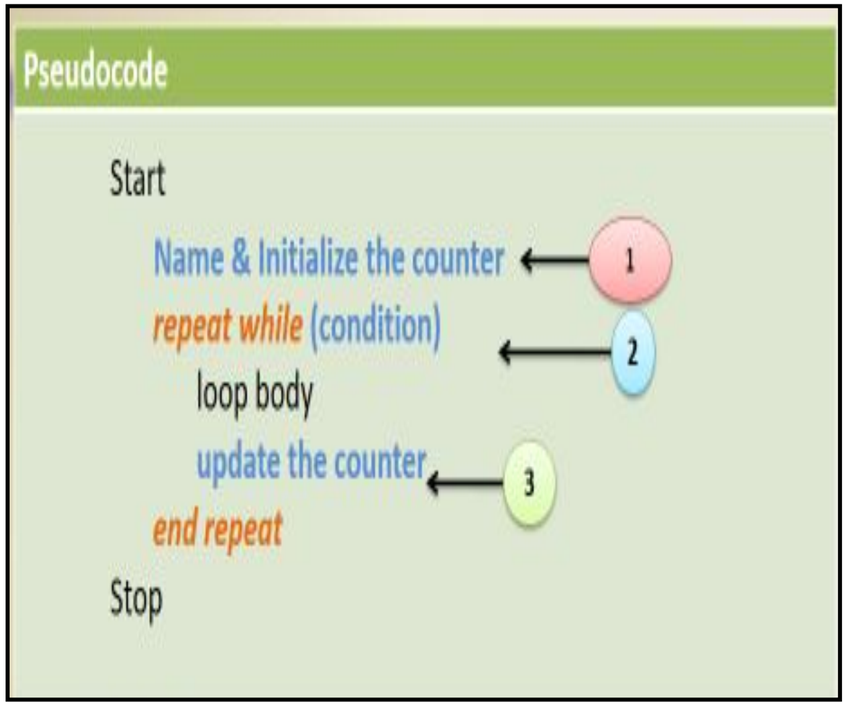
while (condition)
    loop body
end while

Note

- within a looping structure, the condition is first evaluated, if it evaluates to (TRUE), a block of statements called loop body executes and the expression is evaluated again
- the loop body can be sequence or selection

