

## Algorithme Convertisseur Décimal => Binaire

Algo : convDecBin()

Déclaration :

nb, d, r : entier

res : chaîne de caractères

Début :

Afficher ("Entrer un nombre : ")

Saisir nb

res  $\leftarrow$  ''

d  $\leftarrow$  1

TantQue (d  $\neq$  0) Faire

r  $\leftarrow$  nb%2

d  $\leftarrow$  nb//2

nb  $\leftarrow$  d

res  $\leftarrow$  str(r) + res

FinTantQue

Afficher ("Voici le résultat :", res)

FinAlgo : convDecBin()

# Algorithme

## Les Principaux opérateurs

### Les opérateurs classiques :

+ = addition  
- = soustraction  
\* = multiplication  
/ = division  
\*\* = puissance

### Le Dividende : //

Il permet de récupérer la partie entière dans une division. Exemple :

$20 / 3 = 6,666667$

$20 // 3 = 6$

### Le Modulo : %

Il permet de récupérer le reste entier dans une division. Exemple :

$20 \% 3 = 2$

C'est-à-dire :  $20 = 3 \times 6 + 2$  (Division Euclidienne)

## Téléchargement et utilisation de Python

Il vous suffit de **taper** dans la **barre de recherche** de votre **navigateur** :

"télécharger python"

Et de **cliquer** sur le **premier lien** qui vous propose de **télécharger** le **logiciel Python**.

Ou d'aller sur **ce lien** :

<https://www.python.org/downloads/>

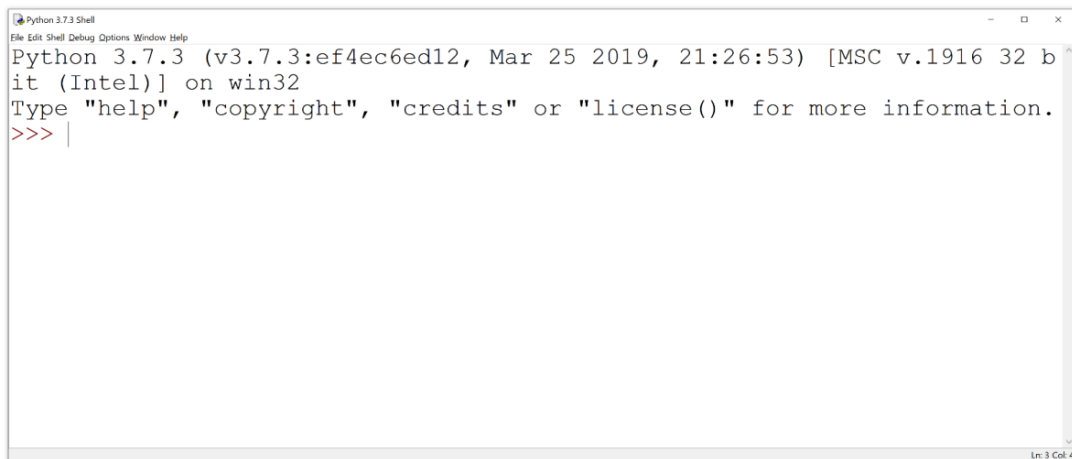
Une fois le logiciel **téléchargé** et **installé**, vous devez **chercher** le programme :

"IDLE"

Touche "Windows" et taper "IDLE" dans la barre de recherche de "Windows".

Et vous **exécutez**.

La **fenêtre suivante** va apparaître :

A screenshot of a Python 3.7.3 Shell window. The title bar reads "Python 3.7.3 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following information: "Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32", followed by "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>> |". The status bar at the bottom right shows "Ln: 3 Col: 4".

