

## Partie I

*Introduction à la programmation*

# Informatique 1

## Introduction à la programmation

Mission 1 : INTRODUCTION

*Programmes, Variables & Valeurs, Expressions & Instructions*



Kim Mens Siegfried Nijssen Charles Pecheur

# Python

Langage de programmation **Python**

**L'interpréteur Python**



**Python 3.7.2**

Type "help", "copyright", "credits" or "license" for more information.

>>>  

Invite (prompt)

$$2+2 = 4$$

## Le premier calcul

$$2+2$$

```
Python 3.7.2
```

```
>>> 2+2
```

```
4
```

```
>>>
```

# Hello, world

Le premier programme

```
print("Hello, World!")
```

```
Python 3.7.2
>>> print("Hello, World!")
Hello, World!
>>>
```

# decompte.py

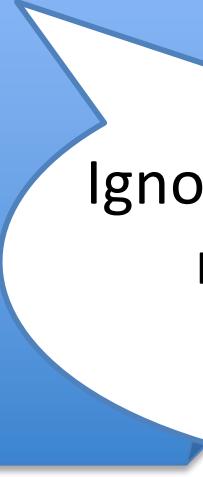
```
# Affiche un compte à rebours.  
# Decompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du decompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Decollage")
```

Dans un fichier  
decompte.py

Python 3.7.2  
>>> import decompte  
5  
4  
3  
2  
1  
0  
Decollage  
>>>

# Les commentaires

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```



Ignorés par Python mais nécessaires pour les humains !

# Les valeurs

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Une programme manipule des valeurs  
(des entiers, des chaînes de caractères, ...)

# Les mots réservés

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	<b>while</b>
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

# Les identifiants

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Des noms qui désignent des éléments du programme  
(une variable, une fonction, ...)

# Les variables

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

# Les variables

Identifient une zone mémoire

Affectation :

$$\begin{array}{l} \textcolor{orange}{i} = 5 \\ \textcolor{orange}{i} = i - 1 \end{array}$$