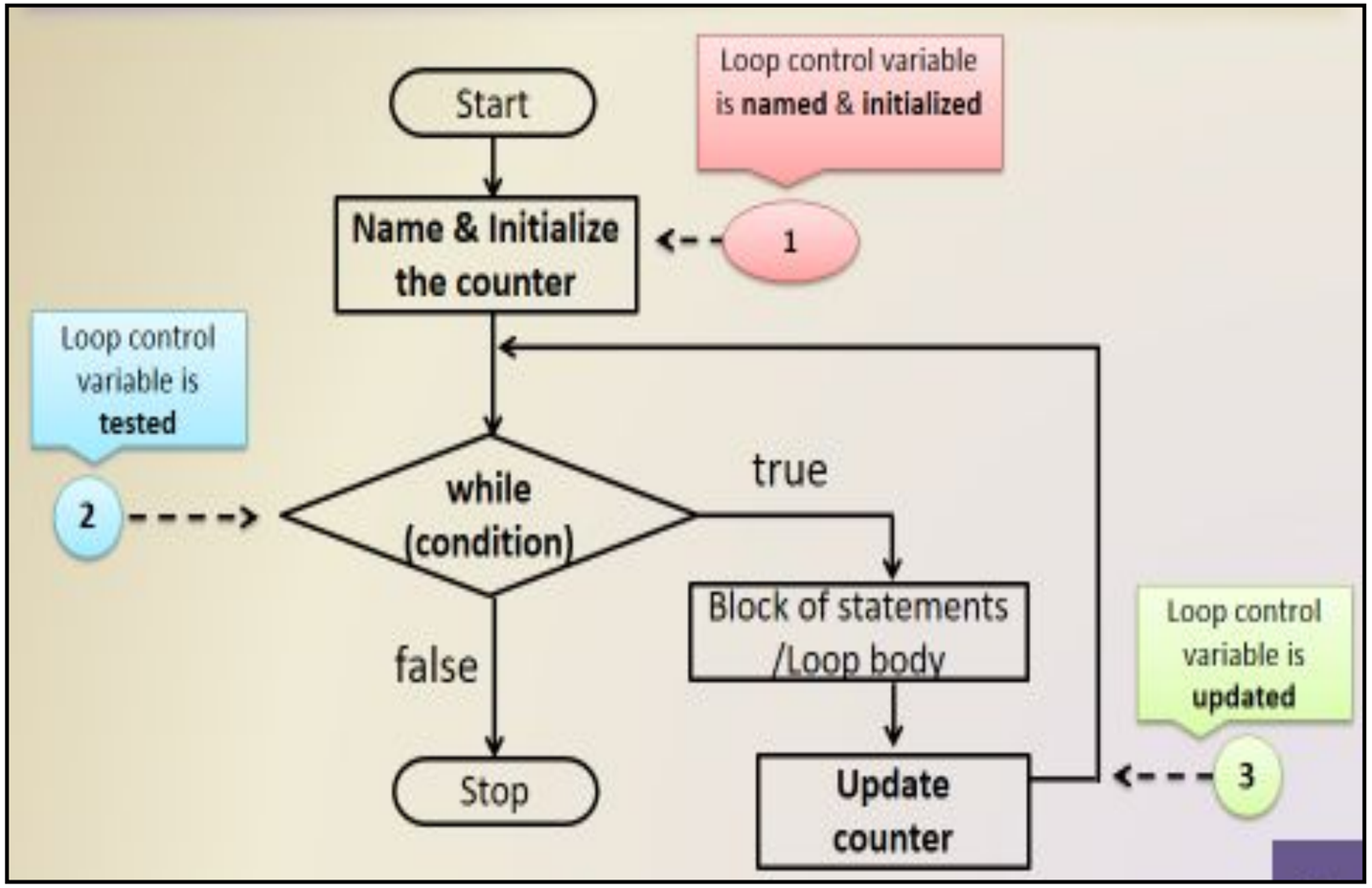


General Format of Pseudocode of a Counter-Controlled Repetition



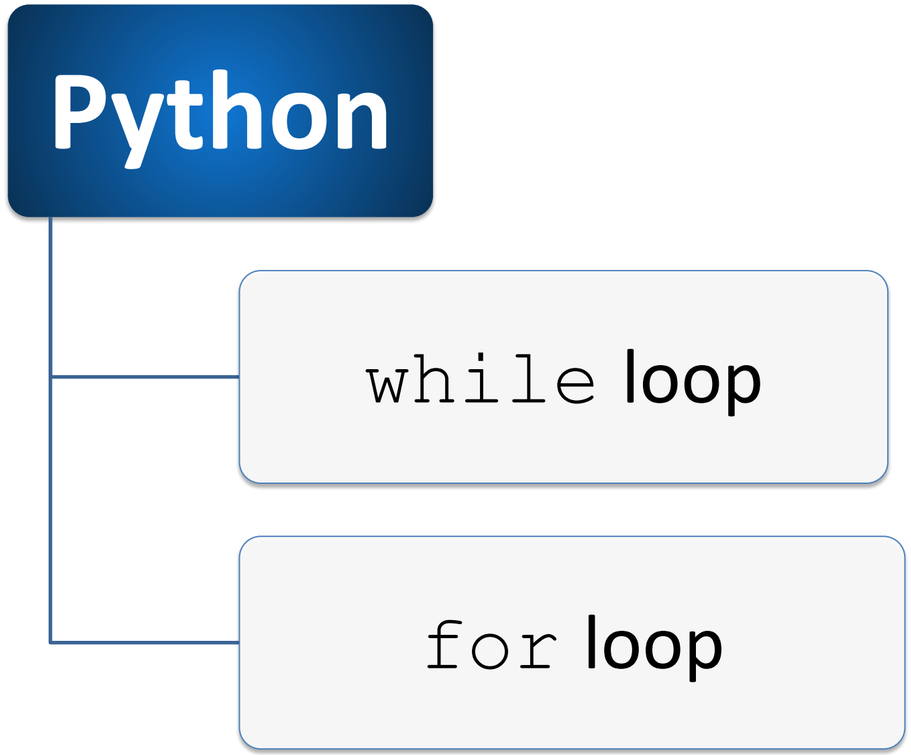
1. **Name & Initialize**
   - the loop counter control variable before the loop begins
2. **Condition**
   - test the loop counter control variable in the **while** condition; if the condition is **true, the loop begins executing**
3. **Update**
   - the value of the loop counter control variable **(increment or decrement)**

General Format of Flowchart of a Counter-Controlled Repetition



Python Loops

Python has two primitive loop commands:



Python Loops

while loop

- With the **while** loop we can execute a set of statements as long as a condition is **true**
- **Syntax:**

```
Name_initialize counter
while condition:
   statements to be processed if the condition is true
   update_counter
#end repeat
```

- **Example 1:**

```
counter= 1   #1.Name & Initialize the counter

while counter <= 10: #2.Condition

   print('Counter is: ', counter)
   counter= counter  + 1 #3.Update the counter
#end repeat
```

while loop

- **Example 2: Calculate the population growth of a certain demographic for a span of 2000 days**

```
time=0    #1.Name & Initialize the counter

population = 1000
growth_rate = 0.21

while population <= 2000: #2.Condition
  population = population + growth_rate * population
  print('Population is: ', population)

  time= time  + 1    #3.Update the counter
#end repeat
```

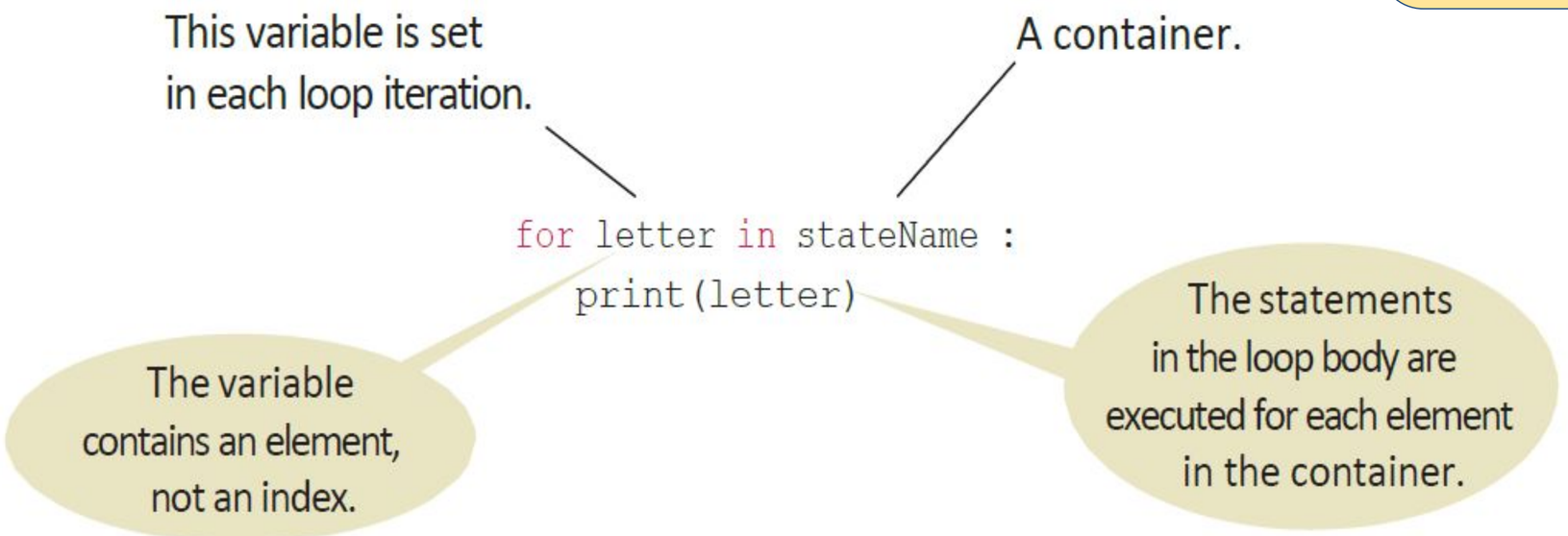
Python Loops

`for` loop

- With the **`for`** loop we can execute a set of statements as long as a condition is **`true`**

Syntax

```
for variable in container :
    statements
```



Example 1:

```
stateName = "Virginia"
for letter in stateName:
    print(letter)
```

Output

```
V
i
r
g
i
n
i
a
```

Python Loops

`for` loop

- With the **`for`** loop we can execute a set of statements as long as a condition is **`true`**
- **Syntax:** `for variable in range(start, stop, step):`

Syntax

```
for variable in range(...) :
    statements
```

This variable is set, at the beginning of each iteration, to the next integer in the sequence generated by the `range` function.