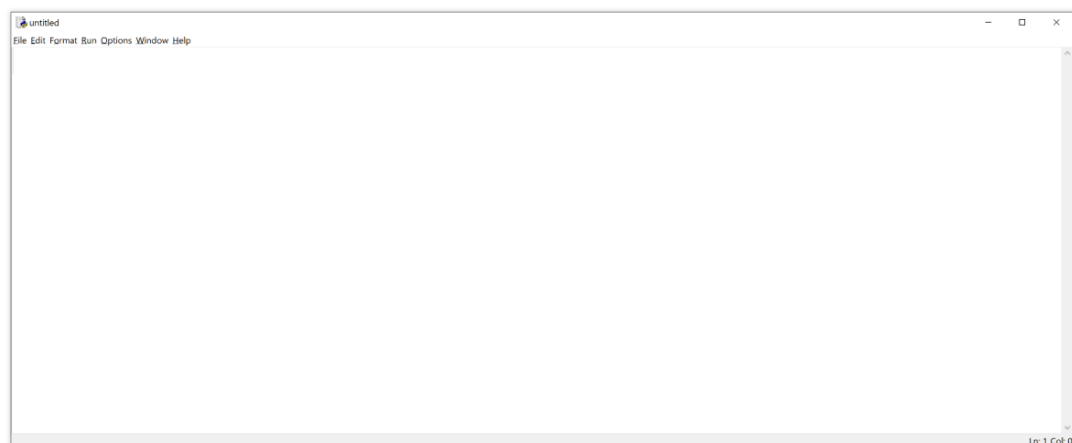
C'est dans **cette fenêtre**, que l'on appelle la "**Console**", que **les résultats** de nos programmes vont **apparaître**.

On peut également effectuer **tous les calculs** ou **opérations** que l'on souhaite.

En revanche, il **ne faut jamais faire de programme** dans la **Console**.

Pour écrire un programme, il faut **cliquer** sur **l'onglet "File"** puis sur **"New File"**.

La **fenêtre suivante** va apparaître :



C'est dans **cette fenêtre** que nous allons **créer nos programmes** ou **fonctions** et que l'on pourra **sauvegarder**.

# Création d'un programme ou d'une fonction en Python

Pour faire **des programmes**, nous allons utiliser **plusieurs fonctions** propres à Python. Ces **fonctions** vont nous permettre de **réaliser différents programmes**.

**NB :** Attention, en **programmation Python**, il faut **respecter** les **MAJUSCULES** et les **minuscules**. On dit que ce langage est "**sensible à la casse**".

## Principales fonctions :

**def** = permet de **définir** le **nom du programme** ou de la **fonction**.

**print()** = fonction qui **permet d'afficher des variables** ou du **texte**.

**input()** = fonction qui **permet de récupérer la saisie** de **l'utilisateur** (ce que vous **tapez sur votre clavier**). **Par défaut**, cette fonction **récupère** la saisie de l'utilisateur dans une **chaîne de caractères**.

**int()** = fonction qui **permet de convertir une chaîne de caractères** en **type entier** (**chiffre**). Cependant, il faut que votre **chaîne de caractères** ne **contienne que des chiffres**. Sinon, cela ne fonctionnera pas.

## Les conditions :

**if()** = **Si. Permet de tester une condition** et **d'exécuter** une ou plusieurs **instructions** si la condition **est vérifiée**. Exemple :

**if**(x == 0):   #se lit : si x est égal à zéro alors...

    [code à exécuter]

    [Exemple de code :  $x = y + 1$ ]

**L'indentation** signifie que **tout ce qui est indenté en dessous** de la **condition** appartient à la **condition**. Si on veut sortir de la condition, il faut désindenter.

On peut faire **autant** de "if()" que l'on souhaite. Dans ce cas, ils vont **tous être testés**.

## Exemple :

**if**(.....):

    [....]

**if**(.....):

    [....]

**if**(.....):

    [....]

On peut aussi faire des **conditions exclusives**. Dans ce cas, il faut **utiliser** la fonction "if()" **suivi** de la fonction "elif()" qui **signifie** "**SinonSi**". Exemple :

```
if(.....):  
    [...]  
elif(.....):  
    [...]  
elif(.....):  
    [...]
```

On peut mettre **autant** de "elif()" que l'on veut. La **particularité** est la suivante : dès qu'une **condition** est **vérifiée**, **toutes les autres** sont **ignorées**.

