

LES DONNEES DE TYPES CONSTRUIITS

1. Les tuples en Python (Activité 1)

2. Les listes en Python

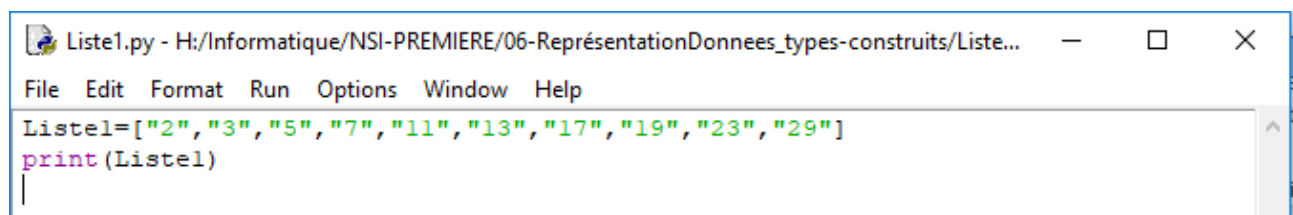
Les tuples (n-uplet) sont des objets non mutables. Une fois créé, un tuple ne peut en aucune manière être modifié.

En python, on dispose d'un autre type d'objet, les listes, qui sont des structures ordonnées de données placées les unes à la suite des autres.

Exemple 1 : Liste des 10 premiers nombres premiers. (2 ;3 ;5 ;7 ;11 ;13 ;17 ;19 ;23 ;29)

Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs. Ces deux diviseurs sont 1 et le nombre considéré, puisque tout nombre a pour diviseurs 1 et lui-même.

[Recopiez le code suivant et testez-le:](#)

A screenshot of a Python IDE window titled 'Liste1.py - H:/Informatique/NSI-PREMIERE/06-ReprésentationDonnees_types-construits/Liste...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
Listel=["2","3","5","7","11","13","17","19","23","29"]
print(Listel)
```

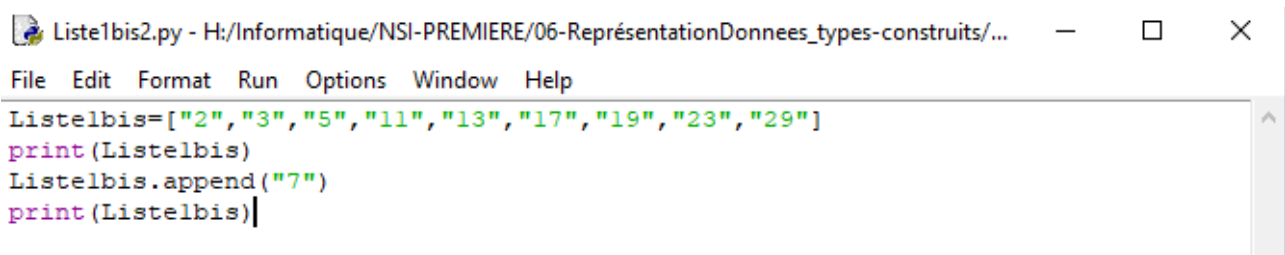
Un élève « distrait » oublie de saisir le nombre 7 !

Ecrivez le code correspondant en le nommant « Listelbis » et testez-le.

Heureusement, contrairement aux tuples, on peut modifier une liste, et on va donc pouvoir corriger l'erreur de cet « élève distrait » !

Pour ajouter un objet dans une liste, on utilise la commande « nom-de-liste.append »

Complétez le code précédent pour ajouter le nombre « 7 »



```
Listelbis2.py - H:/Informatique/NSI-PREMIERE/06-ReprésentationDonnees_types-construits/...
File Edit Format Run Options Window Help
Listelbis=["2","3","5","11","13","17","19","23","29"]
print(Listelbis)
Listelbis.append("7")
print(Listelbis)|
```

Copier-coller le résultat dans le cadre ci-dessous. Que remarquez-vous ?

Le nombre 7 est bien dans la nouvelle liste, mais a été ajouté à la fin.

Pour placer un élément dans une liste, on utilise la commande : « nom-de-liste.insert(i, x) »
Où **i** est l'indice de l'élément à insérer, et **x** la valeur de l'élément.

Modifiez le code précédent pour insérer le nombre « 7 » à la bonne place.

Exemple 2 : Un autre élève tout autant distrait que le premier créé comme liste :

```
Liste2=["1","2","3","5","11","13","17","19","23"]
```

Quelle erreur a-t-il commis ?

La commande pour enlever un élément d'une liste est : « nom-de-liste.remove("x") » où **x** est l'élément à enlever.

Ecrivez le programme qui crée la liste erronée, qui la fait afficher, puis qui corrige l'erreur commise et qui fait afficher la liste correcte.

On peut utiliser les indices pour accéder aux éléments d'une liste.

Quel nombre premier sera affiché par la commande : « `print(Liste2[3])` »

Vérifiez votre réponse en écrivant le code correspondant.

Remarque : on peut repérer un élément à partir de la fin de la liste, en utilisant des indices négatifs.

L'indice « -1 » correspond au dernier élément, l'indice « -2 » à l'avant dernier élément...

Quelle commande (avec indice négatif) faut-il écrire pour faire afficher le nombre « 19 » ?

Il existe un grand nombre de commandes pour travailler avec les listes.

`list.extend(iterable)`

Étend la liste en y ajoutant tous les éléments de « l'itérable ». Qui peut être une autre liste.

`list.pop(i)`

Enlève de la liste l'élément situé à la position indiquée et le renvoie en valeur de retour.

Si aucune position n'est spécifiée, `liste.pop()` enlève et renvoie le dernier élément de la liste

`list.clear()`

Supprime tous les éléments de la liste.

`list.count(x)`

Renvoie le nombre d'éléments ayant la valeur *x* dans la liste.

`list.reverse()`

Inverse l'ordre des éléments de la liste, sur place.

`list.copy()`

Renvoie une copie superficielle de la liste.

Testez ces commandes pour vous les approprier.