The `range` function generates a sequence of integers over which the loop iterates.

```
for i in range(5) :
    print(i)  # Prints 0, 1, 2, 3, 4
```

With one argument, the sequence starts at 0. The argument is the first value NOT included in the sequence.

```
for i in range(1, 5) :
    print(i)  # Prints 1, 2, 3, 4
```

With two arguments, the sequence starts with the first argument.

```
for i in range(1, 11, 2) :
    print(i)  # Prints 1, 3, 5, 7, 9
```

With three arguments, the third argument is the step value.

for loop

- **Example 2:**

```
for i in range(3): # Starts at 0, stops before 3
    print(i)
```

Output

```
0
1
2
```

- **Example 3:**

```
for i in range(8, 0, -2): # Starts at 8, decrements by 2
    print(i)
```

Output

```
8
6
4
2
```

- **Example 4:**

```
for i in range(1, 5): # Start at 1, end before 5
    print(i)
```

Output

```
1
2
3
4
```

for loop

- **Example 5:**

```
for num in range(3, -2, -1):
    print(num)
```

Output

```
3
2
1
0
-1
```

Explanation:

- **range(3, -2, -1)**: The range function takes three arguments:
  - start: 3 (the loop starts at 3).
  - stop: -2 (the loop ends before reaching -2).
  - step: -1 (the loop decrements by 1 on each iteration).

- **Example 6:**

```
for i in range(5):
    print("Hello, World!")
```

Output

```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

Explanation:

1. **range(5)**: This generates a sequence of numbers from 0 to 4.
2. **for i in range(5):**: Iterates through each value in the sequence (0, 1, 2, 3, 4).
3. **print("Hello, World!")**: Executes the print statement for each iteration.

`for` loop examples

Loop	Values of `i`	Comment
`for i in range(6) :`	0, 1, 2, 3, 4, 5	Note that the loop executes 6 times.
`for i in range(10, 16) :`	10, 11, 12, 13, 14, 15	The ending value is never included in the sequence.
`for i in range(0, 9, 2) :`	0, 2, 4, 6, 8	The third argument is the step value.
`for i in range(5, 0, -1) :`	5, 4, 3, 2, 1	Use a negative step value to count down.

Exercise 1: Calculate the sum of 5 numbers entered by the user

Control Structure: Repetition – Sequence

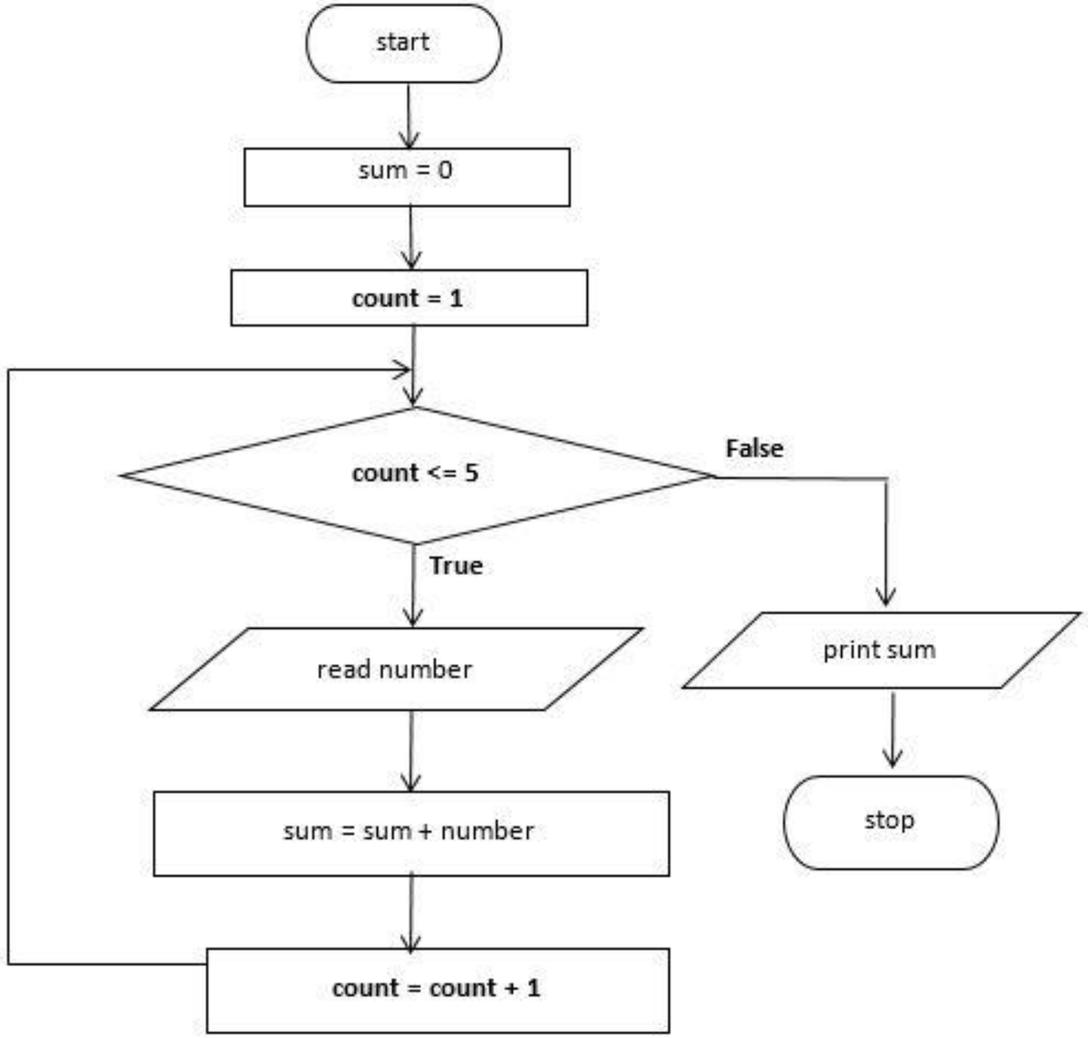
Step 1: Problem Analysis

Input	Process	Output
5 number	**To repeat** enter number and to **calculate** the sum based on the numbers entered for five times	sum

Step 2: Design a Solution (Pseudocode)