Enfin, **on fini généralement** les conditions avec un "else" qui signifie "**Sinon**". Il ne peut y en avoir **qu'un seul** **par groupe de condition**. C'est **mieux** de **toujours finir** avec un "else" mais **ce n'est pas obligatoire**. Il y **des situations** où cela ne **sert à rien**.

La **particularité** du "else" **par rapport** au "if()" et "elif()" c'est qu'il n'y a **aucune condition** dans le "else". Il **s'exécute** quand **aucune des conditions précédentes** n'est **vérifiée**. On l'appelle aussi "**le cas poubelle**". Le "else" est donc **la condition** qui est **exécutée** quand **aucune autre n'est bonne** (n'est validée). Exemple :

```
if(.....):  
    [...]  
if(.....):  
    [...]  
if(.....):  
    [...]  
else:  
    [...]
```

Autre cas avec le "elif()" :

```
if(.....):  
    [...]  
elif(.....):  
    [...]  
elif(.....):  
    [...]  
else:  
    [...]
```

On peut **imbriquer** autant **que l'on veut** les **conditions**. Exemple :

```
if(.....):  
    if(.....):  
        [...]  
    if(.....):  
        [...]  
    else:  
        [...]  
else:  
    [...]
```

### **Affectation et Comparaison :**

Pour **affecter** ou **initialiser** une **variable**, ou pour **comparer** une **variable** à une autre voici ce qu'il **convient de faire** :

Algo	Python	Signification
←	=	<i>Affectation (prend la valeur de ...)</i>
=	==	<i>Comparaison (permet de comparer)</i>

### **Création d'un programme ou d'une fonction :**

Pour **créer** un programme, il faut **le nommer** (lui donner un nom). Pour cela nous allons utiliser la **fonction "def" suivi du nom (pas d'espace ni de caractère spécial)** que vous voulez **donner** à **votre programme**. Il faut **ajouter** des **parenthèses ouvrante et fermante** suivi **immédiatement** d'un **:** (**deux points**) Exemple :  
def test():

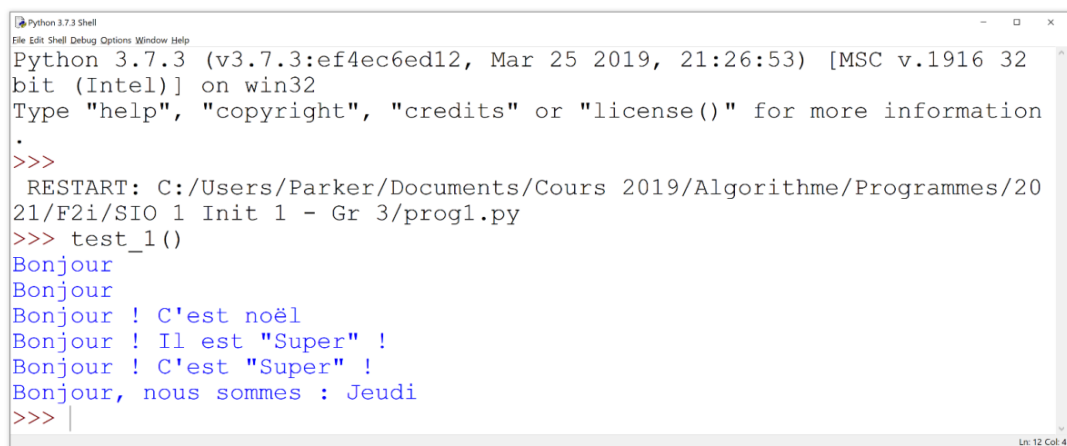
Une fois que cela est fait, quand vous **allez à la ligne** en **appuyant** sur la touche "**Entrer**" vous allez constater qu'il y a **un espace** qui s'est fait **automatiquement**. Cela s'appelle **l'indentation**. Cela veut dire que tout ce qui est **indenté par rapport au nom** de votre programme va **appartenir** à **ce programme**. Si vous voulez **quitter** ce programme pour en faire un autre, il vous suffit de **désindenter** en **utilisant** la touche qui permet **d'effacer**. Vous pouvez maintenant **créer** un **autre programme** avec la **fonction "def xxxx():"** que l'on a vu précédemment.

