

## Instructions conditionnelles

## C5 Initiation à Python avec turtle

### Instructions conditionnelles

- La syntaxe d'une instruction conditionnelle en Python est :

```
1  if <condition >:  
2      <instructions1 >  
3  else:  
4      <instructions2 >
```

Cela permet d'exécuter les <instructions1> si la condition est vérifiée, sinon on exécute les <instructions2>.


## C5 Initiation à Python avec turtle

### Instructions conditionnelles

- La syntaxe d'une instruction conditionnelle en Python est :

```
1  if <condition >:  
2      <instructions1 >  
3  else:  
4      <instructions2 >
```

Cela permet d'exécuter les <instructions1> si la condition est vérifiée, sinon on exécute les <instructions2>.

-  On fera bien attention à la syntaxe du langage, et notamment à l'usage du caractère **:** qui suit la condition (et le else) et à l'**indentation**, c'est à dire le décalage des instructions qui doivent s'exécuter.

## C5 Initiation à Python avec turtle

### Exemples

- Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0

## C5 Initiation à Python avec turtle

### Exemples

- 1 Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0
- 2 On suppose qu'une variable `longueur` peut être positive ou négative, si cette variable est positive alors on fait avancer la tortue de `longueur`, sinon on la fait reculer de `-longueur`. Ecrire les instructions python correspondantes.

## C5 Initiation à Python avec turtle

### Exemples

- ❶ Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0

```
1  if erreur==0:
```

- ❷ On suppose qu'une variable `longueur` peut être positive ou négative, si cette variable est positive alors on fait avancer la tortue de `longueur`, sinon on la fait reculer de `-longueur`. Ecrire les instructions python correspondantes.

```
1      if longueur > 0:
2          crayon.forward(longueur)
3      else:
4          crayon.backward(-longueur)
```

## Boucles while

## C5 Initiation à Python avec turtle

### Boucles while

- La syntaxe d'une boucle **while** en Python est :

```
1 while <condition>:  
2     <instruction>
```

Cela permet d'exécuter les <instructions> tant que la <condition> est vérifiée.



## C5 Initiation à Python avec turtle

### Boucles while

- La syntaxe d'une boucle **while** en Python est :

```
1 while <condition>:  
2     <instruction>
```

Cela permet d'exécuter les <instructions> tant que la <condition> est vérifiée.

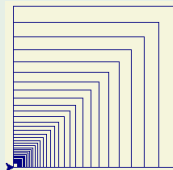
- On ne sait pas a priori combien de fois cette boucle sera exécutée (et elle peut même être infinie), on dit que c'est une boucle **non bornée**.

## C5 Initiation à Python avec turtle

### Exemple d'une boucle `while`

On suppose déjà crée une fonction `carre(c)` qui dessine un carré de côté `c` à partir de la position courante de la tortue. Ecrire un programme Python, permettant de tracer la figure suivante sachant que :

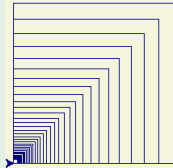
- le carré initial à 200 pixels de côté
- le côté des carrés intérieur diminue de dix pourcents à chaque étape
- le plus petit carré a un côté mesurant plus de 5 pixels.



# C5 Initiation à Python avec turtle

## Exemple d'une boucle while

```
1 cote = 200
2 while cote > 5:
3     carre(cote)
4     cote = cote * 0.9
```



## C5 Initiation à Python avec turtle

### Fonction renvoyant un résultat

En plus d'exécuter un bloc d'instructions, une fonction peut transmettre une valeur au reste du programme à l'aide d'une instruction `return`. On utilise alors la syntaxe suivante :

```
1     def <nom_fonction>(<arguments>):  
2         <instruction>  
3         return <valeur>
```

## C5 Initiation à Python avec turtle

### Exemple de fonction contenant un `return`

La fonction ci-dessous, renvoie la moyenne des deux nombres donnés en argument

```
1  def moyenne(x, y):  
2      m = (x+y)/2  
3      return m
```

## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.

