Nom :	Prénom :	Classe :

SNT — Fonction en Python : révisions & approfondissements

Objectifs :
- Savoir créer (définir) sa propre fonction en Python, à partir d'un descriptif donné.
- Savoir utiliser (appeler) une fonction dans différents contextes.

Cette session de révision est à réaliser en **binôme**.
Réfléchissez **dans le calme** à deux, mais écrivez **votre code** sur votre **copie personnelle**.
Pour gagner du temps, vous pouvez aussi vous répartir les différents exercices entre vous.

➤ Exercice 1 — Créer

Créer la fonction `somme` prenant comme paramètres `a` et `b`. Cette fonction retourne une valeur correspondant à l'addition des valeurs stockées dans `a` et `b`.

```
def somme(a, b):
    resultat = a + b
    return resultat


# Ou, alternative plus courte :

def somme(a, b):
    return a + b
```

➤ Exercice 2 — Enquêter

En gardant en mémoire le code présent dans l'exercice 1, devinez ce que les appels suivants vont afficher :

N°	Code Python	Qu'est-ce qui va s'afficher ?
1	`print( somme(100, 100) )`	200
2	`print( somme(100, -100) )`	0
3	`somme(3, 5)`	**RIEN !** (car pas de print ...)

4	`switch2 = 430`
	`concert_damso = 52`
	`print( somme(switch2, concert_damso) )`
		482

➤ **Exercice 3** — Créer

Créer la fonction `soustraction`, à partir des exemples d'appels de fonction ci-dessous :

```
print( soustraction(10, 100) ) # Affiche « 90 »
print( soustraction(50, 50) )  # Affiche « 0 »
print( soustraction(10, 5) )   # Affiche « -5 »
```

À vous de jouer :

```
def soustraction(a, b):
    resultat = b - a
    return resultat
```

➤ **Exercice 4** — Créer

L'apprentissage anticipé de la conduite (AAC) concerne les personnes âgées de 15 ans ou plus qui souhaitent obtenir le permis B.

Une auto-école vous demande de créer une fonction `conduite_aac` prenant comme paramètre `age`. Cette fonction retourne `True` si l'âge correspond à l'âge requis pour l'AAC, et `False` sinon.

```
def conduite_aac(age):
    if age >= 15:        # Si la valeur d'entrée est supérieure ou égale à 15
        return True      # Alors on quitte la fonction en retournant True
    else:                # Sinon ...
        return False     # On quitte la fonction en retournant False
```

➤ **Exercice 5** — Créer

Une jeune enseignante de mathématiques essaye de voir comment coder une fonction `est_pair` qui prend comme paramètre `a`, et qui retourne `True` si `a` est pair, et `False` sinon.

Pour information, un nombre est pair si, lorsqu'il est divisé par deux, son reste vaut zéro.

Exemples d'appel :

```
print( est_pair(8) ) # Affiche « True »
print( est_pair(9) ) # Affiche « False »
```

À vous de jouer :

```
def est_pair(a):
    if a % 2 == 0:    # Si le calcul « a ÷ 2 » donne un reste de 0
        return True   # Alors on quitte le programme en retournant True
    else:             # Sinon ...
        return False  # On quitte le programme en retournant False
```

➤ **Exercice 6** — Créer

Une entreprise d'origine suédoise spécialisée dans la conception et la vente de mobilier et objets de décoration vous a sollicité pour créer une fonction `volume_pave` permettant de calculer le volume d'un semi-remorque. Cette fonction prend les paramètres `longueur`, `largeur` et `hauteur` et retourne le volume correspondant.

Pour rappel, le volume d'un pavé se calcul en faisant longueur × largeur × hauteur.

Exemples d'appel :

```
print( volume_pave(13.6, 2.5, 4) ) # Affiche « 136.0 »
```

À vous de jouer :

```
def volume_pave(longueur, largeur, hauteur):
    resultat = longueur * largeur * hauteur
    return resultat
```

➤ **Exercice 7.1** — Introduction

Pour contrôler la vitesse d'un véhicule, il existe différents types de radars.
Dans cet exercice, on se penche sur le « radar vitesse moyenne », aussi appelé « radar tronçon ».

Le « radar vitesse moyenne » calcule la vitesse moyenne réalisée sur une portion de route.

Comment ça marche ?

- Une borne A est placée au *début* de la portion de route, et enregistre l'heure de passage.
- Une borne B est placée à *l'arrivée* de la portion de route, et enregistre l'heure de passage.