## Exemple de programme et le résultat obtenu :

### Programme 1 :

```
def test_1():
    print('Bonjour')
    print("Bonjour")
    print("Bonjour ! C'est Noël")
    print('Bonjour ! Il est "Super" !')
    print('Bonjour ! C\'est "Super" !')    #\ = alt Gr + 8
    jour = 'Jeudi'
    print('Bonjour, nous sommes :', jour)
```

### Résultat :

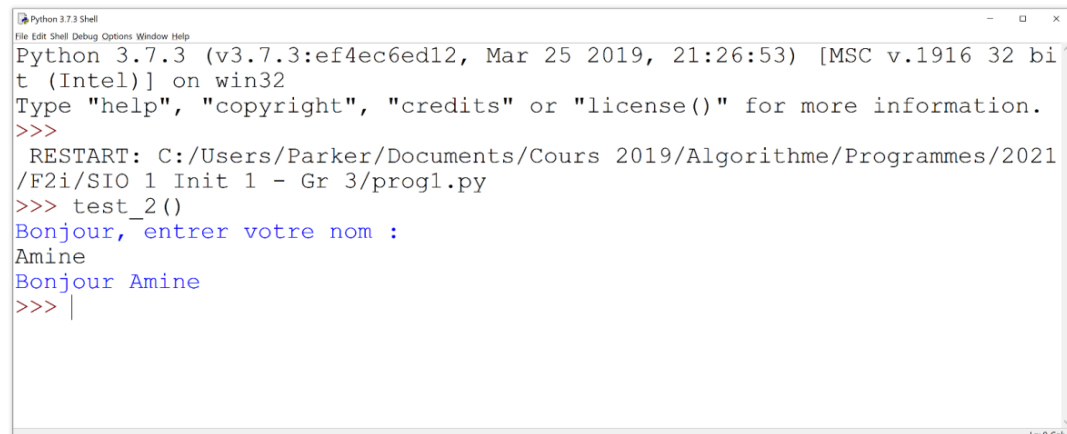


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32
bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information
.>>>
RESTART: C:/Users/Parker/Documents/Cours 2019/Algorithmes/Programmes/20
21/F2i/SIO 1 Init 1 - Gr 3/prog1.py
>>> test_1()
Bonjour
Bonjour
Bonjour ! C'est Noël
Bonjour ! Il est "Super" !
Bonjour ! C'est "Super" !
Bonjour, nous sommes : Jeudi
>>> |
```

### Programme 2 :

```
def test_2():
    print('Bonjour, entrer votre nom :')
    nom = input()    #input() permet de récupérer la saisie utilisateur
    print("Bonjour", nom)
```

### Résultat :



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bi
t (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
.>>>
RESTART: C:/Users/Parker/Documents/Cours 2019/Algorithmes/Programmes/2021
/F2i/SIO 1 Init 1 - Gr 3/prog1.py
>>> test_2()
Bonjour, entrer votre nom :
Amine
Bonjour Amine
>>> |
```

Un **programme** ou une **fonction** est toujours rédigé en premier sous la forme d'un **script** que l'on appelle **un algorithme**. C'est grâce à cet **algorithme** que le développeur peut **créer** le **programme**. Il peut le faire dans **n'importe quel langage de programmation**. **L'algorithme** est donc un **langage universel** pour réaliser **n'importe quel programme** dans le **langage de programmation** que l'on **veut**.

Il faut donc **maîtriser** ce **langage universel**. Il faut savoir que c'est un **langage très tolérant**. Il existe **beaucoup** de **façons différentes** d'écrire en **algorithme**. Il faut donc être **curieux** et **chercher différent algorithme sur internet** pour être **familiariser** avec les **différentes façons de faire**. Néanmoins, voici un **format assez simple** et **assez complet** que je vous recommande de **connaître** et **d'utiliser**.

