"A program without a loop is not worth writting."
- A. Perlis

ITI 1520
Module 3: Loops

General Concepts:

- while Loops
- for Lopps

GeneralObjective: You will implement programs using loops.

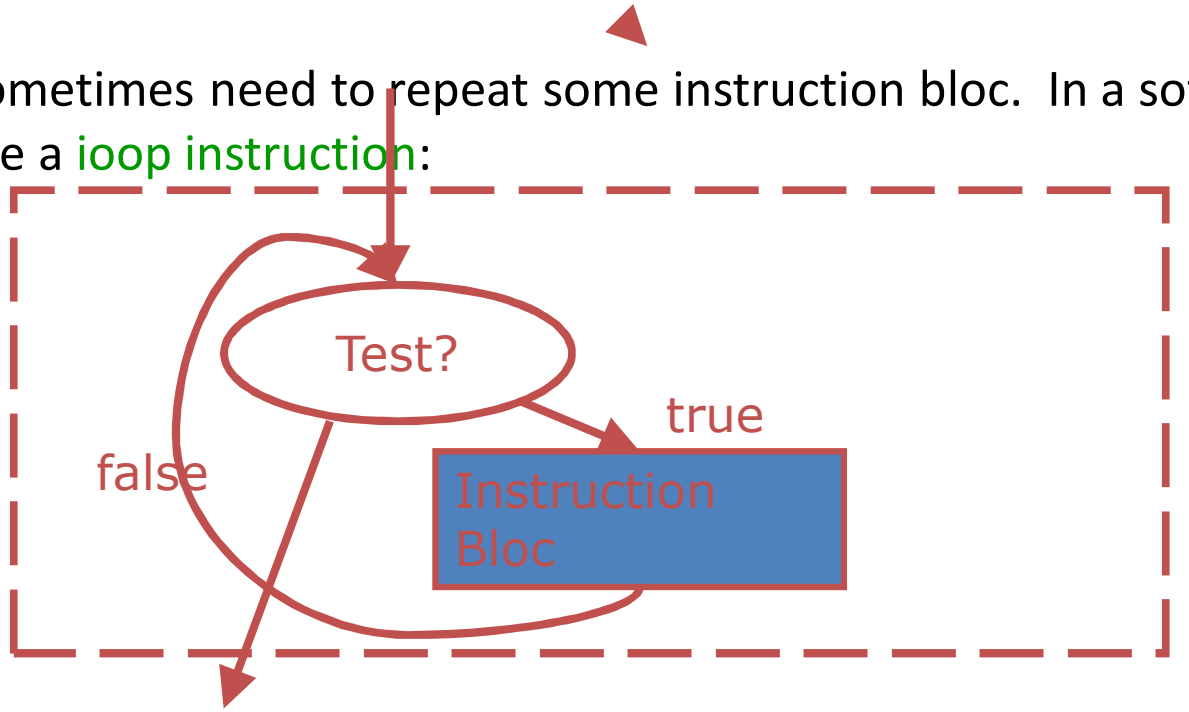
Learning achievements:

1. Solve problems inPython using while loops
2. Solve prolems in Python using for loops

Theme 1. while Loops

Sub-theme: loop instruction – Software model

- We sometimes need to repeat some instruction bloc. In a software model we use a ioop instruction:



- The instruction bloc in the loop is repeated until the test becomes false.

Conception of a loop instruction

1. Initialization
   - Is there any variable to initialize?
   - Those variables will be updated in the loop.
2. Condition to test
   - A condition to determine if we need to repeat the loop instruction bloc or not.
3. Instruction bloc of the loop
   - What arethe steps to be repeated?
   - Finite loop – The number of times the loop is repeated is known – use normally a counter variable.
   - Infinite loop – The number of times it loops is not known – use normally un flag variable.

Sub-theme: while Loop in Python

```
# initialize variables
while test:
    # Instructions bloc
```

- If test is true, execute the instruction bloc, and repeat the processus. If it is false, stop the loop.
- In the instruction bloc you need to modify the variable values so that the test becomes false in the last iterration. Otherwise it will loop infinitly.

Exercise 1: Sum from 1 to N
Loop “finite”

DATA: N

INTERMEDIAIRY: ?

