

PROJET 4 DATA ANALYST

Réalisez une étude de santé publique avec R ou Python

Objectif : étude sur l'alimentation et la sous nutrition dans le monde

Auteur et commanditaire : FAO (Food and Agriculture Organization of the United Nations)

Partie historique : période 2013-2017

Objectifs détaillés :

- Faire un bilan de l'état de sous-nutrition
- Analyse du niveau de disponibilité en ressources alimentaires mondiales
- Comparaison des situations par pays pour : la sous-nutrition, l'aide alimentaire, la disponibilité alimentaire
- Etude complémentaire sur le Manioc en Thaïlande

NOTE :

Toutes les parties indiquées PM (signifiant "Pour Moi") sont des compléments non indispensables qui permettent d'apporter des détails sur les différentes étapes, des tests, ou des variantes de possibilité. Ce projet est de type professionnel, mais est aussi un outil pédagogique, ce qui est la raison d'avoir laissé certaines de ces remarques visibles, ou masquées (démasquable sur le fichier .ipynb).

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

Importation des librairies utilisées

```
#Importation de la librairie Numpy
import numpy as np

#Importation de la librairie Pandas
import pandas as pd

#Importation du module pyplot de la librairie matplotlib (graphiques)
import matplotlib.pyplot as plt

#Importation de la librairie Seaborn (graphiques)
import seaborn as sns
```

1.2 - Chargement des fichiers Excel

Importation de fichier placés dans le même dossier que le Notebook (pour simplifier l'écriture du chemin d'accès aux fichiers .csv)

```
#Importation du fichier population.csv
population = pd.read_csv('population.csv')

#Importation du fichier dispo_alimentaire.csv
dispo_alimentaire = pd.read_csv('dispo_alimentaire.csv')

#Importation du fichier aide_alimentaire.csv
aide_alimentaire = pd.read_csv('aide_alimentaire.csv')

#Importation du fichier sous_nutrition.csv
sous_nutrition = pd.read_csv('sous_nutrition.csv')
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

> PM - Exploration des données

[] ↴ 35 cellules masquées

FIN PM - Exploration des données

✓ 1.Afficher les dimensions du dataset

```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(population.shape[0]))
print("Le tableau comporte {} colonne(s)".format(population.shape[1]))
```

→ Le tableau comporte 1416 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)

✓ 2.Consulter :

-le nombre de colonnes

-La nature des données dans chacune des colonnes

-Le nombre de valeurs présentes dans chacune des colonnes

LECTURE DIRECTE

```
population.info()
```

→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1416 entries, 0 to 1415
Data columns (total 3 columns):
 # Column Non-Null Count Dtype
 --- -----
 0 Zone 1416 non-null object
 1 Année 1416 non-null int64
 2 Valeur 1416 non-null float64
 dtypes: float64(1), int64(1), object(1)
 memory usage: 33.3+ KB

```
def Description_Fichier(FICHIER):
```

```
#Consulter le nombre de colonnes
print("Le tableau comporte ",FICHIER.shape[1]," colonne(s) : ")

for elt in FICHIER.columns :
    print(elt)

print("\n")

#le type de données et Le nombre de valeurs présentes dans chacune des colonnes
for elt in FICHIER.columns :
    print("La colonne ",elt," (est de type ",FICHIER[elt].dtypes," ) et contient :")
    print("\t", len(FICHIER[elt])," lignes")
    print("\t", FICHIER[elt].notna().sum()," valeur(s) non-vide(s)")
    print("\t", FICHIER[elt].isna().sum()," valeur(s) vide(s)")
    print ("", len(FICHIER[elt].value_counts()), " valeur(s) distincte(s).")
    print("\n")
```

```
Description_Fichier(population)
```

→ Le tableau comporte 3 colonne(s) :
Zone
Année
Valeur

La colonne Zone (est de type object) et contient :
1416 lignes
1416 valeur(s) non-vide(s)
0 valeur(s) vide(s)
236 valeur(s) distincte(s).

La colonne Année (est de type int64) et contient :
1416 lignes
1416 valeur(s) non-vide(s)
0 valeur(s) vide(s)

```
6 valeur(s) distincte(s).
```

```
La colonne Valeur (est de type float64 ) et contient :  
1416 lignes  
1416 valeur(s) non-vide(s)  
0 valeur(s) vide(s)  
1413 valeur(s) distincte(s).
```

DATAFRAME EXPLORATOIRE

```
def Description_Fichier_Tableau(FICHIER):  
    #création d'un DataFrame qui va contenir les colonnes du Fichier en Index, et la description des lignes dans les colonnes  
    Tab_Descriptif = pd.DataFrame(index = FICHIER.columns, columns=['Type','Nb lignes','Valeurs non-vides','Valeurs vides','Valeurs dist'  
  
    #remplissage des lignes  
    for elt in FICHIER.columns :  
        Tab_Descriptif.loc[elt,:] = [FICHIER[elt].dtypes,len(FICHIER[elt]) , FICHIER[elt].notna().sum() , FICHIER[elt].isna().sum(), len  
  
    #print(Tab_Descriptif)  
    return Tab_Descriptif
```

```
Description_Fichier_Tableau(population)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	object	1416	1416	0	236
Année	int64	1416	1416	0	6
Valeur	float64	1416	1416	0	1413

```
population['Année'].unique()
```

```
array([2013, 2014, 2015, 2016, 2017, 2018], dtype=int64)
```

3.Affichage des 5 premières lignes de la table

```
#Affichage les 5 premières lignes de la table  
population.head()
```

	Zone	Année	Valeur
0	Afghanistan	2013	32269.589
1	Afghanistan	2014	33370.794
2	Afghanistan	2015	34413.603
3	Afghanistan	2016	35383.032
4	Afghanistan	2017	36296.113

4.HARMONISATION DES UNITES :

Selon le lexique des données la Population est dans la colonne Valeur et exprimée en milliers d'habitants. Nous décidons donc de multiplier la colonne Population par 1_000 pour obtenir le nombre d'habitants.

```
#Nous allons harmoniser les unités. Pour cela, nous avons décidé de multiplier la population par 1000  
#Multiplication de la colonne valeur par 1000  
population['Valeur']*=1000
```

5.changement du nom de la colonne Valeur par Population

Il est plus parlant de renommer la colonne Valeur en Population

```
#changement du nom de la colonne Valeur par Population  
population.rename(columns={'Valeur':'Population'},inplace=True)
```

6.Affichage des 5 premières lignes de la table pour voir les modifications

```
#Affichage les 5 premières lignes de la table pour voir les modifications  
population.head()
```

	Zone	Année	Population
0	Afghanistan	2013	32269589.0
1	Afghanistan	2014	33370794.0
2	Afghanistan	2015	34413603.0
3	Afghanistan	2016	35383032.0
4	Afghanistan	2017	36296113.0

▼ 7.PM : AMELIORATION DU TYPE DES DONNEES :

```
population.dtypes
```

Zone	object
Année	int64
Population	float64

CONVERTIR TOUTES LES COLONNES vers le meilleur format possible : .convert_dtypes()

```
population=population.convert_dtypes()
```

```
population.dtypes
```

Zone	string[python]
Année	Int64
Population	Float64

FORCAGE POUR Passer la colonne Population en INT64

CONVERTIR UNE COLONNES vers un format bien précis : .astype({'colx':'typex','coly':'typey'})

```
population = population.astype({'Population': 'Int64'})
```

```
population.dtypes
```

Zone	string[python]
Année	Int64
Population	Int64

```
population.head()
```

	Zone	Année	Population
0	Afghanistan	2013	32269589
1	Afghanistan	2014	33370794
2	Afghanistan	2015	34413603
3	Afghanistan	2016	35383032
4	Afghanistan	2017	36296113

```
Description_Fichier_Tableau(population)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	1416	1416	0	236
Année	Int64	1416	1416	0	6
Population	Int64	1416	1416	0	1411

2.2 - Analyse exploratoire du fichier disponibilité alimentaire

> PM - Exploration des données

[] ↴ 2 cellules masquées

FIN PM - Exploration des données

✓ 1.Afficher les dimensions du dataset

```
#Afficher les dimensions du dataset
print("Le tableau comporte {} lignes".format(dispo_alimentaire.shape[0]))
print("Le tableau comporte {} colonne(s)".format(dispo_alimentaire.shape[1]))
```

→ Le tableau comporte 15605 lignes
Le tableau comporte 18 colonne(s)

✓ 2. Consulter le nombre de colonnes

```
dispo_alimentaire.info()
```

→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 15605 entries, 0 to 15604
Data columns (total 18 columns):
 # Column Non-Null Count Dtype

 0 Zone 15605 non-null object
 1 Produit 15605 non-null object
 2 Origine 15605 non-null object
 3 Aliments pour animaux 2720 non-null float64
 4 Autres Utilisations 5496 non-null float64
 5 Disponibilité alimentaire (Kcal/personne/jour) 14241 non-null float64
 6 Disponibilité alimentaire en quantité (kg/personne/an) 14015 non-null float64
 7 Disponibilité de matière grasse en quantité (g/personne/jour) 11794 non-null float64
 8 Disponibilité de protéines en quantité (g/personne/jour) 11561 non-null float64
 9 Disponibilité intérieure 15382 non-null float64
 10 Exportations - Quantité 12226 non-null float64
 11 Importations - Quantité 14852 non-null float64
 12 Nourriture 14015 non-null float64
 13 Pertes 4278 non-null float64
 14 Production 9180 non-null float64
 15 Semences 2091 non-null float64
 16 Traitement 2292 non-null float64
 17 Variation de stock 6776 non-null float64
dtypes: float64(15), object(3)
memory usage: 2.1+ MB

```
Description_Fichier_Tableau(dispo_alimentaire)
```

		Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
	Zone	object	15605	15605	0	174
	Produit	object	15605	15605	0	98
	Origine	object	15605	15605	0	2
	Aliments pour animaux	float64	15605	2720	12885	538
	Autres Utilisations	float64	15605	5496	10109	407
	Disponibilité alimentaire (Kcal/personne/jour)	float64	15605	14241	1364	561
	Disponibilité alimentaire en quantité (kg/personne/an)	float64	15605	14015	1590	2919
	Disponibilité de matière grasse en quantité (g/personne/jour)	float64	15605	11794	3811	1127
	Disponibilité de protéines en quantité (g/personne/jour)	float64	15605	11561	4044	1035
	Disponibilité intérieure	float64	15605	15382	223	1795
	Exportations - Quantité	float64	15605	12226	3379	762
	Importations - Quantité	float64	15605	14852	753	871
	Nourriture	float64	15605	14015	1590	1396
	Pertes	float64	15605	4278	11327	434
	Production	float64	15605	9180	6425	1657
	Semences	float64	15605	2091	13514	250
	Traitemet	float64	15605	2292	13313	500
	Variation de stock	float64	15605	6776	8829	400

3. Affichage des 5 premières lignes

```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité protéines quant (g/personne/jo
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.20	C
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.01	C
2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.01	C
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.00	NaN	N
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.70	0.02	C

4.replacement des NaN dans le dataset par des 0

```
#remplacement des NaN dans le dataset par des 0
dispo_alimentaire.fillna(0,inplace=True)
```

```
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité protéines quant
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.00
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.00
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.00
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.00

✓ 5. Multiplication de toutes les lignes contenant des milliers de tonnes en Kg

CONVERSION DES UNITES :

Il existe un certains nombres de colonnes qui sont en milliers de tonnes.

Afin d'homogénéiser les unités nous les passons en kilo (*1_000_000)

dispo_alimentaire.columns

```
Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux',
       'Autres Utilisations', 'Disponibilité alimentaire (Kcal/personne/jour)',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)',
       'Disponibilité de protéines en quantité (g/personne/jour)',
       'Disponibilité intérieure', 'Exportations - Quantité',
       'Importations - Quantité', 'Nourriture', 'Pertes', 'Production',
       'Semences', 'Traitement', 'Variation de stock'],
      dtype='object')
```

```
liste_col_en_mill_tonnes=['Aliments pour animaux',
    'Autres Utilisations',
    'Disponibilité intérieure', 'Exportations - Quantité',
    'Importations - Quantité', 'Nourriture', 'Pertes', 'Production',
    'Semences', 'Traitement', 'Variation de stock']
```

Pour passer de MILLIERS de tonnes à kg, il faut multiplier par 1_000_000

```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
for elt in liste_col_en_mill_tonnes :
    dispo_alimentaire[elt]*=1 000 000
```

MODIFICATION DU NOM DES COLONNES :

Les unités utilisées étant confuses, nous allons également expliciter les unités en les ajoutant dans les colonnes.

Le nom des colonnes que l'on souhaiterait avoir (liste utile plus bas)

```
New_liste_col_en_kg=[]
for elt in liste_col_en_mill_tonnes:
    New_liste_col_en_kg.append(elt+"(kg)")

print(New_liste_col_en_kg)

→ ['Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Disponibilité intérieure(kg)', 'Exportations - Quantité(kg)', 'Importation(kg)']

for elt in liste_col_en_mill_tonnes:
    dispo_alimentaire.rename(columns={elt:elt+"(kg)"},inplace=True)

dispo_alimentaire.columns

→ Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux(kg)',
       'Autres Utilisations(kg)',
       'Disponibilité alimentaire (Kcal/personne/jour)',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)'].
```

```
'Disponibilité de protéines en quantité (g/personne/jour)',
'Disponibilité intérieure(kg)', 'Exportations - Quantité(kg)',
'Importations - Quantité(kg)', 'Nourriture(kg)', 'Pertes(kg)',
'Production(kg)', 'Semences(kg)', 'Traitement(kg)',
'Variation de stock(kg)'],
dtype='object')
```

6.Affichage des 5 premières lignes de la table

```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	

7.PM : AMELIORATION DU TYPE DES DONNEES :

```
dispo_alimentaire.dtypes
```

Zone	object
Produit	object
Origine	object
Aliments pour animaux(kg)	float64
Autres Utilisations(kg)	float64
Disponibilité alimentaire (Kcal/personne/jour)	float64
Disponibilité alimentaire en quantité (kg/personne/an)	float64
Disponibilité de matière grasse en quantité (g/personne/jour)	float64
Disponibilité de protéines en quantité (g/personne/jour)	float64
Disponibilité intérieure(kg)	float64
Exportations - Quantité(kg)	float64
Importations - Quantité(kg)	float64
Nourriture(kg)	float64
Pertes(kg)	float64
Production(kg)	float64
Semences(kg)	float64
Traitement(kg)	float64
Variation de stock(kg)	float64
dtype: object	

```
dispo_alimentaire=dispo_alimentaire.convert_dtypes()
```

```
→ D:\Programmes2\Anaconda3\Lib\site-packages\pandas\core\dtypes\cast.py:1057: RuntimeWarning: invalid value encountered in cast
  if (arr.astype(int) == arr).all():
D:\Programmes2\Anaconda3\Lib\site-packages\pandas\core\dtypes\cast.py:1081: RuntimeWarning: invalid value encountered in cast
  if (arr.astype(int) == arr).all():
```

```
dispo_alimentaire.dtypes
```

Zone	string[python]
Produit	string[python]
Origine	string[python]
Aliments pour animaux(kg)	Float64
Autres Utilisations(kg)	Float64
Disponibilité alimentaire (Kcal/personne/jour)	Int64
Disponibilité alimentaire en quantité (kg/personne/an)	Float64
Disponibilité de matière grasse en quantité (g/personne/jour)	Float64
Disponibilité de protéines en quantité (g/personne/jour)	Float64
Disponibilité intérieure(kg)	Float64
Exportations - Quantité(kg)	Float64
Importations - Quantité(kg)	Float64
Nourriture(kg)	Float64
Pertes(kg)	Float64
Production(kg)	Float64
Semences(kg)	Float64
Traitement(kg)	Float64

```
Variation de stock(kg)                                     Float64
dtype: object
```

FORCAGE POUR Passer les colonnes en kg en INT64

```
for elt in New_liste_col_en_kg :
    dispo_alimentaire = dispo_alimentaire.astype({elt : 'Int64'})
```

```
dispo_alimentaire.dtypes
```

Zone	string[python]
Produit	string[python]
Origine	string[python]
Aliments pour animaux(kg)	Int64
Autres Utilisations(kg)	Int64
Disponibilité alimentaire (Kcal/personne/jour)	Int64
Disponibilité alimentaire en quantité (kg/personne/an)	Float64
Disponibilité de matière grasse en quantité (g/personne/jour)	Float64
Disponibilité de protéines en quantité (g/personne/jour)	Float64
Disponibilité intérieure(kg)	Int64
Exportations - Quantité(kg)	Int64
Importations - Quantité(kg)	Int64
Nourriture(kg)	Int64
Pertes(kg)	Int64
Production(kg)	Int64
Semences(kg)	Int64
Traitemen(kg)	Int64
Variation de stock(kg)	Int64
dtype: object	

```
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Utilisations(kg)	Autres	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Disponibilité intérieure(kg)
0	Afghanistan	Abats Comestible	animale	0	0	0	5	1.72	0.0	0.2	0
1	Afghanistan	Agrumes, Autres	vegetale	0	0	0	1	1.29	0.0	0.01	0
2	Afghanistan	Aliments pour enfants	vegetale	0	0	0	1	0.06	0.0	0.01	0
3	Afghanistan	Ananas	vegetale	0	0	0	0	0.0	0.0	0.0	0
4	Afghanistan	Bananes	vegetale	0	0	0	4	2.7	0.0	0.02	0

```
Description_Fichier_Tableau(dispo_alimentaire)
```

CSV

		Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
	Zone	string[python]	15605	15605	0	174
	Produit	string[python]	15605	15605	0	98
	Origine	string[python]	15605	15605	0	2
	Aliments pour animaux(kg)	Int64	15605	15605	0	537
	Autres Utilisations(kg)	Int64	15605	15605	0	406
	Disponibilité alimentaire (Kcal/personne/jour)	Int64	15605	15605	0	560
	Disponibilité alimentaire en quantité (kg/personne/an)	Float64	15605	15605	0	2918
	Disponibilité de matière grasse en quantité (g/personne/jour)	Float64	15605	15605	0	1126
	Disponibilité de protéines en quantité (g/personne/jour)	Float64	15605	15605	0	1034
	Disponibilité intérieure(kg)	Int64	15605	15605	0	1794
	Exportations - Quantité(kg)	Int64	15605	15605	0	761
	Importations - Quantité(kg)	Int64	15605	15605	0	870
	Nourriture(kg)	Int64	15605	15605	0	1395
	Pertes(kg)	Int64	15605	15605	0	433
	Production(kg)	Int64	15605	15605	0	1656
	Semences(kg)	Int64	15605	15605	0	249
	Traitement(kg)	Int64	15605	15605	0	499
	Variation de stock(kg)	Int64	15605	15605	0	399

2.3 - Analyse exploratoire du fichier aide alimentaire

➤ PM - Exploration des données

[] ↴ 1 cellule masquée

FIN PM - Exploration des données

▼ 1.Afficher les dimensions du dataset

```
#Afficher les dimensions du dataset
FICHIER = aide_alimentaire
print("Le tableau comporte {} lignes".format(FICHIER.shape[0]))
print("Le tableau comporte {} colonne(s)".format(FICHIER.shape[1]))
```

CSV
Le tableau comporte 1475 lignes
Le tableau comporte 4 colonne(s)

▼ 2.Consulter le nombre de colonnes

```
aide_alimentaire.info()
```

CSV
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1475 entries, 0 to 1474
Data columns (total 4 columns):
 # Column Non-Null Count Dtype
 --- --
 0 Pays bénéficiaire 1475 non-null object
 1 Année 1475 non-null int64
 2 Produit 1475 non-null object
 3 Valeur 1475 non-null int64
 dtypes: int64(2), object(2)
 memory usage: 46.2+ KB

```
Description_Fichier_Tableau(aide_alimentaire)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Pays bénéficiaire	object	1475	1475	0	76
Année	int64	1475	1475	0	4
Produit	object	1475	1475	0	16
Valeur	int64	1475	1475	0	1086

```
aide_alimentaire['Année'].unique()
array([2013, 2014, 2015, 2016], dtype=int64)
```

3.Affichage des 5 premières lignes de la table

```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

	Pays bénéficiaire	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682
1	Afghanistan	2014	Autres non-céréales	335
2	Afghanistan	2013	Blé et Farin	39224
3	Afghanistan	2014	Blé et Farin	15160
4	Afghanistan	2013	Céréales	40504

4.changement du nom de la colonne Pays bénéficiaire par Zone

```
#changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire.rename(columns={'Pays bénéficiaire':'Zone'}, inplace=True)
aide_alimentaire.head()
```

	Zone	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682
1	Afghanistan	2014	Autres non-céréales	335
2	Afghanistan	2013	Blé et Farin	39224
3	Afghanistan	2014	Blé et Farin	15160
4	Afghanistan	2013	Céréales	40504

5.Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000 pour avoir des kg

CONVERSION DES UNITES :

```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000 pour avoir des kg
aide_alimentaire['Valeur']*=1000
aide_alimentaire.head()
```

	Zone	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

MODIFICATION DU NOM DE COLONNE :

Les noms et unités étant confuses, nous allons expliciter le nom de la dernière colonne :

```
aide_alimentaire.rename(columns={'Valeur':'Aide_alimentaire_annuelle(kg)'}, inplace=True)
```

✓ 6.Affichage des 5 premières lignes de la table

```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

	Zone	Année	Produit	Aide_alimentaire_annuelle(kg)
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

✓ 7.PM : AMELIORATION DU TYPE DES DONNEES :

```
aide_alimentaire.dtypes
```

Zone	object
Année	int64
Produit	object
Aide_alimentaire_annuelle(kg)	int64
dtype:	object

```
aide_alimentaire=aide_alimentaire.convert_dtypes()
```

```
aide_alimentaire.dtypes
```

Zone	string[python]
Année	Int64
Produit	string[python]
Aide_alimentaire_annuelle(kg)	Int64
dtype:	object

```
Description_Fichier_Tableau(aide_alimentaire)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	1475	1475	0	76
Année	Int64	1475	1475	0	4
Produit	string[python]	1475	1475	0	16
Aide_alimentaire_annuelle(kg)	Int64	1475	1475	0	1086

2.4 - Analyse exploratoire du fichier sous nutrition

> PM - Exploration des données

```
[ ] ↴ 1 cellule masquée
```

FIN PM - Exploration des données

✓ 1.Afficher les dimensions du dataset

```
#Afficher les dimensions du dataset
FICHIER = sous_nutrition
print("Le tableau comporte {} lignes".format(FICHIER.shape[0]))
print("Le tableau comporte {} colonne(s)".format(FICHIER.shape[1]))
```

```
↳ Le tableau comporte 1218 lignes
    Le tableau comporte 3 colonne(s)
```

✓ 2.Consulter le nombre de colonnes

```
sous_nutrition.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Zone     1218 non-null   object  
 1   Année    1218 non-null   object  
 2   Valeur   624 non-null   object  
dtypes: object(3)
memory usage: 28.7+ KB
```

Description_Fichier_Tableau(sous_nutrition)

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	object	1218	1218	0	203
Année	object	1218	1218	0	6
Valeur	object	1218	624	594	140

sous_nutrition['Année'].unique()

```
↳ array(['2012-2014', '2013-2015', '2014-2016', '2015-2017', '2016-2018',
       '2017-2019'], dtype=object)
```

3.Affichage des 5 premières lignes de la table

```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

	Zone	Année	Valeur
0	Afghanistan	2012-2014	8.6
1	Afghanistan	2013-2015	8.8
2	Afghanistan	2014-2016	8.9
3	Afghanistan	2015-2017	9.7
4	Afghanistan	2016-2018	10.5

4.Conversion de la colonne sous nutrition en numérique

(avec l'argument errors=coerce qui permet de convertir automatiquement les lignes qui ne sont pas des nombres en NaN)

Puis remplacement des NaN en 0

CONVERSION de la COLONNE Valeur(Sous_nutrition) en NUMERIQUE :

Analyse du type de données dans la table

```
#PM : type de données dans la table
sous_nutrition.dtypes
```

```
↳ Zone      object
     Année    object
     Valeur   object
dtype: object
```

Première essai de conversion avec la fonction fournie : sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'])
et Message d'erreur

```
#Conversion de la colonne sous nutrition en numérique
#sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'])
#MSG ERREUR => ValueError: Unable to parse string "<0.1" at position 60
```

ANALYSE DE L'ERREUR

sous_nutrition.iloc[60,2]

```
↳ '<0.1'
```

VISUALISATION DES DONNEES QUI POSENT PROBLEME :

Le format <0,1 ne peut pas être converti en format numérique par la fonction pd.to_numeric().