## Algorithme "Parité d'un nombre"

```
Algo : pariter()
|
Déclaration :
|          nb : entier
|
Début :
|    Afficher ("Entrer un nombre : ")
|    Saisir nb
|    Si (nb%2 = 0) Alors
|    |    Afficher ("Votre nombre",nb,"est pair.")
|    FinSi
|    Sinon
|    |    Afficher ("Votre nombre",nb,"est impair.")
|    FinSinon
FinAlgo : pariter()
```

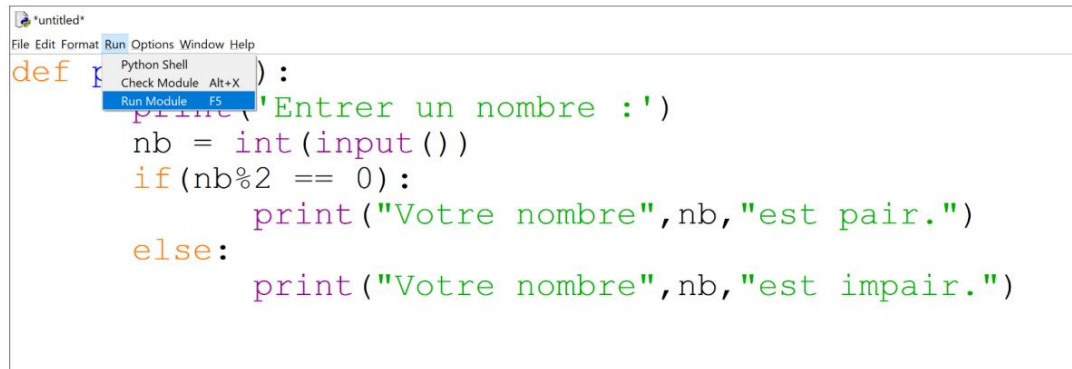
## Programme Python "Parité d'un nombre"

```
def pariter():
    print('Entrer un nombre :')
    nb = int(input())
    if (nb%2 == 0):
        print("Votre nombre :",nb,"est pair.")
    else:
        print("Votre nombre :",nb,"est impair.")
```



## Exécution d'un programme ou d'une fonction en Python

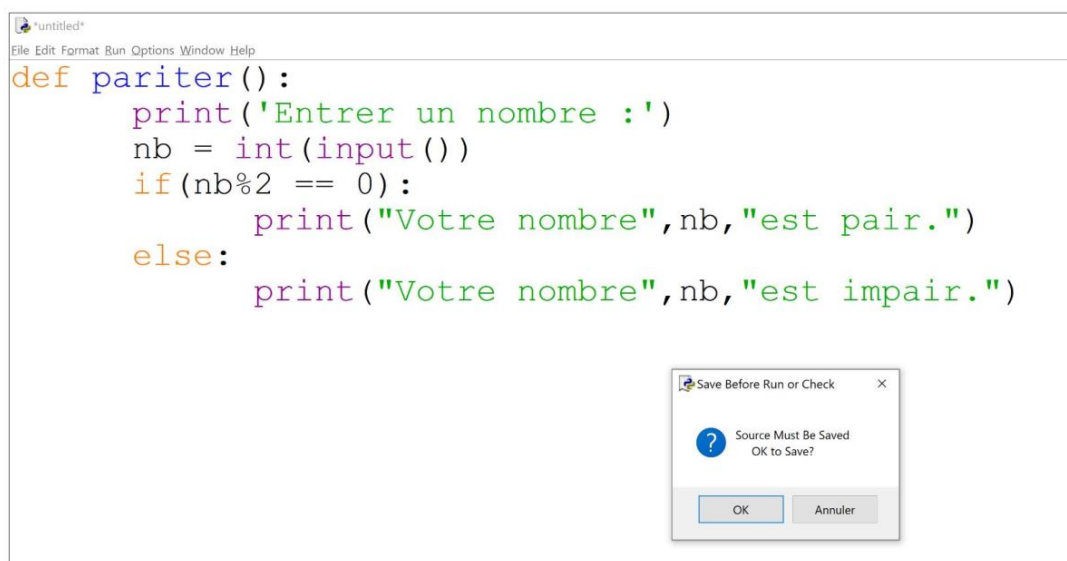
Pour exécuter **votre programme**, il suffit de cliquer sur l'onglet "**run**" puis "**run module**" ou bien la touche raccourci "**F5**" de votre clavier.



```
def pariter():
    print('Entrer un nombre :')
    nb = int(input())
    if (nb%2 == 0):
        print("Votre nombre", nb, "est pair.")
    else:
        print("Votre nombre", nb, "est impair.")
```

