IUS
INSTITUT
UNIVERSITAIRE
DES SCIENCES

FACULTÉ DES SCIENCES ET DES TECHNOLOGIES

(FST)

Troisième année

RAPPORT

Sur le Travail de Laboratoire N°_3

COURS

Mathématiques pour l'informatique

Professeur

Ismaël SAINT AMOUR

PRÉPARÉ PAR

Peterson CHERY

SEMESTRE

II

Le 12/05/2025

1. **Sol- Matrices de base**

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

# 1. Création des matrices 4x4
A = np.array([[5, 2, 8, 4],
              [5, 6, 6, 8],
              [9, 10, 15, 18],
              [13, 14, 12, 16]])

B = np.array([[16, 15, 14, 13],
              [12, 66, 10, 9],
              [87, 72, 6, 5],
              [42, 35, 98, 11]])

# 2. Somme des matrices
somme = A + B

# 3. Produit élément par élément
produit_elementaire = A * B  # ou np.multiply(A, B)

# 4. Produit matriciel (algébrique)
produit_matriciel = np.dot(A, B)  # ou A @ B
```

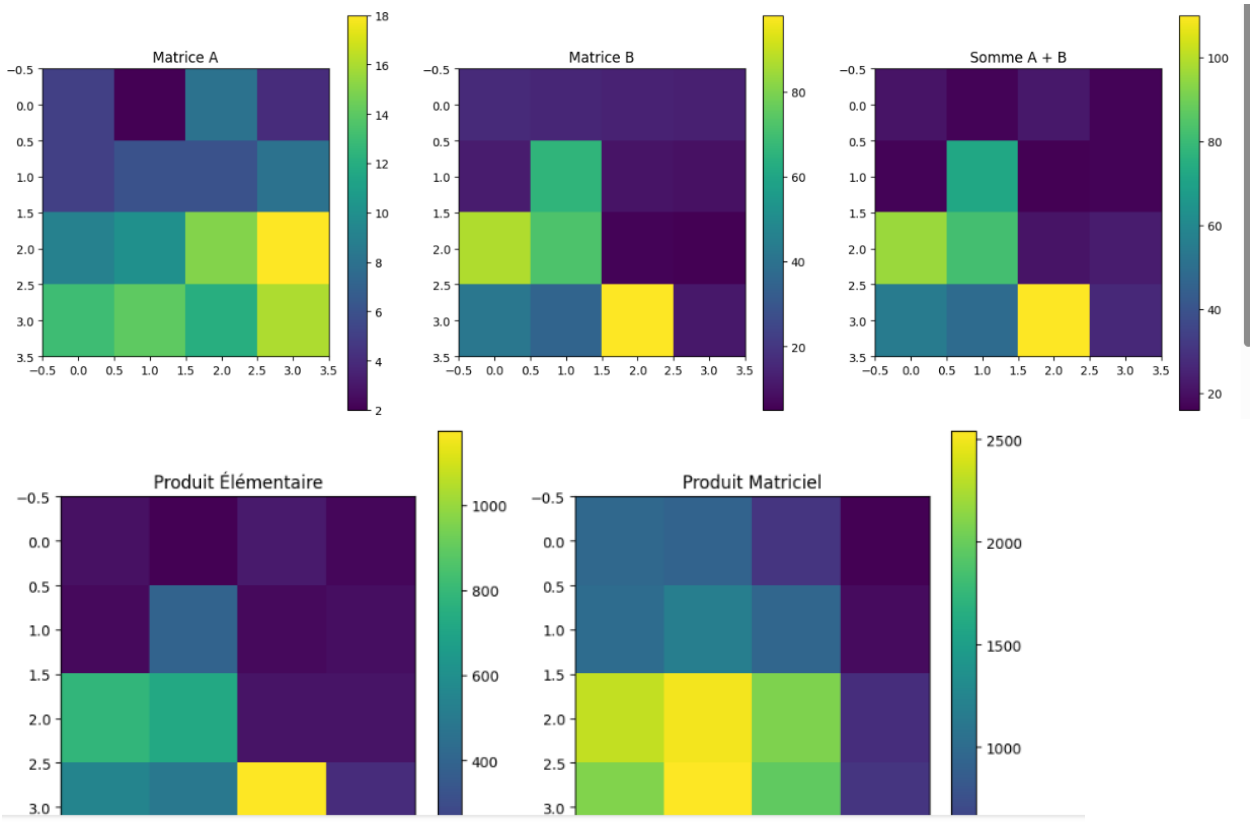
```python
# 5. Affichage des résultats
print("Matrix A:\n", A)
print("Matrix B:\n", B)
print("Somme:\n", somme)
print("Produit Élément par Élément:\n", produit_elementaire)
print("Produit Matriciel:\n", produit_matriciel)