Vérifions qu'il n'y a pas d'autres valeurs non convertissables dans le fichier.

```
sous_nutrition['Valeur'].unique()
```

```
array(['8.6', '8.8', '8.9', '9.7', '10.5', '11.1', '2.2', '2.5', '2.8',
       '3', '3.1', '3.3', '0.1', '1.3', '1.2', 'nan', '7.6', '6.2', '5.3',
       '5.6', '5.8', '5.7', '1.5', '1.6', '1.1', '1.7', '<0.1', '21.7',
       '22.4', '23.3', '22.3', '21.5', '20.9', '0.8', '2', '1.9', '1.8',
       '0.4', '0.5', '0.3', '0.2', '3.2', '3.4', '3.6', '3.8', '2.1',
       '2.3', '2.4', '0.6', '0.7', '0.9', '3.9', '2.7', '1.4', '4.8',
       '4.6', '4.9', '5', '4.4', '4.3', '4.2', '4.5', '26.2', '24.3',
       '21.3', '21.1', '2.9', '5.1', '5.2', '5.4', '203.8', '198.3',
       '193.1', '190.9', '190.1', '189.2', '23.6', '24', '24.1', '3.7',
       '7.3', '7.8', '8.4', '9', '9.1', '10.1', '10', '10.7', '11.5',
       '11.9', '11.8', '8.7', '10.3', '11', '1', '5.5', '6.8', '7.9',
       '5.9', '7', '9.2', '9.4', '9.6', '6.7', '7.1', '7.2', '14.7',
       '17.4', '20.2', '22.2', '22.8', '24.6', '31.1', '28.5', '25.4',
       '24.8', '26.1', '14.5', '15.4', '16.5', '15.8', '15.7', '10.8',
       '11.2', '11.6', '12', '12.2', '13.5', '13.2', '12.8', '13', '13.4',
       '14.1', '4.1', '6.1', '6', '6.5', '2.6', '8', '8.3'], dtype=object)
```

On note la présence de nan mais pas d'autres éléments non convertissables en numérique.

Conversion de la colonne (avec l'argument errors='coerce' qui permet de convertir automatiquement les lignes qui ne sont pas des nombres en NaN).

```
#Conversion de la colonne (avec l'argument errors='coerce' qui permet de convertir automatiquement les lignes qui ne sont pas des nombres
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'],errors='coerce')
sous_nutrition.head()
```

	Zone	Année	Valeur
0	Afghanistan	2012-2014	8.6
1	Afghanistan	2013-2015	8.8
2	Afghanistan	2014-2016	8.9
3	Afghanistan	2015-2017	9.7
4	Afghanistan	2016-2018	10.5

```
sous_nutrition.dtypes
```

```
Zone      object
Année     object
Valeur    float64
dtype: object
```

```
sous_nutrition.iloc[60,2]
```

```
nan
```

Puis REEMPLACEMENT des NaN en 0

```
#Puis remplacement des NaN en 0
sous_nutrition.fillna(0,inplace=True)
```

```
sous_nutrition.iloc[60,2]
```

```
0.0
```

Remarque:

Attention cela signifie aussi que les autres nan du fichier auront une valeur 0.

J'en ai conscience; c'est le choix qui a été fait par Julien plus expérimenté.

✓ **5.Changement du nom de la colonne Valeur par Sous_nutrition**

```
#changement du nom de la colonne Valeur par sous_nutrition  
sous_nutrition.rename(columns={'Valeur':'Sous_nutrition'},inplace=True)  
sous_nutrition.head()
```

	Zone	Année	Sous_nutrition
0	Afghanistan	2012-2014	8.6
1	Afghanistan	2013-2015	8.8
2	Afghanistan	2014-2016	8.9
3	Afghanistan	2015-2017	9.7
4	Afghanistan	2016-2018	10.5

▼ 6.Multiplication de la colonne Sous_nutrition par 1_000_000

```
#Multiplication de la colonne sous_nutrition par 1000000  
sous_nutrition['Sous_nutrition']*=1_000_000
```

▼ 7.Affichage des 5 premières lignes de la table

```
#Afficher les 5 premières lignes de la table  
sous_nutrition.head()
```

	Zone	Année	Sous_nutrition
0	Afghanistan	2012-2014	8600000.0
1	Afghanistan	2013-2015	8800000.0
2	Afghanistan	2014-2016	8900000.0
3	Afghanistan	2015-2017	9700000.0
4	Afghanistan	2016-2018	10500000.0

▼ 8.PM : AMELIORATION DU TYPE DES DONNEES :

```
sous_nutrition.dtypes
```

```
Zone          object  
Année         object  
Sous_nutrition float64  
dtype: object
```

CONVERTIR TOUTES LES COLONNES vers le meilleur format possible :

```
sous_nutrition=sous_nutrition.convert_dtypes()
```

```
sous_nutrition.dtypes
```

```
Zone          string[python]  
Année         string[python]  
Sous_nutrition Float64  
dtype: object
```

FORCAGE POUR Passer la colonne Sous_nutrition en INT64

```
sous_nutrition = sous_nutrition.astype({'Sous_nutrition': 'Int64'})
```

```
sous_nutrition.dtypes
```

```
Zone          string[python]  
Année         string[python]  
Sous_nutrition Int64  
dtype: object
```

```
Description_Fichier_Tableau(sous_nutrition)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	1218	1218	0	203
Année	string[python]	1218	1218	0	6
Sous_nutrition	Int64	1218	1218	0	139

3.1 - Proportion de personnes en sous nutrition

▼ 1.jointure entre la table population et la table sous nutrition, en ciblant l'année 2017 :

```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'année 2017
```

➤ PM : ETUDE PREALABLE A LA JOINTURE

[] ↴ 8 cellules masquées

PM : FIN ETUDE PREALABLE A LA JOINTURE

▼ REMARQUE : NOMBRE DE PAYS

```
#CONSTATATION NOMBRE DE PAYS :
print("dans le fichier population il y a ", len((population.loc[population['Année']== 2017 ,:] )['Zone'].unique())),
      "valeurs distinctes pour la colonne 'Zone' et l'année 2017.")
print("dans le fichier sous_nutrition il y a ", len((sous_nutrition.loc[sous_nutrition['Année']=='2016-2018',:] )['Zone'].unique())),
      "valeurs distinctes pour la colonne 'Zone' et la période 2016-2018.")
```

☞ dans le fichier population il y a 236 valeurs distinctes pour la colonne 'Zone' et l'année 2017.
dans le fichier sous_nutrition il y a 203 valeurs distinctes pour la colonne 'Zone' et la période 2016-2018.

CHOIX DU TYPE DE JOINTURE

Pour 2017, dans le fichier population, il y a 236 pays, dans le fichier sous_nutrition il y a 203 Pays.

Etant donné que l'on veut calculer un pourcentage de personnes en état de sous-nutrition.

Il semble plus judicieux de ne garder

que les pays pour lesquels nous avons à la fois l'information de la population totale et de la population en sous-nutrition en 2017.

Cela correspond donc à un INNER JOIN

```
population_avc_sous_nutrition_2017 = pd.merge(left= population.loc[population['Année']== 2017 ,:],
                                              right= sous_nutrition.loc[sous_nutrition['Année']=='2016-2018',:],
                                              on='Zone', how='inner')
```

▼ 2.Affichage du dataset

```
#Affichage du dataset
display(population_avc_sous_nutrition_2017)
```

	Zone	Année_x	Population	Année_y	Sous_nutrition
0	Afghanistan	2017	36296113	2016-2018	10500000
1	Afrique du Sud	2017	57009756	2016-2018	3100000
2	Albanie	2017	2884169	2016-2018	100000
3	Algérie	2017	41389189	2016-2018	1300000
4	Allemagne	2017	82658409	2016-2018	0
...
198	Venezuela (République bolivarienne du)	2017	29402484	2016-2018	8000000
199	Viet Nam	2017	94600648	2016-2018	6500000
200	Yémen	2017	27834819	2016-2018	0
201	Zambie	2017	16853599	2016-2018	0
202	Zimbabwe	2017	14236595	2016-2018	0

203 rows × 5 columns

```
Description_Fichier_Tableau(population_avc_sous_nutrition_2017)
```

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	203	203	0	203
Année_x	Int64	203	203	0	1
Population	Int64	203	203	0	203
Année_y	string[python]	203	203	0	1
Sous_nutrition	Int64	203	203	0	50

REMARQUE :

Dans le fichier population en 2017 il y a 236 valeurs distinctes pour la colonne 'Zone'.

Dans le fichier sous_nutrition en 2016-2018 il y a 203 valeurs distinctes pour la colonne 'Zone'

Dans la Jointure interne avec un INNER JOIN on a 203 lignes.

Cela signifie donc que pour les 203 Pays connus dans le fichier sous-nutrition, nous connaissons la population totale dans le fichier population.

3. Calcul et affichage du nombre de personnes en état de sous nutrition

Le nombre de personne en état de sous nutrition correspond à la somme de toutes les personnes en état de sous-nutrition connus.

C'est à dire à la somme de la colonne Sous_nutrition.

REMARQUE :

Ici, on pourrait aussi bien le faire sur le fichier sous_nutrition.loc[sous_nutrition['Année']=='2016-2018',:] que sur le dataset (avec le même résultat attendu, car tous les pays du fichier sous_nutrition ont été inclus dans la jointure).

```
#Calcul et affichage du nombre de personnes en état de sous nutrition
Total_pers_SN = population_avc_sous_nutrition_2017['Sous_nutrition'].sum()
```

```
print("le nombre de personnes en état de sous-nutrition connus en 2017 est de : ",
      round(Total_pers_SN/1_000_000 ,1) , "millions .")
```

⤓ le nombre de personnes en état de sous-nutrition connus en 2017 est de : 535.7 millions .

```
#PM COMPLEMENT CALCUL DU POURCENTAGE DE LA POPULATION EN SOUS NUTRITION dans le DATASET:
```

```
Total_pers_SN = population_avc_sous_nutrition_2017['Sous_nutrition'].sum()
```

```
Total_pop_dataset = population_avc_sous_nutrition_2017['Population'].sum()
```

```
print("le nombre de personnes en état de sous-nutrition connus en 2017 est de : ",
      round(Total_pers_SN/1_000_000 ,1) , "millions .")
```

```
print("La population totale du DataSet est de :",
      round(Total_pop_dataset/1_000_000 ,1) , "millions .")
```

```
print("Dans les pays du Dataset, le pourcentage de la population en sous nutrition est de :",
      round(((Total_pers_SN*100)/Total_pop_dataset),2)," .%.")
```

⤓ le nombre de personnes en état de sous-nutrition connus en 2017 est de : 535.7 millions .

La population totale du DataSet est de : 7543.8 millions .

Dans les pays du Dataset, le pourcentage de la population en sous nutrition est de : 7.1 %.

4. PM : VISUALISATION Calcul et affichage du nombre de personnes en état de sous nutrition

```
Visualisation_population_avc_sous_nutrition_2017 = population_avc_sous_nutrition_2017[['Population','Sous_nutrition']].sum()
Visualisation_population_avc_sous_nutrition_2017.head()
```

```
⤓ Population      7543798769
Sous_nutrition   535700000
dtype: Int64
```

```
type(Visualisation_population_avc_sous_nutrition_2017)
```

⤓ pandas.core.series.Series

```
Visualisation_population_avc_sous_nutrition_2017['Population_nutrie']=Visualisation_population_avc_sous_nutrition_2017['Population']-Vi
```

```
Visualisation_population_avc_sous_nutrition_2017.head()
```

```
⤓ Population      7543798769
Sous_nutrition   535700000
Population_nutrie 7008098769
dtype: Int64
```

```
del Visualisation_population_avc_sous_nutrition_2017['Population']
```

```
Visualisation_population_avc_sous_nutrition_2017
```

```
→ Sous_nutrition      535700000  
Population_nutrie    7008098769  
dtype: Int64
```

```
Visualisation_population_avc_sous_nutrition_2017_2 = pd.DataFrame(Visualisation_population_avc_sous_nutrition_2017)  
Visualisation_population_avc_sous_nutrition_2017_2
```

```
→  
0  
Sous_nutrition      535700000  
Population_nutrie   7008098769  
|—————>
```

```
Visualisation_population_avc_sous_nutrition_2017_2.reset_index(inplace=True)  
Visualisation_population_avc_sous_nutrition_2017_2
```

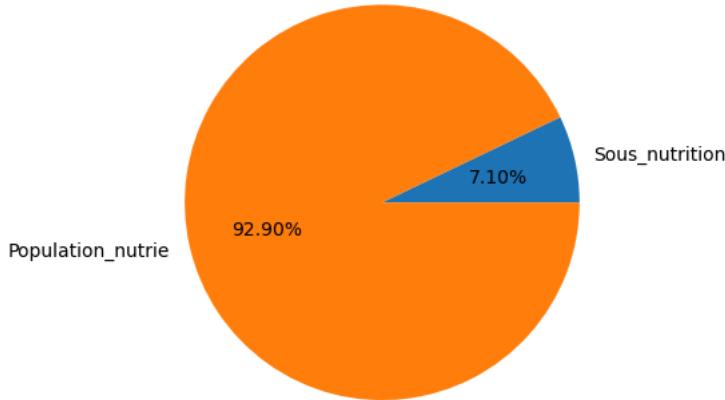
```
→ index      0  
0 Sous_nutrition 535700000  
1 Population_nutrie 7008098769  
|—————>
```

```
Visualisation_population_avc_sous_nutrition_2017_2.rename(columns={'index':'Label',0:'Quantité'},inplace=True)  
Visualisation_population_avc_sous_nutrition_2017_2.head()
```

```
→ Label      Quantité  
0 Sous_nutrition 535700000  
1 Population_nutrie 7008098769  
|—————>
```

```
plt.pie(x= Visualisation_population_avc_sous_nutrition_2017_2['Quantité'], labels=Visualisation_population_avc_sous_nutrition_2017_2['Label'],  
        autopct='%.2f %', startangle=90)  
plt.title("Proportion de la population sous_nutrie en 2017",  
         fontdict = {'fontsize' : 16, 'color':'green', 'verticalalignment': 'baseline'})  
plt.show()
```

→ Proportion de la population sous_nutrie en 2017



3.2 - Nombre théorique de personnes qui pourraient être nourries

✓ 1. Combien mange en moyenne un être humain ?

```
#Combien mange en moyenne un être humain ? Source =>
```

Les Apports en énergie des adultes

Pour un homme adulte, l'apport conseillé en énergie est, en moyenne, de 2 400 à 2 600 calories par jour, selon l'activité.
Pour une femme adulte, il est de 1 800 à 2 200 calories.

Considérons donc un apport moyen conseillé de **2500 calories/jr pour un homme**, et **2000 calories/jr pour une femme** et donc de **2250 calories/jr en moyenne**.

source VIDAL.fr :

<https://www.vidal.fr/sante/nutrition/equilibre-alimentaire-adulte/recommandations-nutritionnelles-adulte.html#:~:text=Pour%20un%20homme%20adulte%2C%201,800%20%C3%A0%202%2000%20calories>

Rappel 1calories =1kcal

▼ 2. Jointure DataFrame population et DataFrame Dispo_alimentaire

```
#On commence par faire une jointure entre le data frame population et Dispo_alimentaire afin d'ajouter dans ce dernier la population
```

➤ PM : ETUDE PREALABLE A LA JOINTURE

```
[ ] ↴ 12 cellules masquées
```

PM : FIN ETUDE PREALABLE A LA JOINTURE

▼ REMARQUE SUR NOMBRE DE PAYS :

```
#CONSTATATION NOMBRE DE PAYS :
```

```
print("dans le fichier population il y a ", len((population.loc[population['Année']== 2017 ,:])[['Zone']].unique()),  
      "valeurs distinctes pour la colonne 'Zone' et l'année 2017.")  
print("dans le fichier dispo_alimentaire il y a ", len(dispo_alimentaire[['Zone']] .unique()),  
      "valeurs distinctes pour la colonne 'Zone' en 2017.")
```

→ dans le fichier population il y a 236 valeurs distinctes pour la colonne 'Zone' et l'année 2017.
dans le fichier dispo_alimentaire il y a 174 valeurs distinctes pour la colonne 'Zone' en 2017.

Ajoutons la population en 2017 dans le DataFrame dispo_alimentaire

▼ CHOIX DU TYPE DE JONCTION

Ici on veut compléter le fichier dispo_alimentaire avec la Population de 2017.

Et on veut faire un travail qui tiendra compte de la population.

Nous devons donc conserver uniquement les lignes du fichier dispo_alimentaire pour lesquels nous avons la population.

Inversement nous n'avons besoin de la population que pour les lignes où nous connaissons la disponibilité alimentaire.

Nous faisons donc le choix de faire à nouveau un **INNER JOIN**

```
dispo_alimentaire_avec_pop2017 = pd.merge(left= dispo_alimentaire,  
                                         right = population.loc[population['Année']==2017,['Zone','Population']],  
                                         on = 'Zone',  
                                         how = 'inner'  
)
```

```
Description_Fichier_Tableau( dispo_alimentaire_avec_pop2017)
```



	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	15416	15416	0	172
Produit	string[python]	15416	15416	0	98
Origine	string[python]	15416	15416	0	2
Aliments pour animaux(kg)	Int64	15416	15416	0	530
Autres Utilisations(kg)	Int64	15416	15416	0	399
Disponibilité alimentaire (Kcal/personne/jour)	Int64	15416	15416	0	556
Disponibilité alimentaire en quantité (kg/personne/an)	Float64	15416	15416	0	2891
Disponibilité de matière grasse en quantité (g/personne/jour)	Float64	15416	15416	0	1116
Disponibilité de protéines en quantité (g/personne/jour)	Float64	15416	15416	0	1025
Disponibilité intérieure(kg)	Int64	15416	15416	0	1773
Exportations - Quantité(kg)	Int64	15416	15416	0	748
Importations - Quantité(kg)	Int64	15416	15416	0	852
Nourriture(kg)	Int64	15416	15416	0	1379
Pertes(kg)	Int64	15416	15416	0	431
Production(kg)	Int64	15416	15416	0	1638
Semences(kg)	Int64	15416	15416	0	248
Traitement(kg)	Int64	15416	15416	0	490
Variation de stock(kg)	Int64	15416	15416	0	395
Population	Int64	15416	15416	0	172

Remarque :

Nous avons 172 pays dans la jointure, ce qui signifie qu'il ya 2 pays qui étaient présent dans le fichier dispo_alimentaire et qui ne le sont pas dans le fichier population.

➤ PM Remarque sur le manque d'homogénéité des noms de pays

[] ↴ 33 cellules masquées

FIN PM Remarque sur le manque d'homogénéité des noms de pays

➤ 3. Affichage du nouveau dataframe