The screenshot shows a Python IDE window titled "untitled\*". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The "Run" menu is open, showing options like Python Shell, Check Module, and Run Module (F5). The "Run Module" option is highlighted. The code editor contains a function definition for "pariter" that prompts the user for a number and checks if it is even or odd.

Vous allez avoir un message vous invitant à **sauvegarder votre fichier**. Pensez bien à **créer un dossier** où vous rangerez **tous les fichiers Python** que nous allons faire cette année.



```
def pariter():
    print('Entrer un nombre :')
    nb = int(input())
    if (nb%2 == 0):
        print("Votre nombre", nb, "est pair.")
    else:
        print("Votre nombre", nb, "est impair.")
```

The screenshot shows the same Python IDE window, but now a dialog box titled "Save Before Run or Check" is open. The dialog box contains a question mark icon and the text "Source Must Be Saved OK to Save?". There are two buttons: "OK" and "Annuler".

Une fois que vous avez **validez l'enregistrement** dans l'**emplacement souhaité**, la **console (IDLE)** va s'ouvrir **à nouveau (même si vous l'aviez déjà ouvert)**. Vous aurez donc **2 consoles IDLE**. Il faut **absolument utiliser** celle qui **vient d'apparaître** car votre programme a été compilé dans **cette dernière**. Ce qui veut dire que pour **exécuter un programme**, vous êtes **obligé de le compiler** avec "**run module**" avant de pouvoir l'utiliser dans la **console qui vient de s'ouvrir**.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Desktop/Algorithmique/pariter.py =====
>>> |
```

Pour **tester votre programme**, il faut maintenant **l'exécuter** en tapant sur le clavier **le nom** que vous lui avait **donner**. Dans notre exemple, le nom du programme est "**pariter()**". Il faut donc taper sur le clavier : `pariter()`



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Parker/Desktop/Algorithmique/pariter.py =====
>>> pariter()|
```

Puis exécuter en appuyant sur la touche "Entrée" de votre clavier. La phrase "Entrer un nombre :" va apparaître, il vous suffit de saisir un nombre entier sur votre clavier. Nous allons essayer avec le nombre "23". Et enfin, il suffit à nouveau d'exécuter en appuyant sur la touche "Entrée" de votre clavier. Le résultat va apparaître immédiatement :

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Parker/Desktop/Algorithmique/pariter.py =====
>>> pariter()
Entrer un nombre :
23
Votre nombre 23 est impair.
>>> |
```

## Algorithme "Table de multiplication"

Algo : multi()  
|  
Déclaration :  
| nb, i : entier  
|  
Début :  
| Afficher ("Entrer un nombre : ")  
| Saisir nb  
| Pour i allant de (0, 10, +1) Faire  
| | Afficher (i,"x",nb,"=",i\*nb)  
| FinPour  
|  
FinAlgo : multi()

## Python "Table de multiplication"