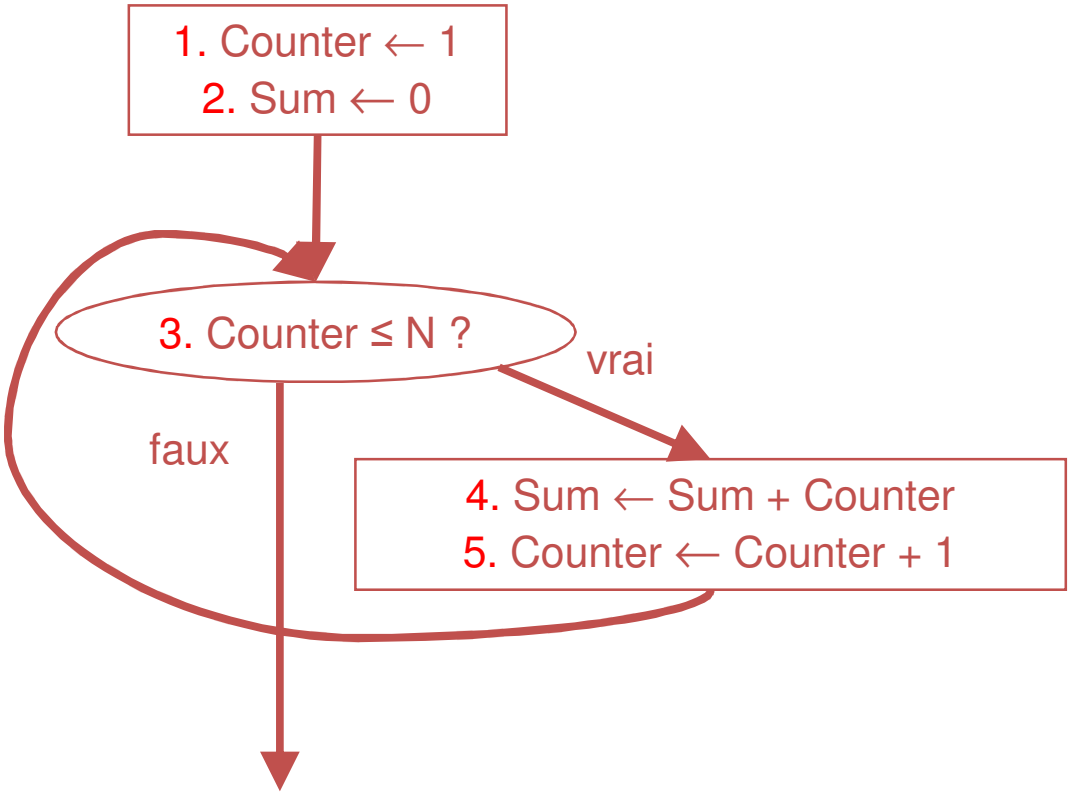
RESULT: Sum

HEADER: Sum ← Sum from1to N(N)

MODULE: ?

Exercise 1 – *Solution*: Sum from 1 to N

DATA: N *(a positive number)*
INTERMEDIAIRY: Counter *(counter from to N)*
RESULT: Sum *(sum of integers from 1 to N)*
HEADER: Sum ← Sum1toN)
MODULE:

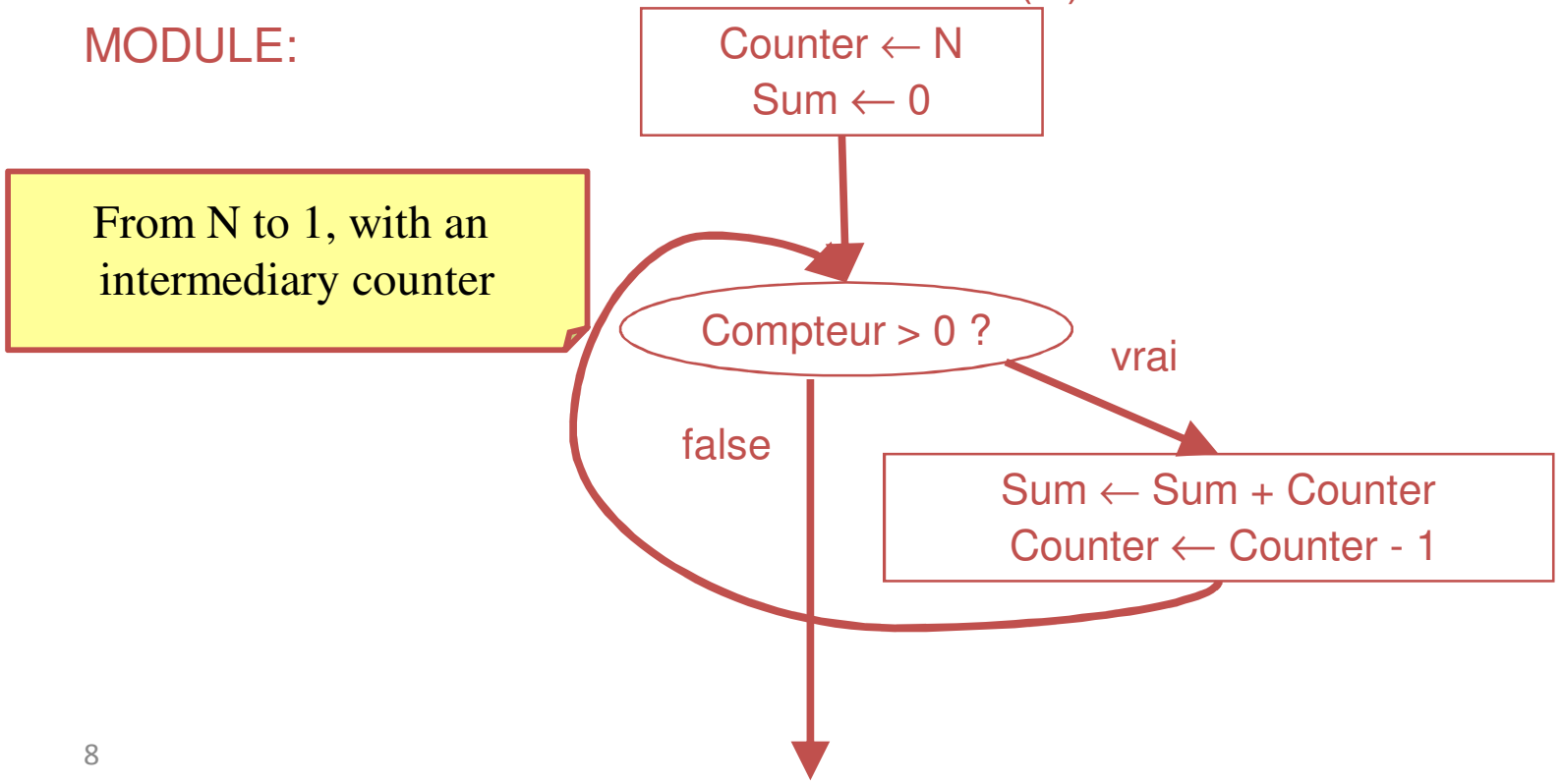


Exercise 1: Somme1àN(3) Trace

Instructions	N	Counter	Sum
Init.	3	?	?
1. Counter ← 1		1	
2. Sum ← 0			0
3. Counter ≤ N ? True (Vrai)			
4. Sum ← Sum+Counter			1
5. . Counter ← Counpter + 1		2	
3. Conter ≤ N ? True (Vrai)			
4. Sum ← Sum+Counter			3
5. Counter ← Counter + 1		3	
3. Counter ≤ N ? True (Vrai)			
4. Sum ← Sum+Counter			6
5. Counter ← Counter + 1		4	
3. Counter ≤ N ? False (Faux)			