```
#Affichage du nouveau dataframe  
display(dispo_alimentaire_avec_pop2017)
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Di (g/
0	Afghanistan	Abats Comestible	animale	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	4	2.7	0.02	
...
15411	îles Salomon	Viande de Suides	animale	0	0	45	4.7	4.28	
15412	îles Salomon	Viande de Volailles	animale	0	0	11	3.34	0.69	
15413	îles Salomon	Viande, Autre	animale	0	0	0	0.06	0.0	
15414	îles Salomon	Vin	vegetale	0	0	0	0.07	0.0	
15415	îles Salomon	Épices, Autres	vegetale	0	0	4	0.48	0.21	

15416 rows x 19 columns

▼ 4. Création de la colonne dispo_kcal (/jour) avec calcul des kcal disponibles mondialement

CREATION DE LA COLONNE 'dispo_kcal (/jour)'

A partir de la colonne 'Disponibilité alimentaire (Kcal/personne/jour)',

A partir de la population du pays en 2017,

Créons la colonne 'dispo_kcal (/jour)' :

disponibilité en kcal/jour = disponibilité en Kcal/personne/jour * Nombre de personnes

```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
dispo_alimentaire_avec_pop2017['dispo_kcal (/jour)'] = (dispo_alimentaire_avec_pop2017['Disponibilité alimentaire (Kcal/personne/jour)']
                                                       *dispo_alimentaire_avec_pop2017['Population'])
```

dispo_alimentaire_avec_pop2017.head()

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Dispon pr (g/pers
0	Afghanistan	Abats Comestible	animale	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	4	2.7	0.02	

CALCUL DES KCAL DISPONIBLES MONDIALEMENT PAR JOUR

Il s'agit de la somme de toutes les Kcal/jour disponibles dans toutes les catégories.

```
#le nombre de Kcal disponibles mondialement par jour est de :
Total_mond_kcal_jr_dispo = dispo_alimentaire_avec_pop2017['dispo_kcal (/jour)'].sum()
print (Total_mond_kcal_jr_dispo)
```

```
→ 20918984609007
```

```
print("le nombre de Kcal disponibles mondialement par jour est de :",
      round( Total_mond_kcal_jr_dispo / 1_000_000_000 ,3),
      " milliards de Kcal/jour .")
→ le nombre de Kcal disponibles mondialement par jour est de : 20918.985 milliards de Kcal/jour .
```

✓ 5. Calcul du nombre d'humains pouvant être nourris

CALCUL DU NOMBRE D'ETRE HUMAIN POUVANT ETRE NOURRI PAR JOUR :

On rappelle que la notation abusives kcal correspond à 1 calorie. (Voir Lexique des données)

On rappelle que le besoin moyen d'une personne est de 2250 calories par jour. (Source Vidal)

Nous avons donc :

Nombre de personnes pouvant être nourrie = Nbr de Kcal dispo par jour/Nbre de Kcal nécessaire par personne par jour

```
#le besoin journalier en kcal (=calories) d'une personne (Source Vidal.fr):
Besoin_kcal_jr_pers = 2250
```

```
#le nombre de Kcal disponibles mondialement par jour est de :
Total_mond_kcal_jr_dispo
```

```
#Calcul du nombre d'humains pouvant être nourris (par jour):
Nb_hum_nourrissable_jr = Total_mond_kcal_jr_dispo / Besoin_kcal_jr_pers
print(Nb_hum_nourrissable_jr)
```

```
→ 9297326492.892
```

```
#Calcul du nombre d'humains pouvant être nourris
print("le nombre d'êtres humains pouvant être nourris est de :",
      round ( Nb_hum_nourrissable_jr / 1_000_000_000 ,3),
      " milliards.")
```

```
→ le nombre d'êtres humains pouvant être nourris est de : 9.297 milliards.
```

Calcul de la population totale des pays dans le fichier JOINTURE dispo_alimentaire avec_pop2017

Remarque :

-tous les pays du fichier population ne sont pas présents dans notre fichier dispo_alimentaire_avec_pop2017.

Nous ne pouvons donc pas utiliser la population totale du fichier de base population

-Dans notre fichier dispo_alimentaire_avec_pop2017 les pays apparaissent plusieurs fois.

Nous ne pouvons donc pas sommer la population de toutes les lignes,

ni regrouper par pays, car en regroupant, la population d'un même pays sera reprise autant de fois que ce pays est présent.

Nous allons donc **extraire la base des pays présents dans le fichier-jointure dispo_alimentaire_avec_pop2017**

puis **calculer sa population totale**.

Liste des pays dans dispo_alimentaire_avec_pop2017

```
dispo_alimentaire_avec_pop2017['Zone'].unique()
```

```
→ <StringArray>
[      'Afghanistan',      'Afrique du Sud',      'Albanie',
      'Algérie',          'Allemagne',          'Angola',
      'Antigua-et-Barbuda', 'Arabie saoudite',     'Argentine',
      'Arménie',
...
      'Viet Nam',           'Yémen',           'Zambie',
      'Zimbabwe',          'Égypte',          'Émirats arabes unis',
      'Équateur',          "États-Unis d'Amérique", 'Éthiopie',
      'îles Salomon']
Length: 172, dtype: string
```

```
#DataFrame avec les pays présents dans df_pays_ds_disp_alim_2017
df_pays_ds_disp_alim_2017 = pd.DataFrame( dispo_alimentaire_avec_pop2017['Zone'].unique() , columns=['Zone'])
df_pays_ds_disp_alim_2017.head()
```

	Zone
0	Afghanistan
1	Afrique du Sud
2	Albanie
3	Algérie
4	Allemagne

```
#Complétons le df_pays_ds_disp_alim_2017 avec les populations de 2017 :
df_pays_ds_disp_alim_2017_pop = pd.merge(left= df_pays_ds_disp_alim_2017,
                                         right = population.loc[population['Année']==2017,['Zone','Population']],
                                         on = 'Zone',
                                         how = 'left'
                                         )

df_pays_ds_disp_alim_2017_pop.head()
```

	Zone	Population
0	Afghanistan	36296113
1	Afrique du Sud	57009756
2	Albanie	2884169
3	Algérie	41389189
4	Allemagne	82658409

Description_Fichier_Tableau(df_pays_ds_disp_alim_2017_pop)

	Type	Nb lignes	Valeurs non-vides	Valeurs vides	Valeurs distinctes
Zone	string[python]	172	172	0	172
Population	Int64	172	172	0	172

> PM une autre façon, de créer le fichier : un filtre sur le fichier population

[] ↴ 2 cellules masquées

✗ FIN PM une autre façon, de créer le fichier : un filtre sur le fichier population

```
#Calculons la population totale des pays dans dispo_alimentaire_avec_pop2017 :
Total_pop_ds_disp_alim_2017 = df_pays_ds_disp_alim_2017_pop['Population'].sum()
print(Total_pop_ds_disp_alim_2017)
```

⤵ 7291900824

```
print ("la population totale des pays dans le fichier-jointure dispo_alimentaire_avec_pop2017 est de ",
      round(Total_pop_ds_disp_alim_2017 / 1_000_000_000,3), " milliards.")
```

⤵ la population totale des pays dans le fichier-jointure dispo_alimentaire_avec_pop2017 est de 7.292 milliards.

COMPARAISON dans le fichier jointure, pour 2017 de la Population Totale et Population pouvant être nourrie par jour:

```
print("dans ce fichier-jointure dispo_alimentaire_avec_pop2017 :")

print("le nombre d'êtres humains pouvant être nourris est de :",
      round ( Nb_hum_nourrissable_jr / 1_000_000_000 ,3),
      " milliards.")

print ("la population totale des pays de ce fichier-jointure est de ",
      round(Total_pop_ds_disp_alim_2017 / 1_000_000_000,3), " milliards.")

print ("Soit ",
      round(Nb_hum_nourrissable_jr*100 / Total_pop_ds_disp_alim_2017,2), " % de la population peut être nourrie.")

⤵ dans ce fichier-jointure dispo_alimentaire_avec_pop2017 :
le nombre d'êtres humains pouvant être nourris est de : 9.297 milliards.
la population totale des pays de ce fichier-jointure est de 7.292 milliards.
Soit 127.5 % de la population peut être nourrie.
```

CONCLUSION :

Concernant les pays pour lesquels nous connaissons la disponibilité alimentaire et la population en 2017,
la disponibilité alimentaire globale permettrait de nourrir la population globale de ces pays, sur la base des apports nécessaire moyen pour une personne.

3.3 - Nombre théorique de personnes qui pourraient être nourries avec les produits végétaux

Principe :

Il s'agit de refaire l'étude en restreignant les données aux apports végétaux à l'aide de la colonne Origine.

▼ PM TRAVAIL PERSO

Vérifions les différentes valeur possibles dans la colonne 'Origine':

```
dispo_alimentaire_avec_pop2017['Origine'].unique()
```

```
→ <StringArray>
  ['animale', 'vegetale']
  Length: 2, dtype: string
```

Il n'y a bien que 2 valeurs possibles ['animale', 'vegetale']

FIN PM TRAVAIL PERSO

▼ 1. Transfert des données avec les végétaux dans un nouveau dataframe

A partir du DataFrame dispo_alimentaire_avec_pop2017 , créons un nouveau DataFrame réstrent aux origines végétales :

```
#Transfert des données avec les végétaux dans un nouveau dataframe
dispo_alimentaire_VEGETALE_avec_pop2017 = dispo_alimentaire_avec_pop2017.loc[dispo_alimentaire_avec_pop2017['Origine']=='vegetale',:]
dispo_alimentaire_VEGETALE_avec_pop2017.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité prot (g/personne/jour)
1	Afghanistan	Agrumes, Autres	vegetale	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	4	2.7	0.02	
6	Afghanistan	Bière	vegetale	0	0	0	0.09	0.0	

▼ 2. Calcul du nombre de kcal disponible pour les végétaux

```
#Calcul du nombre de kcal disponible pour les végétaux
```

CALCUL DES KCAL VEGETALES DISPONIBLES MONDIALEMENT PAR JOUR

Il s'agit de la somme de toutes les Kcal/jour VEGETALES disponibles dans toutes les catégories.

```
#le nombre de Kcal VEGETALES disponibles mondialement par jour est de :
Total_mond_kcal_jr_dispo_VEGETAL = dispo_alimentaire_VEGETALE_avec_pop2017 ['dispo_kcal (/jour)'].sum()
print (Total_mond_kcal_jr_dispo_VEGETAL)
```

```
→ 17260764197424
```

```
print("le nombre de Kcal VEGETALES disponibles mondialement par jour est de :",
      round( Total_mond_kcal_jr_dispo_VEGETAL / 1_000_000_000 ,3),
```

```
" milliards de Kcal/jour .")
```

→ le nombre de Kcal VEGETALES disponibles mondialement par jour est de : 17260.764 milliards de Kcal/jour .

✓ 3. Calcul du nombre d'humains pouvant être nourris avec les végétaux

```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
```

On rappelle que la notation abusives kcal correspond à 1 calorie. (Voir Lexique des données)

On rappelle que le besoin moyen d'une personne est de 2250 calories par jour. (Source Vidal)

Nombre de personnes pouvant être nourrie = Nbr de Kcal dispo par jour/Nbre de Kcal nécessaire par personne par jour

```
#le besoin journalier en kcal (=calories) d'une personne (Source Vidal.fr):  
# déjà implémenté : Besoin_kcal_jr_pers = 2250
```

```
#le nombre de Kcal VEGETALES disponibles mondialement par jour est de :  
Total_mond_kcal_jr_dispo_VEGETAL
```

```
#Calcul du nombre d'humains pouvant être nourris (par jour) en VEGETAL:  
Nb_hum_nourrissable_jr_VEGETAL = Total_mond_kcal_jr_dispo_VEGETAL / Besoin_kcal_jr_pers  
print(Nb_hum_nourrissable_jr_VEGETAL)
```

→ 7671450754.410666

```
#Calcul du nombre d'humains pouvant être nourris  
print("Avec des produits végétaux, le nombre d'êtres humains pouvant être nourris est de :"  
     round ( Nb_hum_nourrissable_jr_VEGETAL / 1_000_000_000 ,3),  
     " milliards.")
```

→ Avec des produits végétaux, le nombre d'êtres humains pouvant être nourris est de : 7.671 milliards.

Calcul de la population totale des pays dans la fichier dispo_alimentaire_avec_pop2017

Nous n'avons pas besoin de reprendre ici les calculs déjà effectués concernant la population totale.

Ils sont les mêmes que l'on considère toutes origines, ou Origine végétale.

En effet même si un pays ne produisait pas de végétal, sa population aurait tout de même besoin d'être nourrie !

POUR RAPPEL :

```
print ("la population totale des pays dans dispo_alimentaire_avec_pop2017 est de ",  
      round(Total_pop_ds_disp_alim_2017 / 1_000_000_000,3), " milliards.")
```

→ la population totale des pays dans dispo_alimentaire_avec_pop2017 est de 7.292 milliards.

COMPARAISON pour 2017 de la Population Totale et Population pouvant être nourrie par jour avec les végétaux:

```
print("dans ce fichier-jointure dispo_alimentaire_VEGETALE_avec_pop2017 :")
```

```
print("le nombre d'êtres humains pouvant être nourris est de :"  
     round ( Nb_hum_nourrissable_jr_VEGETAL / 1_000_000_000 ,3),  
     " milliards.")
```

```
print ("la population totale des pays de ce fichier-jointure est de ",  
      round(Total_pop_ds_disp_alim_2017 / 1_000_000_000,3), " milliards.")
```

```
print ("Soit ",  
      round(Nb_hum_nourrissable_jr_VEGETAL*100 / Total_pop_ds_disp_alim_2017,2),  
      " % de la population peut être nourrie avec l'alimentation d'origine végétale.")
```

→ dans ce fichier-jointure dispo_alimentaire_VEGETALE_avec_pop2017 :
le nombre d'êtres humains pouvant être nourris est de : 7.671 milliards.
la population totale des pays de ce fichier-jointure est de 7.292 milliards.
Soit 105.21 % de la population peut être nourrie avec l'alimentation d'origine végétale.

CONCLUSION :

Concernant les pays pour lesquels nous connaissons la disponibilité alimentaire et la population en 2017,

la disponibilité alimentaire globale, d'origine uniquement végétale, permettrait de nourrir la population globale de ces pays, sur la base des apports nécessaire moyen pour une personne.

3.4 - Utilisation de la disponibilité intérieure

dispo_alimentaire.head()

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité (g/pers)
0	Afghanistan	Abats Comestible	animale	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	4	2.7	0.02	

POUR RAPPEL :

Il existe 3 manière de calculer la Disponibilité intérieure :

Calcul direct

Disponibilité intérieure = Production + Importations - Exportations + Variation de stock

Disponibilité intérieure = Semences + Pertes + Nourriture + Aliments pour animaux + Traitement + Autres utilisations

▼ 1. Calcul de la disponibilité totale

▼ CALCUL de LA DISPO INTERIEURE - METHODE 1 DIRECTE

IL Y A UNE COLONNE DISPONIBILITE INTERIEURE, on peut donc l'utiliser directement :

#Calcul de la disponibilité totale

```
dispo_interieure_totale_1 = dispo_alimentaire['Disponibilité intérieure(kg)'].sum()
print("En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : ",
      round( dispo_interieure_totale_1 /(1_000_000*1_000),3),
      " millions de tonnes.")
```

→ En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : 984.899 millions de tonnes.

FIN CALCUL de LA DISPO INTERIEURE - METHODE 1 DIRECTE

▼ CALCUL de LA DISPO INTERIEURE - METHODE 2 par Flux Prod+Import-Export+Var

```
#Calcul de la disponibilité totale
dispo_interieure_totale_2 = (
    dispo_alimentaire['Production(kg)'].sum()
    +dispo_alimentaire['Importations - Quantité(kg)'].sum()
    - dispo_alimentaire['Exportations - Quantité(kg)'].sum()
    + dispo_alimentaire['Variation de stock(kg)'].sum()
)
print(dispo_interieure_totale_2)
```

→ 9849173000000

```
print("En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : ",
      round(dispo_interieure_totale_2 /(1_000_000*1_000),3),
      " millions de tonnes.")
```

→ En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : 984.917 millions de tonnes.

FIN CALCUL de LA DISPO INTERIEURE - METHODE 2 par Flux Prod+Import-Export+Var

✓ A UTILISER -CALCUL de LA DISPO INTERIEURE - METHODE 3 par Type d'Utilisation

```
#Calcul de la disponibilité totale
dispo_interieure_totale_3 = (
    dispo_alimentaire['Aliments pour animaux(kg)'].sum()
    +dispo_alimentaire['Autres Utilisations(kg)'].sum()
    + dispo_alimentaire['Nourriture(kg)'].sum()
    + dispo_alimentaire['Pertes(kg)'].sum()
    + dispo_alimentaire['Semences(kg)'].sum()
    + dispo_alimentaire['Traitement(kg)'].sum()
)
print(dispo_interieure_totale_3)

→ 9858592000000
```

```
print("En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : ",
      round(dispo_interieure_totale_3 /(1_000_000*1_000),3),
      " millions de tonnes.")
```

→ En 2017, la disponibilité intérieure totale pour l'ensemble des pays est de : 985.859 millions de tonnes.

✓ FIN A UTILISER -CALCUL de LA DISPO INTERIEURE - METHODE 3 par Type d'Utilisation

Remarque :

les chiffres ne sont pas exactement les mêmes, mais ils sont globalement du même ordre de grandeur.

Et si on veut faire des pourcentages en fonction de l'utilisation, pour arriver à 100, il faudra prendre cette 3 eme méthode de calcul !!!

✓ 2. création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux, pertes, nourritures,

```
#création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux, pertes, nourritures,
```

```
#Rappel des colonnes :
dispo_alimentaire.columns
```

```
→ Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Disponibilité alimentaire (Kcal/personne/jour)', 'Disponibilité alimentaire en quantité (kg/personne/an)', 'Disponibilité de matière grasse en quantité (g/personne/jour)', 'Disponibilité de protéines en quantité (g/personne/jour)', 'Disponibilité intérieure(kg)', 'Exportations - Quantité(kg)', 'Importations - Quantité(kg)', 'Nourriture(kg)', 'Pertes(kg)', 'Production(kg)', 'Semences(kg)', 'Traitement(kg)', 'Variation de stock(kg)'],
       dtype='object')
```

```
Liste_col_utilisation_dispo = [ 'Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Nourriture(kg)', 'Pertes(kg)', 'Semences(kg)', 'Traitement(kg)' ]
```

➤ SI ON VEUT FAIRE DE LA LECTURE DIRECTE :

```
[ ] ↴ 1 cellule masquée
```

FIN SI ON VEUT FAIRE DE LA LECTURE DIRECTE :

✓ SI ON VEUT CONSTRUIRE UN DATA FRAME EXPLOITABLE POUR VISUALISATION :

```
Liste_utilisation_dispo_Volumes_kg =[]
for elt in Liste_col_utilisation_dispo :
    Liste_utilisation_dispo_Volumes_kg.append(dispo_alimentaire[elt].sum())

print(Liste_utilisation_dispo_Volumes_kg)
```

→ [130424500000, 865023000000, 487625800000, 453698000000, 154681000000, 220468700000]

```

#VERIFICATION TOTAL :
Somme=0
for elt in Liste_utilisation_dispo_Volumes_kg :
    Somme+=elt

print (Somme == dispo_interieure_totale_3)
→ True

Liste_utilisation_dispo_Pourcentage =[]
for elt in Liste_col_utilisation_dispo :
    Liste_utilisation_dispo_Pourcentage.append(
        (dispo_alimentaire[elt].sum())*100/dispo_interieure_totale_3)

print(Liste_utilisation_dispo_Pourcentage)
→ [13.229526082426375, 8.774305702071858, 49.46201242530373, 4.602056764292508, 1.5689968709527689, 22.363102154952756]

```

```

#Mettons tout cela dans un DataFrame pour visualisation :
d = {'Volumes_kg' : Liste_utilisation_dispo_Volumes_kg , 'Pourcentage' : Liste_utilisation_dispo_Pourcentage}
df_utilisation_dispo = pd.DataFrame(data=d, index = Liste_col_utilisation_dispo)
display(df_utilisation_dispo)

```

	Volumes_kg	Pourcentage
Aliments pour animaux(kg)	1304245000000	13.229526
Autres Utilisations(kg)	865023000000	8.774306
Nourriture(kg)	4876258000000	49.462012
Pertes(kg)	453698000000	4.602057
Semences(kg)	154681000000	1.568997
Traitement(kg)	2204687000000	22.363102

LA LECTURE

```

#Mettons tout cela dans un DataFrame pour visualisation : ET EN TRANPOSE :
df_utilisation_dispo_transposed = df_utilisation_dispo.transpose()
display(df_utilisation_dispo_transposed)

```

	Aliments pour animaux(kg)	Autres Utilisations(kg)	Nourriture(kg)	Pertes(kg)	Semences(kg)	Traitement(kg)
Volumes_kg	1.304245e+12	8.650230e+11	4.876258e+12	4.536980e+11	1.546810e+11	2.204687e+12
Pourcentage	1.322953e+01	8.774306e+00	4.946201e+01	4.602057e+00	1.568997e+00	2.236310e+01

```
df_utilisation_dispo_transposed.columns.tolist()
```

```

→ ['Aliments pour animaux(kg)',
 'Autres Utilisations(kg)',
 'Nourriture(kg)',
 'Pertes(kg)',
 'Semences(kg)',
 'Traitement(kg)']

```