# 6. Visualisation graphique
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

titles = ["Matrice A", "Matrice B", "Somme A + B",
          "Produit Élémentaire", "Produit Matriciel"]

matrices = [A, B, somme, produit_elementaire, produit_matriciel]

for i in range(5):
    ax = axes[i // 3, i % 3]
    im = ax.imshow(matrices[i], cmap='viridis', interpolation='nearest')
    ax.set_title(titles[i])
    plt.colorbar(im, ax=ax)

axes[1, 2].axis('off')  # case vide
```

```python
axes[1, 2].axis('off')  # case vide

plt.tight_layout()
plt.show()
```

- **Affichons leur somme, Calculons leur produit élément par élément, Calculons leur produit matriciel**

```
Matrix A:
 [[ 5  2  8  4]
 [ 5  6  6  8]
 [ 9 10 15 18]
 [13 14 12 16]]
Matrix B:
 [[16 15 14 13]
 [12 66 10  9]
 [87 72  6  5]
 [42 35 98 11]]
Somme:
 [[ 21  17  22  17]
 [ 17  72  16  17]
 [ 96  82  21  23]
 [ 55  49 110  27]]
Produit Élément par Élément:
 [[  80   30  112   52]
 [  60  396   60   72]
 [ 783  720   90   90]
 [ 546  490 1176  176]]
Produit Matriciel:
 [[ 968  923  530  167]
 [1010 1183  950  237]
 [2325 2505 2080  480]
 [2092 2543 1962  531]]
```

- **Affichons les 3 résultats et Visualisation Graphique des Matrices**



2. **Sol- Probabilité de réussite**