**i** prend la valeur 5  
**PAS** ~~i égale 5~~

**i**

**5**

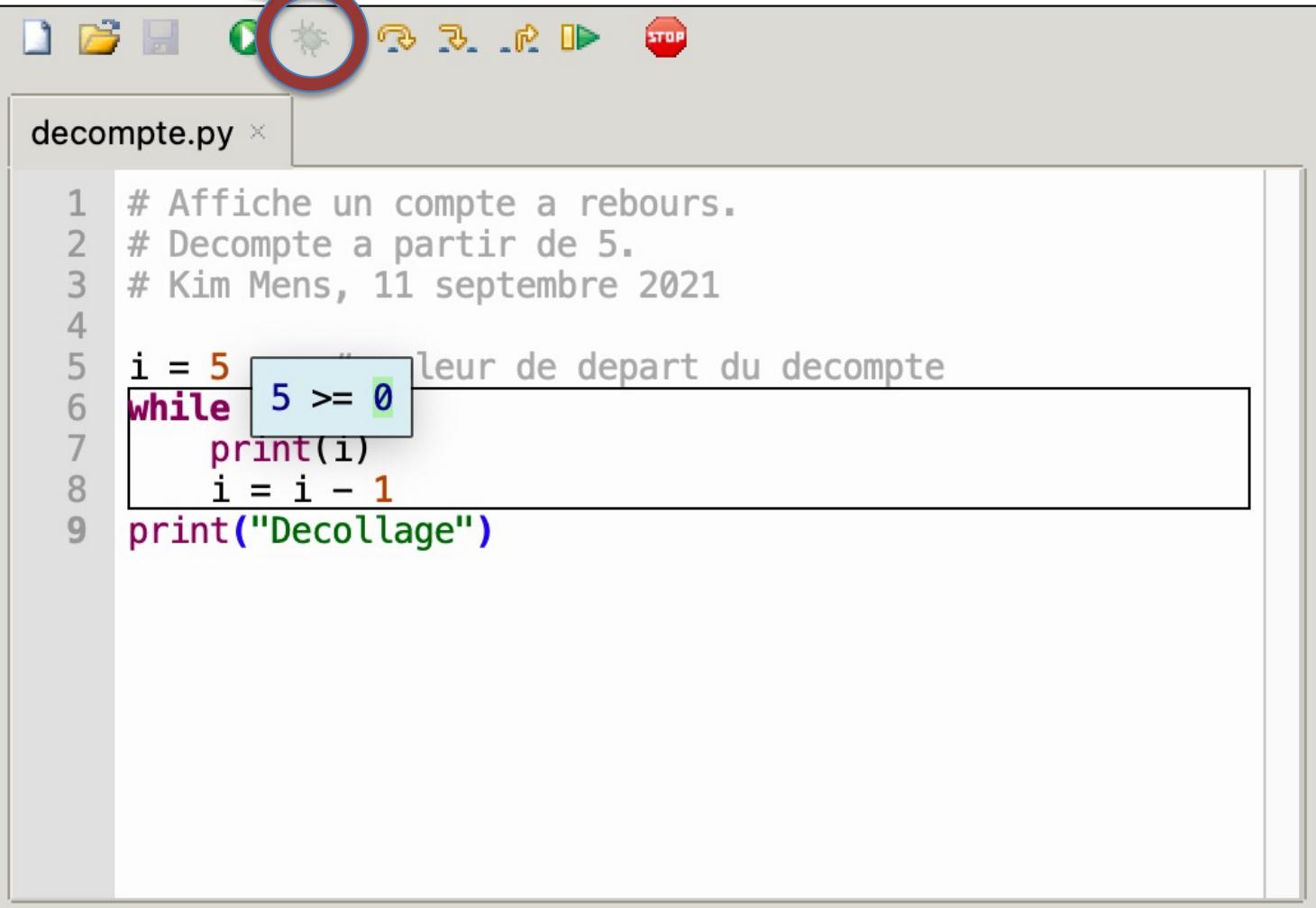
Mémoire

La première affectation crée la variable  
Les autres affectations mettent à jour sa valeur

Utilisation :

```
while i >= 0 :  
    print(i)  
    i = i - 1
```

# Thonny Debugger



The screenshot shows the Thonny Python IDE interface. At the top, there's a toolbar with various icons: file, folder, run, undo, redo, and stop. Below the toolbar, a window titled "decompte.py" displays the following Python code:

```
1 # Affiche un compte à rebours.
2 # Décompte à partir de 5.
3 # Kim Mens, 11 septembre 2021
4
5 i = 5 # valeur de départ du décompte
6 while 5 >= 0
7     print(i)
8     i = i - 1
9 print("Décollage")
```

A red arrow points to the green play button icon in the toolbar, which has a sunburst symbol on it. This icon is circled in red, indicating it is the debugger button. The code editor shows the script "decompte.py". The code itself is a simple countdown script that prints numbers from 5 down to 0, followed by the text "Décollage". The variable "i" is initialized to 5, and the condition in the while loop is 5 >= 0.

# PythonTutor

Python 3.6  
([known limitations](#))

---

```
1 # Affiche un compte à rebours.
2 # Décompte à partir de 5.
3 # Kim Mens, 11 septembre 2021
4
5 i = 5      # valeur de départ du décompte
6 while i >= 0:
7     print(i)
8     i = i - 1
9 print("Décollage")
```

---

[Edit this code](#)

→ line that just executed  
→ next line to execute

Step 2 of 21

Print output (drag lower right corner to resize)

Frames Objects

Global frame	
i	5

# Les instructions

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021
```

```
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Une instruction **produit un effet**

# Instructions simples

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021
```

```
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Affiche un message à l'écran

Effectuent une opération  
(une affectation, un affichage à l'écran, ...)

# Instructions simples

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021
```

```
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

ou, plus court:  
`i -= 1`

Effectuent une opération  
(une affectation, un affichage à l'écran, ...)