```

print ("En 2017, dans le fichier dispo_alimentaire :")
for elt in df_utilisation_dispo_transposed.columns.tolist() :
    print("l'utilisation pour ",elt," représente :")
    print("\t un volume de ",round(df_utilisation_dispo_transposed.loc['Volumes_kg',elt] /(1_000_000*1_000),3),
          " millions de tonnes.")
    print("\t un part de ",round(df_utilisation_dispo_transposed.loc['Pourcentage',elt],2),
          "% de la disponibilité intérieure.")
    print()

```

```

→ En 2017, dans le fichier dispo_alimentaire :
l'utilisation pour Aliments pour animaux(kg) représente :
    un volume de 130.424 millions de tonnes.
    un part de 13.23 % de la disponibilité intérieure.

```

```

l'utilisation pour Autres Utilisations(kg) représente :
    un volume de 86.502 millions de tonnes.
    un part de 8.77 % de la disponibilité intérieure.

```

```

l'utilisation pour Nourriture(kg) représente :
    un volume de 487.626 millions de tonnes.
    un part de 49.46 % de la disponibilité intérieure.

```

l'utilisation pour Pertes(kg) représente :
un volume de 45.37 millions de tonnes.
un part de 4.6 % de la disponibilité intérieure.

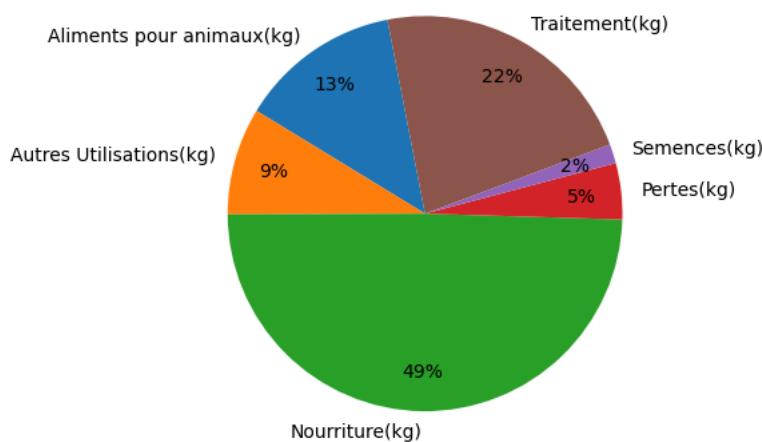
l'utilisation pour Semences(kg) représente :
un volume de 15.468 millions de tonnes.
un part de 1.57 % de la disponibilité intérieure.

l'utilisation pour Traitement(kg) représente :
un volume de 220.469 millions de tonnes.
un part de 22.36 % de la disponibilité intérieure.

LA VISUALISATION

```
plt.pie(x= df_utilisation_dispo['Volumes_kg'],  
        labels=df_utilisation_dispo.index, autopct='%.0f%%',  
        pctdistance =0.8,  
        startangle=101  
)  
  
plt.title("Repartition de l'utilisation de la disponibilité intérieure",  
          fontdict = {'fontsize' : 16, 'color':'green', 'verticalalignment': 'baseline'})  
plt.show()
```

Repartition de l'utilisation de la disponibilité intérieure



FIN SI ON VEUT CONSTRUIRE UN DATA FRAME EXPLOITABLE POUR VISUALISATION :

SI ON VEUT CONSTRUIRE UN DATA FRAME EXPLOITABLE POUR VISUALISATION METHODE 2 :

↳ ↓ 9 cellules masquées

FIN SI ON VEUT CONSTRUIRE UN DATA FRAME EXPLOITABLE POUR VISUALISATION METHODE 2 :

SI ON VEUT CONSTRUIRE UN HISTOGRAMME :

```
df_utilisation_dispo
```

	Volumes_kg	Pourcentage
Aliments pour animaux(kg)	1304245000000	13.229526
Autres Utilisations(kg)	865023000000	8.774306
Nourriture(kg)	4876258000000	49.462012
Pertes(kg)	453698000000	4.602057
Semences(kg)	154681000000	1.568997
Traitement(kg)	2204687000000	22.363102

```
df_utilisation_dispo_2 = df_utilisation_dispo.reset_index().rename(columns={'index':'Utilisation'})  
df_utilisation_dispo_2
```

	Utilisation	Volumes_kg	Pourcentage
0	Aliments pour animaux(kg)	1304245000000	13.229526
1	Autres Utilisations(kg)	865023000000	8.774306
2	Nourriture(kg)	4876258000000	49.462012
3	Pertes(kg)	453698000000	4.602057
4	Semences(kg)	154681000000	1.568997
5	Traitement(kg)	2204687000000	22.363102

```
df_utilisation_dispo_2['Volumes_kg']=round( df_utilisation_dispo_2['Volumes_kg']/(1_000_000*1_000),3)
df_utilisation_dispo_2
```

	Utilisation	Volumes_kg	Pourcentage
0	Aliments pour animaux(kg)	130.424	13.229526
1	Autres Utilisations(kg)	86.502	8.774306
2	Nourriture(kg)	487.626	49.462012
3	Pertes(kg)	45.370	4.602057
4	Semences(kg)	15.468	1.568997
5	Traitement(kg)	220.469	22.363102

```
df_utilisation_dispo_2.rename(columns={'Volumes_kg':'Volumes_Millions_de_tonnes'},inplace=True)
df_utilisation_dispo_2
```

	Utilisation	Volumes_Millions_de_tonnes	Pourcentage
0	Aliments pour animaux(kg)	130.424	13.229526
1	Autres Utilisations(kg)	86.502	8.774306
2	Nourriture(kg)	487.626	49.462012
3	Pertes(kg)	45.370	4.602057
4	Semences(kg)	15.468	1.568997
5	Traitement(kg)	220.469	22.363102

Liste_col_utilisation_dispo

```
['Aliments pour animaux(kg)',  
 'Autres Utilisations(kg)',  
 'Nourriture(kg)',  
 'Pertes(kg)',  
 'Semences(kg)',  
 'Traitement(kg)']
```

```
df_utilisation_dispo_2['Utilisation']= ['Aliments animaux', 'Autres',  
                                         'Nourriture','Pertes',  
                                         'Semences', 'Traitement']
```

df_utilisation_dispo_2

	Utilisation	Volumes_Millions_de_tonnes	Pourcentage
0	Aliments animaux	130.424	13.229526
1	Autres	86.502	8.774306
2	Nourriture	487.626	49.462012
3	Pertes	45.370	4.602057
4	Semences	15.468	1.568997
5	Traitement	220.469	22.363102

```
plt.figure(figsize=(10,6))
sns.set_theme(style='dark', palette='muted')

sns.barplot(data=df_utilisation_dispo_2,
            x='Utilisation',
            y='Volumes_Millions_de_tonnes',
            errorbar=None,estimator='sum',hue='Utilisation')
plt.grid(axis='y')
plt.ylabel("Volume en Millions de tonnes")
```

```

plt.xlabel("Type d'utilisation")

Volumes = df_utilisation_dispo_2['Volumes_Millions_de_tonnes'].tolist()
for i in range(len(Volumes)) :
    plt.text(i-0.2, Volumes[i],round(Volumes[i],1))

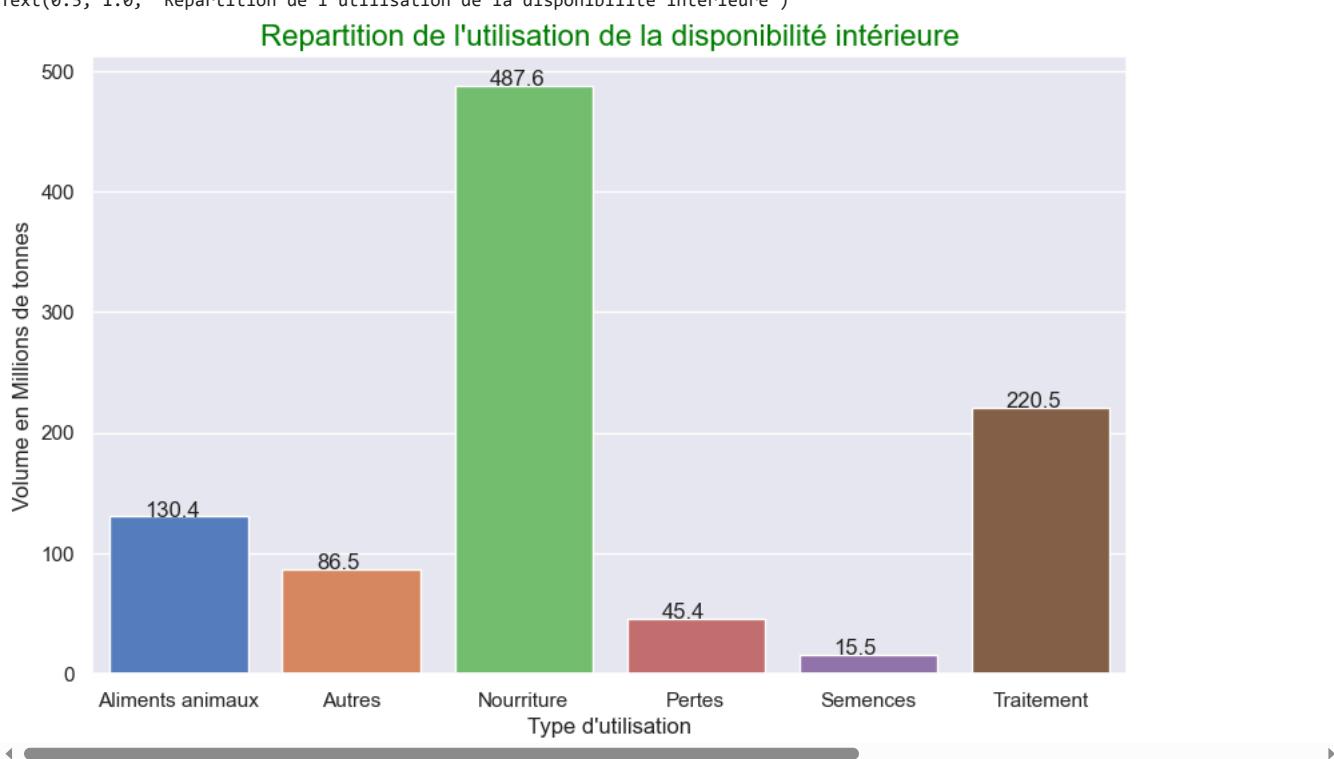
```

```

plt.title("Repartition de l'utilisation de la disponibilité intérieure",
          fontdict={'fontsize' : 16, 'color':'green'})

→ Text(0.5, 1.0, "Repartition de l'utilisation de la disponibilité intérieure")

```



Les 3 principales Utilisations de la disponibilité sont :

- Nourriture
- Traitement
- Aliments pour animaux

Remarque : Signification de Traitement :

Quantité du produit utilisée pour la transformation, comme par exemple pour la production d'huiles, de farine, ou d'autres produits transformés.

3.5 - Utilisation des céréales

Nous n'aurons pas besoin de la population,
nous pouvons utiliser le fichier dispo_alimentaire de base.

```
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Utilisations(kg)	Autres	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité pr (g/pers)
0	Afghanistan	Abats Comestible	animale	0	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	0	4	2.7	0.02	

▼ 1. Création d'une liste avec toutes les variables

```
#Création d'une liste avec toutes les variables
```

```
# Liste des Produits:  
Liste_Produits = dispo_alimentaire['Produit'].unique().tolist()  
print(Liste_Produits)
```

→ ['Abats Comestible', 'Agrumes, Autres', 'Aliments pour enfants', 'Ananas', 'Bananes', 'Beurre, Ghee', 'Bière', 'Blé', 'Boissons Alco...']

```
#Liste des Céréales :  
Liste_cereales=['Blé' , 'Riz (Eq Blanchi)' , 'Orge' , 'Maïs' , 'Seigle', 'Avoine' ,  
'Millet' , 'Sorgho' , 'Céréales, Autres']
```

▼ 2. Création d'un dataframe avec les informations uniquement pour ces céréales

```
dispo_alimentaire_cereales = dispo_alimentaire.loc[dispo_alimentaire['Produit'].isin(Liste_cereales) ,:]  
display(dispo_alimentaire_cereales)
```

→

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disp
7	Afghanistan	Blé	vegetale	0	0	1369	160.23	4.69	
12	Afghanistan	Céréales, Autres	vegetale	0	0	0	0.0	0.0	
32	Afghanistan	Maïs	vegetale	200000000	0	21	2.5	0.3	
34	Afghanistan	Millet	vegetale	0	0	3	0.4	0.02	
40	Afghanistan	Orge	vegetale	360000000	0	26	2.92	0.24	
...
15545	îles Salomon	Céréales, Autres	vegetale	0	0	0	0.0	0.0	
15568	îles Salomon	Maïs	vegetale	0	0	1	0.15	0.01	
15575	îles Salomon	Orge	vegetale	0	0	0	0.07	0.0	
15591	îles Salomon	Riz (Eq Blanchi)	vegetale	0	12000000	623	63.76	1.36	
15593	îles Salomon	Sorgho	vegetale	0	0	0	0.0	0.0	

1497 rows × 18 columns

▼ 3. Affichage de la proportion d'alimentation animale

Choix des colonnes :

il s'agit à nouveau de s'intéresser aux différentes utilisations faites des Produits

```
dispo_alimentaire_cereales.columns
```

→ Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux(kg)',
'Autres Utilisations(kg)',
'Disponibilité alimentaire (Kcal/personne/jour)',
'Disponibilité alimentaire en quantité (kg/personne/an)',
'Disponibilité de matière grasse en quantité (g/personne/jour)',
'Disponibilité de protéines en quantité (g/personne/jour)',
'Disponibilité intérieure(kg)', 'Exportations - Quantité(kg)',
'Importations - Quantité(kg)', 'Nourriture(kg)', 'Pertes(kg)',
'Production(kg)', 'Semences(kg)', 'Traitement(kg)',
'Variation de stock(kg)'),
dtype='object')

RESTREIGNONS le DataFrame aux colonnes qui nous intéressent :

```
dispo_alimentaire_cereales_USAGES = dispo_alimentaire_cereales[['Zone', 'Produit', 'Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Nourriture(kg)', 'Pertes(kg)', 'Semences(kg)', 'Traitement(kg)']]
dispo_alimentaire_cereales_USAGES.head()
```

Zone	Produit	Aliments pour animaux(kg)	Autres Utilisations(kg)	Nourriture(kg)	Pertes(kg)	Semences(kg)	Traitement(kg)
7	Afghanistan	Blé	0	0	4895000000	775000000	322000000
12	Afghanistan	Céréales, Autres	0	0	0	0	0
32	Afghanistan	Maïs	200000000	0	76000000	31000000	5000000
34	Afghanistan	Millet	0	0	12000000	1000000	0

Liste_col_utilisation_dispo

```
['Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Nourriture(kg)', 'Pertes(kg)', 'Semences(kg)', 'Traitement(kg)']
```

```
#Affichage de la proportion d'alimentation animale
df_cereales_1 = dispo_alimentaire_cereales_USAGES[Liste_col_utilisation_dispo].sum()
df_cereales_1.head()
```

```
Aliments pour animaux(kg)      873535000000
Autres Utilisations(kg)        234787000000
Nourriture(kg)                1029010000000
Pertes(kg)                    107120000000
Semences(kg)                  68538000000
dtype: Int64
```

df_cereales_1.sum()

```
2407579000000
```

LECTURE DIRECTE:

```
df_cereales_2 = pd.DataFrame(df_cereales_1,columns = ['Volumes_cereales_kg'])
display(df_cereales_2)
```

Volumes_cereales_kg	
Aliments pour animaux(kg)	873535000000
Autres Utilisations(kg)	234787000000
Nourriture(kg)	1029010000000
Pertes(kg)	107120000000
Semences(kg)	68538000000
Traitement(kg)	94589000000

```
df_cereales_3 = df_cereales_2.transpose()
display(df_cereales_3)
```

	Aliments pour animaux(kg)	Autres Utilisations(kg)	Nourriture(kg)	Pertes(kg)	Semences(kg)	Traitement(kg)
Volumes_cereales_kg	873535000000	234787000000	1029010000000	107120000000	68538000000	94589000000

```
Somme_2=0
for elt in df_cereales_3.columns.tolist() :
    Somme_2+=df_cereales_3.loc['Volumes_cereales_kg',elt]
```

```
print("Somme total des Volumes de céréales en 2017 est de ",Somme_2,"kg")
```

```
Somme_2=0
for elt in df_cereales_3.columns.tolist() :
    Somme_2+=df_cereales_3.loc['Volumes_cereales_kg',elt]
```

```
print ("En 2017, dans le fichier dispo_alimentaire :")
```

```

for elt in df_cereales_3.columns.tolist() :
    print("l'utilisation des céréales pour ",elt," représente :")
    print("\t un volume de ",round(df_cereales_3.loc['Volumes_cereales_kg',elt] /(1_000_0000*1_000),3),
          " millions de tonnes.")
    print("\t un part de ",round((df_cereales_3.loc['Volumes_cereales_kg',elt] *100)/Somme_2,2),
          "% de la disponibilité intérieure.")
    print()

→ En 2017, dans le fichier dispo_alimentaire :
    l'utilisation des céréales pour Aliments pour animaux(kg) représente :
        un volume de 87.354 millions de tonnes.
        un part de 36.28 % de la disponibilité intérieure.

    l'utilisation des céréales pour Autres Utilisations(kg) représente :
        un volume de 23.479 millions de tonnes.
        un part de 9.75 % de la disponibilité intérieure.

    l'utilisation des céréales pour Nourriture(kg) représente :
        un volume de 102.901 millions de tonnes.
        un part de 42.74 % de la disponibilité intérieure.

    l'utilisation des céréales pour Pertes(kg) représente :
        un volume de 10.712 millions de tonnes.
        un part de 4.45 % de la disponibilité intérieure.

    l'utilisation des céréales pour Sémenices(kg) représente :
        un volume de 6.854 millions de tonnes.
        un part de 2.85 % de la disponibilité intérieure.

    l'utilisation des céréales pour Traitement(kg) représente :
        un volume de 9.459 millions de tonnes.
        un part de 3.93 % de la disponibilité intérieure.

```

▼ VISUALISATION

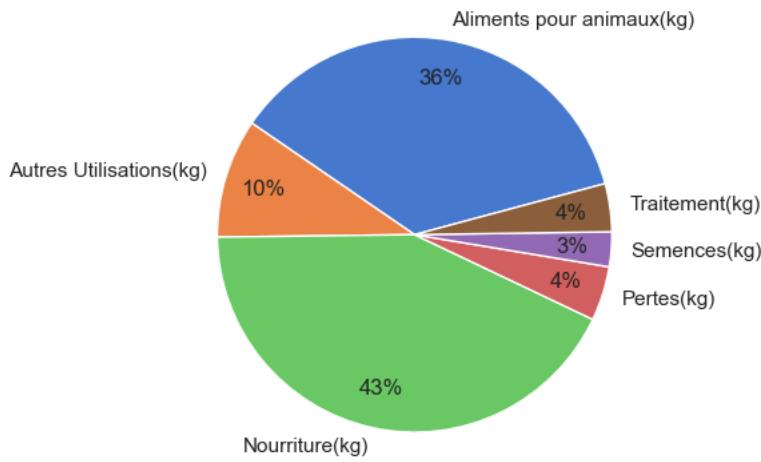
```

plt.pie(x= df_cereales_2['Volumes_cereales_kg'],
       labels=df_cereales_2.index, autopct='%.0f%%',
       pctdistance =0.8,
       startangle=15
       )

plt.title("Repartition de l'utilisation de la disponibilité intérieure \ndes principales céréales",
          fontdict = {'fontsize' : 16, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```

→ Repartition de l'utilisation de la disponibilité intérieure des principales céréales



▼ 4. Affichage de la proportion d'alimentation animale

DOUBLON

```
#Affichage de la proportion d'alimentation animale
```

▼ PM 5. Affichage comparatif Usage Animal/humain pour les céréales

REPRENONS La disponibilité alimentaire RESTREINTE AUX CEREALES :

```
dispo_alimentaire_cereales_USAGES.head()
```

	Zone	Produit	Aliments pour animaux(kg)	Autres Utilisations(kg)	Nourriture(kg)	Pertes(kg)	Semences(kg)	Traitement(kg)
7	Afghanistan	Blé	0	0	4895000000	775000000	322000000	0
12	Afghanistan	Céréales, Autres	0	0	0	0	0	0
32	Afghanistan	Maïs	200000000	0	76000000	31000000	5000000	0
34	Afghanistan	Millet	0	0	12000000	1000000	0	0

```
dispo_alimentaire_cereales_USAGES.columns
```

```
Index(['Zone', 'Produit', 'Aliments pour animaux(kg)',  
       'Autres Utilisations(kg)', 'Nourriture(kg)', 'Pertes(kg)',  
       'Semences(kg)', 'Traitement(kg)'],  
      dtype='object')
```

▼ REGROUPONS PAR PRODUIT et CONSERVONS UNIQUEMENT LES COLONNES UTILES

```
df_cereales_comparaison_ANIM_HOMME_2 = dispo_alimentaire_cereales_USAGES[['Produit',  
                           'Aliments pour animaux(kg)',  
                           'Nourriture(kg)']]  
df_cereales_comparaison_ANIM_HOMME_2 = df_cereales_comparaison_ANIM_HOMME_2.groupby('Produit').sum()
```