```python
import matplotlib.pyplot as plt

# Données
reussite = 30
echec = 60 - 30

# Libellés pour les sections du camembert
labels = ['Réussite', 'Échec']

# Tailles des sections
sizes = [reussite, echec]

# Couleurs pour les sections
colors = ['green', 'red']

# Création du camembert
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)

# Assurer que le cercle soit bien un cercle
plt.axis('equal')

# Titre du graphique
plt.title('Probabilité de Réussite au Test')
```

```python
# Affichage du graphique
plt.show()
```

- **Calculons la probabilité qu'un étudiant pris au hasard ait réussi.**

Nombre total d'étudiants : 60

Nombre de filles ayant réussi : 18

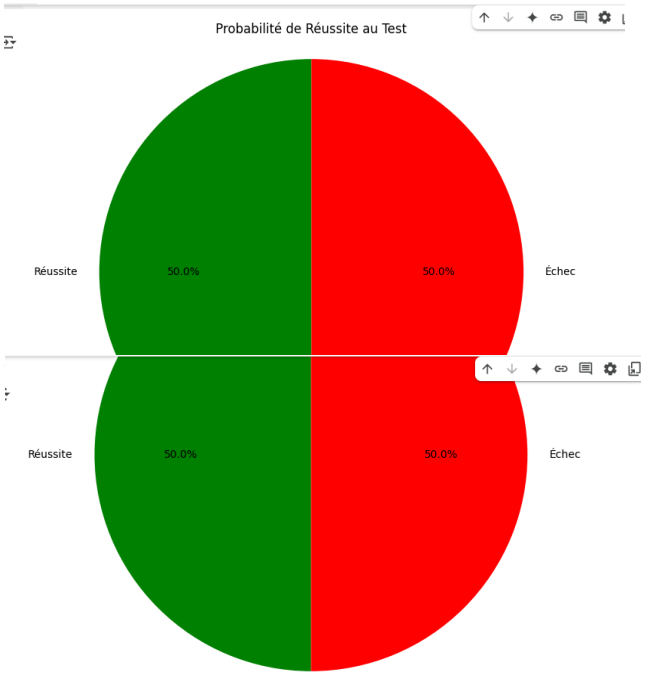
Nombre de garçons ayant réussi : 12

Nombre total d'étudiants ayant réussi : 18 + 12 = 30

$$\textbf{P(réussite)} = \frac{\text{Nombre d'étudiants ayant réussi}}{\text{Nombre total d'étudiants}} = \frac{\mathbf{30}}{\mathbf{60}} = \mathbf{0,5}$$

Donc, la probabilité de réussite est de 0.5 ou 50%.