# Instructions de contrôle

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
  
print("Décollage")
```

**Contrôlent l'exécution** du programme

# Les expressions

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Une expression **calcule une valeur**

# Types de valeurs

Entiers	int
Réels	float
Text	str
Booléens	bool
...	list
	tuple
	range
	dict

# Les entiers

Valeurs :

$\dots, -2, -1, 0, 1, 2, \dots$

Illimités :

$10 ** 1000$

mille chiffres

1000...000

Opérations :

$x = 2$

La puissance

$2 ** 5 = 32$

$3 * x ** 2 + 4 * x - 2 = 18$

$(42 // 5) * 5 + 42 \% 5 = 42$

Division entière  
 $42 // 5 = 8$  mais  
 $42 / 5 = 8.4$

Reste de la division entière  
 $42 \% 5 = 2$  (modulo)

# La priorité des opérations arithmétiques

Priorité :

1. d'abord  $\ast \ast$
2. ensuite  $\ast, \text{//}, \text{\%}$
3. finalement  $+, -$

- () pour forcer la priorité
- de gauche à droite en cas de priorité égale

Exemples:

3 $\ast$ 2 $\ast \ast$ 2 $+$ 4 $\ast$ 2 $-$ 2	18
( (3 $\ast$ (2 $\ast \ast$ 2)) $+$ (4 $\ast$ 2)) $-$ 2	18
((((3 $\ast$ 2) $\ast \ast$ 2) $+$ 4) $\ast$ 2) $-$ 2	78
(42 $\text{//}$ 5) $\ast$ 5 $+$ 42 $\%$ 5	42
((42 $\text{//}$ 5) $\ast$ 5) $+$ (42 $\%$ 5)	42
((((42 $\text{//}$ 5) $\ast$ 5) $+$ 42) $\%$ 5	2

# Les nombres réels

$$3 \cdot 10^8 = 300000000$$

$$1,602 \cdot 10^{-19} = 0.00000000000000000001602$$

Valeurs :

0.0, 1.0, 0,01, 3e8, 1.602e-19, ...

Représentés en « virgule flottante »

Stocké comme      mантисса \* 10 <sup>exposant</sup>

Nombres très grands ou très petits (exposant)

Précision constante de la mantisse (risque d'**erreurs d'arrondi**)

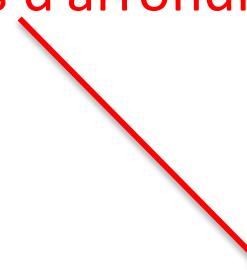
Opérations :

$$x = 0.5$$

$$3.0 * x ** 2 + 4.0 * x - 2.0 \quad 18$$

$$(42 / 5) * 3$$

$$25.200000000000003$$



Division réelle

$$42 / 5 = 8.4 \text{ mais}$$

$$42 // 5 = 8$$

# Les conditions

```
# Affiche un compte à rebours.  
# Décompte à partir de 5.  
# Kim Mens, 11 septembre 2021  
  
i = 5      # valeur de départ du décompte  
while i >= 0:  
    print(i)  
    i = i - 1  
print("Décollage")
```

Une condition est une expression logique.

Elle **calcule une valeur logique VRAI (True) ou FAUX (False)**

# Les Booléens

Deux valeurs :

**True, False**

Opérations : **and, or, not**

True **and** False      **False**

True **or** False      **True**

**not** True      **False**

Comparaisons : **==, !=, <, >, <=, >=**

5 **==** 3+2      **True**

5 **!=** 6      **True**

Attention :

i = 5    i prend la valeur 5  
i == 5    i égale 5 ?

# Instructions conditionnelles

```
# Imprimer les chiffres pairs et impairs.  
# Kim Mens, 12 septembre 2021
```

```
n = 10  
i = 0  
while i <= n :  
    if i % 2 == 0 :  
        print(i, " est pair")  
    else :  
        print(i, " est impair")  
    i = i + 1
```