### Exemples

## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.
- Une liste se note entre crochets : `[ et ]`

### Exemples

## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.
- Une liste se note entre crochets : [ et ]
- Les éléments sont séparés par des virgules

### Exemples



## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.
- Une liste se note entre crochets : [ et ]
- Les éléments sont séparés par des virgules

### Exemples

- Une liste `main` qui contient les noms des cinq doigts :

## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.
- Une liste se note entre crochets : [ et ]
- Les éléments sont séparés par des virgules

### Exemples

- Une liste main qui contient les noms des cinq doigts :  

```
main = ["pouce", "index", "majeur", "annulaire", "auriculaire"]
```

## C5 Initiation à Python avec turtle

### Les listes de Python

- Les listes de Python sont des structures contenant zéro, une ou plusieurs valeurs.
- Une liste se note entre crochets : [ et ]
- Les éléments sont séparés par des virgules

### Exemples

- Une liste main qui contient les noms des cinq doigts :  

```
main = ["pouce", "index", "majeur", "annulaire", "auriculaire"]
```
- Une liste l contenant un unique élément : 12  

```
l = [12]
```

## C5 Initiation à Python avec turtle

### Indice d'un élément

- Les éléments d'une liste sont repérés par leur position dans la liste, on dit leur **indice**

## C5 Initiation à Python avec turtle

### Indice d'un élément

- Les éléments d'une liste sont repérés par leur position dans la liste, on dit leur **indice**
- Attention, la numérotation commence à zéro, l'indice du premier élément de la liste est donc zéro

## C5 Initiation à Python avec turtle

### Indice d'un élément

- Les éléments d'une liste sont repérés par leur position dans la liste, on dit leur **indice**
- Attention, la numérotation commence à zéro, l'indice du premier élément de la liste est donc zéro
- On peut accéder à un élément en indiquant le nom de la liste puis l'indice de cet élément entre crochet

## C5 Initiation à Python avec turtle

### Indice d'un élément

- Les éléments d'une liste sont repérés par leur position dans la liste, on dit leur **indice**
- Attention, la numérotation commence à zéro, l'indice du premier élément de la liste est donc zéro
- On peut accéder à un élément en indiquant le nom de la liste puis l'indice de cet élément entre crochet
- L'erreur `IndexError` indique qu'on tente d'accéder à un indice qui n'existe pas.

Une liste L :

Eléments	L[0]	L[1]	L[2]	L[3]	L[4]
	↓	↓	↓	↓	↓
Indices	0	1	2	3	4

## C5 Initiation à Python avec turtle

### Exemple

On considère la liste de prénoms suivants :

```
prenoms = ["Alex", "Pierre", "Marie", "Jimmy", "Elise"]
```

- Compléter le schéma suivant permettant de représenter cette liste



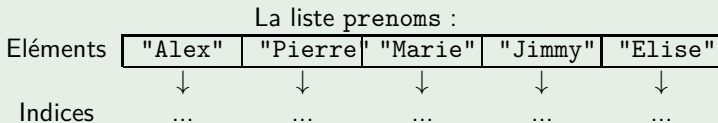
## C5 Initiation à Python avec turtle

### Exemple

On considère la liste de prénoms suivants :

```
prenoms = ["Alex", "Pierre", "Marie", "Jimmy", "Elise"]
```

- Compléter le schéma suivant permettant de représenter cette liste



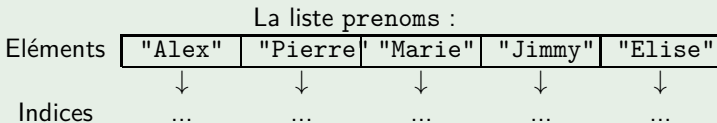
## C5 Initiation à Python avec turtle

### Exemple

On considère la liste de prénoms suivants :

```
prenoms = ["Alex", "Pierre", "Marie", "Jimmy", "Elise"]
```

- Compléter le schéma suivant permettant de représenter cette liste



- Que contient `prenoms[2]` ?

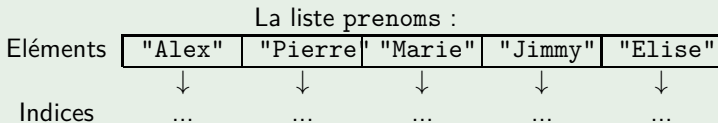
## C5 Initiation à Python avec turtle

### Exemple

On considère la liste de prénoms suivants :

```
prenoms = ["Alex", "Pierre", "Marie", "Jimmy", "Elise"]
```

- Compléter le schéma suivant permettant de représenter cette liste



- Que contient `prenoms[2]` ?
- Comment accéder au premier élément de cette liste (c'est à dire "Alex") ?