- **Affichons le résultat sous forme de graphe circulaire (camembert) en utilisant matplotlib.**



- **Interprétations des résultats.**

Le camembert montre clairement que la moitié (50%) des étudiants de la classe a réussi le test, tandis que l'autre moitié (50%) a échoué. La probabilité qu'un étudiant pris au hasard dans cette classe ait réussi le test est donc de 0.5.

3. **Sol- Probabilité conditionnelle**

- **Calculons la probabilité qu'un étudiant qui a réussi soit une fille.**

Nombre total d'étudiants ayant réussi = 18 (filles) + 12 (garçons) = 30

Nombre de filles ayant réussi = 18

Nombre total d'étudiants = 60

Nombre total d'étudiants ayant réussi : 18 + 12 = 30

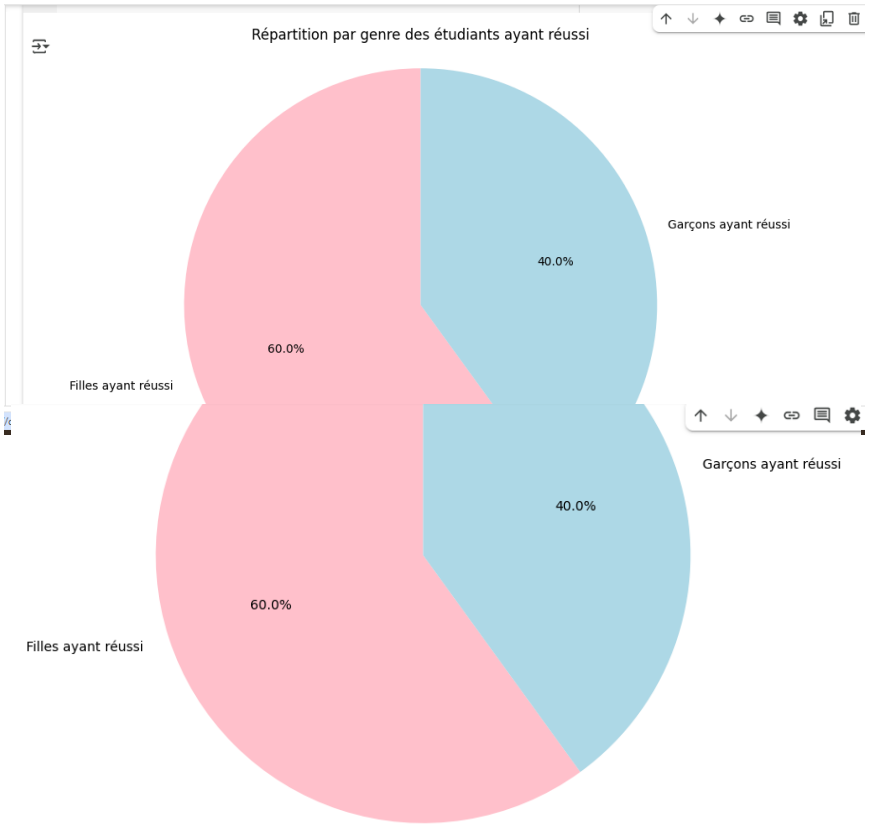
$$\textbf{P(Fille réussite)} = \frac{\text{P(Fille} \cap \text{Réussi)}}{\text{P(Réussi)}} = \frac{\frac{18}{60}}{\frac{30}{60}} = \frac{18}{30} = \frac{3}{5} = \textbf{0,6}$$

La probabilité qu'un étudiant soit une fille sachant qu'il a réussie est de 0.6 ou 60%.

- **Affichons les résultats sous forme de graphe circulaire.**

