

DS bases de la programmation

Durée de l'épreuve : **01h50**

L'usage de la calculatrice n'est pas autorisé.

Le candidat répond sur feuilles doubles numérotées et garde l'énoncé.

La qualité de la rédaction, la clarté et la précision seront prises en compte.

Les traces de recherche, même incomplètes ou infructueuses, seront valorisées.

Exercice 1 (9 points)

1. Types et valeurs de variables (3 pt)

Donner le type des variables finales (**res_1**, **res_2** et **res_3**) et déterminer leur valeur après exécution des programmes ci-dessous :

```
# programme 1
a = 17
b = 4
res_1 = a / b + a // b
```

```
# programme 2
chaine = "15"
n = 3
res_2 = chaine * n
```

```
# programme 3
x = 8
y = 12
res_3 = (x > 5) and not (y < 10)
```

2. Suivi de variables (3 pt)

Écrire ce qui s'affichera en console lors de l'exécution des programmes ci-dessous :

```
# programme 1
a = 7
b = 3
c = a + b
print(a, b, c)
a = a * b
print(a, b, c)
b = c - 2
print(a, b, c)
```

```
# programme 2
x = 2
y = 0
for i in range(3):
    y = y + x
    x = x + 1
    print(i, x, y)
```

```
# programme 3
val_a = 1
val_b = 10
compteur = 1
while val_b > val_a:
    val_a = val_a + 2
    val_b = val_b - 1
    print(compteur, val_a, val_b, val_b > val_a)
    compteur = compteur + 1
```

3. Débogage (3 pt)

Identifiez les erreurs des programmes ci-dessous et les corriger :

```
# programme 1
age = input("Entrez votre âge : ")
annees_restantes = 18 - age
print(f"Plus que {annees_restantes} ans avant la majorité.")
```

```
# programme 2
note1 = 14
note2 = "16"
moyenne = (note1 + Note2) / 2
print("La moyenne est " + moyenne)
```

```
# programme 3
def calculer_ttc(prix_ht)
    prix_ttc = prix_ht * 1.20
    print(prix_ttc)

calculer_ttc(100)
print(f"Le prix final est de {prix_ttc} euros.")
```

Exercice 2 (6 points)

Une entreprise de logistique calcule le **tarif de livraison** d'un colis en fonction de son poids (un entier exprimé en grammes). Les règles de tarification de base sont les suivantes :

- Tout poids inférieur ou égal à **250 g** est facturé **4 €**.
- Tout poids supérieur à **250 g** et inférieur ou égal à **1 kg** (1000 g) est facturé **7 €**.
- Tout poids strictement supérieur à **1 kg** est facturé **12 €**.

1. Version initiale : sans option

Écrire une fonction Python *prix_simple(poids : float) → float* qui prend en paramètre le poids du colis (en grammes) et qui renvoie son tarif de base (en euros) en respectant les règles ci-dessus.

```
assert prix_simple(150) == 4
assert prix_simple(500) == 7
assert prix_simple(1200) == 12
```

2. Version enrichie : avec options Express et International

On ajoute deux options indépendantes et cumulatives à notre système de livraison :

- Si l'option **Express** est activée (paramètre *express* à *True*), un **supplément fixe de 5 €** est ajouté au tarif de base.
- Si la destination est **Internationale** (paramètre *internationale* à *True*), le prix de livraison calculé (tarif de base + éventuel supplément Express) est **augmenté de 50 %** (c'est-à-dire multiplié par 1.5).

Écrire une fonction Python *prix_options(poids, express, internationale)* qui prend en paramètres le poids (int), l'option express (bool) ainsi que l'option internationale (bool) et qui renvoie le prix total de la livraison.

```
# Colis de 150g (4 euros de base) + Pas d'Express + National
assert prix_options(150, False, False) == 4.0

# Colis de 500g (7 euros de base) + Express (+5 euros) + National
assert prix_options(500, True, False) == 12.0

# Colis de 1200g (12 euros de base) + Pas d'Express + Internationale (12 * 1.5)
assert prix_options(1200, False, True) == 18.0

# Colis de 500g (7 euros de base) + Express (+5 euros) + Internationale ((7 + 5) * 1.5)
assert prix_options(500, True, True) == 18.0
```

3. Version globale : traitement d'une liste de colis sans option

L'entreprise souhaite maintenant traiter un lot contenant plusieurs colis d'un coup, sans option express ni internationale

Écrivez une fonction *total_lot* qui prend en paramètre une liste contenant les poids de plusieurs colis à envoyer (list) et qui renvoie la somme totale à payer pour l'ensemble de ces colis.

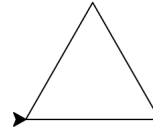
```
assert total_lot([150, 500, 1200]) == 23 # car 4 + 7 + 12
assert total_lot([100, 200]) == 8        # car 4 + 4
assert total_lot([]) == 0                 # lot vide
```

Exercice 3 (5 points)

On utilise ici le module *turtle* de Python pour dessiner des figures géométriques.

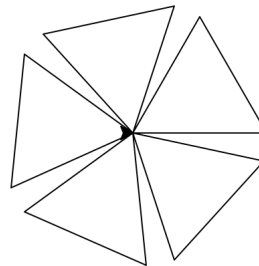
1. Écrire une fonction *triangle* qui prend en paramètre une longueur (*int*) et qui dessine un triangle équilatéral dont le côté à cette longueur :

```
triangle(100)
```



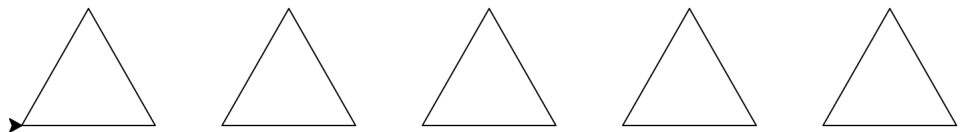
2. Écrire une fonction *rotation_triangles* qui prend en paramètres un nombre de triangles (*int*) et une longueur (*int*) et qui dessine la figure ci-dessous :

```
rotation_triangles(5, 100)
```



3. Écrire une fonction *ligne_triangles* qui prend en paramètres un nombre de triangles (*int*) et une longueur (*int*) et qui dessine la figure ci-dessous :

```
ligne_triangles(5, 100)
```



4. Question bonus (optionnel)

Écrire une fonction *escaliers_triangles* qui prend en paramètres un nombre de lignes (*int*), un nombre de triangles (*int*) et une longueur (*int*) et qui dessine la figure ci-dessous :

```
escaliers_triangles(3, 13, 20)
```