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste à partir de la fin en utilisant des index négatifs. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste **à partir de la fin** en utilisant des index **négatifs**. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction **len** renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste à partir de la fin en utilisant des index négatifs. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction `len` renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste à partir de la fin en utilisant des index négatifs. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction `len` renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

- Que renvoie `len(voyelles)` ?

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste à partir de la fin en utilisant des index négatifs. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction `len` renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

- Que renvoie `len(voyelles)` ?
- Que va afficher `print(voyelles[-2])` ?



## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste **à partir de la fin** en utilisant des index **négatifs**. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction **len** renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

- Que renvoie `len(voyelles)` ?
- Que va afficher `print(voyelles[-2])` ?
- Que va afficher `print(voyelles[2])` ?

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste **à partir de la fin** en utilisant des index **négatifs**. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction **len** renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

- Que renvoie `len(voyelles)` ?
- Que va afficher `print(voyelles[-2])` ?
- Que va afficher `print(voyelles[2])` ?
- Que va afficher `print(voyelles[6])` ?

## C5 Initiation à Python avec turtle

### Longueur et index négatif

- On peut accéder aux éléments d'une liste à partir de la fin en utilisant des index négatifs. L'indice  $-1$  est le dernier élément,  $-2$  l'avant dernier, ...
- La fonction `len` renvoie la longueur d'une liste, c'est à dire son nombre d'éléments.

### Exemples

On considère la liste `voyelles = ["a","e","i","o","u","y"]`

- Que renvoie `len(voyelles)` ?
- Que va afficher `print(voyelles[-2])` ?
- Que va afficher `print(voyelles[2])` ?
- Que va afficher `print(voyelles[6])` ?
- Donner deux façons d'afficher le dernier élément de cette liste.

## C5 Initiation à Python avec turtle

### Opérations sur les listes

Les opérations suivantes permettent de manipuler les listes (ajout, suppression, insertion d'éléments). On fera bien attention à la syntaxe on met le nom de la liste suivi d'un point suivi de l'opération à effectuer (voir exemples)

- **remove** permet de supprimer un élément d'une liste. Par exemple :  
`ma_liste.remove(elt)` va enlever `elt` de `ma_liste`.

## C5 Initiation à Python avec turtle

### Opérations sur les listes

Les opérations suivantes permettent de manipuler les listes (ajout, suppression, insertion d'éléments). On fera bien attention à la syntaxe on met le nom de la liste suivi d'un point suivi de l'opération à effectuer (voir exemples)

- **remove** permet de supprimer un élément d'une liste. Par exemple :  
`ma_liste.remove(elt)` va enlever `elt` de `ma_liste`.
- **append** permet d'ajouter un élément à la fin d'une liste. Par exemple :  
`ma_liste.append(elt)` va ajouter `elt` à la fin de `ma_liste`.

## C5 Initiation à Python avec turtle

### Opérations sur les listes

Les opérations suivantes permettent de manipuler les listes (ajout, suppression, insertion d'éléments). On fera bien attention à la syntaxe on met le nom de la liste suivi d'un point suivi de l'opération à effectuer (voir exemples)

- **remove** permet de supprimer un élément d'une liste. Par exemple :  
`ma_liste.remove(elt)` va enlever `elt` de `ma_liste`.
- **append** permet d'ajouter un élément à la fin d'une liste. Par exemple :  
`ma_liste.append(elt)` va ajouter `elt` à la fin de `ma_liste`.
- **insert** permet d'insérer un élément à un indice donnée. Par exemple :  
`ma_liste.insert(indice,elt)` va insérer `elt` dans `ma_liste` à l'index `indice`.