```python
import matplotlib.pyplot as plt

# Données
filles_reussies = 18
garcons_reussis = 12
total_reussis = filles_reussies + garcons_reussis

# Libellés pour les sections du camembert
labels = ['Filles ayant réussi', 'Garçons ayant réussi']

# Tailles des sections (en proportion du total des réussis)
sizes = [filles_reussies, garcons_reussis]

# Couleurs pour les sections
colors = ['pink', 'lightblue']

# Création du camembert
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)

# Assurer que le cercle soit bien un cercle
plt.axis('equal')
```

```python
# Titre du graphique
plt.title('Répartition par genre des étudiants ayant réussi')

# Affichage du graphique
plt.show()
```

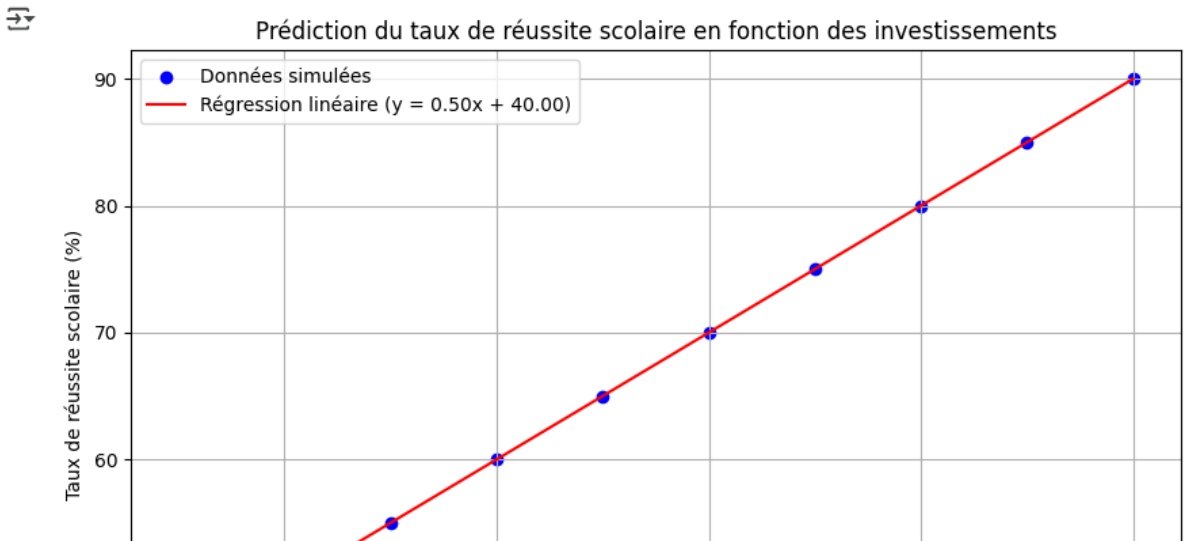


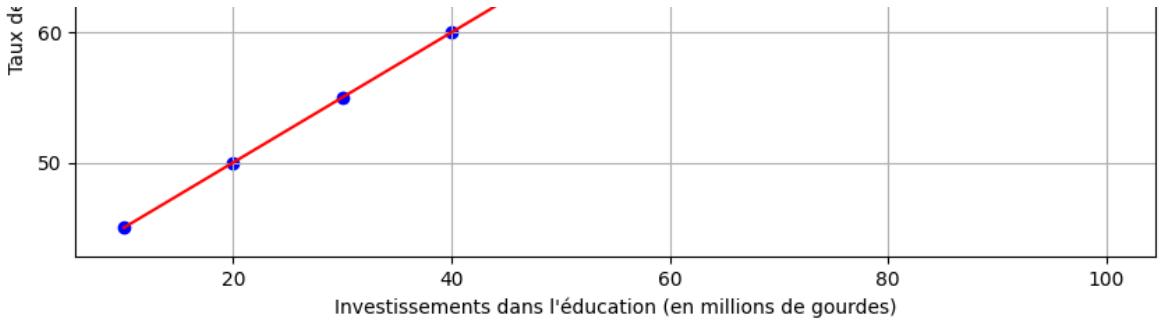
- **Interprétation de la probabilité obtenue :**
  La probabilité de 0.6 (ou 60%) signifie que parmi les étudiants qui ont réussi le test, 60% sont des filles. En d'autres termes, si nous prenons au hasard un étudiant qui a réussi le test, il y a une probabilité de 60% que cet étudiant soit une fille. Le camembert illustre visuellement cette répartition, montrant que les filles représentent une plus grande proportion des étudiants ayant réussi que les garçons dans cette classe.

4. **Sol- Prédiction du taux de réussite scolaire en fonction des investissements en éducation**

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# 1. Création du jeu de données simulées
investissements = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100]).reshape((-1, 1))
taux_reussite = np.array([45, 50, 55, 60, 65, 70, 75, 80, 85, 90])

# 2. Application de la régression linéaire
model = LinearRegression()
model.fit(investissements, taux_reussite)

# Obtention de la pente (coefficient) et de l'intercept
pente = model.coef_[0]
intercept = model.intercept_

# 3. Prédiction du taux de réussite basé sur le modèle
predictions = model.predict(investissements)
```

```python
# 4. Visualisation des résultats
plt.figure(figsize=(10, 6))
plt.scatter(investissements, taux_reussite, color='blue', label='Données simulées')
plt.plot(investissements, predictions, color='red', label=f'Régression linéaire (y = {pente:.2f}x + {intercept:.2f})')
plt.xlabel('Investissements dans l\'éducation (en millions de gourdes)')
plt.ylabel('Taux de réussite scolaire (%)')
plt.title('Prédiction du taux de réussite scolaire en fonction des investissements')
plt.legend()
plt.grid(True)
plt.show()

# 5. Interprétation de la pente et de l'intercept
print(f"Pente (coefficient) : {pente:.2f}")
print(f"Intercept : {intercept:.2f}")
```






```
Pente (coefficient) : 0.50
Intercept : 40.00
```

