

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°3 : Structures de branchement et des boucles répétitives (suite)

BTS CIEL
2 ^{ème} année
Page 1 sur 3

Boucle for :

Saisir le programme suivant et lancer le :

```

nom = 'KORRI' # Déclaration de variable de type str
langage = 'Python' # Déclaration d'une autre variable de type str

print("\n1. #####")
for i in range(len(nom)): # pour i in longueur de la chaine
    print(nom[i], end=" ") # Affiche successivement les caractères

print("\n\n2. #####")
for i in range(len(langage)): # pour i in longueur de la chaine
    print(langage[i], end=" ") # end = " " : pour ne pas sauter de ligne
print(f"\n{len(langage)}") # Affiche la longueur de la chaine

print("\n3. #####")
for i in range(6): # collection de 0 à 6-1
    print(i)

print("\n4. #####")
for i in range(5): # Collection de 0 à 5-1
    print("KORRI")

print("\n5. #####")
for i in range(4, 8): # départ 4 à la valeur finale 8-1
    print(i)

print("\n6. #####")
for i in range(8, 4, -1): # départ 8, finale 4+1 (-1 : décrémenter de 1)
    print(i)

print("\n7. #####")
for i in range(0, 10, 2): # de 0 à 9 en incrémentant de 2
    print(i)

print("\n8. #####")
for element in langage: # caractère par caractère
    print(element)

```

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°3 : Structures de branchement et des boucles répétitives (suite)

BTS CIEL
2 ^{ème} année
Page 2 sur 3

Exercice 1 : Tentatives de connexion brute-force (boucle while)

Écrire un script qui simule la détection de plusieurs tentatives de connexion d'un même utilisateur. Il doit compter le nombre de fois qu'un utilisateur, identifié par un nom ou un ID, essaie de se connecter à un serveur en entrant plusieurs noms d'utilisateur successifs, et arrêter la détection après un nombre défini de tentatives échouées.

Script de départ :

```
# Utilisateur correct prédéfini
utilisateur_correct = "admin"

# Nombre maximum de tentatives
max_tentatives = 3

# Compteur de tentatives
tentatives = 0
```

Exercice 2 : Surveillance de capteurs IoT avec des alertes

Écrire un programme qui permet de simuler la collecte des données provenant de plusieurs capteurs de température IoT. Le programme doit parcourir les données transmises par ces capteurs et générer une alerte si une des valeurs dépasse un seuil prédéfini.

Étapes :

1. Générer 10 valeurs de température envoyées par un capteur IoT.
2. Utiliser la boucle **for** pour parcourir les 10 valeurs collectées (simulées).
3. Si une valeur dépasse un seuil critique (par exemple, 35°C), une alerte est déclenchée.
4. Le programme affichera les valeurs normales et les alertes.

Pour afficher deux chiffres après la virgule :

```
print(f"Température mesurée : {temperature:.2f}°C")
```

TRAVAUX PRATIQUES
La programmation en Python
T.P. n°3 : Structures de branchement et des boucles répétitives (suite)

BTS CIEL
2 ^{ème} année
Page 3 sur 3

Exercice 3 : Vérification de mots de passe

Écrire un programme qui demande à l'utilisateur de définir un mot de passe, puis de vérifier si l'utilisateur est capable de se reconnecter avec ce même mot de passe. Le programme limite le nombre de tentatives.

Étapes :

1. Le programme demande à l'utilisateur de créer un mot de passe.
2. L'utilisateur a 3 tentatives pour entrer le mot de passe correct avant que l'accès ne soit refusé.
3. Selon les tentatives, afficher si la connexion est réussie ou échouée.

Exercice 4 : Vérification de format d'une clé Wi-Fi (WPA/WPA2)

Écrire un programme qui demande à l'utilisateur d'entrer une clé Wi-Fi WPA ou WPA2 et de vérifier si la clé est valide en fonction de sa longueur. Une clé WPA/WPA2 doit contenir entre 8 et 63 caractères alphanumériques.

Étapes :

1. Demander à l'utilisateur d'entrer une clé Wi-Fi.
2. Vérifier que la longueur de la clé est comprise entre 8 et 63 caractères.
3. Vérifier que tous les caractères sont alphanumériques (lettres et chiffres uniquement).
4. Afficher un message indiquant si la clé est valide ou non.

`caractere.isalnum()` : **permet de vérifier que chaque caractère est alphanumérique**