Puis un programme calcule la **durée du parcours** :

Durée du parcours = Heure d'arrivée − Heure de départ

Ce qui lui permet de trouver la **vitesse moyenne** :

Vitesse moyenne = Distance du parcours × 60 ÷ Durée du parcours

➤ **Exercice 7.2** — Enquêter

Sur une distance de parcours de 10 kms, une Lamborghini™ a franchi la borne A à 2 minutes, et la borne B à 7 minutes. À quelle vitesse moyenne roulait-elle ? **Détailler les calculs.**

Durée du parcours = 7 - 2 soit **5**.

Vitesse moyenne = 10 × 60 ÷ 5 soit **120**.

Sur cette même distance de parcours de 10 kms, une Clio™ a franchi la borne A à 2 minutes, et la borne B à 5 minutes. À quelle vitesse moyenne roulait-elle ? **Détailler les calculs.**

Durée du parcours = 5 - 2 soit **3**.

Vitesse moyenne = 10 × 60 ÷ 3 soit **200**.

➤ **Exercice 7.3** — Créer

Créer une fonction `radar_v1` qui prend comme paramètres `minutes_depart`, `minutes_arrivee` et `distance`. Cette fonction retourne la valeur correspondant à la vitesse moyenne d'un véhicule.

Exemples d'appel :

```
print( radar_v1(2, 7, 10) ) # Affiche « 120.0 » - donc ce véhicule roule à 120 km/h
print( radar_v1(2, 5, 10) ) # Affiche « 200.0 » - OOPS, ce véhicule roule à 200 km/h
```

À vous de jouer :

```
def radar_v1(minutes_depart, minutes_arrivee, distance):
    duree = minutes_arrivee - minutes_depart
    vitesse_moyenne = distance * 60 / duree
    return vitesse_moyenne
```

🌶️ Supplément "chili pepper"

Ce supplément est facultatif. À faire uniquement si vous avez terminé les exercices précédents.

➤ **Exercice 8** — Créer

Particulièrement satisfaite de votre fonction `radar_v1`, une équipe liée à la sécurité routière vous demande d'en reprendre le code, pour en créer une nouvelle fonction baptisée `radar_v2`, en intégrant les nouvelles fonctionnalités suivantes :

- Cette fonction accepte un paramètre supplémentaire `limite_vitesse`.
- Cette fonction retourne désormais un booléen : Elle retourne `True`, si la `vitesse_moyenne` est inférieure ou égale à `limite_vitesse`, sinon elle retourne `False`.

Exemples d'appel :

```
print( radar_v2(2, 7, 10, 130) ) # Affiche « True »
print( radar_v2(2, 5, 10, 130) ) # Affiche « False »
```

À vous de jouer :

```
def radar_v2(minutes_depart, minutes_arrivee, distance, limite_vitesse):
    duree = minutes_arrivee - minutes_depart
    vitesse_moyenne = distance * 60 / duree
    if vitesse_moyenne <= limite_vitesse:
        return True
    else:
        return False
```

🌶🌶🌶 Supplément "Dragon's Breath"

*Ce supplément est **hors-programme**. À faire uniquement si vous avez terminé les exercices précédents et que vous n'avez pas peur de monter sur l'échelle de Scoville[1].*

➤ Exercice 9 — Créer

Un collègue de la jeune enseignante de mathématiques s'est adressé à vous, car il souhaiterait coder une fonction `est_premier` qui prend comme paramètre `n`, et qui retourne `True` si `n` est un nombre premier, `False` sinon.

*Pour information, un nombre est considéré comme premier s'il n'est divisible que par 1 **et** lui-même.*

```
# Première proposition
def est_premier(n):
    a = 2
    p = True # Pas de diviseur trouvé, par défaut
    while p == True and a < n:
        if n % a == 0:
            p = False # Diviseur trouvé
        a = a + 1
    return p

# Deuxième proposition : Afin de réduire le nombre de lignes, on peut utiliser
# une boucle bornée, et sortir de la fonction dès qu'un diviseur est trouvé.
def est_premier(n):
    for a in range(2, n):
        if n % a == 0:
            return False # Diviseur trouvé
    return True # Pas de diviseur trouvé

# Troisième proposition : Les nombres < 2 ne sont pas des nombres premiers.
def est_premier(n):
    if n < 2:
        return False
    for a in range(2, n):
        if n % a == 0:
            return False
    return True

# Quatrième proposition : Pour réduire le nombre d'itérations, on ne teste
# que les diviseurs allant jusqu'à l'entier de √n (inclus)
def est_premier(n):
    if n < 2:
        return False
    for a in range(2, int(n ** 0.5) + 1):
        if n % a == 0:
            return False
    return True
```

1 Voir l'article « Échelle de Scoville » sur Wikipédia.