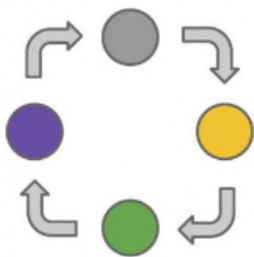




La Plateforme

Runtrack Python

Python is powerful... and fast; and open; and ... many other things.



En Python, les boucles et les conditions sont des structures de contrôle fondamentales qui facilitent la gestion du flux d'exécution dans un programme. Les boucles **"for"** et **"while"** permettent d'itérer à travers des séquences de données ou d'exécuter un bloc de code tant qu'une condition spécifiée est vraie.

Les conditions, implémentées avec des déclarations **"if"**, **"else"**, et **"elif"** (**else if**), permettent d'établir des branches conditionnelles dans le code. Une instruction **"if"** teste une condition et exécute un bloc de code si cette condition est vraie. L'instruction **"else"** peut être utilisée pour spécifier un bloc de code à exécuter lorsque la condition de l'instruction **"if"** est fausse. De plus, les instructions **"elif"** permettent d'ajouter des conditions alternatives à considérer.

Job 01

Créez un programme qui parcourt les nombres de **0** à **20**, **affichez** chacun des chiffres dans le terminal.

Job 02

```
0
2
4
6
8
10
12
14
16
18
```

Créer un programme qui parcourt les nombres de **0** à **20**, affichez **1 nombre sur 2** dans le terminal.

Job 03

Créer un programme qui affiche tous les nombres de **0** à **100** compris SAUF **26, 37, 88**

Job 04

```
Entrez un entier supérieur à zéro (N) :
Table de multiplication de 1 :
```

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

```
Table de multiplication de 2 :
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Créer un programme qui affiche en console **les tables de multiplication** de **1** à **N**. N étant un **entier** supérieur à zéro saisi par l'utilisateur.

N'oubliez pas de vérifier tout ce qui est nécessaire pour assurer la fiabilité de votre programme.

Job 05

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
```

Écrire un programme qui itère les nombres entiers de **1** à **100**. Pour les multiples de **trois**, afficher "**Fizz**" au lieu du nombre et pour les multiples de **cinq** afficher "**Buzz**". Pour les nombres qui sont des multiples de **trois** et **cinq**, afficher "**FizzBuzz**".

...

Job 06

```
2
3
5
7
11
13
17
19
23
29
31
37
41
```

Écrire un programme qui affiche les nombres premiers jusqu'à **1000**.

Job 07

```
a
abc
abcde
abcdefg
abcdefghi
abcdefghijk
abcdefghijklm
abcdefghijklmno
abcdefghijklmnopq
abcdefghijklmnopqrs
abcdefghijklmnopqrstu
abcdefghijklmnopqrstuvw
abcdefghijklmnopqrstuvwxy
abcdefghijklmnopqrstuvwxyza
abcdefghijklmnopqrstuvwxyzaabc
```

À partir de la chaîne "**abcdefghijklmnopqrstuvwxy**" *
10, écrivez un programme qui **récupère** et **affiche** autant
de caractères que possible de cette chaîne sous forme
de suite pyramidale.

... pour aller plus loin.

Créer un programme qui demande à l'utilisateur **trois longueurs** a, b, c. À l'aide de ces
trois longueurs, déterminer s'il est possible de construire un triangle.

Déterminer ensuite si ce triangle est rectangle, isocèle, équilatéral ou quelconque.

Attention : un triangle rectangle peut être isocèle.

Rendu

Créer sur github un répertoire nommé “runtrack-python”. Créer dans ce répertoire un dossier “**jour02**” et pour chaque étape, un dossier “**jobXX**” où XX est le numéro de l’étape.

Compétences visées

- Maîtriser les bases de python
 - Implémenter un algorithme
-

Base de connaissances

- [Les bases du développement en python](#)
- [Les boucles](#)
- [Les conditions](#)