```
display(df_cereales_comparaison_ANIM_HOMME_2)
```

Produit	Aliments pour animaux(kg)	Nourriture(kg)
Avoine	16251000000	3903000000
Blé	129668000000	457824000000
Céréales, Autres	19035000000	5324000000
Maïs	546116000000	125184000000
Millet	3306000000	23040000000
Orge	92658000000	6794000000
Riz (Eq Blanchi)	33594000000	377286000000
Seigle	8099000000	5502000000
Sorgho	24808000000	24153000000

▼ REFORMATONS LE FICHIER POUR LECTURE ET VISUALISATION

```
df_cereales_comparaison_ANIM_HOMME_2.reset_index(inplace=True)  
df_cereales_comparaison_ANIM_HOMME_2
```

	Produit	Aliments pour animaux(kg)	Nourriture(kg)
0	Avoine	16251000000	3903000000
1	Blé	129668000000	457824000000
2	Céréales, Autres	19035000000	5324000000
3	Maïs	546116000000	125184000000
4	Millet	3306000000	23040000000
5	Orge	92658000000	6794000000
6	Riz (Eq Blanchi)	33594000000	377286000000
7	Seigle	8099000000	5502000000
8	Sorgho	24808000000	24153000000

```
df_cereales_comparaison_ANIM_HOMME_3 = df_cereales_comparaison_ANIM_HOMME_2.melt(id_vars='Produit',  
                           value_vars=['Aliments pour animaux(kg)', 'Nourriture(kg)'])
```

```
display(df_cereales_comparaison_ANIM_HOMME_3)
```

	Produit	variable	value
0	Avoine	Aliments pour animaux(kg)	16251000000
1	Blé	Aliments pour animaux(kg)	129668000000
2	Céréales, Autres	Aliments pour animaux(kg)	19035000000
3	Maïs	Aliments pour animaux(kg)	546116000000
4	Millet	Aliments pour animaux(kg)	33060000000
5	Orge	Aliments pour animaux(kg)	92658000000
6	Riz (Eq Blanchi)	Aliments pour animaux(kg)	33594000000
7	Seigle	Aliments pour animaux(kg)	8099000000
8	Sorgho	Aliments pour animaux(kg)	24808000000
9	Avoine	Nourriture(kg)	3903000000
10	Blé	Nourriture(kg)	457824000000
11	Céréales, Autres	Nourriture(kg)	5324000000
12	Maïs	Nourriture(kg)	125184000000
13	Millet	Nourriture(kg)	23040000000
14	Orge	Nourriture(kg)	6794000000
15	Riz (Eq Blanchi)	Nourriture(kg)	377286000000
16	Seigle	Nourriture(kg)	5502000000
17	Sorgho	Nourriture(kg)	24153000000

```
df_cereales_comparaison_ANIM_HOMME_3.rename( columns ={'variable':'Usage','value':'Valeur'}, inplace=True)  
display(df_cereales_comparaison_ANIM_HOMME_3)
```

	Produit	Usage	Valeur
0	Avoine	Aliments pour animaux(kg)	16251000000
1	Blé	Aliments pour animaux(kg)	129668000000
2	Céréales, Autres	Aliments pour animaux(kg)	19035000000
3	Maïs	Aliments pour animaux(kg)	546116000000
4	Millet	Aliments pour animaux(kg)	33060000000
5	Orge	Aliments pour animaux(kg)	92658000000
6	Riz (Eq Blanchi)	Aliments pour animaux(kg)	33594000000
7	Seigle	Aliments pour animaux(kg)	8099000000
8	Sorgho	Aliments pour animaux(kg)	24808000000
9	Avoine	Nourriture(kg)	3903000000
10	Blé	Nourriture(kg)	457824000000
11	Céréales, Autres	Nourriture(kg)	5324000000
12	Maïs	Nourriture(kg)	125184000000
13	Millet	Nourriture(kg)	23040000000
14	Orge	Nourriture(kg)	6794000000
15	Riz (Eq Blanchi)	Nourriture(kg)	377286000000
16	Seigle	Nourriture(kg)	5502000000
17	Sorgho	Nourriture(kg)	24153000000

```
#Passage à une colonne Valeur_Millions_tonnes  
df_cereales_comparaison_ANIM_HOMME_3['Valeur_Millions_tonnes']=df_cereales_comparaison_ANIM_HOMME_3['Valeur']/(1_000_000 * 1_000)  
del df_cereales_comparaison_ANIM_HOMME_3['Valeur']  
df_cereales_comparaison_ANIM_HOMME_3
```

	Produit	Usage	Valeur_Millions_tonnes
0	Avoine	Aliments pour animaux(kg)	16.251
1	Blé	Aliments pour animaux(kg)	129.668
2	Céréales, Autres	Aliments pour animaux(kg)	19.035
3	Maïs	Aliments pour animaux(kg)	546.116
4	Millet	Aliments pour animaux(kg)	3.306
5	Orge	Aliments pour animaux(kg)	92.658
6	Riz (Eq Blanchi)	Aliments pour animaux(kg)	33.594
7	Seigle	Aliments pour animaux(kg)	8.099
8	Sorgho	Aliments pour animaux(kg)	24.808
9	Avoine	Nourriture(kg)	3.903
10	Blé	Nourriture(kg)	457.824
11	Céréales, Autres	Nourriture(kg)	5.324
12	Maïs	Nourriture(kg)	125.184
13	Millet	Nourriture(kg)	23.04
14	Orge	Nourriture(kg)	6.794
15	Riz (Eq Blanchi)	Nourriture(kg)	377.286
16	Seigle	Nourriture(kg)	5.502
17	Sorgho	Nourriture(kg)	24.153

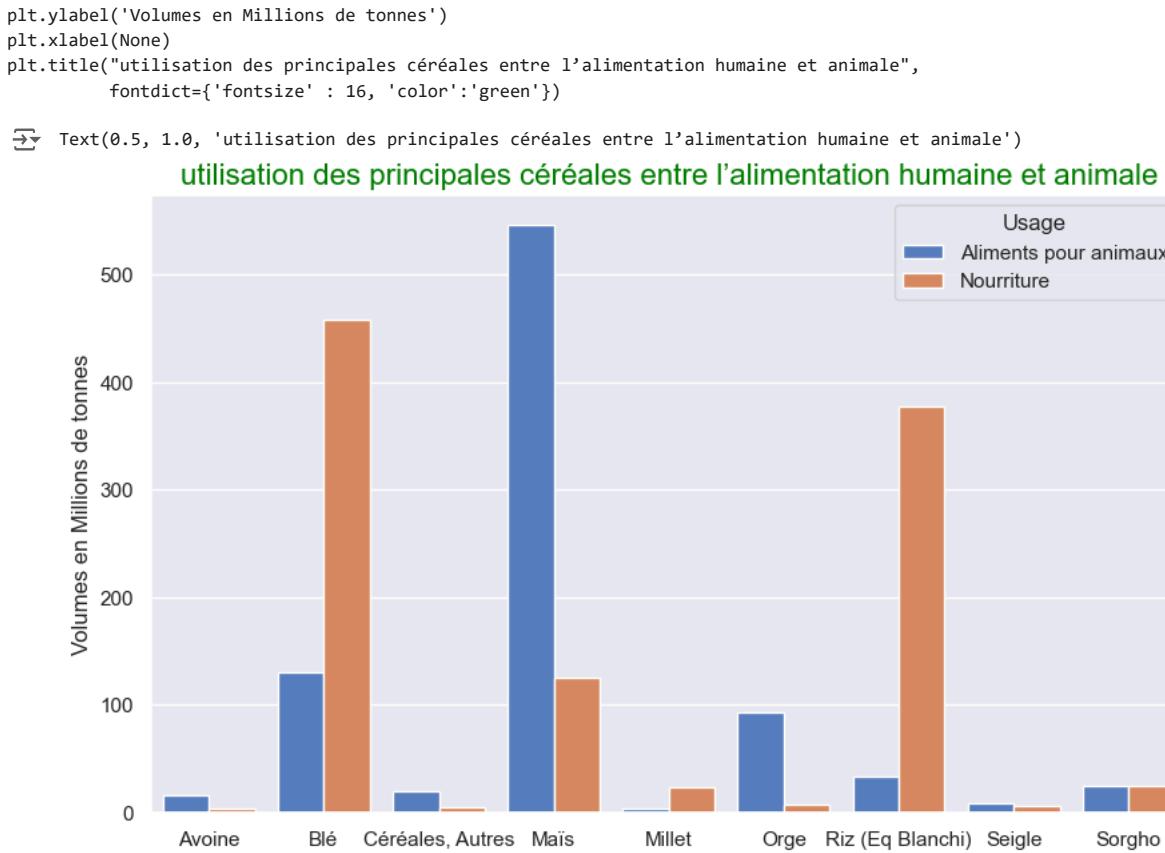
```
#conversion des labels de 'Usage'
df_cereales_comparaison_ANIM_HOMME_3.loc[ df_cereales_comparaison_ANIM_HOMME_3['Usage']=='Aliments pour animaux(kg)' , 'Usage' ] = 'Aliment'
df_cereales_comparaison_ANIM_HOMME_3.loc[ df_cereales_comparaison_ANIM_HOMME_3['Usage']=='Nourriture(kg)' , 'Usage' ] = 'Nourriture'
df_cereales_comparaison_ANIM_HOMME_3
```

	Produit	Usage	Valeur_Millions_tonnes
0	Avoine	Aliments pour animaux	16.251
1	Blé	Aliments pour animaux	129.668
2	Céréales, Autres	Aliments pour animaux	19.035
3	Maïs	Aliments pour animaux	546.116
4	Millet	Aliments pour animaux	3.306
5	Orge	Aliments pour animaux	92.658
6	Riz (Eq Blanchi)	Aliments pour animaux	33.594
7	Seigle	Aliments pour animaux	8.099
8	Sorgho	Aliments pour animaux	24.808
9	Avoine	Nourriture	3.903
10	Blé	Nourriture	457.824
11	Céréales, Autres	Nourriture	5.324
12	Maïs	Nourriture	125.184
13	Millet	Nourriture	23.04
14	Orge	Nourriture	6.794
15	Riz (Eq Blanchi)	Nourriture	377.286
16	Seigle	Nourriture	5.502
17	Sorgho	Nourriture	24.153

▼ VISUALISATION

```
plt.figure(figsize=(10,6))
sns.set_theme(style='dark', palette='muted')

sns.barplot(data=df_cereales_comparaison_ANIM_HOMME_3,
            x='Produit',y='Valeur_Millions_tonnes',
            errorbar=None,hue='Usage')
plt.grid(axis='y')
```



```

plt.figure(figsize=(10,6))
sns.set_theme(style='dark', palette='muted')

sns.barplot(data=df_cereales_comparaison_ANIM_HOMME_3,
            x='Produit',y='Valeur_Millions_tonnes',
            errorbar=None,hue='Usage')

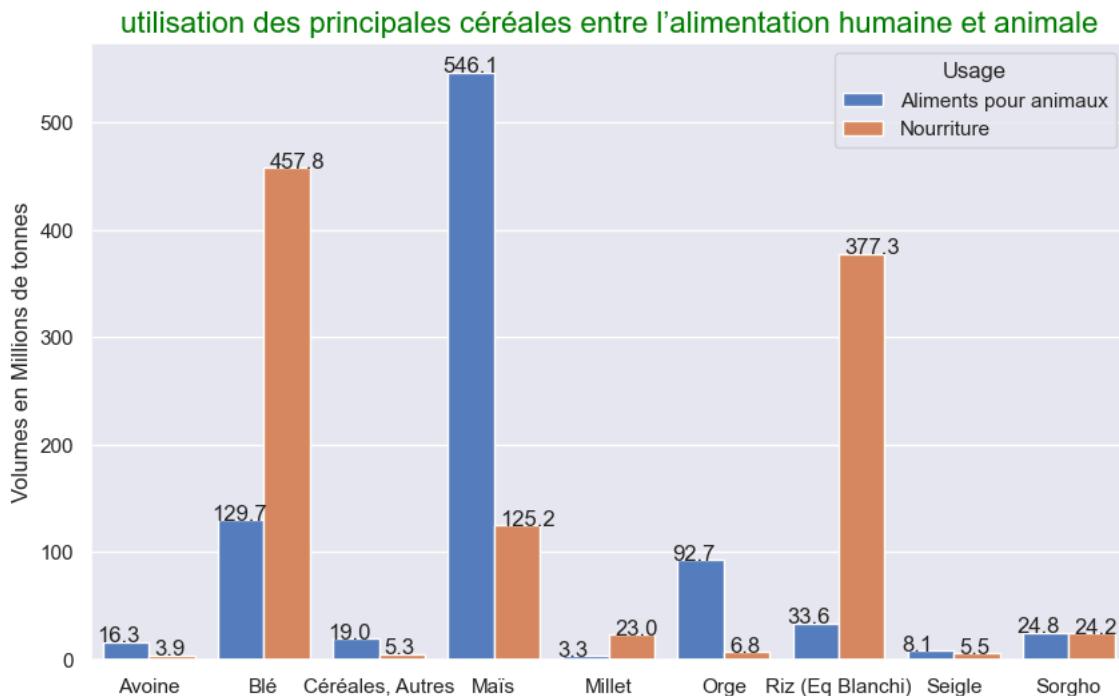
Valeurs_animaux = df_cereales_comparaison_ANIM_HOMME_3.loc[0:8,'Valeur_Millions_tonnes'].tolist()
for i in range (0,len(Valeurs_animaux)) :
    plt.text(i-0.45, Valeurs_animaux[i],round(Valeurs_animaux[i],1))

Valeurs_hommes = df_cereales_comparaison_ANIM_HOMME_3.loc[9:17,'Valeur_Millions_tonnes'].tolist()
for i in range (0,len(Valeurs_hommes)) :
    plt.text(i+0.05, Valeurs_hommes[i],round(Valeurs_hommes[i],1))

plt.grid(axis='y')
plt.ylabel('Volumes en Millions de tonnes')
plt.xlabel(None)
plt.title("utilisation des principales céréales entre l'alimentation humaine et animale",
          fontdict={'fontsize' : 16, 'color':'green'})

```

➡ Text(0.5, 1.0, 'utilisation des principales céréales entre l'alimentation humaine et animale')



CONCLUSION :

3 céréales ont les productions les plus importantes.

2 avec une production nettement pour l'humain :

- Blé
- Riz

1 avec une production nettement pour les animaux :

- Maïs

3.6 - Pays avec la proportion de personnes sous-alimentée la plus forte en 2017

▼ 1. Création de la colonne proportion par pays

#Création de la colonne proportion par pays

RAPPEL DU FICHIER contenant la sous-nutrition :

```
#fichier de base
sous_nutrition.head()
```

➡

	Zone	Année	Sous_nutrition
0	Afghanistan	2012-2014	8600000
1	Afghanistan	2013-2015	8800000
2	Afghanistan	2014-2016	8900000
3	Afghanistan	2015-2017	9700000
4	Afghanistan	2016-2018	10500000

```
#Fichiers fusionnés pour 2017
population_avc_sous_nutrition_2017.head()
```

	Zone	Année_x	Population	Année_y	Sous_nutrition
0	Afghanistan	2017	36296113	2016-2018	10500000
1	Afrique du Sud	2017	57009756	2016-2018	3100000
2	Albanie	2017	2884169	2016-2018	100000
3	Algérie	2017	41389189	2016-2018	1300000
4	Allemagne	2017	82658409	2016-2018	0

Retravaillons le fichier

```
population_avc_sous_nutrition_2017.columns
```

```
Index(['Zone', 'Année_x', 'Population', 'Année_y', 'Sous_nutrition'], dtype='object')
```

```
ds_sous_nutrition_2 = population_avc_sous_nutrition_2017[['Zone', 'Année_x', 'Population', 'Sous_nutrition']]
```

```
ds_sous_nutrition_2.rename(columns={'Année_x': 'Année'}, inplace=True)
ds_sous_nutrition_2.head()
```

```
C:\Users\matth\AppData\Local\Temp\ipykernel_21284\670923433.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus

```
ds_sous_nutrition_2.rename(columns={'Année_x': 'Année'}, inplace=True)
```

	Zone	Année	Population	Sous_nutrition
0	Afghanistan	2017	36296113	10500000
1	Afrique du Sud	2017	57009756	3100000
2	Albanie	2017	2884169	100000
3	Algérie	2017	41389189	1300000
4	Allemagne	2017	82658409	0

```
#Création de la colonne proportion par pays
ds_sous_nutrition_2['Proportion_SN_%']=round((ds_sous_nutrition_2['Sous_nutrition']*100)/ds_sous_nutrition_2['Population'],2)
ds_sous_nutrition_2.head()
```

	Zone	Année	Population	Sous_nutrition	Proportion_SN_%
0	Afghanistan	2017	36296113	10500000	28.93
1	Afrique du Sud	2017	57009756	3100000	5.44
2	Albanie	2017	2884169	100000	3.47
3	Algérie	2017	41389189	1300000	3.14
4	Allemagne	2017	82658409	0	0.0

2. affichage après tri des 10 pays avec la proportion sous-nutrie la plus élevée

```
#affichage après tri des 10 pires pays
ds_sous_nutrition_2_trie = ds_sous_nutrition_2.sort_values('Proportion_SN_%', ascending=False)
ds_sous_nutrition_2_trie.head(10)
```

		Zone	Année	Population	Sous_nutrition	Proportion_SN_%
78		Haïti	2017	10982366	5300000	48.26
157	République populaire démocratique de Corée		2017	25429825	12000000	47.19
108		Madagascar	2017	25570512	10500000	41.06
103		Libéria	2017	4702226	1800000	38.28
100		Lesotho	2017	2091534	800000	38.25
183		Tchad	2017	15016753	5700000	37.96
161		Rwanda	2017	11980961	4200000	35.06
121		Mozambique	2017	28649018	9400000	32.81
186		Timor-Leste	2017	1243258	400000	32.17
0		Afghanistan	2017	36296113	10500000	28.93

```
ds_sous_nutrition_3_trie = ds_sous_nutrition_2_trie.head(10)
display(ds_sous_nutrition_3_trie)
```

		Zone	Année	Population	Sous_nutrition	Proportion_SN_%
78		Haïti	2017	10982366	5300000	48.26
157	République populaire démocratique de Corée		2017	25429825	12000000	47.19
108		Madagascar	2017	25570512	10500000	41.06
103		Libéria	2017	4702226	1800000	38.28
100		Lesotho	2017	2091534	800000	38.25
183		Tchad	2017	15016753	5700000	37.96
161		Rwanda	2017	11980961	4200000	35.06
121		Mozambique	2017	28649018	9400000	32.81
186		Timor-Leste	2017	1243258	400000	32.17
0		Afghanistan	2017	36296113	10500000	28.93

Renommons 'République populaire démocratique de Corée' en 'Corée du Nord',
car le premier terme est trop long pour afficher dans un graphique

```
ds_sous_nutrition_3_trie.loc[ds_sous_nutrition_3_trie['Zone']=='République populaire démocratique de Corée' , 'Zone'] = 'Corée du Nord'
```

```
display(ds_sous_nutrition_3_trie)
```

		Zone	Année	Population	Sous_nutrition	Proportion_SN_%
78		Haïti	2017	10982366	5300000	48.26
157	Corée du Nord		2017	25429825	12000000	47.19
108		Madagascar	2017	25570512	10500000	41.06
103		Libéria	2017	4702226	1800000	38.28
100		Lesotho	2017	2091534	800000	38.25
183		Tchad	2017	15016753	5700000	37.96
161		Rwanda	2017	11980961	4200000	35.06
121		Mozambique	2017	28649018	9400000	32.81
186		Timor-Leste	2017	1243258	400000	32.17
0		Afghanistan	2017	36296113	10500000	28.93

