Listes et fonctions Python

LISTES

TERMINALE STI2D / SPÉCIALITÉ SIN / FICHE ÉLÈVE

Nom : ..... Classe : ..... Date : .....

Objectif • stocker des mesures et créer des fonctions pour un radar de recul simulé. Prévoir les résultats, compléter la fiche, puis tester les programmes sur ordinateur. Les exercices de programmation sont à réaliser dans l'éditeur Python.

1 Lire une liste

Une liste regroupe des valeurs entre crochets. L'indice indique leur position et commence à 0. len(liste) donne le nombre de valeurs.

```
temperatures = [21, 23, 22, 27, 25]
print(temperatures[0])
print(temperatures[2])
print(len(temperatures))
```

1. Prévoir les trois valeurs affichées, dans l'ordre. .....

2. Écrire l'instruction qui affiche la dernière température. .....

3. Pourquoi temperatures[5] provoque-t-il une erreur ? .....

2 Parcourir une liste

La variable temperature prend successivement chaque valeur de la liste.

```
temperatures = [21, 23, 22, 27, 25]
for temperature in temperatures:
    print(temperature)
```

4. Combien de fois print est-il exécuté ? .....

5. Dans la boucle, ajouter une condition qui affiche "ALERTE" lorsque la température est strictement supérieure à 25 °C. Tester.

6. Combien d'alertes obtient-on ? La valeur 25 déclenche-t-elle une alerte ?
.....

3 Ajouter une valeur avec append

[] crée une liste vide. liste.append(valeur) ajoute une valeur à la fin.

```
mesures = []
mesures.append(18)
mesures.append(22)
mesures.append(25)
```

7. Compléter le contenu de mesures après chaque ajout.

Après append(18)	Après append(22)	Après append(25)
.....	.....	.....

8. Compléter, puis tester avec les valeurs 18, 22 et 25.

```
mesures = []
for i in range(3):
    valeur = int(input("Mesure : "))
    mesures._____(_____)
print(mesures)
```

9. Ajouter 30 à la liste après la boucle avec append, puis afficher len(mesures).

Quelle longueur obtient-on ? .....

Listes et fonctions Python

FONCTIONS

TERMINALE STI2D / SPÉCIALITÉ SIN / FICHE ÉLÈVE

4 Définir et appeler des fonctions

def définit une fonction : son corps est indenté. L'appel avec des parenthèses exécute ce corps. Un paramètre reçoit la valeur passée lors de l'appel.

Sans paramètre puis avec un paramètre

```
def demarrer_radar():
    print("Radar actif")

def afficher_distance(distance):
    print("Distance :", distance, "cm")

demarrer_radar()
afficher_distance(80)
```

10. Souligner les deux appels. Quel est le nom du paramètre ? .....

11. Ajouter un appel qui affiche une distance de 35 cm. Tester. Puis supprimer les trois appels : un message est-il encore affiché ? .....

Renvoyer un résultat avec return

return renvoie une valeur et termine l'appel. Le résultat peut être mémorisé ; il ne s'affiche pas automatiquement. Ici, on convertit des cm en m.

```
def convertir_en_metres(distance_cm):
    return distance_cm / 100

resultat = convertir_en_metres(150)
print(resultat)
```

12. Que vaut resultat ? Quelle ligne l'affiche ? .....

13. Tester la conversion de 45 cm. Puis écrire corriger_distance(distance_cm), qui renvoie la mesure augmentée de 2 cm. Appeler la fonction avec 48 et afficher le résultat : .....

5 Réaliser un radar de recul simulé

Les distances du capteur sont simulées en cm. Règle choisie pour l'exercice : en dessous de 50 cm, renvoyer "STOP" ; sinon, renvoyer "LIBRE".

14. Compléter le programme : la fonction décide du message, puis la boucle traite les distances et stocke les messages dans une nouvelle liste.

```
def choisir_message(distance):
    if distance < _____________:
        return "_____________"
    else:
        return "_____________"

distances = [120, 75, 50, 35, 20]
messages = []
for distance in _____________:
    message = choisir_message(_____________)
    messages._____________(message)
print(messages)
```

15. Prévoir la liste messages, puis vérifier sur ordinateur. .....

16. Tester avec distances = [49, 50, 51]. Messages obtenus, dans l'ordre :
.....

17. Ajouter 10 à distances avec append, avant la boucle. Tester : les listes distances et messages ont-elles la même longueur ? Expliquer oralement.

Bilan à expliquer • à quoi servent un indice, len, append, un paramètre et return ?