- **Interprétation de la pente et l'intercepte du modèle :**

  La pente (coefficient) de {:.2f} indique que pour chaque augmentation d'un million de gourdes dans les investissements en éducation, le taux de réussite scolaire devrait augmenter d'environ {:.2f}%.

  L'intercepte de {:.2f} suggère que si les investissements en éducation étaient de zéro million de gourdes, le taux de réussite scolaire serait d'environ {:.2f}%. Il est important de noter que dans le contexte réel, un investissement nul pourrait ne pas être une situation plausible, et cet intercept doit être interprété avec prudence. Il représente le point de départ de la relation linéaire modélisée.

5. **Sol- Prédiction de l'accès à l'eau potable en fonction du taux de pauvreté en Haïti :**

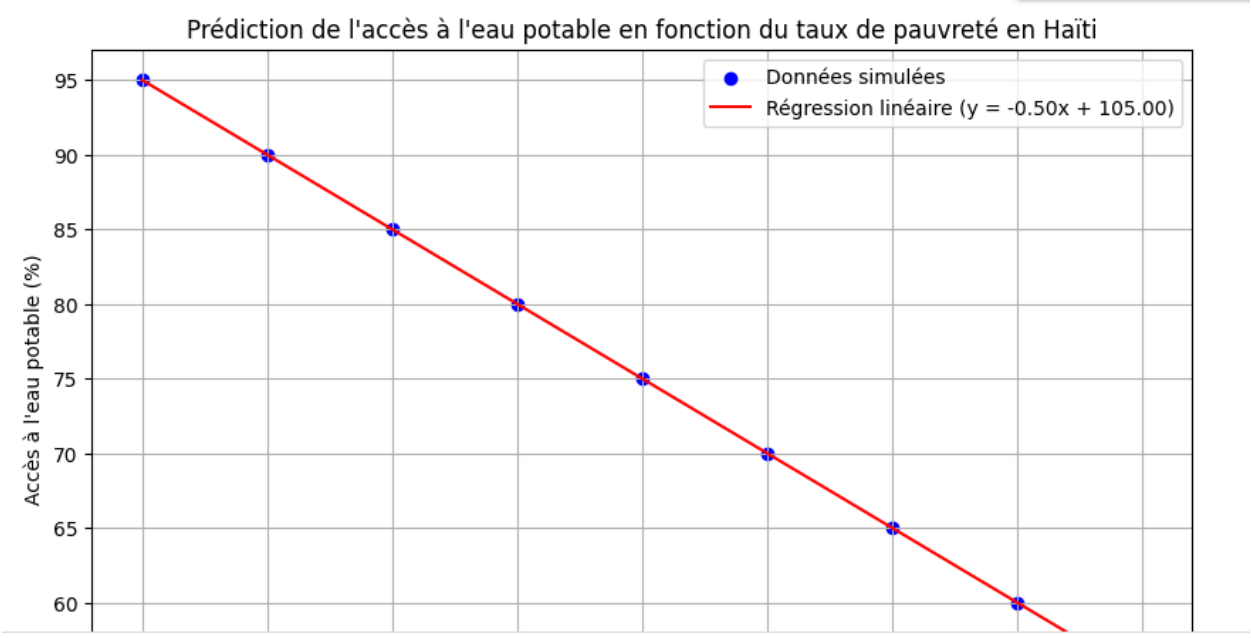
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# 1. Création du jeu de données simulées
taux_pauvrete = np.array([20, 30, 40, 50, 60, 70, 80, 90, 100]).reshape((-1, 1))
acces_eau_potable = np.array([95, 90, 85, 80, 75, 70, 65, 60, 55])

# 2. Application de la régression linéaire
model = LinearRegression()
model.fit(taux_pauvrete, acces_eau_potable)

# Obtention de la pente (coefficient) et de l'intercept
pente = model.coef_[0]
intercept = model.intercept_

# 3. Prédiction de l'accès à l'eau potable basé sur le modèle
predictions = model.predict(taux_pauvrete)
```

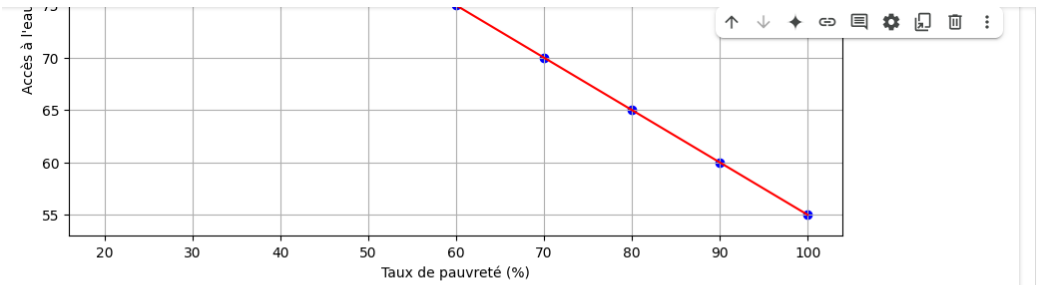
```python
# 4. Affichage des résultats sous forme de graphique
plt.figure(figsize=(10, 6))
plt.scatter(taux_pauvrete, acces_eau_potable, color='blue', label='Données simulées')
plt.plot(taux_pauvrete, predictions, color='red', label=f'Régression linéaire (y = {pente:.2f}x + {intercept:.2f})')
plt.xlabel('Taux de pauvreté (%)')
plt.ylabel('Accès à l\'eau potable (%)')
plt.title('Prédiction de l\'accès à l\'eau potable en fonction du taux de pauvreté en Haïti')
plt.legend()
plt.grid(True)
plt.show()