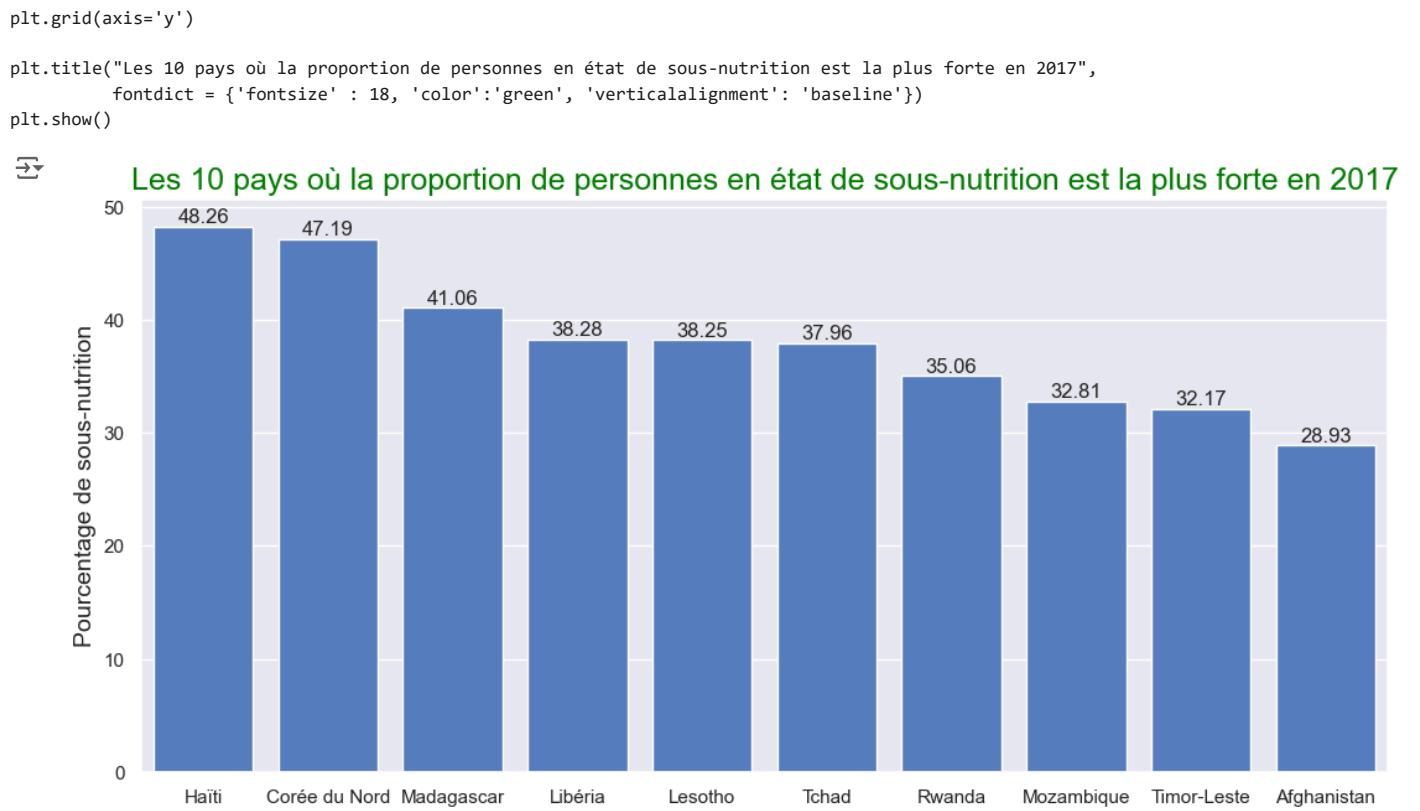
```
plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = ds_sous_nutrition_3_trie , x='Zone', y = 'Proportion_SN_%', errorbar=None)

ax.bar_label(ax.containers[0])

plt.xlabel('')
plt.ylabel('Pourcentage de sous-nutrition',fontdict = {'fontsize' : 14})
```



```

#avec de la couleur (hue) et maintien des ax.bar_label

plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = ds_sous_nutrition_3_trie , x='Zone', y = 'Proportion_SN_%', errorbar=None,hue='Zone')

for x in range(0,10) :
    ax.bar_label(ax.containers[x])

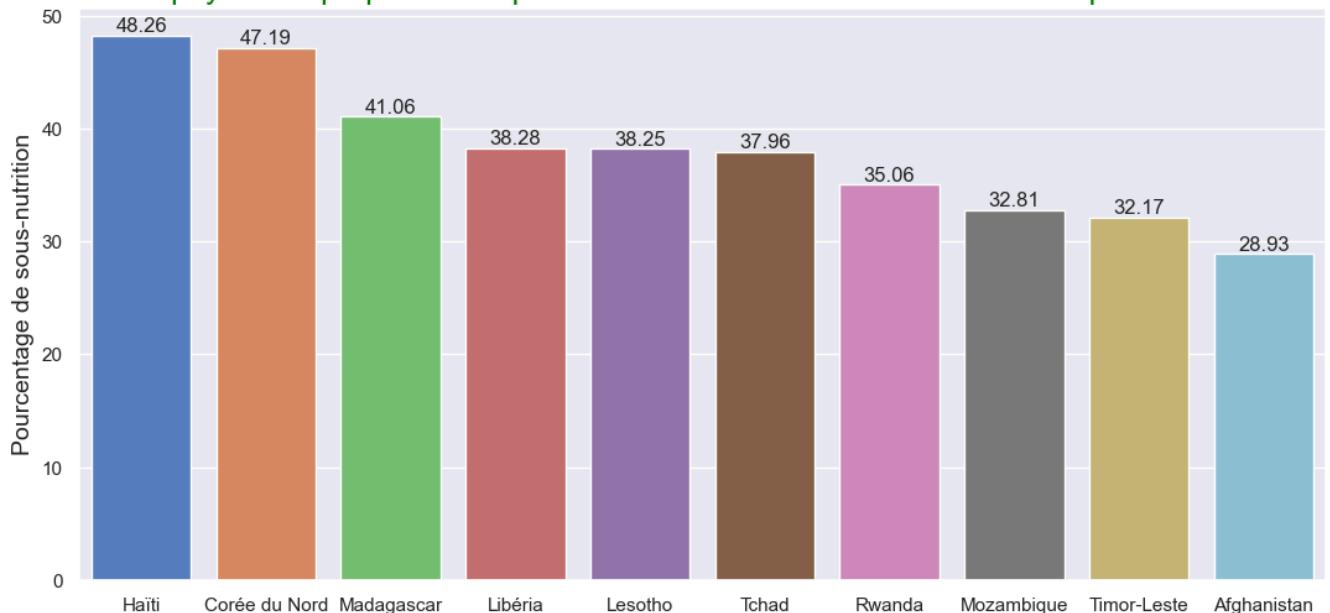
plt.xlabel('')
plt.ylabel('Pourcentage de sous-nutrition', fontdict = {'fontsize' : 14})
plt.grid(axis='y')

plt.title("Les 10 pays où la proportion de personnes en état de sous-nutrition est la plus forte en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```



Les 10 pays où la proportion de personnes en état de sous-nutrition est la plus forte en 2017



3.7 - Pays qui ont le plus bénéficié d'aide alimentaire depuis 2013

On souhaite connaître la liste des 10 pays qui ont le plus bénéficié de l'aide alimentaire entre **2013 et 2016**

▼ 1.calcul du total de l'aide alimentaire par pays

```
#calcul du total de l'aide alimentaire par pays
```

RAPPEL FICHIER AIDE

```
aide_alimentaire.head()
```

	Zone	Année	Produit	Aide_alimentaire_annuelle(kg)
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

```
aide_alimentaire['Année'].unique()
```

```
→ <IntegerArray>
[2013, 2014, 2015, 2016]
Length: 4, dtype: Int64
```

Remarque le jeu de données regroupe uniquement les valeurs de 2013 à 2016 que nous souhaitons étudier.

Nous n'avons donc pas de sélection à faire sur l'année.

```
aide_alimentaire_bilan_1 = aide_alimentaire[ ['Zone', 'Aide_alimentaire_annuelle(kg)'] ].groupby('Zone').sum()
aide_alimentaire_bilan_1.head()
```



Aide_alimentaire_annuelle(kg)

Zone

Afghanistan	185452000
Algérie	81114000
Angola	5014000
Bangladesh	348188000
Bhoutan	2666000

```
aide_alimentaire_bilan_1.reset_index(inplace=True)  
aide_alimentaire_bilan_1.head()
```



Zone Aide_alimentaire_annuelle(kg)

0	Afghanistan	185452000
1	Algérie	81114000
2	Angola	5014000
3	Bangladesh	348188000
4	Bhoutan	2666000

Le cumul est un cumul sur les années 2013 à 2016 et non annuel. Renommons la colonne.

```
aide_alimentaire_bilan_1.rename(columns={'Aide_alimentaire_annuelle(kg)':'Aide_alimentaire_(kg)'},inplace=True)  
aide_alimentaire_bilan_1.head()
```



Zone Aide_alimentaire_(kg)

0	Afghanistan	185452000
1	Algérie	81114000
2	Angola	5014000
3	Bangladesh	348188000
4	Bhoutan	2666000

✓ 2. affichage après tri des 10 pays qui ont bénéficié le plus de l'aide alimentaire

```
#affichage après tri des 10 pays qui ont bénéficié le plus de l'aide alimentaire
```

Préparation du fichier pour affichage :

```
#Mnt=Million de tonnes  
aide_alimentaire_bilan_2 = aide_alimentaire_bilan_1  
aide_alimentaire_bilan_2['Aide_alimentaire_(Mnt)'] = aide_alimentaire_bilan_2['Aide_alimentaire_(kg)']/1_000_000_000  
aide_alimentaire_bilan_2.head()
```



Zone Aide_alimentaire_(kg) Aide_alimentaire_(Mnt)

0	Afghanistan	185452000	0.185452
1	Algérie	81114000	0.081114
2	Angola	5014000	0.005014
3	Bangladesh	348188000	0.348188
4	Bhoutan	2666000	0.002666

```
del aide_alimentaire_bilan_2['Aide_alimentaire_(kg)']  
aide_alimentaire_bilan_2.head()
```

	Zone	Aide_alimentaire_(Mnt)
0	Afghanistan	0.185452
1	Algérie	0.081114
2	Angola	0.005014
3	Bangladesh	0.348188
4	Bhoutan	0.002666

```
aide_alimentaire_bilan_2_trie = aide_alimentaire_bilan_2.sort_values('Aide_alimentaire_(Mnt)', ascending=False)
aide_alimentaire_bilan_2_trie
```

	Zone	Aide_alimentaire_(Mnt)
50	République arabe syrienne	1.858943
75	Éthiopie	1.381294
70	Yémen	1.206484
61	Soudan du Sud	0.695248
60	Soudan	0.669784
...
73	Égypte	0.001122
69	Vanuatu	0.000802
67	Timor-Leste	0.000116
24	Géorgie	0.00007
5	Bolivie (État plurinational de)	0.000006

76 rows × 2 columns

```
aide_alimentaire_bilan_3_trie = aide_alimentaire_bilan_2_trie.head(10)
display(aide_alimentaire_bilan_3_trie)
```

	Zone	Aide_alimentaire_(Mnt)
50	République arabe syrienne	1.858943
75	Éthiopie	1.381294
70	Yémen	1.206484
61	Soudan du Sud	0.695248
60	Soudan	0.669784
30	Kenya	0.552836
3	Bangladesh	0.348188
59	Somalie	0.292678
53	République démocratique du Congo	0.288502
43	Niger	0.276344

```
aide_alimentaire_bilan_3_trie.reset_index(inplace=True)
aide_alimentaire_bilan_3_trie.head()
```

	index	Zone	Aide_alimentaire_(Mnt)
0	50	République arabe syrienne	1.858943
1	75	Éthiopie	1.381294
2	70	Yémen	1.206484
3	61	Soudan du Sud	0.695248
4	60	Soudan	0.669784

```
del aide_alimentaire_bilan_3_trie['index']
display(aide_alimentaire_bilan_3_trie)
```

		Zone	Aide_alimentaire_(Mnt)
0	République arabe syrienne		1.858943
1	Éthiopie		1.381294
2	Yémen		1.206484
3	Soudan du Sud		0.695248
4	Soudan		0.669784
5	Kenya		0.552836
6	Bangladesh		0.348188
7	Somalie		0.292678
8	République démocratique du Congo		0.288502
9	Niger		0.276344

```

aide_alimentaire_bilan_3_trie.loc[aide_alimentaire_bilan_3_trie['Zone']=='République arabe syrienne','Zone'] = 'Syrie'
aide_alimentaire_bilan_3_trie.loc[aide_alimentaire_bilan_3_trie['Zone']=='Soudan du Sud','Zone']= 'Sud-Soudan'
aide_alimentaire_bilan_3_trie.loc[aide_alimentaire_bilan_3_trie['Zone']=='République démocratique du Congo','Zone'] = 'RDC'
display(aide_alimentaire_bilan_3_trie)

```

	Zone	Aide_alimentaire_(Mnt)
0	Syrie	1.858943
1	Éthiopie	1.381294
2	Yémen	1.206484
3	Sud-Soudan	0.695248
4	Soudan	0.669784
5	Kenya	0.552836
6	Bangladesh	0.348188
7	Somalie	0.292678
8	RDC	0.288502
9	Niger	0.276344

```
plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = aide_alimentaire_bilan_3_trie ,
                  x='Zone',
                  y = round (aide_alimentaire_bilan_3_trie['Aide_alimentaire_(Mnt)'] , 3),
                  errorbar=None)

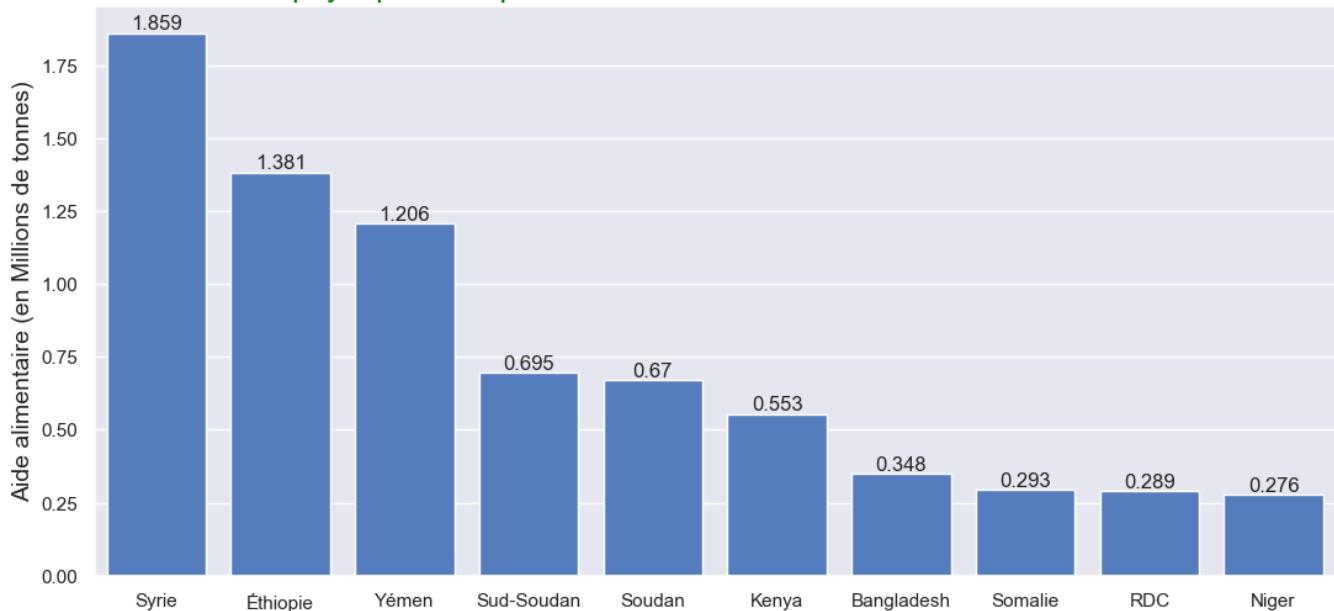
ax.bar_label(ax.containers[0])

plt.xlabel('')
plt.ylabel('Aide alimentaire (en Millions de tonnes)', fontdict = {'fontsize' : 14})
plt.grid(axis='y')

plt.title("Les 10 pays qui ont le plus bénéficié de l'aide alimentaire entre 2013 et 2016",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```



Les 10 pays qui ont le plus bénéficié de l'aide alimentaire entre 2013 et 2016



```
#avec de la couleur (hue) et maintien des ax.bar_label
```

```
plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = aide_alimentaire_bilan_3_trie ,
                  x='Zone',
                  y = round (aide_alimentaire_bilan_3_trie['Aide_alimentaire_(Mnt)'] , 3),
                  errorbar=None,hue='Zone')

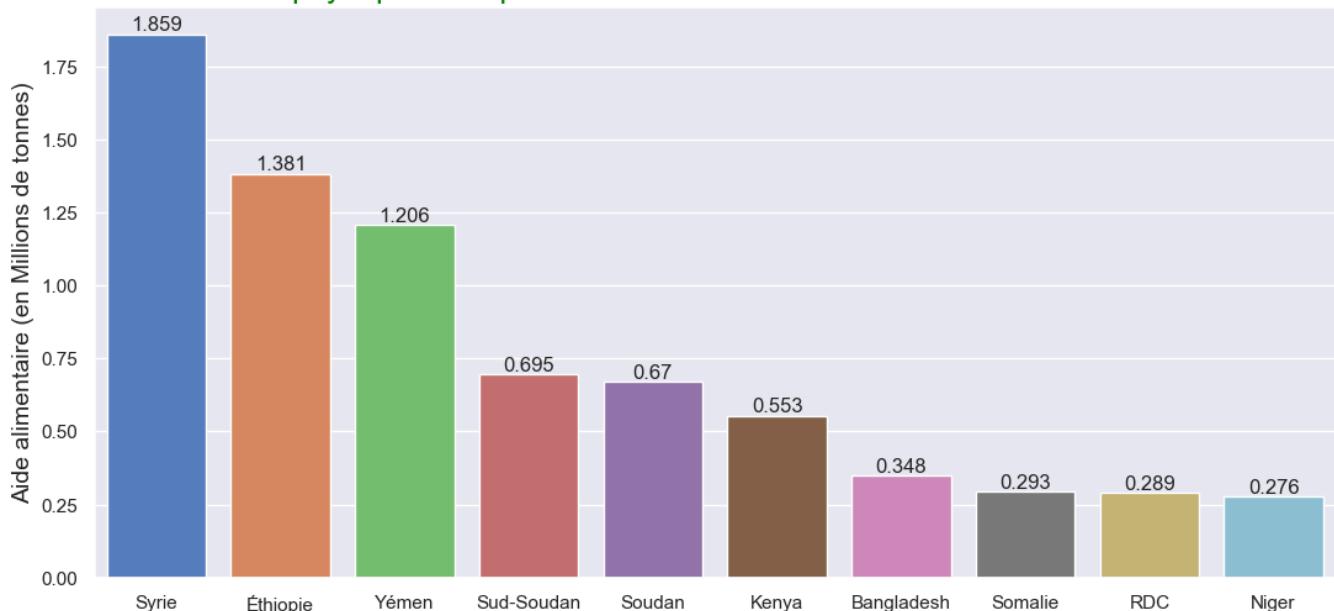
for x in range(0,10) :
    ax.bar_label(ax.containers[x])

plt.xlabel('')
plt.ylabel('Aide alimentaire (en Millions de tonnes)', fontdict = {'fontsize' : 14})
plt.grid(axis='y')

plt.title("Les 10 pays qui ont le plus bénéficié de l'aide alimentaire entre 2013 et 2016",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```



Les 10 pays qui ont le plus bénéficié de l'aide alimentaire entre 2013 et 2016



CONCLUSION :

On peut noter que :

- la plupart des pays ayant reçu de l'aide sont des pays qui ont connu des Guerres (Syrie, Ethiopie, Yémen, Sud-Soudan, Soudan)
- Ce ne sont pas forcément les pays les moins bien nutris

3.8 - Evolution des 5 pays qui ont le plus bénéficiés de l'aide alimentaire entre 2013 et 2016

✓ 1. Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année

```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année
```

```
aide_alimentaire.head()
```

	Zone	Année	Produit	Aide_alimentaire_annuelle(kg)
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

```
aide_alimentaire.columns
```

```
Index(['Zone', 'Année', 'Produit', 'Aide_alimentaire_annuelle(kg)'], dtype='object')
```

```
aide_alimentaire_bilan_annuelle_A = aide_alimentaire[['Zone', 'Année', 'Aide_alimentaire_annuelle(kg)']].groupby(['Zone', 'Année']).sum()
aide_alimentaire_bilan_annuelle_A.head()
```

	Aide_alimentaire_annuelle(kg)	
	Zone	Année
Afghanistan	2013	128238000
	2014	57214000
Algérie	2013	35234000
	2014	18980000
	2015	17424000

```
aide_alimentaire_bilan_annuelle_A.reset_index(inplace=True)
aide_alimentaire_bilan_annuelle_A.head()
```

	Zone	Année	Aide_alimentaire_annuelle(kg)
0	Afghanistan	2013	128238000
1	Afghanistan	2014	57214000
2	Algérie	2013	35234000
3	Algérie	2014	18980000
4	Algérie	2015	17424000

✓ 2. Création d'une liste contenant les 5 pays qui ont le plus bénéficiés de l'aide alimentaire

```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficiés de l'aide alimentaire
Liste_Top5_Pays_ben_aide_alim = ['République arabe syrienne', 'Éthiopie', 'Yémen', 'Soudan du Sud', 'Soudan']
```

✓ 3. On filtre sur le dataframe avec notre liste

```
#On filtre sur le dataframe avec notre liste COMPLEXE
aide_alimentaire_bilan_annuelle_A = aide_alimentaire_bilan_annuelle_A.loc[ aide_alimentaire_bilan_annuelle_A['Zone'].isin(Liste_Top5_Pays_ben_aide_alim) ]
display(aide_alimentaire_bilan_annuelle_A)
```

⤵

	Zone	Année	Aide_alimentaire_annuelle(kg)
157	République arabe syrienne	2013	563566000
158	République arabe syrienne	2014	651870000
159	République arabe syrienne	2015	524949000
160	République arabe syrienne	2016	118558000
189	Soudan	2013	330230000
190	Soudan	2014	321904000
191	Soudan	2015	17650000
192	Soudan du Sud	2013	196330000
193	Soudan du Sud	2014	450610000
194	Soudan du Sud	2015	48308000
214	Yémen	2013	264764000
215	Yémen	2014	103840000
216	Yémen	2015	372306000
217	Yémen	2016	465574000
225	Éthiopie	2013	591404000
226	Éthiopie	2014	586624000
227	Éthiopie	2015	203266000

◀ ▶

⌄ 4. Affichage des pays avec l'aide alimentaire par année

```
# Affichage des pays avec l'aide alimentaire par année
```

MODIFICATION DES DONNEES POUR AFFICHAGE

