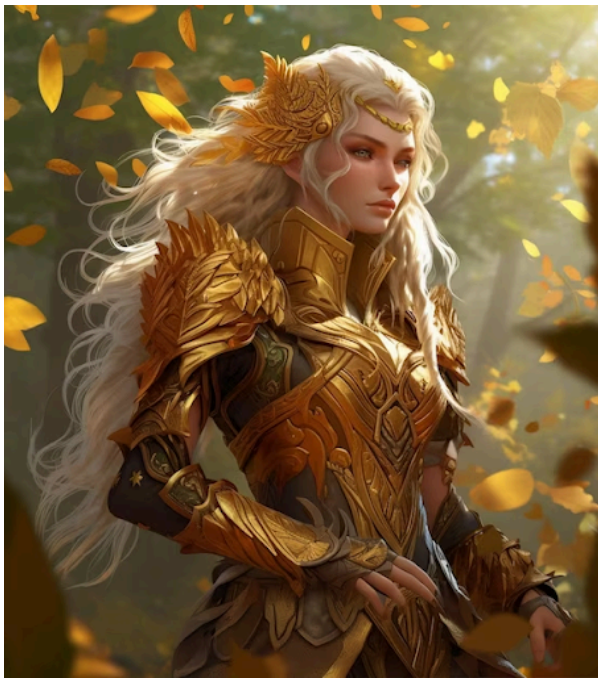
Runtrack Python

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

Dans le vaste jardin du code informatique, les conditions jouent le rôle de chemins, tracés avec les déclarations "**if**", "**else**", et "**elif**" ("else if"). Elles sont les portes d'entrée vers des destinations différentes selon les circonstances.



L'instruction "**if**" sert de gardienne, scrutant une condition donnée.

Si cette condition se révèle vraie, elle ouvre la voie au code qui la suit, permettant ainsi l'exécution d'actions spécifiques.

Lorsque la réalité contredit la condition initiale de "**if**", l'instruction "**else**" se lève, prête à prendre en charge une alternative. Elle guide le flot du programme vers un bloc de code spécifié, répondant ainsi à une situation où la première voie n'est plus viable.

Les instructions "**elif**" enrichissent ce processus de décision en introduisant des options alternatives. Comme des sentiers secondaires dans une forêt dense, elles offrent des chemins supplémentaires à explorer si les conditions précédentes ne correspondent pas. Chaque "**elif**" apporte une nouvelle perspective, permettant au programme de considérer d'autres critères avant de prendre une décision finale.

Job 01

Créer un programme qui demande deux valeurs à l'utilisateur et qui stocke les valeurs dans des variables nommées : "**valeur1**" et "**valeur2**". Si le contenu de la variable "**valeur1**" est égal à la "**valeur2**" afficher : "Valeur1 est égal à valeur2" sinon "Les deux valeurs ne sont pas égales".

Job 02

Initialiser une variable contenant un chiffre entier représentant l'age d'une personne. Si la personne **au moins 18** ans **afficher** "Tu peux voter" "Tu ne peux pas voter".

Job 03

Créez un programme qui affiche dans le terminal tous les nombres de **0** à **100** compris SAUF **26**, **37**, **88**.

Job 04

```
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**, affichez « **Fizz** » au lieu du nombre et pour les multiples de **cinq**, affichez « **Buzz** ». Pour les nombres qui sont des multiples de **trois** et de **cinq**, affichez « **FizzBuzz** ».

...

Job 05

```
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 06

Écrire un programme qui vérifie si un nombre renseigné par l'utilisateur est **pair**. Si le nombre est **pair,** afficher le message "Le nombre X est pair".
Ajouter ensuite a votre programme un message lorsque le nombre est **impair**.

Job 07

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

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

Job 08

Créez un programme qui demande à l'utilisateur **trois longueurs**, a, b, c. À l'aide de ces trois longueurs, déterminez s'il est possible de construire un triangle. Déterminez ensuite si ce triangle est rectangle, isocèle, équilatéral ou quelconque.
Attention : un triangle rectangle peut être isocèle.

Rendu

Que vous aillez fini vos jobs ou pas, rendre sur le projet sur votre intranet et remplir le QCM.

Compétences visées

- ➔ Maîtriser les bases de Python
- ➔ Implémenter un algorithme

Base de connaissances

- ➔ Les bases du développement en Python
- ➔ Les boucles
- ➔ Les conditions