# Les fonctions en Python

SNT – Seconde • Fiche d'exercices et résumé de cours

## Rappel de cours — La syntaxe d'une fonction

Une fonction est un **bloc de code réutilisable** auquel on donne un nom. On la définit une seule fois, puis on l'**appelle** autant de fois qu'on le souhaite.

```
def nom_de_la_fonction(parametre):
    resultat = ...          # on fait le calcul
    return resultat         # on renvoie le résultat
```

| | |
|---|---|
| **`def`** | Mot-clé qui démarre la définition d'une fonction |
| **`parametre`** | La valeur que l'on transmet à la fonction (il peut y en avoir plusieurs) |
| **`return`** | Renvoie le résultat — la ligne qui termine le travail de la fonction |
| **indentation** | Le corps de la fonction doit être décalé de 4 espaces (obligatoire en Python !) |

*Remarques:*

- *Le choix du nom de la fonction est libre, mais ne doit pas commencer par un chiffre ni contenir d'espace*
- *Les informations placées après le # sont des commentaires et ne sont pas executés par le programme.*

## Partie 1 — Créer une fonction simple

### Exemple étudié en classe

La fonction **`metres_en_km`** convertit une distance en mètres en kilomètres (on divise par 1000) :

```
def metres_en_km(distance_m):
    distance_km = distance_m / 1000
    return distance_km
```

Pour l'**appeler** (**utiliser**), on écrit :

```
print(metres_en_km(500))    # affiche 0.5
print(metres_en_km(42195))  # affiche 42.195
```

| Nom / Prénom : | | Classe : | |
|---|---|---|---|

# Partie 1 — Créer une fonction simple

### ✏️ Exercice 1 — Écrire une première fonction

Sur le même modèle, écris la fonction **`secondes_en_heures(duree_s)`** qui convertit une durée en secondes en heures (on divise par 3600).

```
def secondes_en_heures(duree_s):
    duree_h = ...           # à compléter
    return ...
```

Teste ta fonction en complétant les cases ci-dessous :

| Appel | Calcul (montre les étapes) | Résultat attendu |
|---|---|---|
| `secondes_en_heures(3600)` | 3600 / 3600 = | **1.0 h** |
| `secondes_en_heures(5400)` | | **1.5 h** |
| `secondes_en_heures(45)` | | |

# Partie 2 — Fonction avec plusieurs paramètres

## Exemple étudié en classe

La fonction **`vitesse_kmh`** calcule une vitesse en km/h à partir d'une distance en mètres et d'un temps en secondes. Elle réutilise les deux fonctions précédentes :

```
def vitesse_kmh(distance_m, temps_s):
    d_km = metres_en_km(distance_m)      # convertit en km
    t_h  = secondes_en_heures(temps_s)   # convertit en heures
    v    = d_km / t_h                    # calcule la vitesse
    return v
```

Exemple d'appel :

```
v_voiture = vitesse_kmh(500, 45)
print(f"Voiture : {v_voiture:.1f} km/h")   # affiche : Voiture : 40.0
km/h
```

### ✏️ Exercice 2 — Appeler une fonction à deux paramètres

Lors d'une sortie sportive, trois élèves ont chacun couru 400 m. Voici leurs temps :

| Élève | Distance (m) | Temps (s) | Vitesse (km/h) |
|---|---|---|---|
| Alice | 400 | 72 | |
| Bruno | 400 | 68 | |
| Chloé | 400 | 75 | |

Complète les appels de fonction, puis calcule le résultat à la main et écris-le dans la dernière colonne du tableau.

```
v_alice = vitesse_kmh(..., ...)
v_bruno = vitesse_kmh(..., ...)
v_chloe = vitesse_kmh(..., ...)

print(f"Alice : {v_alice:.1f} km/h")
print(f"Bruno : {v_bruno:.1f} km/h")
print(f"Chloe : {v_chloe:.1f} km/h")
```

## Partie 3 — Documenter une fonction

Une bonne fonction s'accompagne d'une **docstring** : une courte description entre triple guillemets `"""` qui explique ce qu'elle fait, ses paramètres, et ce qu'elle renvoie.

```
def vitesse_kmh(distance_m, temps_s):
    """
    Calcule la vitesse en km/h.
    Parametres :
      distance_m : distance parcourue en metres
      temps_s    : duree du trajet en secondes
    Renvoie : la vitesse en km/h (float)
    """
    d_km = metres_en_km(distance_m)
    t_h  = secondes_en_heures(temps_s)
    return d_km / t_h
```

### ✏️ Exercice 3 — Ajouter une docstring

La fonction **`secondes_en_heures`** ci-dessous n'a pas de docstring. Complète-la dans l'espace prévu :

```
def secondes_en_heures(duree_s):
    """
    ...
    ...
    ...
    """
    duree_h = duree_s / 3600
    return duree_h
```

# Exercice Final — Analyse d'une course de natation

Lors d'une compétition, on a relevé les temps de trois nageurs sur plusieurs distances.

| Nageur | 50 m (s) | 100 m (s) | 200 m (s) |
|---|---|---|---|
| Léo | 28.4 | 59.1 | 128.5 |
| Maya | 27.9 | 58.3 | 125.2 |
| Théo | 29.1 | 61.0 | 131.7 |

**Question 1** Calcule la vitesse de chaque nageur sur 50 m en utilisant la fonction **`vitesse_kmh`**. Complète le code et le tableau.

```
# Vitesses sur 50 m
v_leo_50  = vitesse_kmh(50, ...)
v_maya_50 = vitesse_kmh(..., ...)
v_theo_50 = vitesse_kmh(..., ...)
```

| Nageur | Calcul détaillé | Vitesse sur 50 m (km/h) |
|---|---|---|
| Léo | | |
| Maya | | |
| Théo | | |

**Question 2** Écris la fonction **`temps_moyen_par_50m(temps_total_s, nb_longueurs)`** qui calcule le temps moyen pour une longueur de 50 m. Formule : temps_total / nb_longueurs

```
def temps_moyen_par_50m(temps_total_s, nb_longueurs):
    """
    ...
    """
    temps_moyen = ...           # à compléter
    return temps_moyen

# Test : Leo sur 200 m = 4 longueurs de 50 m
t_moyen_leo = temps_moyen_par_50m(..., ...)
print(f"Leo — temps moyen par 50 m : {t_moyen_leo:.2f} s")
```

| **Résultat pour Léo :** | |
|---|---|

**Question 3 (Bonus)** Qui a la vitesse moyenne la plus élevée sur 200 m ? Complète le code.

```
v_leo_200  = vitesse_kmh(200, ...)
v_maya_200 = vitesse_kmh(200, ...)
v_theo_200 = vitesse_kmh(200, ...)

vitesse_max = max(v_leo_200, v_maya_200, v_theo_200)
print(f"Vitesse maximale : {vitesse_max:.2f} km/h")
```

| **Nageur** | Léo | Maya | Théo |
|---|---|---|---|
| **Vitesse sur 200 m (km/h)** | | | |

## Bilan — Ce que tu as appris

| | |
|---|---|
| `def` | Définir une nouvelle fonction avec un nom et des paramètres |
| `return` | Renvoyer un résultat depuis une fonction |
| **Appeler** | Utiliser une fonction avec des valeurs concrètes |
| **Composer** | Appeler une fonction depuis une autre fonction |
| `"""docstring"""` | Documenter une fonction pour expliquer son rôle |