```
Start
    count = 1
    sum = 0
    while (count <= 5)
        Read number
        sum = sum + number
        count = count + 1
    end repeat
    print number
Stop
```

Step 2: Design a Solution (Flowchart)



Counter-Controlled Table

count =1	while (count <= 5)	Read number	sum = sum + number	count = count + 1
1	True	1	sum = 0 + 1 sum = 1	2
2	True	2	sum = 1 + 2 sum = 3	3
3	True	3	sum = 3 + 3 sum = 6	4
4	True	4	sum = 6 + 4 sum = 10	5
5	True	5	sum = 10 + 5 sum = 15	6
6	False	end while		

sum will be display after end while (finished repeat)

Step 3: Implementation

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
count = 1	`count = 1`
sum = 0	`sum = 0`
***while* (count <= 5)**	`while count < 5:`
read number	`number = int(input("Enter a number: "))`
sum = sum + number	`sum = sum + number`
count = count + 1	`count += 1`
end while	`#end while`
print sum	`print("Sum :", sum)`
Stop	

Step 3: Implementation

while loops

```
count = 1
sum = 0
while count <= 5:
    number = int(input("Enter a number: "))
    sum = sum + number
    count += 1
#end while
print("Sum: " , sum)
```

for loops

```
sum = 0

for count in range (1,6,1):
    number = int(input("Enter a number: "))
    sum = sum + number

print("Sum: " , sum)
```

Step 4: Testing

```
Enter a number: 3
Enter a number: 6
Enter a number: 1
Enter a number: 8
Enter a number: 2
Sum:  20
```

Testing

```
Enter a number: 2
Enter a number: 3
Enter a number: 7
Enter a number: 8
Enter a number: 1
Sum:  21
```

Exercise 2: Calculate the area of four circles only if the radius entered is a positive value.

Control Structure: Repetition – Selection if

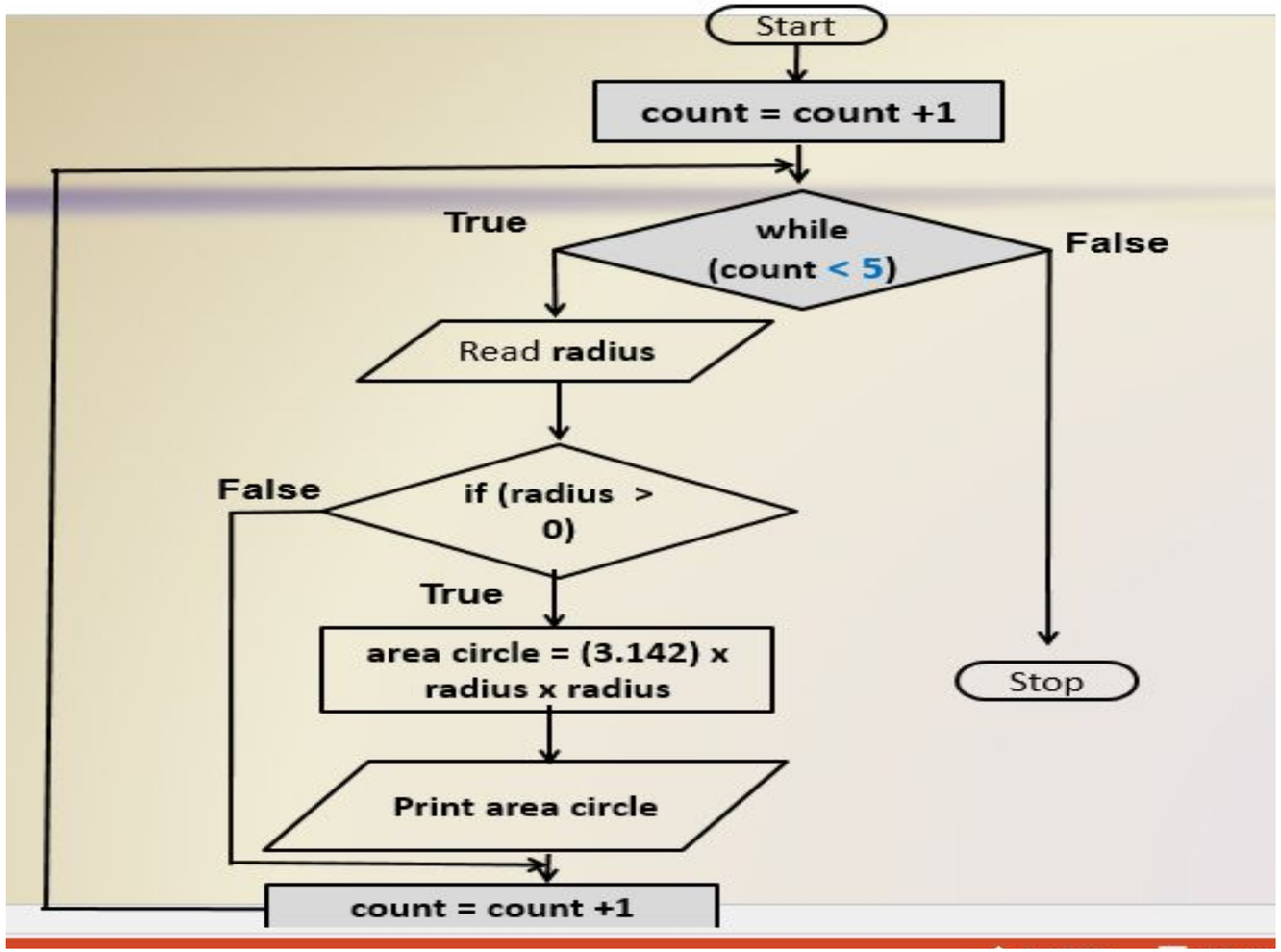
Step 1: Problem Analysis

Input	Process	Output
radius	**Determine** the radius **and calculate** area of a circle **based on** the radius and *repeat for 4 times*	area circle

Step 2: Design a Solution (Pseudocode)

```
Start
    count = 1
    while (count < 5)
        Read radius
            if (radius > 0)
                area circle =  (3.142) x radius x radius
                Print area circle
    count = count + 1
    end while
Stop
```

Step 2: Design a Solution (Flowchart)



Counter-Controlled Table

count =1	while (count < 5)	Read radius	if (radius > 0)	area circle = (3.142) x radius x radius	Print area circle	count = count + 1
1	True	89	True	area circle = (3.142) x 89 x 89 = 24887	24887	2
	True	21	True	area circle = (3.142) x 21x 21 = 1385	1385	3
	True	0	False	-	-	4
	True	-8	False	-	-	5
	False	**end while**				

Step 3: Implementation

Step 2: Design a Solution (Pseudocode)	Step 3: Implementation
Start	
count = 1	`count = 1`
***while* (count < 5)**	`while count < 5:`
read radius	`radius = float(input("Enter a positive radius: "))`
if (radius > 0)	`if radius >= 0:`
area circle = (3.142) x radius x radius	`areacircle = 3.142 * radius * radius`
print area circle	`print("Area of the circle:", areacircle)`
count = count + 1	`count += 1`
end while	**`#end while`**
Stop	

Step 3: Implementation

while loops