## C5 Initiation à Python avec turtle

### Opérations sur les listes

Les opérations suivantes permettent de manipuler les listes (ajout, suppression, insertion d'éléments). On fera bien attention à la syntaxe on met le nom de la liste suivi d'un point suivi de l'opération à effectuer (voir exemples)

- **remove** permet de supprimer un élément d'une liste. Par exemple :  
`ma_liste.remove(elt)` va enlever `elt` de `ma_liste`.
- **append** permet d'ajouter un élément à la fin d'une liste. Par exemple :  
`ma_liste.append(elt)` va ajouter `elt` à la fin de `ma_liste`.
- **insert** permet d'insérer un élément à un indice donnée. Par exemple :  
`ma_liste.insert(indice,elt)` va insérer `elt` dans `ma_liste` à l'index `indice`.
- **pop** permet de récupérer un élément de la liste tout en le supprimant de la liste. Par exemple `elt=ma_liste.pop(2)` va mettre dans `elt` `ma_liste[2]` et dans le même temps supprimer cet élément de la liste.

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`



## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

- Supprimer 'J' de cette liste ?

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

- Supprimer 'J' de cette liste ?
- Ajouter 'N' en fin de liste

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

- Supprimer 'J' de cette liste ?
- Ajouter 'N' en fin de liste
- Insérer 'Y' en indice 1

## C5 Initiation à Python avec turtle

### Exemples

On considère la liste `ex = ['P', 'T', 'O', 'J']`

- Supprimer 'J' de cette liste ?
- Ajouter 'N' en fin de liste
- Insérer 'Y' en indice 1
- Insérer 'H' en indice 3

## C5 Initiation à Python avec turtle

### Exemples

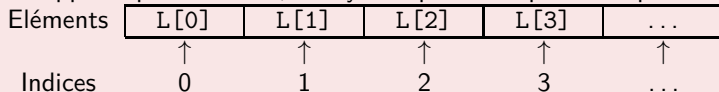
On considère la liste `ex = ['P', 'T', 'O', 'J']`

- Supprimer 'J' de cette liste ?
- Ajouter 'N' en fin de liste
- Insérer 'Y' en indice 1
- Insérer 'H' en indice 3
- Quel sera l'effet de l'instruction `lettre = ex.pop(3)` ?

## C5 Initiation à Python avec turtle

### Parcours d'une liste

On rappelle qu'une liste `L`, en Python peut se représenter par le schéma suivant :



On peut parcourir cette liste :



## C5 Initiation à Python avec turtle

### Parcours d'une liste

On rappelle qu'une liste `L`, en Python peut se représenter par le schéma suivant :

Eléments	L[0]	L[1]	L[2]	L[3]	...
	↑	↑	↑	↑	↑
Indices	0	1	2	3	...

On peut parcourir cette liste :

- **Par indice** (on se place sur la seconde ligne du schéma ci-dessus) et on crée une variable (un entier) qui va parcourir la liste des indices :

```
for indice in range(len(L))
```

Il faut alors accéder aux éléments en utilisant leurs indices.

## C5 Initiation à Python avec turtle

### Parcours d'une liste

On rappelle qu'une liste `L`, en Python peut se représenter par le schéma suivant :

Éléments	L[0]	L[1]	L[2]	L[3]	...
	↑	↑	↑	↑	↑
Indices	0	1	2	3	...

On peut parcourir cette liste :

- **Par indice** (on se place sur la seconde ligne du schéma ci-dessus) et on crée une variable (un entier) qui va parcourir la liste des indices :

```
for indice in range(len(L))
```

Il faut alors accéder aux éléments en utilisant leurs indices.

- **Par élément** (on se place sur la première ligne du schéma ci-dessus) et on crée une variable qui va parcourir directement la liste des éléments :

```
for element in L
```

La variable de parcours (ici `element`) contient alors directement les éléments).

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.

Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.  
Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre "n"

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.  
Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre "n"
- Le parcours par élément peut aussi se faire sur une chaîne de caractères.

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.  
Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre "n"
- Le parcours par élément peut aussi se faire sur une chaîne de caractères.  
Pour afficher chaque lettre du mot "Génial", on peut donc écrire :

## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.  
Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre "n"
- Le parcours par élément peut aussi se faire sur une chaîne de caractères.  
Pour afficher chaque lettre du mot "Génial", on peut donc écrire :  

```
for lettre in mot:
```



## C5 Initiation à Python avec turtle

### Liste et chaîne de caractères

- La notation avec les crochets permettant d'accéder aux éléments d'une liste s'utilise aussi avec les chaînes de caractères.