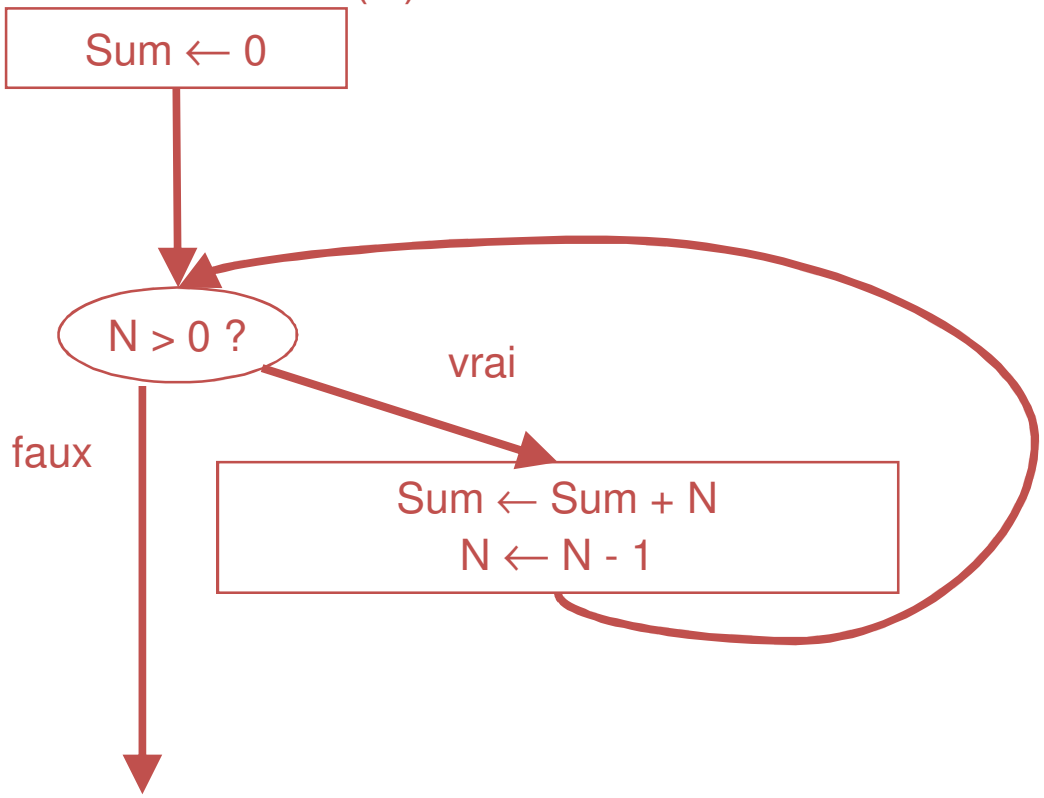
Exercise 1: Sum from 1 to N: Other flavor (I)

DATA: N *(a positive integer)*
INTERMEDIAIRY: Counter *(counter from N to 1)*
RESULT: Sum *(sum of integers from 1 to N)*
EN-TÊTE: Sum ← Sum1toN(N)
MODULE:



Exercice 1: Somme de 1 à N: Autre variante (II)

DATA: N *(a positive integer)*
INTERMÉDIAIRY: *(None)*
RESULTAT: Sum *(sum of integers from 1 to N)*
EN-TÊTE: Sum ← Sum1toN(N)
MODULE:



Attention:
N (*given*) is modified
locally only.
The approch functions,
but can be confusing.

Exercise 1 Translate the algorithm of sum1toN in Python

```
n = int(input(« Please enter n: "))

sum=0
counter = 1    # to count from 1 to n

while counter <= n:
   sum = sum + counter  #accumule the sum
   counteur = counteur + 1

print(sum)
```

Exercise 2: Product of 1 to N

Develop a Python program to compute the product 1 * 2 * … * n. read n from the keyboard. Print out the result.

What is the difference between product from 1 to n and et factorial(n)?
n! = 1 * 2 * … * n (n >=0)
Almost the same but by definition 1! = 1, and 0! = 1.
The product cannot be computed for n=0. n>=1.

Exercise 2 - *Solution*

```
n = int(input(« Please enter n: "))

product = 1
counter = 1
while counter <= n:
  product = product * counter
  counter = coinpter + 1
print(product)
```

Exercise 1 and 2 alternative Solution– with reusable functions

```
def sum1toN(n):
 sum=0
 counter = 1
 while counter <= n:
  sum = sum + counter
 counter = counter +1
 return(sum)
```

```
def product1toN(n):
 product = 1
 counter = 1
 while counter <= n:
   product = product * counter
 counter = counter + 1
 return(product)


n = int(input(“Please enter n:"))

print(‘The sum is', sum1toN(n))
print(‘The product
is',product1toN(n))
```

Exercise 3: Loop “undefined” Integer part of the logarithm

DATA: **aNumber (number)**
divisor (divisor – base dofu log)