```
count = 1 #1.Name & Initialize the counter

while count < 5: #2.Condition

   radius = float(input("Enter a positive radius: "))
   if radius >= 0:
        areacircle = 3.142 * radius * radius
        print("Area of the circle:", areacircle)
        print("")
   count += 1 #3.Update the counter

  #End while
```

for loops

```
for count in range (0,4,1): #initialization,condition,update counter



    #read input
    radius = float(input("Enter a positive radius:"))


    #Selection if
    if radius >= 0:
        areacircle = 3.142 * radius * radius
        #Print output
        print("Area of the circle:", areacircle, "\n ")

#end repeat
```

Step 4: Testing

Testing

```
Enter a positive radius: 12
Area of the circle: 452.448

Enter a positive radius: 56
Area of the circle: 9853.312

Enter a positive radius: -9
Enter a positive radius: 23
Area of the circle: 1662.1179999999997
```

```
Enter a positive radius:5
Area of the circle: 78.55

Enter a positive radius:9
Area of the circle: 254.50199999999998

Enter a positive radius:2
Area of the circle: 12.568

Enter a positive radius:1
Area of the circle: 3.142
```

Create a program that calculates and displays the pay of 5 employees given hours worked and hourly pay rate.

Sample input

hours worked	hourly pay
1	1
2	2
3	3
4	4
5	5

Output:

1

4

9

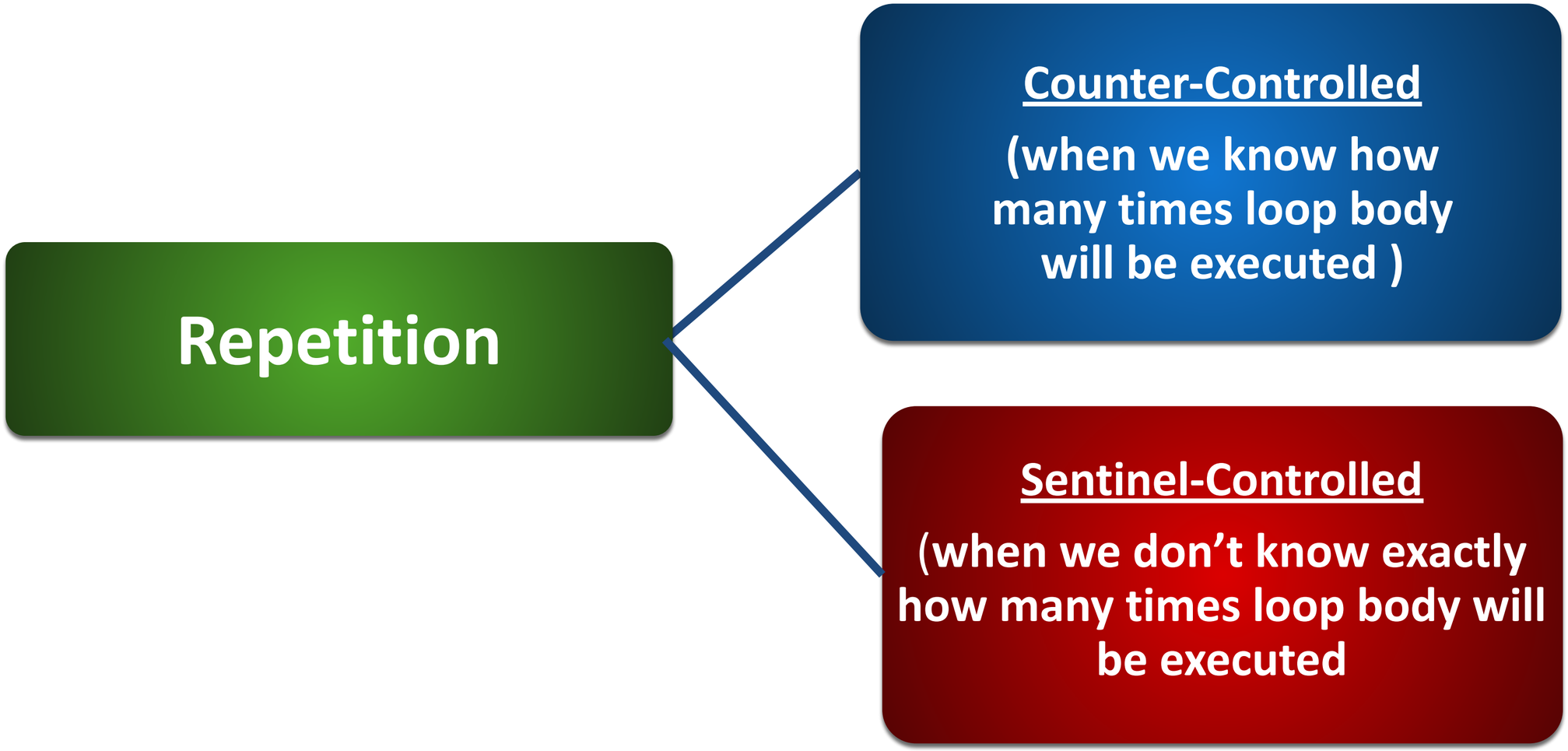
16

25

Topic 7: Use of Repetition Control Structure

Learning Outcomes:

(a) (a) Identify types of repetition (counter-controlled and **sentinel-controlled** (2nd hour)


Repetition
Counter-Controlled
(when we know how many times loop body will be executed)
Sentinel-Controlled