```
>>> %Run pair_impair.py  
0 est pair  
1 est impair  
2 est pair  
3 est impair  
4 est pair  
5 est impair  
6 est pair  
7 est impair  
8 est pair  
9 est impair  
10 est pair
```

Permettent de faire des choix dans l'exécution d'un programme en fonction de la valeur d'une condition.

# Instructions conditionnelles

```
# Imprimer les chiffres pairs et impairs.
```

```
# Kim Mens, 12 septembre 2021
```

```
n = 10
i = 0
while i <= n :
    if i % 2 == 0 :
        print(i," est pair")
    else :
        print(i," est impair")
    i = i + 1
```

```
pair_impaire.py ×
1  # Imprimer les chiffres pairs et impairs
2  # Kim Mens, 12 septembre 2021
3
4  n = 10
5  i = 0
6  while i
7      if 2 % 2 == 0
8          print(i," est pair")
9      else :
10         print(i," est impair")
11     i = i + 1
12
```

```
Shell ×
10  est pair
>>>
>>> %Debug pair_impaire.py
0  est pair
1  est impair
```

# Instruction conditionnelle (1)

```
i = 0  
# Deux cas :  
# i pair, i impair
```

(1) Cas

# Instruction conditionnelle (1)

```
i = 0  
# Deux cas :  
# i pair, i impair  
  
if i % 2 == 0 :  
    print(i,"est pair")
```

(1) Cas  
(3) if

# Instruction conditionnelle (1)

```
i = 0  
# Deux cas :  
# i pair, i impair  
if i % 2 == 0 :  
    print(i,"est pair")  
else :  
    print(i,"est impair")  
#  
# La suite
```

(1) Cas

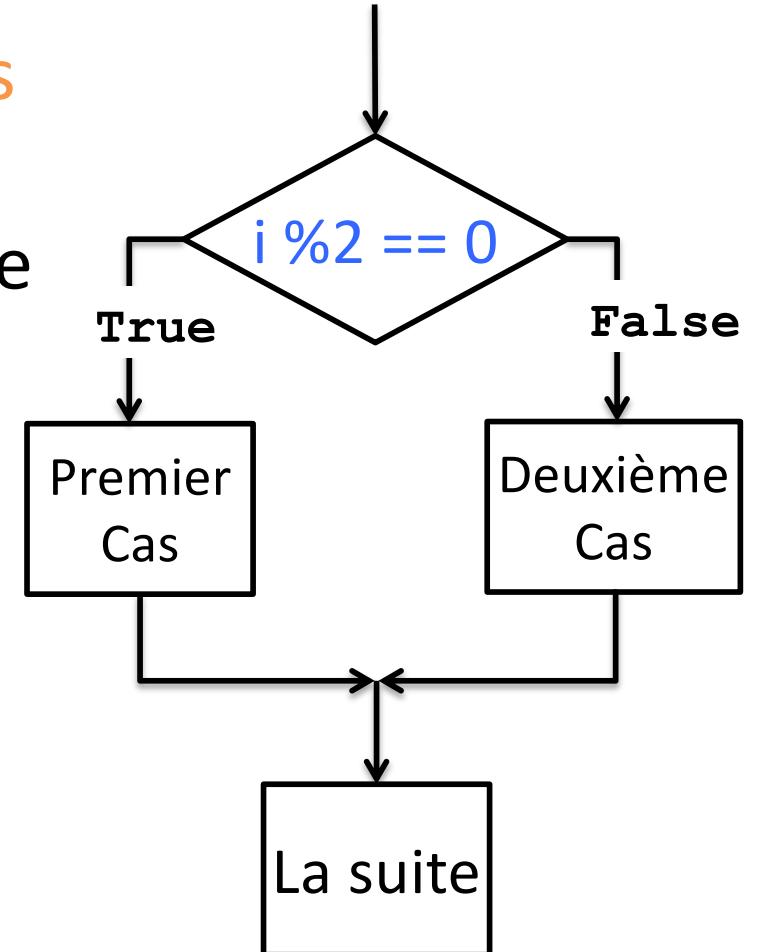
(3) if

(5) else

# Instruction conditionnelle (1)

