02_structures_controle_suite

October 10, 2025

Exercice 1 : Supposons que vous souhaitiez développer un programme permettant à un élève de première année de s'entraîner à la soustraction. Le programme génère de manière aléatoire deux entiers d'un seul chiffre, number1 et number2, avec number1 >= number2, et pose à l'élève une question telle que "Quel est 9 - 2 ?" Après que l'élève ait saisi la réponse, le programme affiche un message indiquant si elle est correcte. Ecrire un programme qui génère cinq questions et, après qu'un élève y ait répondu, rapporte le nombre de réponses correctes. Le programme affiche également le temps passé sur le test, comme le montre l'exécution d'exemple. Pour mesurer le temps, il faut importer la bibliothèque *Time* (exemple d'utilisation: temps_de_depart = time.time()).

[ ]:

Exercice 2 : L'exemple précédent exécute la boucle cinq fois. Si vous souhaitez que l'utilisateur décide s'il souhaite prendre une autre question, vous pouvez proposer une confirmation à l'utilisateur (en tapant 'Y' pour continuer).

[ ]:

Exercice 3 : (Trouver les nombres divisibles par 5 et 6) Écrivez un programme qui affiche, dix nombres par ligne, tous les nombres de 100 à 1 000 qui sont divisibles par 5 et 6. Les nombres sont séparés par exactement un espace.

[ ]:

Exercice 4 : Écrivez un programme qui joue au populaire jeu ciseaux-pierre-papier. (Un ciseau peut couper du papier, une pierre peut écraser un ciseau, et du papier peut envelopper une pierre.) Le programme génère aléatoirement un nombre 0, 1 ou 2, représentant respectivement ciseaux, pierre et papier. Ensuite, le programme demande à l'utilisateur d'entrer un nombre 0, 1 ou 2, puis affiche un message indiquant si l'utilisateur gagne, perd ou obtient un match nul par rapport à l'ordinateur. Le programme permet à l'utilisateur de jouer en continu jusqu'à ce que l'utilisateur ou l'ordinateur gagne plus de deux fois.

[ ]:

Exercice 5 : Écrivez un programme pour vérifier la validité des mots de passe saisis par les utilisateurs.

```
-Au moins 1 lettre entre [a-z] et 1 lettre entre [A-Z].
-Au moins 1 chiffre entre [0-9].
-Au moins 1 caractère parmi [$#@].
-Longueur minimale de 6 caractères.
```

-Longueur maximale de 16 caractères.

La fonction *re.search()* du module *re* en Python est utilisée pour rechercher un motif (expression régulière) dans une chaîne de caractères. Voici la syntaxe simplifiée de re.search() : re.search(pattern, string)

-pattern : C'est le motif (expression régulière) que vous souhaitez rechercher dans la chaîne de caractères.
-string : C'est la chaîne de caractères dans laquelle vous souhaitez effectuer la recherche.

La fonction re.search() renvoie un objet de correspondance (match object) si le motif est trouvé dans la chaîne de caractères. Si le motif n'est pas trouvé, elle renvoie None.

[ ]:

Exercice 6 : Écrivez un premier programme qui permet de chiffrer une chaîne de caractères en utilisant un décalage de x lettres vers la droite dans l'ordre de l'alphabet. Attention, x est un entier dont les valeurs sont comprises entre 0 et 25.

Exemple : Si on fait un décalage de 3, le mot "bonjour" devient "erqmrxu"

Dans un deuxième temps, écrivez également le programme qui déchiffre le mot, en retrouvant le mot initial à partir d'un mot chiffré et de l'entier utilisé pour le décalage.