(when we don’t know exactly how many times loop body will be executed

Definition

Sentinel-Controlled Repetition

- A **sentinel**-controlled loop **uses a sentinel value to stop a loop**.

* **Sentinel value** means user uses special input value to signifies the end of the input

Concept

The sentinel-controlled has 3 important parts:

1. **Prime Input –** Initializes the loop's control variable

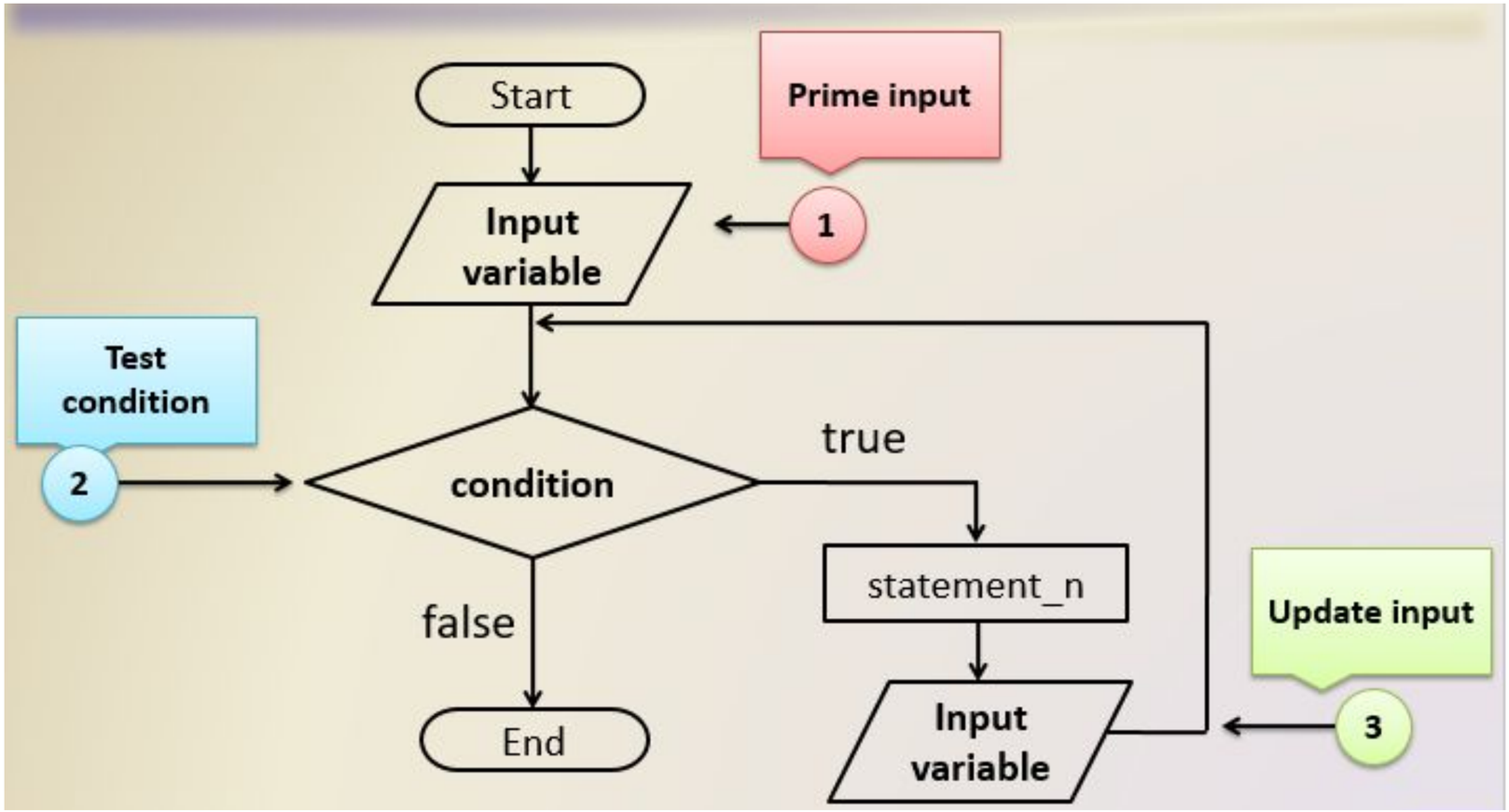
2. **Condition –** Test the loop control variable whether the loop should continue or stop

3. **Update Input–** Could be increment or decrement where the value of the loop control variable is updated

General Format of Pseudocode of a Sentinel Controlled REPETITION

Pseudocode	
Start **Prime input** **`while` condition** statement_n **Update input** **`endwhile`** End	1. **Prime input**; **provides a starting value** of loop control variable that will control the loop. 2. **Test** the loop control variable in the **`while`** condition; if the condition is **true**, **the loop begins executing**. 3. **Update input**; updates the value of the loop control variable.

General Format of Flowchart of a Sentinel Controlled REPETITION



Problem Statement

Problem statement

Create a program that calculates and displays the area of a circle. It allows the user to enter the **radius** repeatedly, **then 0 to end the program**.

Step 1 : Problem Analysis

Problem Analysis	
Input	**radius**
Process	Repeat calculate **circle area** based on radius **until user enters 0**
Output	**circle area**

Step 2 : Design a solution (Pseudocode)

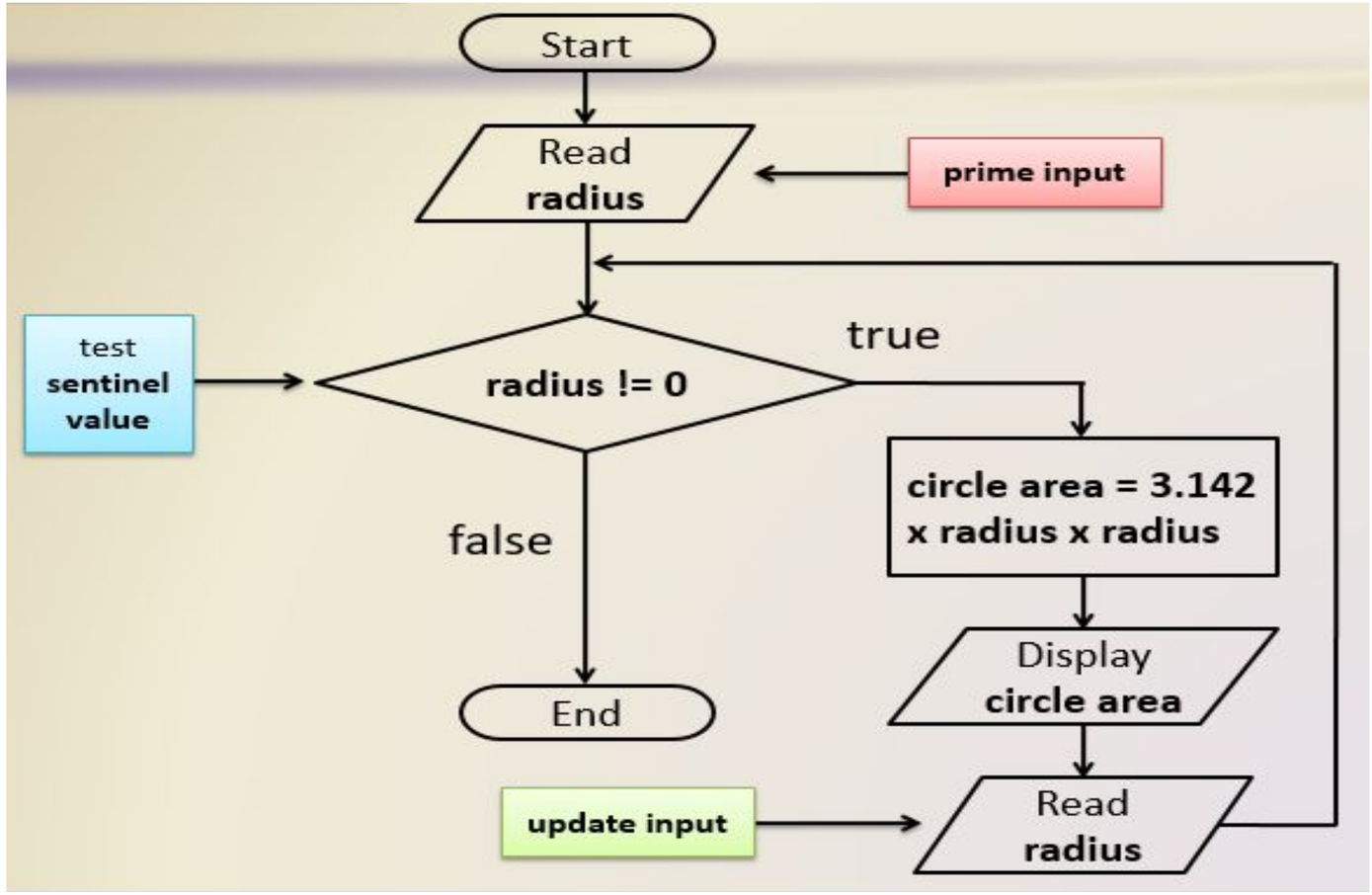
```
Start