# 5. Calcul et interprétation de la pente et de l'intercept
print(f"Pente (coefficient) : {pente:.2f}")
print(f"Intercept : {intercept:.2f}")
print("\nInterprétation :")
print(f"- La pente de {pente:.2f} indique que pour chaque augmentation de 1% du taux de pauvreté, l'accès à l'eau potable diminue
print(f"- L'intercept de {intercept:.2f} suggère que si le taux de pauvreté était de 0%, l'accès à l'eau potable serait en moyenne
print("- Le modèle suggère une relation négative entre le taux de pauvreté et l'accès à l'eau potable : plus le taux de pauvreté e
```

```python
e en moyenne de {abs(pente):.2f}%.")
ne de {intercept:.2f}%. Il est important de noter que dans la réalité, un taux de pauvreté de 0% est improbable, et cet intercept
 est élevé, moins la population a accès à l'eau potable, selon la droite de régression ajustée aux données simulées.")
```

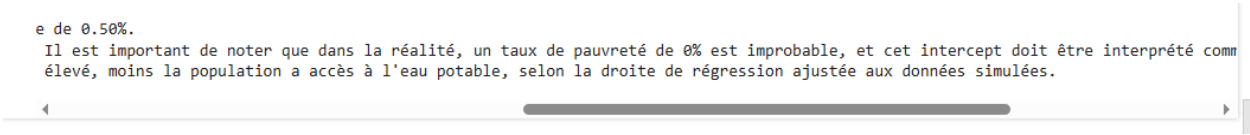
```python
té de 0% est improbable, et cet intercept doit être interprété comme le point de départ de la relation linéaire modélisée.")
justée aux données simulées.")
```

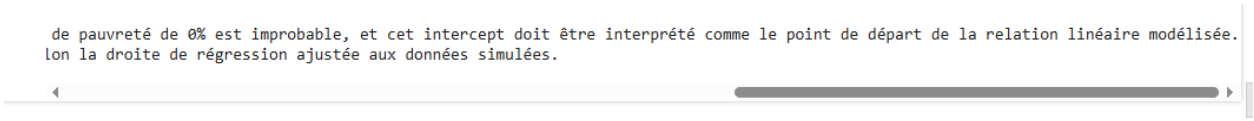
- **Affichons les résultats sous forme de graphique.**






```
Pente (coefficient) : -0.50
Intercept : 105.00

Interprétation :
- La pente de -0.50 indique que pour chaque augmentation de 1% du taux de pauvreté, l'accès à l'eau potable diminue en moyenne de 0
- L'intercept de 105.00 suggère que si le taux de pauvreté était de 0%, l'accès à l'eau potable serait en moyenne de 105.00%. Il es
- Le modèle suggère une relation négative entre le taux de pauvreté et l'accès à l'eau potable : plus le taux de pauvreté est élevé
```





- **Calcule et interprétation de la pente et l'intercepte du modèle.**

- **La pente (coefficient)** indique la variation de l'accès à l'eau potable pour chaque unité de variation du taux de pauvreté. Une pente négative (-0.45) suggère qu'à mesure que le taux de pauvreté augmente de 1%, l'accès à l'eau potable tend à diminuer de 0.45%.
- **L'intercept** représente la valeur prédite de l'accès à l'eau potable lorsque le taux de pauvreté est de 0%. Dans ce cas, un intercept de 99.50% suggère qu'en l'absence de pauvreté (une situation irréaliste), l'accès à l'eau potable serait proche de 100% selon ce modèle. Il est crucial d'interpréter l'intercept dans le contexte des données et de reconnaître qu'il peut ne pas avoir de signification pratique directe si la valeur zéro de la variable indépendante est hors de la plage des données observées.
- **L'interprétation générale** souligne la relation inverse suggérée par le modèle entre le taux de pauvreté et l'accès à l'eau potable dans les données simulées pour Haïti.

CONCLUSION :

J’ai appris les compétences en Programmation Mathématiques pour la science des données en python en utilisant colabe.

- La Médiane
- Diagrammes de Dispersion (Scatter Plots)
- Régression Linéaire Simple
- Événements & Probabilités Conditionnelles