```
i = 0  
# Deux cas :  
# i pair, i impair  
if i % 2 == 0 :  
    print(i,"est pair")  
else :  
    print(i,"est impair")  
#  
# La suite
```

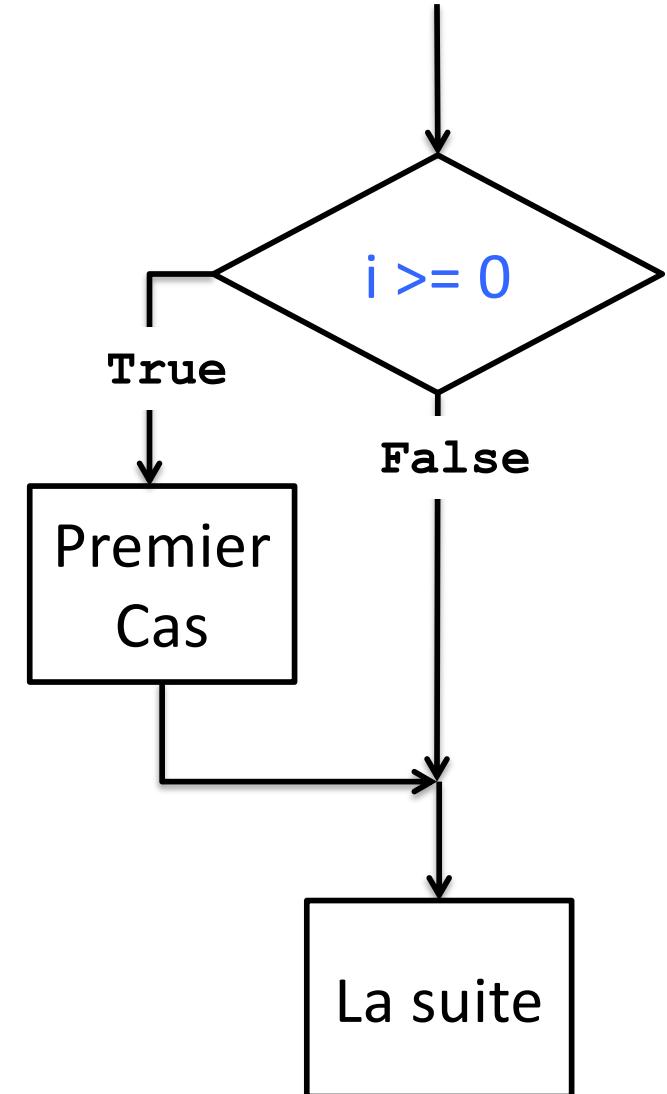
(1) Cas  
(3) if  
(5) else



# Instruction conditionnelle (2)

```
i = 0  
# Un seul cas :  
# i >= 0  
  
if i >= 0 :  
    print(i,"est pair")  
  
#  
# La suite
```

(1) Cas  
(3) if



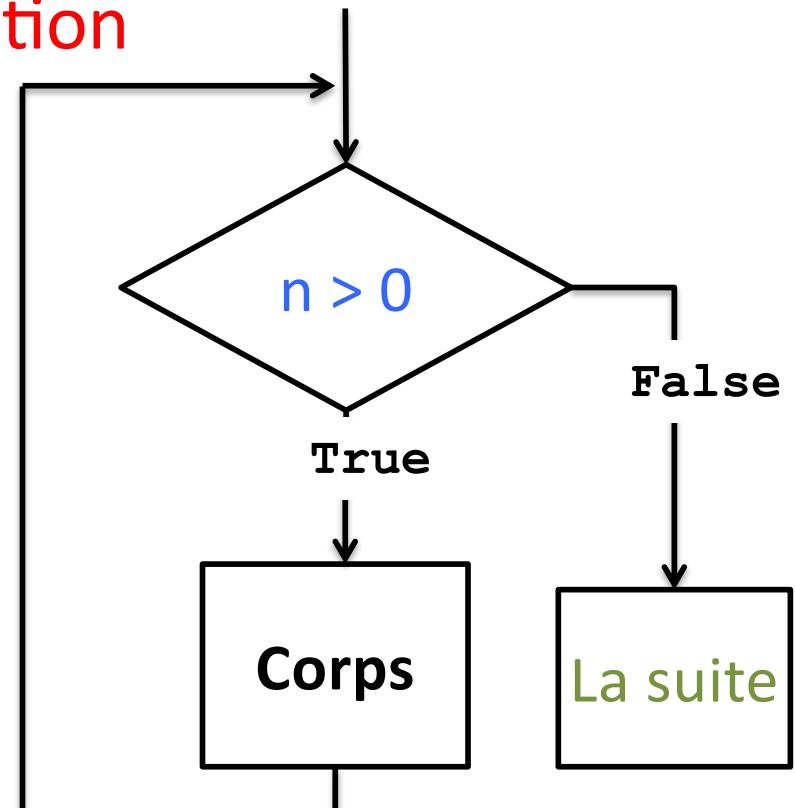
# Instruction de boucle

# Quel est la somme des 10 premiers carrés ?