    Read radius
    while radius != 0
        Calculate circle area:
            circle area = 3.142 x radius x radius
        Display circle area
        Read radius
    endwhile

End
```

prime input

test **radius** for **sentinel value**

update input

Step 2 : Design a solution (Flowchart)



Step 3: Implementation (while syntax)

```
# Start

# Read radius
radius = float(input("Enter the radius of the circle (Enter 0 to stop): "))

# while loop to continue until radius is 0
while radius != 0:
    # Calculate circle area
    circle_area = 3.142 * radius * radius

    # Display circle area
    print("The area of the circle is:", circle_area)

    # Read radius again to check if the loop should continue
    radius = float(input("Enter the radius of the circle (Enter 0 to stop): "))

# End
```

Desk-check

radius	radius != 0	circle area = 3.142 x radius x radius	circle area	radius
9	true	3.142 x 9 x 9	**254.502**	7
	true	3.142 x 7 x 7	**153.958**	5
	true	3.142 x 5 x 5	**78.55**	0
	false	endwhile		

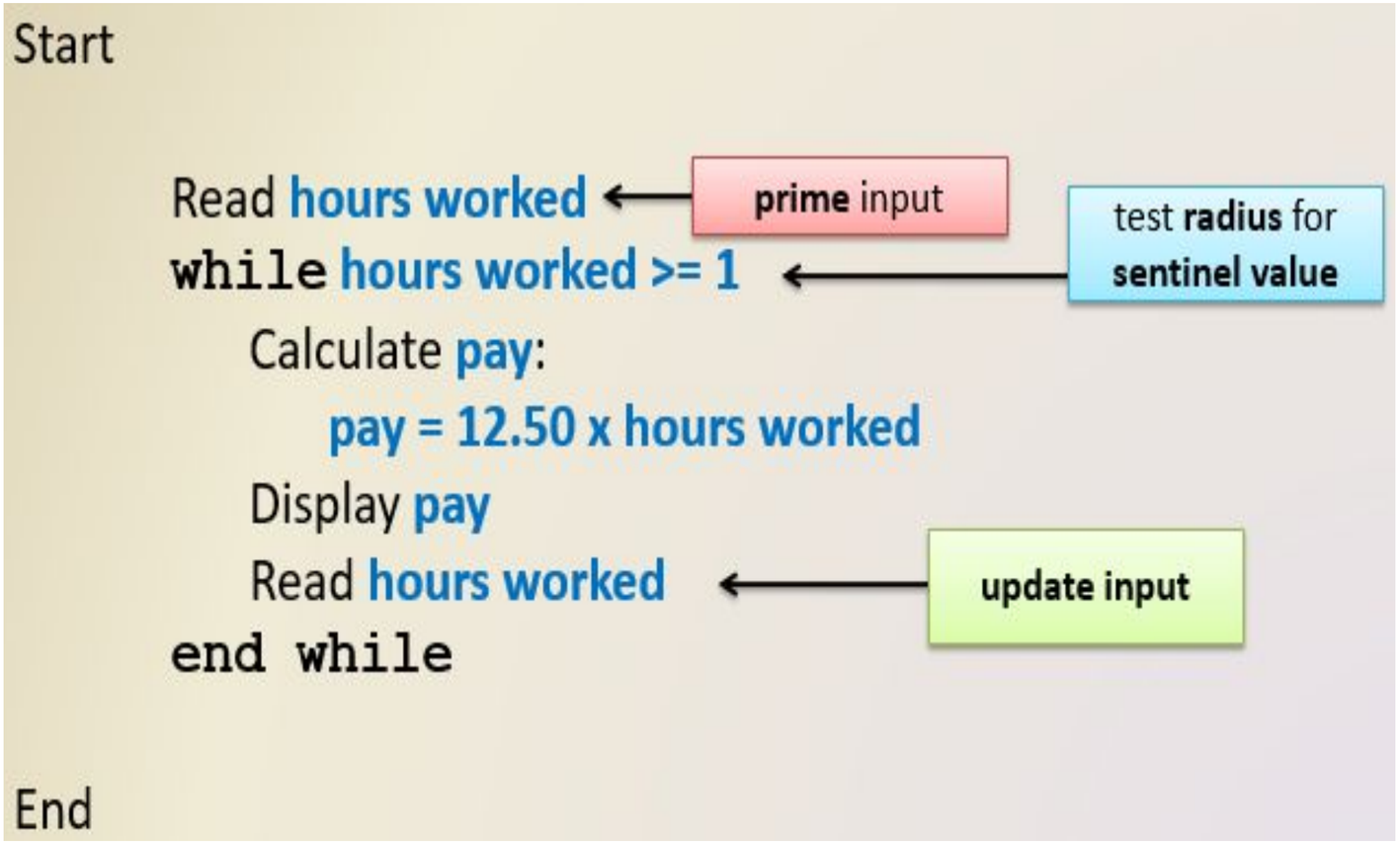
Problem statement

The owner of the Totally Sweet Shoppe hired part-time employees, who earn **RM12.50 per hour**. The owner wants a program that calculates and displays the **pay based on hours worked for as many employees as needed** without having to run the program more than once, and the **hours worked will always be positive numbers and greater than 0**.

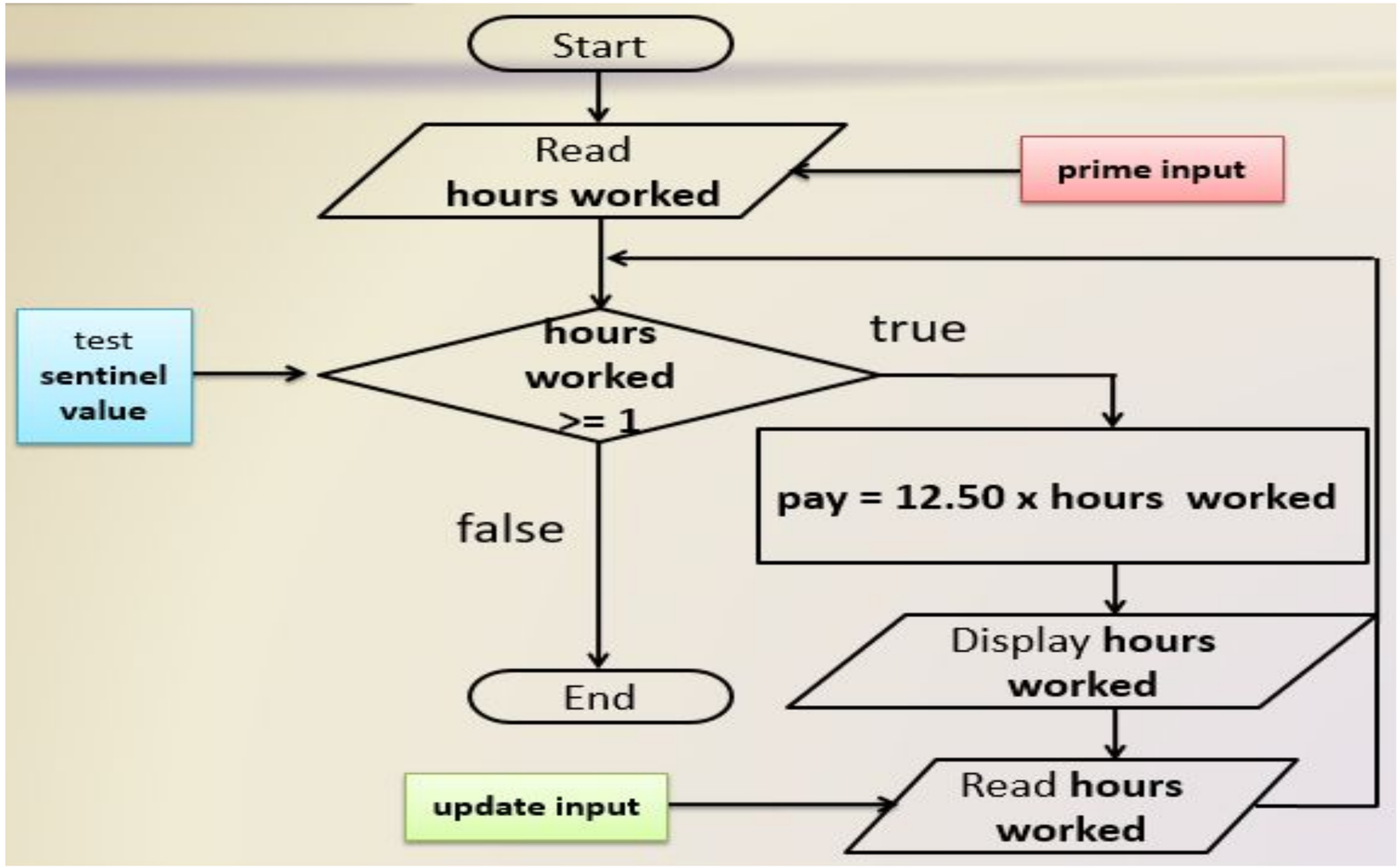
Step 1: Problem Analysis

Problem Analysis	
Input	**hours worked**
Process	Calculate **pay** and repeat while **hours worked greater than 0**
Output	**pay**

Step 2: Design a Solution (Pseudocode)



Step 2: Design a Solution (Flowchart)



Step 3: Implementation

