# Devoir

# Matière : Python

# Statut : individuel

## Etudiant

**Nom : SOW**

**Prénom : MAMADOU DIAN**

**Département : informatique**

**Licence : 3**

**Université : Labé**

**Année Scolaire : 2024-2025**

CORRECTION DU DEVOIR

Exercice 1 :

```
def fibonacci_sup(n):
    """
    Trouve le premier nombre de Fibonacci supérieur à n.

    Args:
        n (int): Nombre limite.

    Returns:
        int: Premier nombre de Fibonacci supérieur à n.
    """
    # Initialisation des deux premiers nombres de Fibonacci
    F1, F2 = 1, 1

    # Boucle pour générer les nombres de Fibonacci jusqu'à dépasser n
    while F2 <= n:
        F1, F2 = F2, F1 + F2

    return F2

# Test avec les valeurs demandées
print("Premier Fibonacci > 75 :", fibonacci_sup(75))
print("Premier Fibonacci > 50 :", fibonacci_sup(50))
print("Premier Fibonacci > 100 :", fibonacci_sup(100))
```

Exercice 2 :

```
# Définition des précipitations mensuelles (en pouces)
BOS = [2.67, 1.00, 1.21, 3.09, 3.43, 4.71, 3.88, 3.08, 4.10, 2.62, 1.01, 5.93]
MER = [6.83, 3.63, 7.20, 2.68, 2.05, 2.96, 1.04, 0.00, 0.03, 6.71, 8.28, 6.85]

# Liste des mois pour affichage
mois = ["Jan", "Fév", "Mar", "Avr", "Mai", "Juin", "Juil", "Août", "Sep",
"Oct", "Nov", "Déc"]

# a) Calcul des précipitations totales et moyennes
total_BOS = sum(BOS)
total_MER = sum(MER)
moyenne_BOS = total_BOS / len(BOS)
moyenne_MER = total_MER / len(MER)

# b) Nombre de mois où les précipitations sont supérieures à la
moyenne
mois_sup_BOS = sum(1 for pluie in BOS if pluie > moyenne_BOS)
mois_sup_MER = sum(1 for pluie in MER if pluie > moyenne_MER)

# c) Mois où Boston a eu moins de pluie que Seattle
mois_inferieurs = [mois[i] for i in range(len(BOS)) if BOS[i] < MER[i]]
nombre_mois_inferieurs = len(mois_inferieurs)

# Affichage des résultats
print(f"Précipitation totale pour l'année à Boston : {total_BOS:.2f}
pouces")
```

print(f"Précipitation totale pour l'année à Seattle : {total_MER:.2f} pouces")

print(f"Précipitation moyenne mensuelle à Boston : {moyenne_BOS:.2f} pouces")

print(f"Précipitation moyenne mensuelle à Seattle : {moyenne_MER:.2f} pouces")

print(f"Nombre de mois avec précipitations supérieures à la moyenne à Boston : {mois_sup_BOS}")

print(f"Nombre de mois avec précipitations supérieures à la moyenne à Seattle : {mois_sup_MER}")

print(f"Boston a eu moins de pluie que Seattle pendant {nombre_mois_inferieurs} mois.")

print("Mois concernés :", ", ".join(mois_inferieurs))

Exercice 3 :

```
import numpy as np

from scipy.stats import norm


def correlation(X, Y):
    """
    Calcule le coefficient de corrélation de Pearson entre X et Y.
    """
    return np.corrcoef(X, Y)[0, 1]


def fisher_transform(r):
    """
```

```
    Applique la transformation Z de Fisher.
    """
    return 0.5 * np.log((1 + r) / (1 - r))

def test_correlation(X, Y, r0=0):
    """
    Effectue un test de corrélation avec hypothèse H0 : r = r0.

    Args:
        X (list): Liste des valeurs de X.
        Y (list): Liste des valeurs de Y.
        r0 (float): Valeur de corrélation nulle à tester.

    Returns:
        float: Coefficient de corrélation r
        float: Valeur p du test statistique
    """
    n = len(X)
    r = correlation(X, Y)
    Z = fisher_transform(r)
    Z0 = fisher_transform(r0)

    # Calcul de la statistique de test
    Z_prime = (Z - Z0) * np.sqrt(n - 3)

    # Calcul de la valeur p
    p_value = 2 * (1 - norm.cdf(abs(Z_prime)))
```

```
    return r, p_value

# Exemple de données
X = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Y = [8, 18, 28, 38, 48, 58, 68, 78, 88, 98]

# Test de corrélation pour r0 = 0 et r0 = 0.6
r1, p1 = test_correlation(X, Y, r0=0)
r2, p2 = test_correlation(X, Y, r0=0.6)

# Affichage des résultats
print(f"Corrélation r : {r1:.3f}, p-value pour r0=0 : {p1:.5f}")
print(f"Corrélation r : {r2:.3f}, p-value pour r0=0.6 : {p2:.5f}")
```

NB : la bibliothèque SciPy, numpy doit être installé dans votre projet pour que ce code marche .

Exercice 4 :

```
import numpy as np
import matplotlib.pyplot as plt

# Initialisation des paramètres
R = 2      # Taux de reproduction des hôtes
a = 0.05   # Zone d'attaque du parasitoïde
b = 1      # Nombre de parasitoïdes produits par hôte parasité
generations = 30  # Nombre de générations
```

```
# Conditions initiales
H = [20]  # Population des hôtes
P = [2]   # Population des parasitoïdes

# Simulation sur 30 générations
for t in range(generations):
    H_next = H[-1] * R * np.exp(-a * P[-1])  # Équation pour les hôtes
    P_next = H[-1] * (1 - np.exp(-a * P[-1])) * b  # Équation pour les parasitoïdes

    H.append(H_next)
    P.append(P_next)

# Affichage des résultats sous forme de tableau
print("Génération | Hôtes (H) | Parasitoïdes (P)")
print("-" * 35)
for t in range(generations + 1):
    print(f"{t:10} | {H[t]:10.2f} | {P[t]:10.2f}")

# Tracé des courbes
plt.figure(figsize=(8, 5))
plt.plot(range(generations + 1), H, label="Hôtes (Insectes)", marker='o', linestyle='-')
plt.plot(range(generations + 1), P, label="Parasitoïdes", marker='s', linestyle='--')
plt.xlabel("Générations")
plt.ylabel("Population")
```

```
plt.title("Évolution des populations (Modèle Nicholson-Bailey)")
plt.legend()
plt.grid()
plt.show()
```

NB : la bibliothèque  Matplotlib, numpy doit être installé dans votre projet pour que ce code marche .