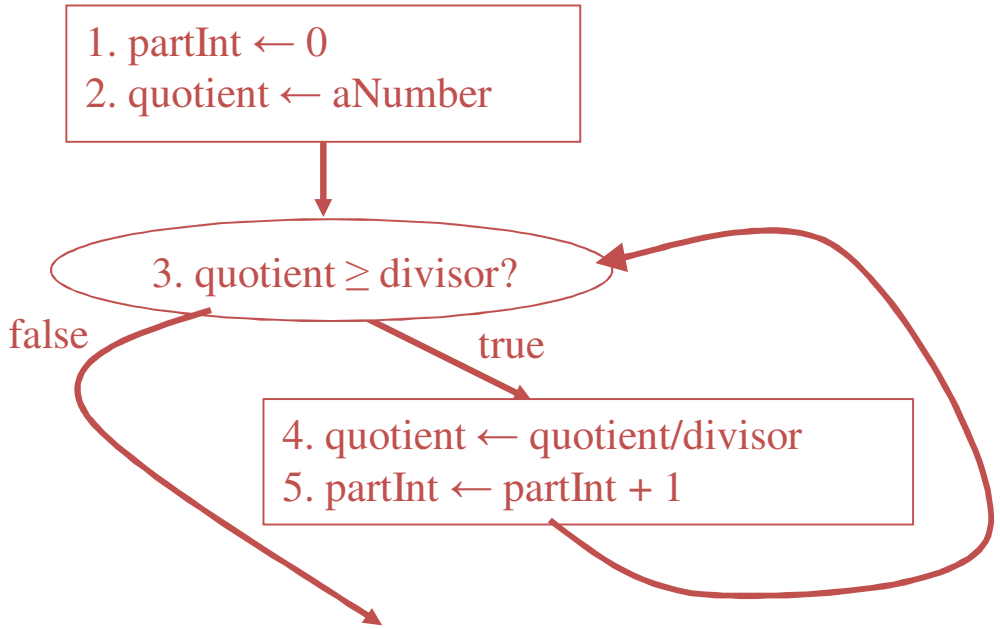
RESULT: **partInt (integer part of the log, number of times that a Number is divisible by divisor)**

INTERMEDIARY: **quotient (quotient of the division)**

HYPOTHESES: **aNumber and the divisor are positive.**

HEADER: **partInt ← logInt (aNumber, divisor)**

MODULE:



Exercise 3 – *Solution*: Integer part of the logarithm

```
# Integer part of the logarithm

aNumber = int(input(« Please enter n "))

diviseur = int(input(« Please enter the base of the
logarithm "))

partInt = 0
quotient = aNumber

while (quotient >= divisor):
 quotient = quotient / divisor
 partInt = partInt + 1

print(partInt)
```

Exercise 3 – *Solution, variance with function*: Integer part of the logarithm

```
# Integer part of the logarithm function

def logInt(aNumber, divisor):
  partInt = 0
  quotient = aNumber
  while (quotient >= divisor):
    quotient = quotient / divisor
    partInt = partInt + 1
 return(partInt)

print(logInt(100,10))
print(logInt(100,2))
print(logInt(127,2))
```

Question:

What does the following Python code display?

```
n = 6
i = 1
result = 0
while (i <= n):
  result = result + i
  i = i + 2
print(result)
```

Possible responses (**choose one**):

a) 6

b) 21

c) 9

d) erreur

Question – *Solution*:

What does the following Python code display?

```
n = 6
i = 1
result = 0
while (i <= n):
   result = result + i
   i = i + 2
print(result)
```

Correcte response c)

Explanation: 1 + 3 + 5 = 9

Possibles responses (choose one):