```
#Kt=Milliers de tonnes
aide_alimentaire_bilan_annuelle_A['Aide_alimentaire_annuelle(Kt)']=aide_alimentaire_bilan_annuelle_A['Aide_alimentaire_annuelle(kg)']/1000
aide_alimentaire_bilan_annuelle_A.head()
```

⤵

	Zone	Année	Aide_alimentaire_annuelle(kg)	Aide_alimentaire_annuelle(Kt)
157	République arabe syrienne	2013	563566000	563.566
158	République arabe syrienne	2014	651870000	651.87
159	République arabe syrienne	2015	524949000	524.949
160	République arabe syrienne	2016	118558000	118.558
189	Soudan	2013	330230000	330.23

◀ ▶

```
del aide_alimentaire_bilan_annuelle_A['Aide_alimentaire_annuelle(kg)']
display(aide_alimentaire_bilan_annuelle_A)
```

	Zone	Année	Aide_alimentaire_annuelle(Kt)
157	République arabe syrienne	2013	563.566
158	République arabe syrienne	2014	651.87
159	République arabe syrienne	2015	524.949
160	République arabe syrienne	2016	118.558
189	Soudan	2013	330.23
190	Soudan	2014	321.904
191	Soudan	2015	17.65
192	Soudan du Sud	2013	196.33
193	Soudan du Sud	2014	450.61
194	Soudan du Sud	2015	48.308
214	Yémen	2013	264.764
215	Yémen	2014	103.84
216	Yémen	2015	372.306
217	Yémen	2016	465.574
225	Éthiopie	2013	591.404
226	Éthiopie	2014	586.624
227	Éthiopie	2015	203.266

```
aide_alimentaire_bilan_annuelle_A.reset_index(inplace=True)
```

```
display(aide_alimentaire_bilan_annuelle_A)
```

	index	Zone	Année	Aide_alimentaire_annuelle(Kt)
0	157	République arabe syrienne	2013	563.566
1	158	République arabe syrienne	2014	651.87
2	159	République arabe syrienne	2015	524.949
3	160	République arabe syrienne	2016	118.558
4	189	Soudan	2013	330.23
5	190	Soudan	2014	321.904
6	191	Soudan	2015	17.65
7	192	Soudan du Sud	2013	196.33
8	193	Soudan du Sud	2014	450.61
9	194	Soudan du Sud	2015	48.308
10	214	Yémen	2013	264.764
11	215	Yémen	2014	103.84
12	216	Yémen	2015	372.306
13	217	Yémen	2016	465.574
14	225	Éthiopie	2013	591.404
15	226	Éthiopie	2014	586.624
16	227	Éthiopie	2015	203.266

```
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='République arabe syrienne','Zone'] = 'Syrie'
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Soudan du Sud','Zone'] = 'Sud-Soudan'
display(aide_alimentaire_bilan_annuelle_A)
```

	index	Zone	Année	Aide_alimentaire_annuelle(Kt)
0	157	Syrie	2013	563.566
1	158	Syrie	2014	651.87
2	159	Syrie	2015	524.949
3	160	Syrie	2016	118.558
4	189	Soudan	2013	330.23
5	190	Soudan	2014	321.904
6	191	Soudan	2015	17.65
7	192	Sud-Soudan	2013	196.33
8	193	Sud-Soudan	2014	450.61
9	194	Sud-Soudan	2015	48.308
10	214	Yémen	2013	264.764
11	215	Yémen	2014	103.84
12	216	Yémen	2015	372.306
13	217	Yémen	2016	465.574
14	225	Éthiopie	2013	591.404
15	226	Éthiopie	2014	586.624
16	227	Éthiopie	2015	203.266

Afin de garder une cohérence on va manuellement les classer dans l'ordre du Top5 :

1'Syrie', 2'Éthiopie', 3'Yémen', 4'Sud-Soudan', 5'Soudan'

(Remarque : a priori on aurait pu le faire dans le graphique avec order= et hueorder=)

```
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Syrie','index'] = 1
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Éthiopie','index'] = 2
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Yémen','index'] = 3
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Sud-Soudan','index'] = 4
aide_alimentaire_bilan_annuelle_A.loc[aide_alimentaire_bilan_annuelle_A['Zone']=='Soudan','index'] = 5
display(aide_alimentaire_bilan_annuelle_A)
```

	index	Zone	Année	Aide_alimentaire_annuelle(Kt)
0	1	Syrie	2013	563.566
1	1	Syrie	2014	651.87
2	1	Syrie	2015	524.949
3	1	Syrie	2016	118.558
4	5	Soudan	2013	330.23
5	5	Soudan	2014	321.904
6	5	Soudan	2015	17.65
7	4	Sud-Soudan	2013	196.33
8	4	Sud-Soudan	2014	450.61
9	4	Sud-Soudan	2015	48.308
10	3	Yémen	2013	264.764
11	3	Yémen	2014	103.84
12	3	Yémen	2015	372.306
13	3	Yémen	2016	465.574
14	2	Éthiopie	2013	591.404
15	2	Éthiopie	2014	586.624
16	2	Éthiopie	2015	203.266

```
aide_alimentaire_bilan_annuelle_A.sort_values(by=[ 'index', 'Année'],inplace=True)
display(aide_alimentaire_bilan_annuelle_A)
```

	index	Zone	Année	Aide_alimentaire_annuelle(Kt)
0	1	Syrie	2013	563.566
1	1	Syrie	2014	651.87
2	1	Syrie	2015	524.949
3	1	Syrie	2016	118.558
14	2	Éthiopie	2013	591.404
15	2	Éthiopie	2014	586.624
16	2	Éthiopie	2015	203.266
10	3	Yémen	2013	264.764
11	3	Yémen	2014	103.84
12	3	Yémen	2015	372.306
13	3	Yémen	2016	465.574
7	4	Sud-Soudan	2013	196.33
8	4	Sud-Soudan	2014	450.61
9	4	Sud-Soudan	2015	48.308
4	5	Soudan	2013	330.23
5	5	Soudan	2014	321.904
6	5	Soudan	2015	17.65

```

plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

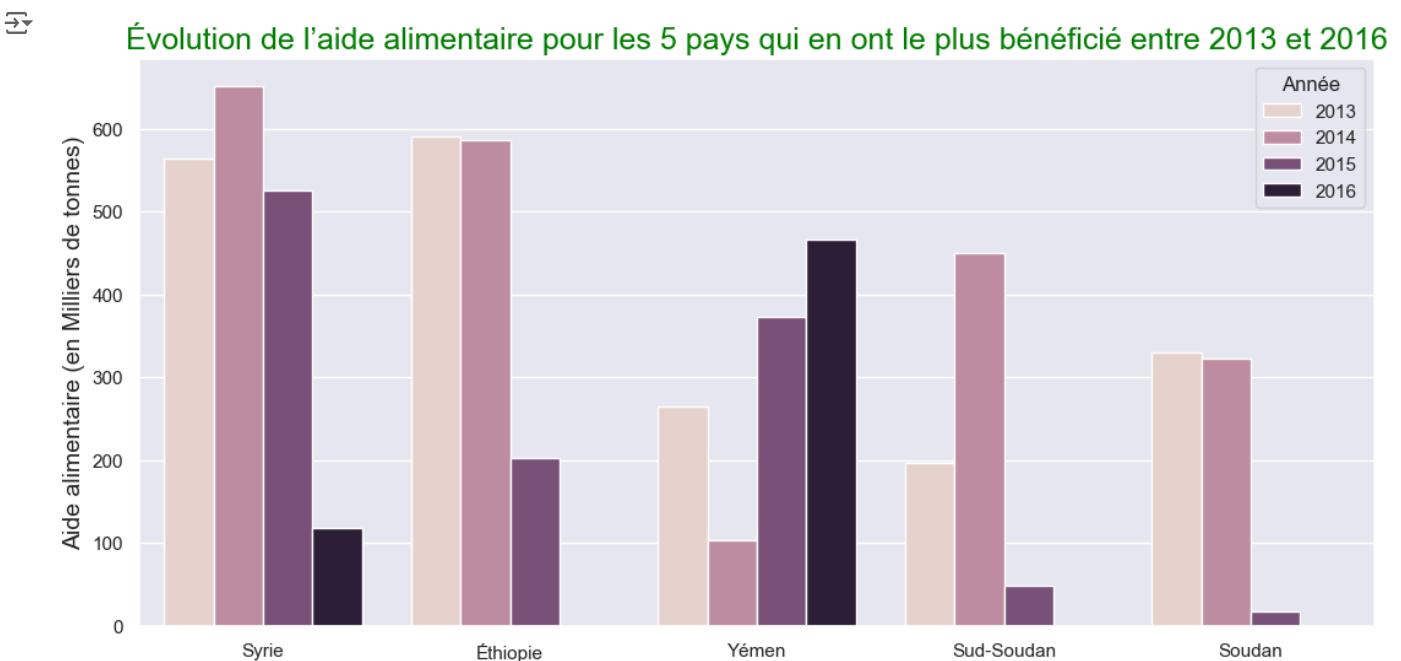
ax = sns.barplot(data = aide_alimentaire_bilan_annuelle_A , x='Zone', y = 'Aide_alimentaire_annuelle(Kt)', errorbar=None,hue='Année')

#ax.bar_label(ax.containers[0])

plt.xlabel('')
plt.ylabel('Aide alimentaire (en Milliers de tonnes)',fontdict = {'fontsize' : 14})
plt.grid(axis='y')

plt.title("Évolution de l'aide alimentaire pour les 5 pays qui en ont le plus bénéficié entre 2013 et 2016",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```



CONCLUSIONS

Les dynamiques d'aide alimentaire fluctuent :

- pour 4 pays l'aide alimentaire a baissé en 2015 et 2016.

- pour le Yémen en revanche l'aide a augmenté en 2015 et 2016.

Il s'agit de fluctuations liées à des crises alimentaires ou événements géopolitiques (guerres, Syrie en 2011, Sud-Soudan en 2013, Yémen en 2014)

▼ VISUALISATION de l'EVOLUTION AVEC DES COURBES

```
plt.figure(figsize=(13,6))

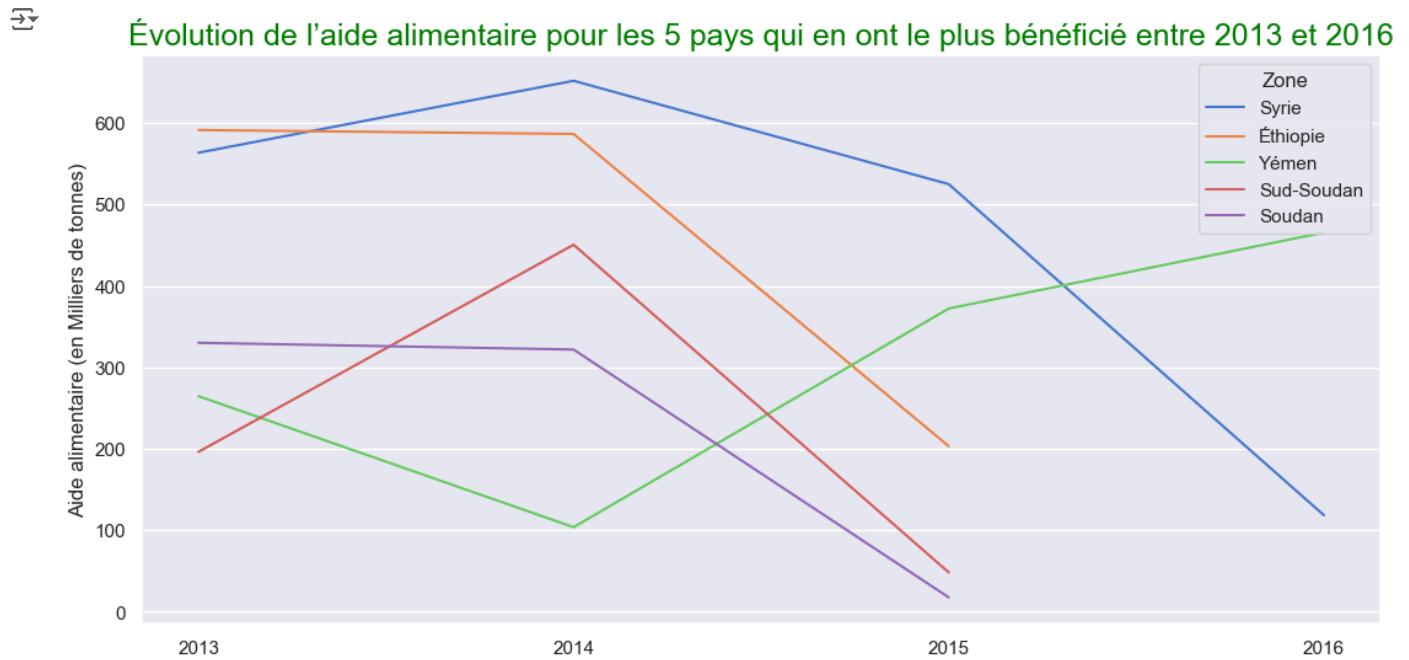
sns.set_theme(style='dark', palette='muted')

ax = sns.lineplot(data = aide_alimentaire_bilan_annuelle_A , x='Année', y = 'Aide_alimentaire_annuelle(Kt)', errorbar=None,hue='Zone')

#ax.bar_label(ax.containers[0])

plt.xlabel('')
plt.ylabel('Aide alimentaire (en Milliers de tonnes)')
plt.grid(axis='y')
plt.xticks([2013,2014,2015,2016])

plt.title("Évolution de l'aide alimentaire pour les 5 pays qui en ont le plus bénéficié entre 2013 et 2016",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```



```
plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.lineplot(data = aide_alimentaire_bilan_annuelle_A , x='Année', y = 'Aide_alimentaire_annuelle(Kt)', errorbar=None,hue='Zone')

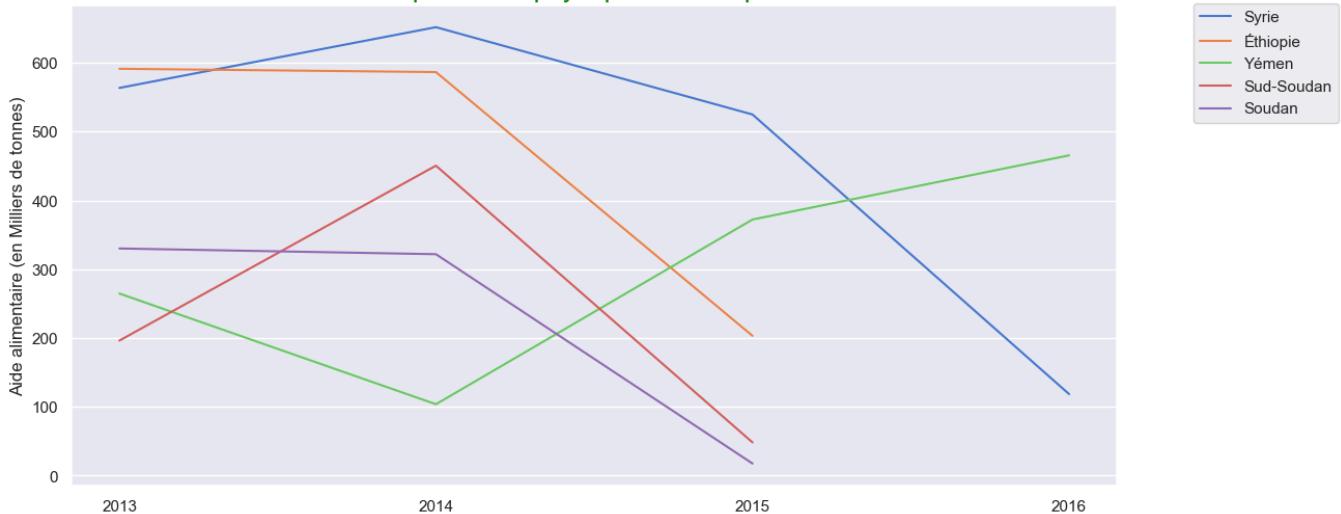
#ax.bar_label(ax.containers[0])

plt.xlabel('')
plt.ylabel('Aide alimentaire (en Milliers de tonnes)')
plt.grid(axis='y')
plt.xticks([2013,2014,2015,2016])
plt.legend(bbox_to_anchor=(1.22,1.02))

plt.title("Évolution de l'aide alimentaire pour les 5 pays qui en ont le plus bénéficié entre 2013 et 2016",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```



Évolution de l'aide alimentaire pour les 5 pays qui en ont le plus bénéficié entre 2013 et 2016



3.9 - Pays avec le moins de disponibilité par habitant

▼ 1. Calcul de la disponibilité en kcal par personne par jour par pays

```
#Calcul de la disponibilité en kcal par personne par jour par pays
```

RAPPEL DU FICHIER :

Fichier pour une seule année (2017)

Nous avons déjà par ligne la disponibilité en kcal par personne par jour (mais pas par pays) : Disponibilité alimentaire (Kcal/personne/jour)

```
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Utilisations(kg)	Autres	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité de pr
0	Afghanistan	Abats Comestible	animale	0	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	0	4	2.7	0.02	

Il nous faut donc simplement rassembler par pays :

```
dispo_alimentaire_par_pers_jr_pays = dispo_alimentaire[ ['Zone','Disponibilité alimentaire (Kcal/personne/jour)']].groupby('Zone').sum()
dispo_alimentaire_par_pers_jr_pays.head()
```



Disponibilité alimentaire (Kcal/personne/jour)

Zone

Zone	Disponibilité alimentaire (Kcal/personne/jour)
Afghanistan	2087
Afrique du Sud	3020
Albanie	3188
Algérie	3293
Allemagne	3503

```
dispo_alimentaire_par_pers_jr_pays.reset_index(inplace=True)
dispo_alimentaire_par_pers_jr_pays.head()
```



Zone Disponibilité alimentaire (Kcal/personne/jour)

0	Afghanistan	2087
1	Afrique du Sud	3020
2	Albanie	3188
3	Algérie	3293
4	Allemagne	3503

▼ 2. Affichage des 10 pays qui ont le moins de dispo alimentaire par personne

```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
```

```
dispo_alimentaire_par_pers_jr_pays_trie = dispo_alimentaire_par_pers_jr_pays.sort_values('Disponibilité alimentaire (Kcal/personne/jour)')
dispo_alimentaire_par_pers_jr_pays_trie.head()
```



Zone Disponibilité alimentaire (Kcal/personne/jour)

128	République centrafricaine	1879
166	Zambie	1924
91	Madagascar	2056
0	Afghanistan	2087
65	Haïti	2089

```
dispo_alimentaire_par_pers_jr_pays_trie.reset_index(inplace=True)
dispo_alimentaire_par_pers_jr_pays_trie.head()
```



index Zone Disponibilité alimentaire (Kcal/personne/jour)

0	128	République centrafricaine	1879
1	166	Zambie	1924
2	91	Madagascar	2056
3	0	Afghanistan	2087
4	65	Haïti	2089

```
del dispo_alimentaire_par_pers_jr_pays_trie['index']
dispo_alimentaire_par_pers_jr_pays_trie.head(10)
```

➡

Zone Disponibilité alimentaire (Kcal/personne/jour)

0	République centrafricaine	1879
1	Zambie	1924
2	Madagascar	2056
3	Afghanistan	2087
4	Haïti	2089
5	République populaire démocratique de Corée	2093
6	Tchad	2109
7	Zimbabwe	2113
8	Ouganda	2126
9	Timor-Leste	2129

```
dispo_alimentaire_par_pers_jr_pays_trie_Last10 = dispo_alimentaire_par_pers_jr_pays_trie.head(10)
display(dispo_alimentaire_par_pers_jr_pays_trie_Last10)
```

➡

Zone Disponibilité alimentaire (Kcal/personne/jour)

0	République centrafricaine	1879
1	Zambie	1924
2	Madagascar	2056
3	Afghanistan	2087
4	Haïti	2089
5	République populaire démocratique de Corée	2093
6	Tchad	2109
7	Zimbabwe	2113
8	Ouganda	2126
9	Timor-Leste	2129

Renommons les Noms trop longs pour le graphique :

```
dispo_alimentaire_par_pers_jr_pays_trie_Last10.loc[dispo_alimentaire_par_pers_jr_pays_trie_Last10['Zone']=='République centrafricaine','Z
dispo_alimentaire_par_pers_jr_pays_trie_Last10.loc[dispo_alimentaire_par_pers_jr_pays_trie_Last10['Zone']=='République populaire démocrat
display(dispo_alimentaire_par_pers_jr_pays_trie_Last10)
```

➡

Zone Disponibilité alimentaire (Kcal/personne/jour)

0	Rep. Centrafricaine	1879
1	Zambie	1924
2	Madagascar	2056
3	Afghanistan	2087
4	Haïti	2089
5	Corée du Nord	2093
6	Tchad	2109
7	Zimbabwe	2113
8	Ouganda	2126
9	Timor-Leste	2129

```
plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = dispo_alimentaire_par_pers_jr_pays_trie_Last10 , x='Zone',
                  y = 'Disponibilité alimentaire (Kcal/personne/jour)', errorbar=None)

#ax.bar_label(ax.containers[0])

