République Tunisienne
Ministère de de l'Enseignement Supérieur
Et de la Recherche
Scientifique

Institut Supérieur des Études
Technologiques de Tozeur

Groupe : Ti
Durée : 1h30min

Enseignant : CHEBBI.M

Année universitaire : 2024/2025

TP 2 : Programmation Python

1. Manipulation de variables simples et des chaînes de caractères

Exercice 1 : Nombre pair ou impair

Créez une fonction qui prend un nombre comme argument et renvoie "pair" pour les nombres pairs et "impair" pour les nombres impairs.

```
print(check(2))  # Résultat attendu : "pair"
print(check(7))  # Résultat attendu : "impair"
```

Exercice 2 : Différence de salaire

Vous avez embauché trois commerciales et vous les payez. Créez une fonction qui prend trois nombres (le salaire horaire de chaque commerciale) et renvoie la différence entre la commerciale la mieux payée et la moins payée.

```
print(getDiff(200, 10, 90))  # Résultat attendu : 190
print(getDiff(56, 29, 16))   # Résultat attendu : 40
```

Exercice 3 : Longueur d'une chaîne de caractères

Écrivez une fonction `longueur_chaine(texte)` qui prend en paramètre une chaîne de caractères et retourne sa longueur (le nombre de caractères dans la chaîne).

```
print(longueur_chaine("Bonjour"))  # Résultat attendu : 7
```

Exercice 4 : Vérification d'une chaîne vide

Écrivez une fonction `chaine_vide(texte)` qui retourne `True` si la chaîne donnée est vide, et `False` sinon.

```
print(chaine_vide(""))  # Résultat attendu : True
print(chaine_vide("Python"))  # Résultat attendu : False
```

Exercice 5 : Transformation en majuscules et minuscules

En utilisant les deux méthodes `.upper() et .lower()`. Écrivez deux fonctions :

- `majuscule(texte)` qui retourne la chaîne en majuscules.
- `minuscule(texte)` qui retourne la chaîne en minuscules.

```
print(majuscule("hello"))  # Résultat attendu : "HELLO"
print(minuscule("HELLO"))  # Résultat attendu : "hello"
```

Exercice 6 : Inverser une chaîne de caractères
Écrivez une fonction `inverser_chaine(texte)` qui retourne la chaîne de caractères inversée.

```
print(inverser_chaine("Python"))  # Résultat attendu : "nohtyP"
```

Exercice 7 : Vérifier divisibilité par 5

Créez une fonction qui renvoie `True` si un entier est divisible par 5, sinon renvoie `False`.

```
print(isDivisibleBy5(5))   # Résultat attendu : True
print(isDivisibleBy5(-5))  # Résultat attendu : True
print(isDivisibleBy5(3))   # Résultat attendu : False
```

Exercice 8 : Déterminer si un mot est au pluriel
Créez une fonction qui prend un mot et détermine s'il est pluriel ou singulier. Un mot pluriel est celui qui se termine par "s". S'il est pluriel, renvoyer `True`, sinon `False`.

```
print(checkIsPlural("enfants"))  # Résultat attendu : True
print(checkIsPlural("filles"))   # Résultat attendu : True
print(checkIsPlural("fille"))    # Résultat attendu : False
```

2. Manipulation des listes

Exercice 1 : Trouver le plus grand élément d'une liste
Créez une fonction qui prend une liste de nombres, et renvoie le plus grand nombre de la liste.

```
print(max_element([6, 9, 1, 2]))    # Résultat attendu : 9
print(max_element([10, 66, 12, 98]))  # Résultat attendu : 98
print(max_element([1, 1, 1, 1, 1]))  # Résultat attendu : 1
```

Exercice 2 : Trouver la somme des éléments d'une liste
Créez une fonction qui prend une liste et renvoie la somme de tous les nombres de la liste.

```
print(sommeL([1, 2, 3, 4]))  # Résultat attendu : 10
print(sommeL([0, 0, 0, 1]))  # Résultat attendu : 1
print(sommeL([-1, -2, 3]))   # Résultat attendu : 0
```

Exercice 3 : Compter les occurrences d'un élément dans une liste
Écrivez une fonction `compter_occurrences(liste, element)` qui prend une liste et un élément, et

retourne le nombre d'occurrences de cet élément dans la liste en utilisant la méthode `.count()`.

```
print(compter_occurrences([1, 2, 3, 1, 1, 4], 1))  # Résultat attendu : 3
```

Exercice 4 : Ajouter un élément à la liste
Écrivez une fonction `ajouter_element(liste, element)` qui ajoute un élément à la fin de la liste en utilisant la méthode `.append()`.

```
ma_liste = [1, 2, 3]
ajouter_element(ma_liste, 4)
print(ma_liste)  # Résultat attendu : [1, 2, 3, 4]
```

Exercice 5 : Supprimer un élément de la liste
Écrivez une fonction `supprimer_element(liste, element)` qui supprime un élément de la liste en utilisant la méthode `.remove()`. Si l'élément n'est pas trouvé, la fonction doit retourner un message d'erreur.

```
ma_liste = [1, 2, 3, 4]
supprimer_element(ma_liste, 3)
print(ma_liste)  # Résultat attendu : [1, 2, 4]
print(supprimer_element(ma_liste, 5))  # Résultat attendu : "L'élément 5 n'est pas dans la liste"
```

3. Combinaison de chaînes et de listes

Exercice 1 : Créer une phrase à partir d'une liste de mots
Écrivez une fonction `creer_phrase(liste_mots)` qui prend en paramètre une liste de mots et retourne une chaîne de caractères formée de ces mots, séparés par des espaces. En utilisant la méthode `.join()`.

```
print(creer_phrase(["Bonjour", "tout", "le", "monde"]))  # Résultat attendu : "Bonjour tout le monde"
```

Exercice 2 : Convertir une chaîne en liste de mots
Écrivez une fonction `chaine_en_liste(texte)` qui prend une chaîne de caractères et retourne une liste de mots. En utilisant la méthode `.split()`.

```
print(chaine_en_liste("Bonjour tout le monde"))  # Résultat attendu : ["Bonjour", "tout", "le", "monde"]
```

4. Fonctions prédéfinies supplémentaires

Exercice 1 : Vérifier si un élément existe dans une liste
Écrivez une fonction `element_existe(liste, element)` qui retourne `True` si l'élément existe dans la liste, sinon retourne `False`.

```
print(element_existe([1, 2, 3], 2))  # Résultat attendu : True
print(element_existe([1, 2, 3], 4))  # Résultat attendu : False
```

Exercice 2 : Trier une liste

Écrivez une fonction `trier_liste(liste)` qui trie une liste de nombres en utilisant la méthode `.sort()`.

```
ma_liste = [3, 1, 2]
trier_liste(ma_liste)
print(ma_liste)  # Résultat attendu : [1, 2, 3]
```

Exercice : Rechercher un caractère dans une chaîne

Créez une fonction `rechercher_caractere(texte, caractere)` qui recherche un caractère spécifique dans une chaîne et retourne son index (ou un message si le caractère n'est pas trouvé). En utilisant la méthode `.find()`.

```
print(rechercher_caractere("Python", "t"))  # Résultat attendu : 2
print(rechercher_caractere("Python", "z"))  # Résultat attendu : -1
```