```
sum = 0
n = 10
while n > 0 :
    sum = sum + n*n
    n = n - 1
```

```
#
# La suite
#
```

- (1) Initialiser
- (2) Condition
- (3) Corps



# Blocs

```
while n > 0 :  
    sum = sum + n*n  
n = n-1
```

```
while n > 0 :  
    sum = sum + n*n  
    n = n-1
```

```
while n > 0 :  
    sum = sum + n*n  
    n = n-1
```

Boucle infinie



Erreur d'indentation



Correct !



Les **blocs** sont définis par l'**indentation**  
=> Attention à l'**indentation** !

# Que fait ce programme?

```
# Imprimer les chiffres pairs et impairs.  
# Kim Mens, 12 septembre 2021
```

```
n = 10  
i = 0  
while i <= n :  
    if i % 2 == 0 :  
        print(i, " est pair")  
    else :  
        print(i, " est impair")  
    i = i + 1
```



```
>>> %Run pair_impair.py
```

```
0 est pair  
0 est pair
```

Attention à l'indentation !

# Quiz

```
capital = 1000
taux = 5
duree = 20
an = 0
while an <= duree:
    print(an, "\t", capital)
    capital = capital * (1 + taux/100)
    an = an+1
```

Combien de fois exécute-t-on la boucle ?

21

Que vaut an lors du premier print ?

0

Que vaut an lors du dernier print ?

20

Que vaut an à la fin de l'exécution ?

21

# Quiz

```
capital = 1000
taux = 5
duree = 20
an = 0
while an <= duree:
    print(an, "\t", capital)
    capital = capital * (1 + taux/100)
    an = an+1
```

Qu'imprime ce programme ?

0	1000
1	1050.0
2	1102.5
3	1157.625
4	1215.5062500000001
5	1276.2815625000003
6	1340.0956406250004
7	1407.1004226562504
8	1477.455443789063
9	1551.3282159785163
10	1628.8946267774422
11	1710.3393581163143
12	1795.8563260221301
13	1885.6491423232367
14	1979.9315994393985
15	2078.9281794113685
16	2182.874588381937
17	2292.018317801034
18	2406.619233691086
19	2526.9501953756403
20	2653.2977051444223



A faire dès  
que possible

Faire la mission 1 "mise en route"

<https://syllabus-interactif.info.ucl.ac.be/index/info1-exercises>

- [Mission 1 - Mise en route](#)
- [Démarrage](#) : QCM + questions ouvertes
- [Réalisation](#) d'un programme

Lire le syllabus du cours

<https://syllabus-interactif.info.ucl.ac.be/index/info1-theory>

- chapitres 1 → 5

# Prochaines échéances

- Mardi
  - Séance tutorée intermédiaire Mission 1
- Jeudi 14h00
  - Avoir terminé la phase de réalisation de la mission 1 et soumis le code sur INGInious
- Vendredi
  - Séance tutorée finale Mission 1
- Vendredi 14h00
  - Cours restructuration mission 1, introduction mission 2