Par exemple si `mot = "Génial"` alors `mot[2]` contient la lettre "n"

- Le parcours par élément peut aussi se faire sur une chaîne de caractères.

Pour afficher chaque lettre du mot "Génial", on peut donc écrire :

```
for lettre in mot:  
    print(lettre)
```

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parcours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parcours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.
- 2 De calculer la somme des éléments de cette liste.

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parcours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.
- 2 De calculer la somme des éléments de cette liste.
- 3 De créer une nouvelle liste à partir de cette liste en ne conservant que les éléments inférieurs à 10

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parcours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.

```
1 notes = [17,12,9,11,13,15,8]
2 for note in notes:
3     if note > 10:
4         print(note)
```

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.
- 2 De calculer la somme des éléments de cette liste

```
1 notes = [17,12,9,11,13,15,8]
2 somme_notes=0
3 for note in notes:
4     somme_notes = somme_notes + note
```

## C5 Initiation à Python avec turtle

### Exemple

Soit la liste Python : `notes = [17,12,9,11,13,15,8]`, en effectuant des parcours de cette liste, écrire un programme permettant :

- 1 D'écrire les éléments de cette liste qui sont supérieurs à 10.
- 2 De calculer la somme des éléments de cette liste
- 3 De créer une nouvelle liste à partir de cette liste en ne conservant que les éléments inférieurs ou égaux à 10

```
1 notes = [17,12,9,11,13,15,8]
2 notes_inf_10 = []
3 for note in notes:
4     if note <= 10:
5         notes_inf_10.append(note)
```

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :



## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

```
bavardages = ["bla"]*4
```

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

```
bavardages = ["bla"]*4
```

- **Par compréhension**, c'est à dire en indiquant la définition des éléments qui composent la liste.

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

```
bavardages = ["bla"]*4
```

- **Par compréhension**, c'est à dire en indiquant la définition des éléments qui composent la liste.

Par exemple la liste `puissances2 = [1, 2, 4, 8, 16, 32, 64, 128]` est constitué des huit premières puissances de 2

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

```
bavardages = ["bla"]*4
```

- **Par compréhension**, c'est à dire en indiquant la définition des éléments qui composent la liste.

Par exemple la liste `puissances2 = [1, 2, 4, 8, 16, 32, 64, 128]` est constitué des huit premières puissances de 2

Elle contient donc  $2^0, 2^1, 2^2, \dots, 2^7$ , ce qui se traduit en Python par :

## C5 Initiation à Python avec turtle

### Création de listes

On peut créer des listes de diverses façons en Python :

- **Par ajout succesif d'élément** on part alors d'une liste (éventuellement vide) et on ajoute chaque élément à l'aide d'instruction `append`.
- **Par répétition du même élément** on utilise alors le caractère `*` pour indiquer le nombre de répétitions.

Par exemple pour créer la liste :

```
bavardages = ["bla", "bla", "bla", "bla"]
```

on peut simplement écrire :

```
bavardages = ["bla"]*4
```

- **Par compréhension**, c'est à dire en indiquant la définition des éléments qui composent la liste.

Par exemple la liste `puissances2 = [1, 2, 4, 8, 16, 32, 64, 128]` est constitué des huit premières puissances de 2

Elle contient donc  $2^0, 2^1, 2^2, \dots, 2^7$ , ce qui se traduit en Python par :

```
puissances2 = [2**k for k in range(8)]
```



## C5 Initiation à Python avec turtle

### Exemple

Créer les listes suivantes par le moyen qui vous semble le plus approprié :

- 1 La liste des 20 premiers multiples de 7

## C5 Initiation à Python avec turtle

### Exemple

Créer les listes suivantes par le moyen qui vous semble le plus approprié :

- 1 La liste des 20 premiers multiples de 7
- 2 La liste constituée de 100 zéros

## C5 Initiation à Python avec turtle

### Exemple

Créer les listes suivantes par le moyen qui vous semble le plus approprié :

- 1 La liste des 20 premiers multiples de 7
- 2 La liste constituée de 100 zéros
- 3 La liste des lettres de l'alphabet