```
def multi():
    print("Entrer un nombre :")
    nb = int(input())
    for i in range(0, 10 + 1):
        print(i, "x", nb, "=", i*nb)
```

## Les Boucles

Il en existe 2 principales :

- La boucle "Pour" => *for (en Python)*
- La boucle "Tant Que" => *while (en Python)*

### La boucle "Pour" ("for" en Python) :

Elle se présente sous cette forme en Algo :

- Pour i allant de (0, nb, +1) Faire

Dans cet exemple, si "**nb**" = **8**, cela veut dire que la variable "i" va **prendre comme valeur** de **départ** la **première valeur** dans la parenthèse (ici 0) et que "i" **s'arrêtera** quand elle sera **égale** à la **seconde valeur** dans la parenthèse. Ici, cette **valeur** peut être **un nombre** mais aussi **une variable** comme dans notre exemple avec "**nb**". Donc, dans cet exemple, la boucle s'arrêtera lorsque la valeur de "i" aura atteint la valeur de "**nb**", c'est-à-dire **8**.

En Python, elle se présente sous la forme :

```
for i in range(0, nb) :  
    [... code ...]
```

#### Fonctionnement du range :

Comme vu précédemment, le premier élément du **range** correspond à la valeur de **départ** de la variable "i".

Le second élément correspond à la valeur de **fin** (ou de sortie de boucle) de la variable "i".

On peut également ajouter une **3<sup>ème</sup> valeur** dans le **range**. Cette valeur correspond **au pas** de "i". C'est-à-dire comment **va évoluer** la variable "i". **Par défaut**, dans Python, la variable "i" avance de **"+1"** en **"+1"**. Mais on peut tout à fait changer cette valeur.

On peut avancer de **"-1"** en **"-1"** ou bien de toute autre valeur.

#### Il y a 3 cas possible dans le range :

1<sup>er</sup> Cas : **3 valeurs** dans le **range**, cela signifie que :

- la **première valeur** est **le départ**
- la **seconde valeur** est **la fin**
- la **dernière valeur** correspond **au pas**, c'est-à-dire à **l'évolution** de i.

2<sup>ème</sup> Cas : **2 valeurs** dans le **range**, cela signifie que :

- la **première valeur** est **le départ**
- la **seconde valeur** est **la fin**

Par conséquent, le pas est par défaut à + 1.

3<sup>ème</sup> Cas : 1 valeur dans le range, cela signifie que :

- la seule valeur est la fin

Par conséquent, le départ est initialisé par défaut à zéro et le pas à + 1.

En Algo, la valeur de fin correspond au nombre que l'on a saisi alors qu'en Python, la valeur de fin correspond au nombre que l'on a saisi moins un. Donc, si l'on souhaite aller jusqu'au nombre saisi, il faut ajouter un en plus.

Exemple :

Si nb = 7

En Algo, pour i allant de (0, nb), i prendra en premier la valeur de 0 puis en dernier la valeur de nb donc la valeur 7.

En Python, pour i allant de (0, nb), i prendra en premier la valeur de 0 puis en dernier la valeur de nb - 1 donc la valeur 6. Si on veut aller jusqu'à 7 en Python, il faut faire cela :  
for i in range (0, nb + 1):

## Les Listes

Nous avons également vu la création (ou initialisation) d'une liste ainsi que l'ajout d'éléments dans une liste.

En Algo :

Création (ou initialisation) d'une liste :

L ← []

Ajout d'un élément dans une liste :

L.apprendre(n)

ou encore

L.ajouter(n)

n peut être un entier, un réel, un booléen, une chaîne de caractères ou encore une liste.

Noter que l'on peut tout à fait ajouter une liste dans une liste.

On peut donc faire :

L.apprendre(17)

L.apprendre(3.75)

L.apprendre(Vrai)

L.apprendre('bonjour')

L.apprendre(a)                    a étant une liste contenant par exemple : [4, 'bleu', 51.7]

En Python :

Création (ou initialisation) d'une liste :

L = []

Ajout d'un élément dans une liste :

L.append(*n*)

*n* peut être un **entier**, un **réel**, un **booléen**, une **chaîne de caractères** ou encore une **liste**.

**Noter** que l'on **peut tout à fait ajouter** une **liste** dans une **liste**.

On peut donc faire :

L.append(**17**)

L L.append(**3.75**)

L.append(**Vrai**)

L L.append(**'bonjour'**)

L.append(*a*)                      *a* étant **une liste** contenant par exemple : [**4**, **'bleu'**, **51.7**]

## Les Programmes faits en cours (Algo + Python)

### Algorithme Nombre Premier

Algo : nbPremier()

|

Déclaration :

|

nb, cpt, i : entier

Début :

|

Afficher ("Entrer un nombre : ")

Saisir nb

cpt ← 0

Pour i allant de (1, nb) Faire

|

Si (nb%i = 0) Alors

|

cpt ← cpt + 1

FinSi

FinPour

Si (cpt = 2) Alors

|

Afficher ("Votre nombre",nb,"est premier.")

FinSi

Sinon

|

Afficher ("Votre nombre",nb,"n'est pas premier.")

FinSinon

FinAlgo : nbPremier ()

## Python Nombre Premier