```
while True:
    hours = float(input("Enter the hours worked by the employee (or
type a negative number or 0 to stop): "))

    if hours <= 0:
        print("Program has ended.")
        break

    if hours >= 1:
        pay = hours * 12.50
        print(f"Employee's pay: RM{pay:.2f}")
    else:
        print("Please enter a valid number of hours (1 or more).")
```

Desk check

hours worked	hours worked >= 1	pay = 12.50 x hours worked	pay	hours worked
40	true	12.50 x 40	**500**	55
	true	12.50 x 55	**687.50**	60
	true	12.50 x 60	**750**	75
	true	12.50 x 75	**937.50**	65
	true	12.50 x 65	**812.50**	-1
	false		endwhile	

Control Flow Statements

`Break`

The break statement is used to exit the loop prematurely when a certain condition is met.

`Continue`

The continue statement skips the current iteration and proceeds to the next iteration of the loop.

Break

Print the numbers 1 to 10. The program will stop when the number's square number is 25

```
# Using break to exit the loop
i = 1
while (i <= 10):
    if i*i == 25:
        break
    print(i)
    i = i + 1
```

Output

```
1
2
3
4
5
```

i	i < 10	i*i = = 25	print(i)	i = i + 1
1	True	1 == 25 False	1	2
2	True	4 == 25 False	2	3
3	True	9 == 25 False	3	4
4	True	16 == 25 False	4	5
5	True	25 == 25 True	5	-
-	-	-	-	-

Continue

Print the numbers 0 to 5. The program will not print when the number's square number is 9

```
# Using continue
 for i in range(6):
     if i*i == 9:
         continue
     print(i)
```

Output

```
0
1
2
4
5
```

i	i < 6	i*i = = 9	print(i)	i increment
0	True	0 == 9 False	0	1
1	True	1 == 9 False	1	2
2	True	4 == 9 False	2	3
3	True	9 == 9 True	-	4
4	True	16 == 9 False	4	5
5	True	25 == 9 False	5	6
6	False	-	-	-