a) 6

b) 21

c) 9

d) erreur

Theme 2. Boucles for

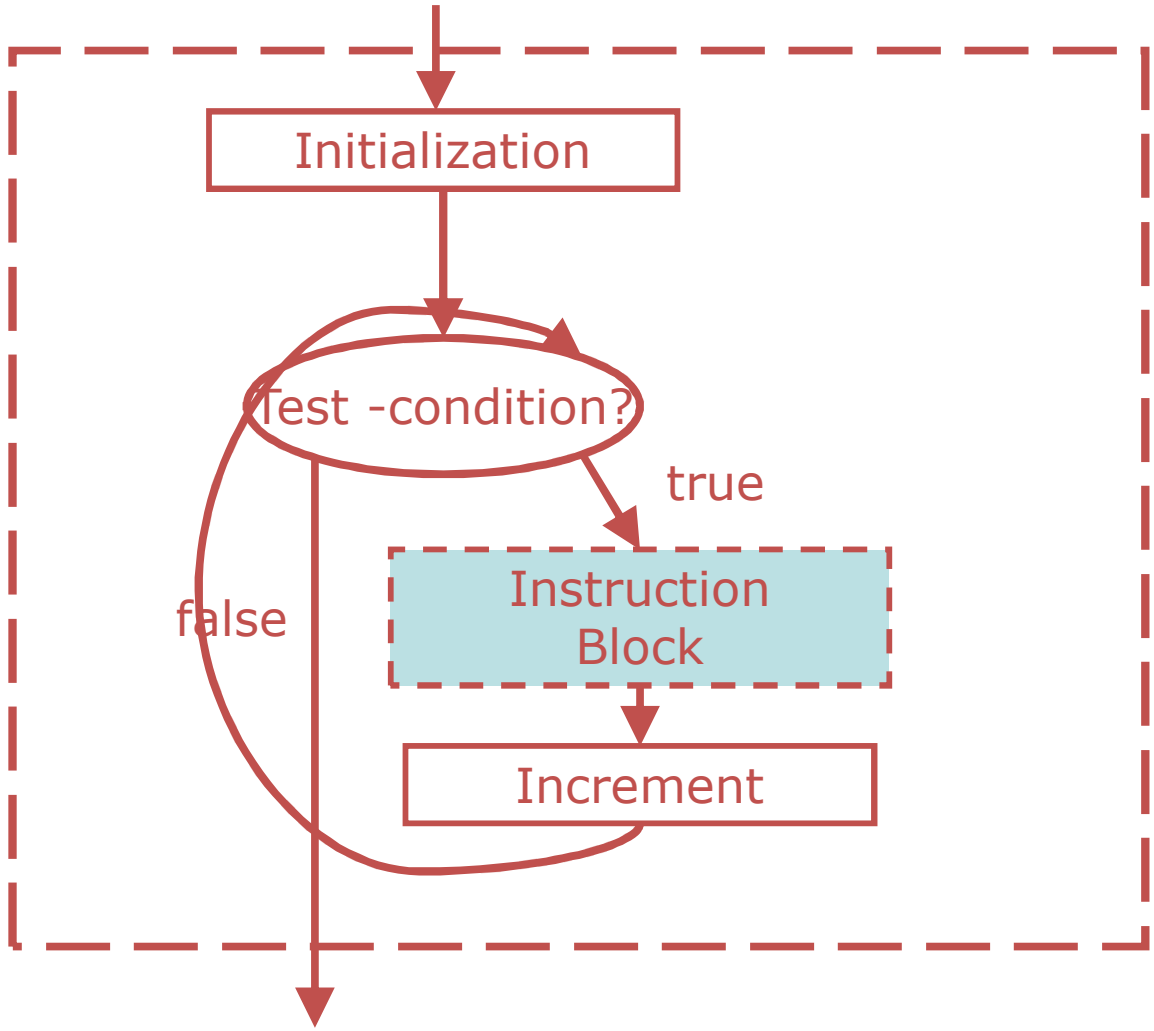
Sub-theme: FOR loop in Python

- Python offers another form of loop, that can replacer a WHILE loop when we know ahead of time how many times it should loop (finite loop).
- The FOR loop has the following format:

```
for counter_variable in list_of_value:
    # instructions
    # instructions
    # …
```

- In most cases:
  - **Initialization** initialize the counter with the first value of the list.
  - **test-condition** test if a counter has reached the last value from the list.
  - **Increment** incremente the counter to the next value in the list.
- Any FOR loop can be reformulated as a WHILE loop but not the other way around.

Sub-theme: software model for a FOR loop



Sub-theme: The range instruction

range(a,b) returns une liste des values: a, a+1, a+2, …, b-1

```
for x in range(0, 3):
    print("Line", x)
```

Displayed result:

Line 0

Line 1

Line 2

range(a,b, i) returns a list of values: a, a+i, a+2i, …

(smaller than b)

(a, b and i are integers)

```
for x in range(1, 10, 2):
    print("Line", x)
```

Résultat affiché:

Line 1

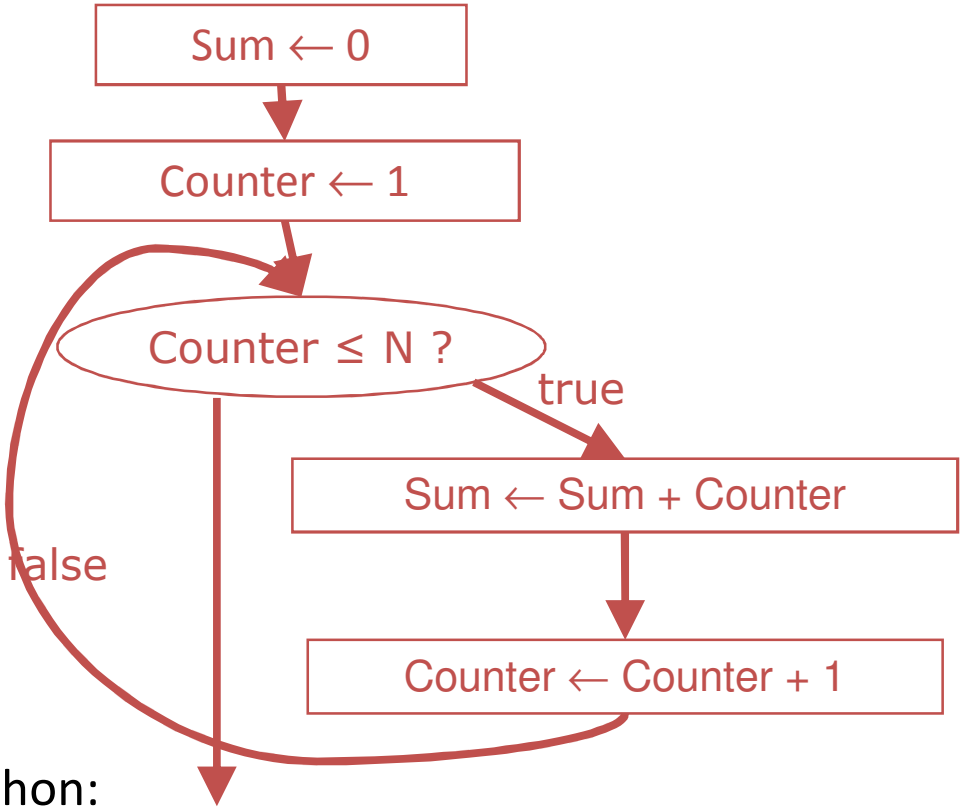
Line 3

Line 5

Line 7

Line 9

Exercise 4: Sum1toN with a FOR loop



- Translation in Python:

Exercise 4 -*Solution*: Sum1toN with a FOR loop

```
n = int(input("Please enter n: "))

sum=0

for counter in range (1, n+1):
    sum = sum + counter

print("The sum is ", sum)
```

Exercise 5 - *Solution*: Product with FOR loop

```
n = int(input(“Please enter n: "))

product = 1
for counter in range (1, n+1):
  product = product * counter

print(‘The product is ', product)
```

Exercise 6: *Fibonacci serie*

- A serie of numbers whose terms are equal o the sum of the 2 terms that preceed it.
- Derive an algorithm to display the first n numbers of the serie.
- Example: The first 10 numbers. (n = 10)

1 2 3 5 8 13 21 34 55 89

Exercise 6 - *Solution*: *Fibonacci serie*

```
n = int(input(“Please enter n "))
a = 1
b = 1
i = 1 # to count until n

print(a, end =" ")  # so not go to the next line

while i < n :
    print(a + b, end =" ")
    temp = b
    b = a + b # b becomes a+b before any change of a
    a = temp  # a takes the value of b
    i = i + 1
```

Exercice 6 - *Solution*: *Fibonacci serie(variance)*

```
n = int(input(“Please enter n "))
a = 1
b = 1
i = 1 # to count until n

while i <= n :
    print(b, end =" ")
    a, b, i = b, a+b, i+1 # The three variables change
                          # at the same time. meme temps
```

Question:

What does the following Python code display on the screen?

```
for i in range(10, 1, -2):
  print(i, end =" ")
```

Possible responses (choose one):

a) 10 8 6 4 2

b) 10 9 8 7 6 5 4 3 2 1

c) 10 9 8 7 6 5 4 3 2

d) error

Question – *Solution*:

What does the following Python code display on the screen?

```
for i in range(10, 1, -2):
  print(i, end =" ")
```

Possible responses (choose-one):

a) 10 8 6 4 2

b) 10 9 8 7 6 5 4 3 2 1

c) 10 9 8 7 6 5 4 3 2

d) error

Correcte response
correcte a)

Explanation: from 10 to 2 with steps of 2

Conclusion

- The concept of the loop is essential to solve complex problems.
- There are defined as well as indefined loops.
- Its execution is dynamic, relatively to the number of times the program will loop.