

La boucle inconditionnelle ou boucle for

Intérêt : répéter une même suite d'instructions , plusieurs fois, dans un contexte à chaque fois différent.

Intérêt : répéter une même suite d'instructions , plusieurs fois, dans un contexte à chaque fois différent.

Boucle inconditionnelle ou boucle for : quand on est capable de dire **a priori** combien de fois un même type de traitement doit être réalisé.

Exemple de traitement répétitif :

Algorithme :

Données :

classe : liste contenant les noms des élèves d'une classe

notes : ensemble des résultats aux devoirs surveillés

Traitement :

pour chaque élève de la classe faire :

calculer la moyenne de l'élève

enregistrer cette moyenne dans le bulletin électronique de l'élève

fin pour

En Python, une première approche

Commandes for et range :

```
for i in range(debut,fin):  
    instruction1  
    ...  
    instructionN
```

En Python, une première approche

Commandes for et range :

```
for i in range(debut,fin):  
    instruction1  
    ...  
    instructionN
```

- utilisation (souvent) de `range`

En Python, une première approche

Commandes for et range :

```
for i in range(debut,fin):  
    instruction1  
    ...  
    instructionN
```

- utilisation (souvent) de **range**
- On ne modifie jamais la valeur contenue dans le compteur **i**

À retenir sur les boucles for



En Python, une instruction composée a une ligne d'en-tête se terminant par deux points,

À retenir sur les boucles for



En Python, une instruction composée a une ligne d'en-tête se terminant par deux points, et elle est suivie par un bloc d'instructions.

À retenir sur les boucles for



En Python, une instruction composée a une ligne d'en-tête se terminant par deux points, et elle est suivie par un bloc d'instructions.



En Python, un bloc d'instructions, est formé par des lignes successives indentées de la même façon.

Indentation significative !

À retenir sur les boucles for



En Python, une instruction composée a une ligne d'en-tête se terminant par deux points, et elle est suivie par un bloc d'instructions.



En Python, un bloc d'instructions, est formé par des lignes successives indentées de la même façon.

Indentation significative !

La commande `range(debut,fin)` fabrique le « nécessaire » pour que, dans une commande `for`, la variable utilisée prenne toutes les valeurs entières **de** `debut` inclus à `fin` exclu.

Exercice

Écrire la boucle Python permettant d'afficher la valeur du carré des dix premiers entiers (à savoir 1, 2, ..., 10).

Exercice

Écrire la boucle Python permettant d'afficher la valeur du carré des dix premiers entiers (à savoir 1, 2, ..., 10).



Exercice

Écrire la boucle Python permettant d'afficher la valeur du carré des dix premiers entiers (à savoir 1, 2, ..., 10).



```
for i in range(1,10+1):  
    print(i**2)
```

Exercice

Écrire la boucle Python permettant d'afficher la valeur du carré des quinze premiers entiers non nuls impairs.



Exercice

Écrire la boucle Python permettant d'afficher la valeur du carré des quinze premiers entiers non nuls impairs.



```
for i in range(0, 14+1): # 0 à 14 : 15 entiers
    impair = 2*i+1
    print(impair**2)
```

Boucles imbriquées

- Une boucle peut elle-même contenir une boucle.
- **Bien respecter la syntaxe.**

À RETENIR

Chaque boucle aura :

Boucles imbriquées

- Une boucle peut elle-même contenir une boucle.
- **Bien respecter la syntaxe.**

À RETENIR

Chaque boucle aura :

- sa propre variable de parcours

Boucles imbriquées

- Une boucle peut elle-même contenir une boucle.
- **Bien respecter la syntaxe.**

À RETENIR

Chaque boucle aura :

- sa propre variable de parcours
- une ligne d'en-tête terminée par “ : ”

Boucles imbriquées

- Une boucle peut elle-même contenir une boucle.
- **Bien respecter la syntaxe.**

À RETENIR

Chaque boucle aura :

- sa propre variable de parcours
- une ligne d'en-tête terminée par “ : ”
- et qui sera suivie de son bloc d'instructions associées **toutes indentées de la même manière.**

Exercice

Écrire le script Python permettant d'afficher les résultats de tous les produits des entiers de 1 à 10 par un entier compris entre 13 et 21. On affichera chaque résultat sur une nouvelle ligne de la forme "3 * 4 = 12".



Exercice

Écrire le script Python permettant d'afficher les résultats de tous les produits des entiers de 1 à 10 par un entier compris entre 13 et 21. On affichera chaque résultat sur une nouvelle ligne de la forme "3 * 4 = 12".



```
for i in range(1, 10+1):  
    for j in range(13, 21+1):  
        print(i, "*", j, "=", i*j)
```

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser :

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser : $s = 0$.

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser : $s = 0$.
- On prend le premier entier (1), on l'ajoute à la somme déjà calculée : $s = 0 + 1$,

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser : $s = 0$.
- On prend le premier entier (1), on l'ajoute à la somme déjà calculée : $s = 0 + 1$,
- On prend le second entier (2), on l'ajoute à la somme déjà calculée : $s = (0 + 1) + 2$,

Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser : $s = 0$.
- On prend le premier entier (1), on l'ajoute à la somme déjà calculée : $s = 0 + 1$,
- On prend le second entier (2), on l'ajoute à la somme déjà calculée : $s = (0 + 1) + 2$,
- Et ainsi de suite...

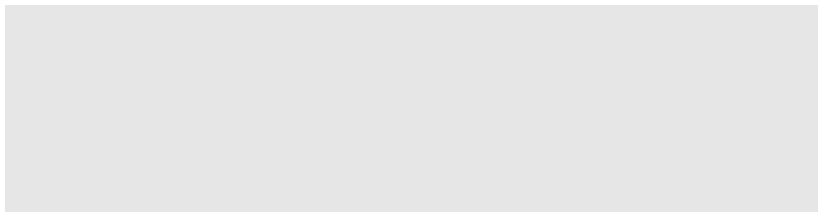
Somme des n premiers entiers

Comment fait-on à la main ?

- On prend en fait une variable s pour stocker la somme (en cours de calcul).
- Il faut l'initialiser : $s = 0$.
- On prend le premier entier (1), on l'ajoute à la somme déjà calculée : $s = 0 + 1$,
- On prend le second entier (2), on l'ajoute à la somme déjà calculée : $s = (0 + 1) + 2$,
- Et ainsi de suite...

On parcourt les entiers à sommer, et on les additionne à la somme partielle dans la variable s .

En Python :



En Python :



```
n = 10
s = 0
for i in range(1, n+1):
    s = s+i
print("La somme vaut", s)
```