```
def nbPremier():  
    print("Entrer un nombre :")  
    nb = int(input())  
    cpt = 0  
    for i in range(1, nb + 1):  
        if (nb%i == 0):  
            cpt = cpt + 1  
    if (cpt == 2):  
        print("Votre nombre", nb, "est premier.")  
    else:  
        print("Votre nombre", nb, "n'est pas premier.")
```

## Algorithme Nombre Premier + Diviseur non Premier

Algo : nbPremier2()

|

Déclaration :

|

L : liste

nb, cpt, i : entier

Début :

|

Afficher ("Entrer un nombre : ")

Saisir nb

cpt ← 0

L ← []

Pour i allant de (1, nb) Faire

|

Si (nb%i = 0) Alors

|

cpt ← cpt + 1

L.append(i)

|

FinSi

FinPour

Si (cpt = 2) Alors

|

Afficher ("Votre nombre", nb, "est premier.")

FinSi

Sinon

|

Afficher ("Votre nombre", nb, "n'est pas premier.")

Afficher ("Voici ses diviseurs :", L)

FinSinon

|

FinAlgo : nbPremier ()

## Python Nombre Premier + Diviseur non Premier

```
def nbPremier2():
    print("Entrer un nombre :")
    nb = int(input())
    cpt = 0
    L = []
    r = ''
    for i in range(1, nb + 1):
        if(nb%i == 0):
            cpt = cpt + 1
            L.append(i)
            r = r + str(i) + ' '
    if(cpt == 2):
        print("Votre nombre", nb, "est premier.")
    else:
        print("Votre nombre", nb, "n'est pas premier.")
        print("Voici ses diviseurs :", L)
        print("Voici ses diviseurs :", r)
```

Dans ce dernier programme, il y a **2 façons** de **recupérer les diviseurs** d'un nombre non premier :

- Avec **une liste** (ici avec la liste **L**)
- Avec **une variable** qui est une chaîne de caractères (ici la variable **r**)

La **fonction** `str()` est une **fonction** qui permet de **transformer un entier** en **chaîne de caractères**. Ainsi, si **i = 2** (i est un **entier**), `str(i)` va **devenir une chaîne de caractères** contenant le caractères '2'.



## Algorithme Nombre Parfait

Algo : nbParfait()

|

Déclaration :

|

nb, s, i : entier

Début :

|

Afficher ("Entrer un nombre : ")

Saisir nb

s ← 0

Pour i allant de (1, nb - 1) Faire

|

Si (nb%i = 0) Alors

|

s ← s + i

|

FinSi

FinPour

Si (s = nb) Alors

|

Afficher ("Votre nombre",nb,"est parfait.")

|

FinSi

Sinon

|

Afficher ("Votre nombre",nb,"n'est pas parfait.")

|

FinSinon

|

FinAlgo : nbParfait()

## Algorithme Nombre Parfait

```
def nbParfait() :  
    print("Entrer un nombre :")  
    nb = int(input())  
    s = 0  
    for i in range(1,nb):  
        if(nb%i == 0):  
            s = s + i  
    if(s == nb):  
        print("Votre nombre",nb,"est parfait.")  
    else:  
        print("Votre nombre",nb,"n'est pas parfait.")
```