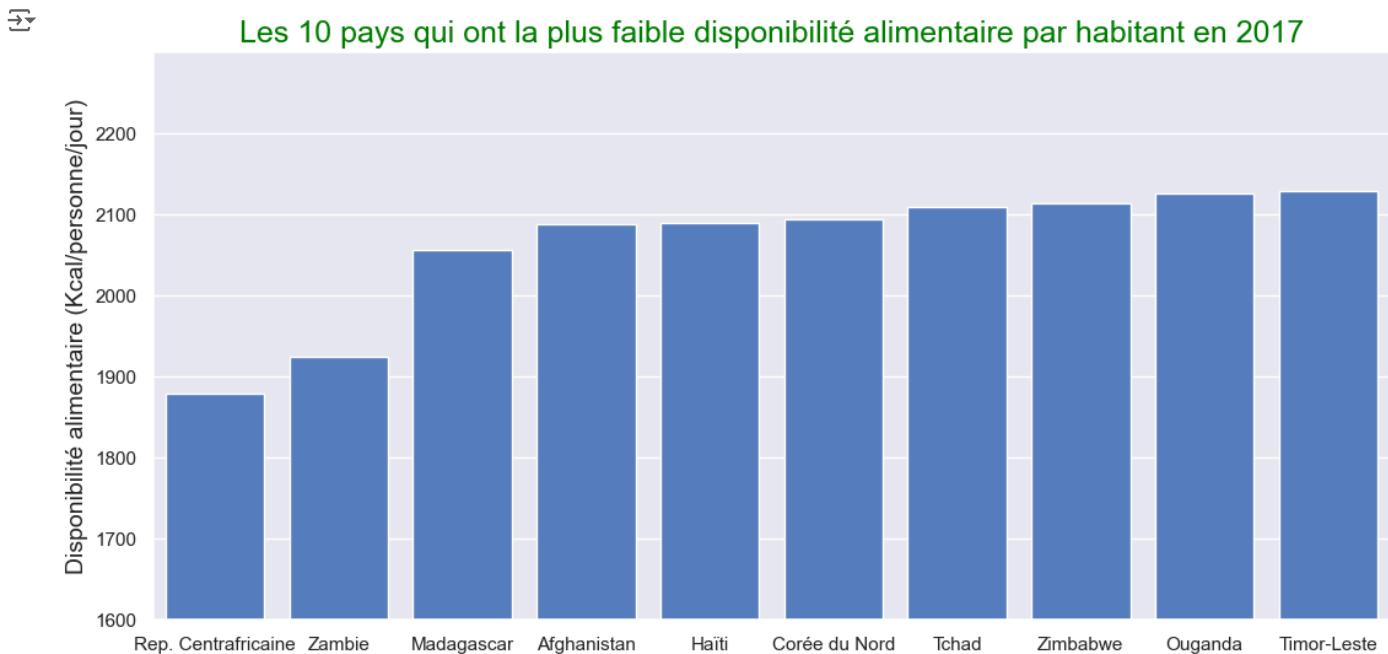
plt.xlabel('')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)',fontdict = {'fontsize' : 14})
plt.grid(axis='y')
plt.ylim(1600,2300)
```

```

plt.yticks(range(1600,2300,100))

plt.title("Les 10 pays qui ont la plus faible disponibilité alimentaire par habitant en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```



```

plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = dispo_alimentaire_par_pers_jr_pays_trie_Last10 , x='Zone',
                  y = 'Disponibilité alimentaire (Kcal/personne/jour)', errorbar=None,
                  hue='Zone')

#for x in range(0,10) :
#    ax.bar_label(ax.containers[x])

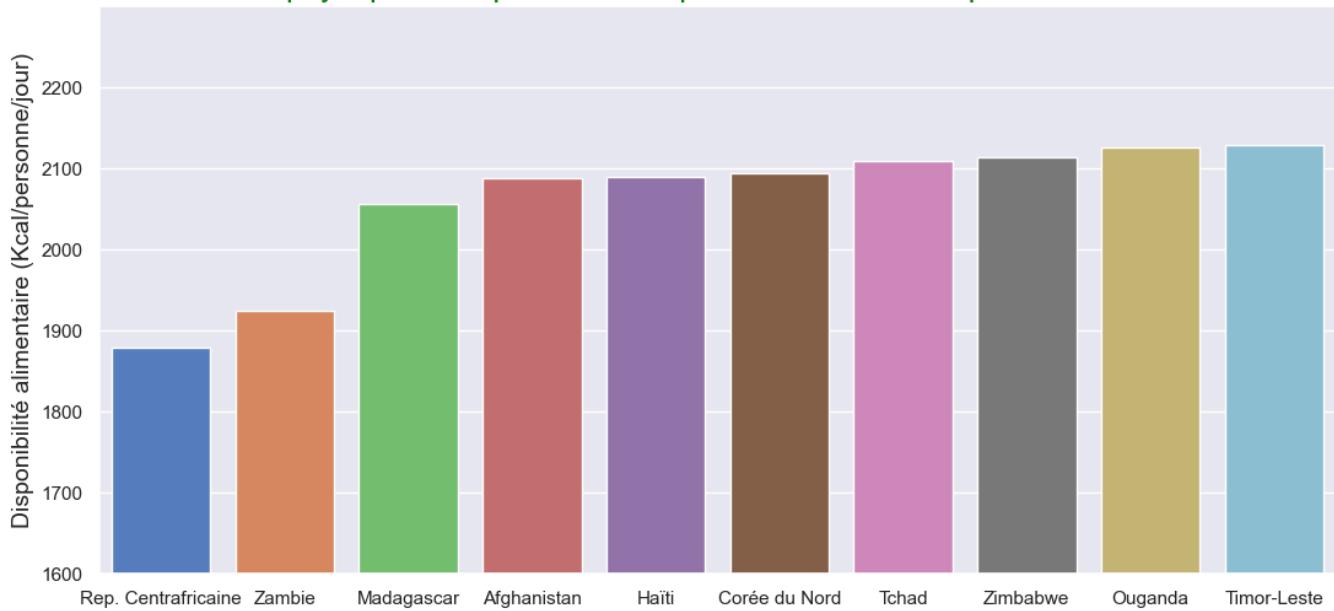
plt.xlabel('')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)',fontdict = {'fontsize' : 14})
plt.grid(axis='y')
plt.ylim(1600,2300)
plt.yticks(range(1600,2300,100))

plt.title("Les 10 pays qui ont la plus faible disponibilité alimentaire par habitant en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```



Les 10 pays qui ont la plus faible disponibilité alimentaire par habitant en 2017



Commencez à coder ou à générer avec l'IA.

3.10 - Pays avec le plus de disponibilité par habitant

▼ 1. Affichage des 10 pays qui ont le plus de dispo alimentaire par personne

```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
```

```
dispo_alimentaire_par_pers_jr_pays_trie_Top10 = dispo_alimentaire_par_pers_jr_pays_trie.tail(10)
display(dispo_alimentaire_par_pers_jr_pays_trie_Top10)
```



Zone Disponibilité alimentaire (Kcal/personne/jour)

164	Allemagne	3503
165	Égypte	3518
166	Luxembourg	3540
167	Italie	3578
168	Irlande	3602
169	Israël	3610
170	États-Unis d'Amérique	3682
171	Turquie	3708
172	Belgique	3737
173	Autriche	3770

Renommons les noms trop longs pour le graphique

```
dispo_alimentaire_par_pers_jr_pays_trie_Top10.loc[dispo_alimentaire_par_pers_jr_pays_trie_Top10['Zone']=="États-Unis d'Amérique",'Zone']= "USA"
display(dispo_alimentaire_par_pers_jr_pays_trie_Top10)
```

```

Zone Disponibilité alimentaire (Kcal/personne/jour)
164 Allemagne 3503
165 Égypte 3518
166 Luxembourg 3540
167 Italie 3578
168 Irlande 3602
169 Israël 3610
170 USA 3682
171 Turquie 3708
172 Belgique 3737
173 Autriche 3770

dispo_alimentaire_par_pers_jr_pays_trie_Top10.sort_values('Disponibilité alimentaire (Kcal/personne/jour)', inplace=True, ascending=False)
display(dispo_alimentaire_par_pers_jr_pays_trie_Top10)

C:\Users\matth\AppData\Local\Temp\ipykernel_21284\1161776623.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus
dispo_alimentaire_par_pers_jr_pays_trie_Top10.sort_values('Disponibilité alimentaire (Kcal/personne/jour)', inplace=True, ascending=False)

Zone Disponibilité alimentaire (Kcal/personne/jour)
173 Autriche 3770
172 Belgique 3737
171 Turquie 3708
170 USA 3682
169 Israël 3610
168 Irlande 3602
167 Italie 3578
166 Luxembourg 3540
165 Égypte 3518
164 Allemagne 3503

plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = dispo_alimentaire_par_pers_jr_pays_trie_Top10 , x='Zone',
                  y = 'Disponibilité alimentaire (Kcal/personne/jour)', errorbar=None)

#ax.bar_label(ax.containers[0])

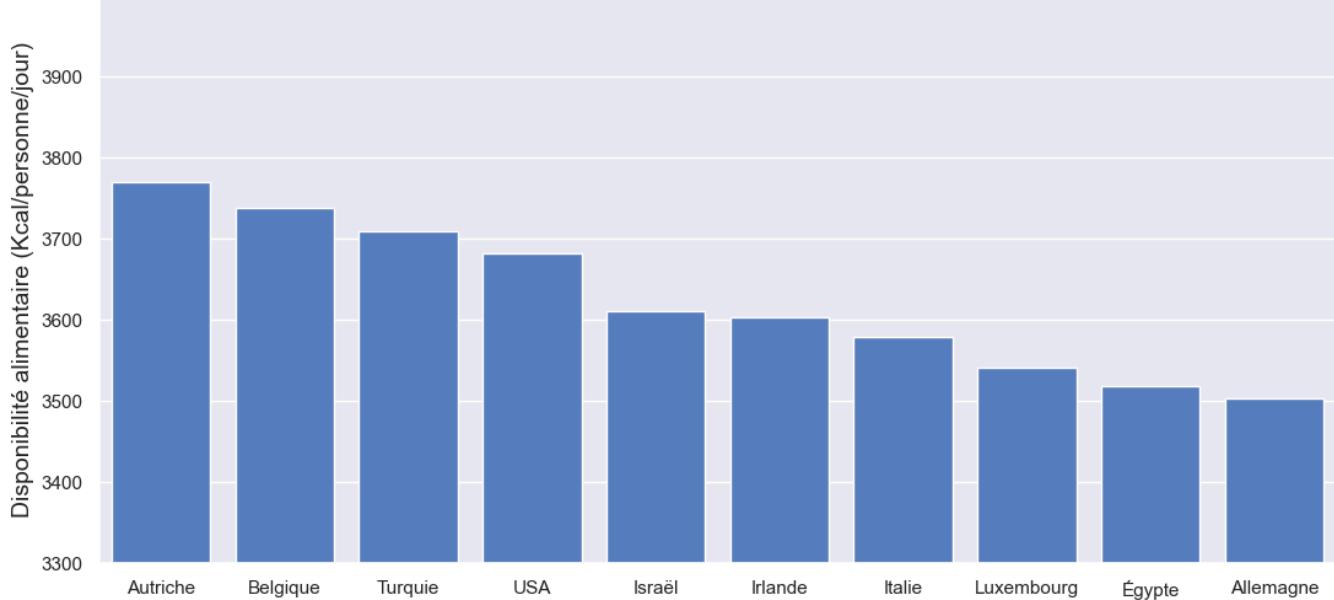
plt.xlabel('')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)', fontdict = {'fontsize' : 14})
plt.grid(axis='y')
plt.ylim(3300,4000)
plt.yticks(range(3300,4000,100))

plt.title("Les 10 pays qui ont la plus forte disponibilité alimentaire par habitant en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```

[

Les 10 pays qui ont la plus forte disponibilité alimentaire par habitant en 2017



```

plt.figure(figsize=(13,6))

sns.set_theme(style='dark', palette='muted')

ax = sns.barplot(data = dispo_alimentaire_par_pers_jr_pays_trie_Top10 , x='Zone',
                  y = 'Disponibilité alimentaire (Kcal/personne/jour)', errorbar=None,
                  hue='Zone')

#for x in range(0,10) :
#    ax.bar_label(ax.containers[x])

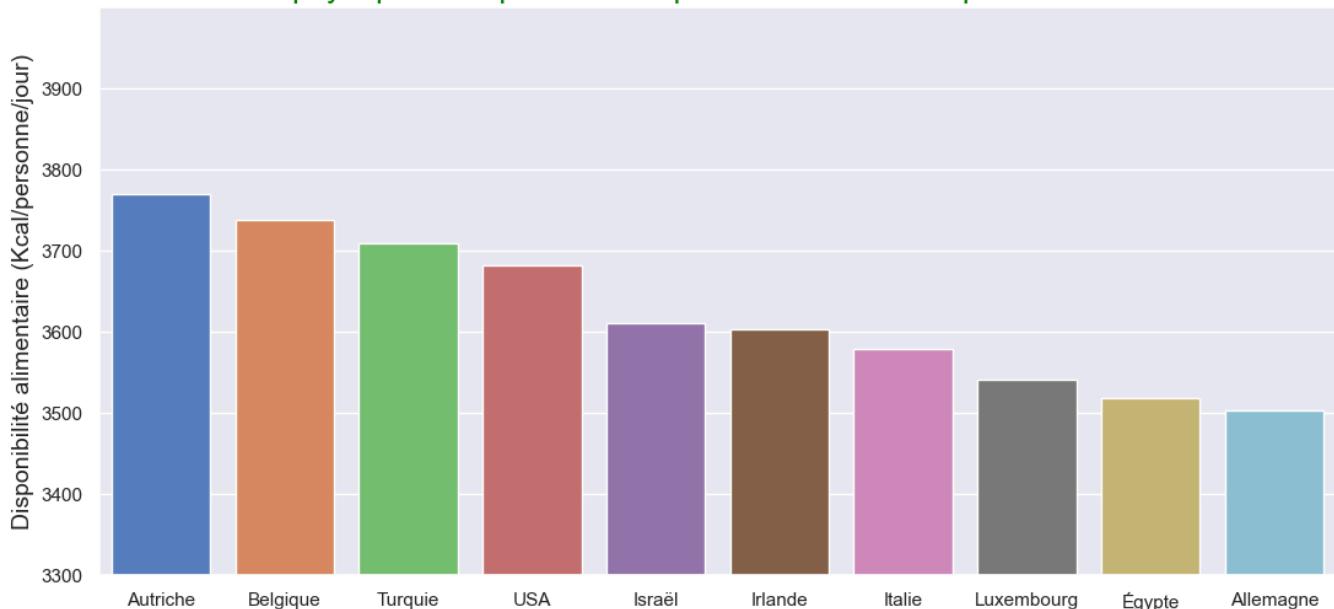
plt.xlabel('')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)', fontdict = {'fontsize' : 14})
plt.grid(axis='y')
plt.ylim(3300,4000)
plt.yticks(range(3300,4000,100))

plt.title("Les 10 pays qui ont la plus forte disponibilité alimentaire par habitant en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()

```

[

Les 10 pays qui ont la plus forte disponibilité alimentaire par habitant en 2017



3.11 - Exemple de la Thaïlande pour le Manioc

✓ 1. Création d'un dataframe avec uniquement la Thaïlande

```
#création d'un dataframe avec uniquement la Thaïlande
```

```
#Rappel des fichiers déjà étudiés :
```

```
#display(population)
```

```
#display(sous_nutrition)
```

```
#display(population_avc_sous_nutrition_2017)
```

```
population_avc_sous_nutrition_2017.head()
```

	Zone	Année_x	Population	Année_y	Sous_nutrition
0	Afghanistan	2017	36296113	2016-2018	10500000
1	Afrique du Sud	2017	57009756	2016-2018	3100000
2	Albanie	2017	2884169	2016-2018	100000
3	Algérie	2017	41389189	2016-2018	1300000
4	Allemagne	2017	82658409	2016-2018	0

```
#Pour voir comment est écrit Thaïlande dans le fichier : => 'Thaïlande'  
#display(population_avc_sous_nutrition_2017.Zone.unique().tolist())
```

```
#Restrictions pour avoir les données utiles à la Thaïlande :
```

```
#avec la notation raccourci du filtre
```

```
population_avc_sous_nutrition_2017_Thaïlande = population_avc_sous_nutrition_2017[population_avc_sous_nutrition_2017.Zone=='Thaïlande']  
display(population_avc_sous_nutrition_2017_Thaïlande)
```

	Zone	Année_x	Population	Année_y	Sous_nutrition
185	Thaïlande	2017	69209810	2016-2018	6200000

```
population_avc_sous_nutrition_2017_Thaïlande.reset_index(inplace=True)  
display(population_avc_sous_nutrition_2017_Thaïlande)
```

	index	Zone	Année_x	Population	Année_y	Sous_nutrition
0	185	Thaïlande	2017	69209810	2016-2018	6200000

```
del population_avc_sous_nutrition_2017_Thaïlande['index']  
display(population_avc_sous_nutrition_2017_Thaïlande)
```

	Zone	Année_x	Population	Année_y	Sous_nutrition
0	Thaïlande	2017	69209810	2016-2018	6200000

✓ 2. Calcul de la sous nutrition en Thaïlande

```
#Calcul de la sous nutrition en Thaïlande
```

```
population_avc_sous_nutrition_2017_Thaïlande.loc[0,'Population']
```

```
69209810
```

```
Proportion_SN_Thaïlande = (population_avc_sous_nutrition_2017_Thaïlande.loc[0,'Sous_nutrition'] *100  
                           )/population_avc_sous_nutrition_2017_Thaïlande.loc[0,'Population']  
print(Proportion_SN_Thaïlande)
```

```
8.958267621309753
```

```
print("En 2017, en Thaïlande, la proportion de la population en sous-nutrition est de",  
     round(Proportion_SN_Thaïlande , 2),  
     "% .")
```

```
En 2017, en Thaïlande, la proportion de la population en sous-nutrition est de 8.96 % .
```

VISUALISATION :

```
dict_col = {'Label': ['Nutrion','Sous-Nutrion'],
            'Population':[population_avc_sous_nutrition_2017_Thailande.loc[0,'Population']-population_avc_sous_nutrition_2017_Thailande.loc[0,'Sous_nutrition'] ]
           }
df_Visual_SN_Thailande = pd.DataFrame(data= dict_col, index=[0,1])
df_Visual_SN_Thailande.head()
```

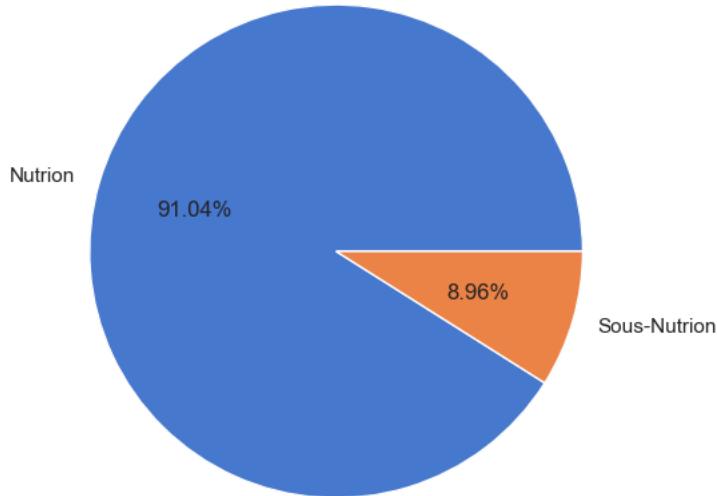
	Label	Population
0	Nutrion	63009810
1	Sous-Nutrion	6200000

```
plt.figure(figsize=(13,6))

plt.pie(x= df_Visual_SN_Thailande['Population'], labels=df_Visual_SN_Thailande['Label'], autopct='%.2f%%')

plt.title("Proportion de sous-nutrition en Thaïlande en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```

☞ Proportion de sous-nutrition en Thaïlande en 2017



▼ 3. Calcul la proportion de Manioc exportée en fonction de la production

```
# On calcule la proportion exportée en fonction de la proportion
```

La phrase "proportion exportée en fonction de la proportion" ne veut rien dire. J'imagine que cela signifie "proportion exportée par rapport à la production"

```
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/pers)
0	Afghanistan	Abats Comestible	animale	0	0	5	1.72	0.2	
1	Afghanistan	Agrumes, Autres	vegetale	0	0	1	1.29	0.01	
2	Afghanistan	Aliments pour enfants	vegetale	0	0	1	0.06	0.01	
3	Afghanistan	Ananas	vegetale	0	0	0	0.0	0.0	
4	Afghanistan	Bananes	vegetale	0	0	4	2.7	0.02	

```
dispo_alimentaire_Thailande = dispo_alimentaire[dispo_alimentaire.Zone == 'Thaïlande']
dispo_alimentaire_Thailande.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/pers)
13759	Thaïlande	Abats Comestible	animale	0	0	3	1.11	0.09	
13760	Thaïlande	Agrumes, Autres	vegetale	0	0	0	0.09	0.0	
13761	Thaïlande	Alcool, non Comestible	vegetale	0	358000000	0	0.0	0.0	
13762	Thaïlande	Aliments pour enfants	vegetale	0	0	2	0.18	0.01	
13763	Thaïlande	Ananas	vegetale	0	0	10	10.02	0.04	

```
dispo_alimentaire_Thailande.columns.tolist()
```

```
['Zone',
 'Produit',
 'Origine',
 'Aliments pour animaux(kg)',
 'Autres Utilisations(kg)',
 'Disponibilité alimentaire (Kcal/personne/jour)',
 'Disponibilité alimentaire en quantité (kg/personne/an)',
 'Disponibilité matière grasse en quantité (g/personne/jour)',
 'Disponibilité de protéines en quantité (g/personne/jour)',
 'Disponibilité intérieure(kg)',
 'Exportations - Quantité(kg)',
 'Importations - Quantité(kg)',
 'Nourriture(kg)',
 'Pertes(kg)',
 'Production(kg)',
 'Semences(kg)',
 'Traitement(kg)',
 'Variation de stock(kg)']
```

▼ POUR LE MANIOC :

```
dispo_alimentaire_Thailande.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disp (g/pe)
13759	Thaïlande	Abats Comestible	animale	0	0	3	1.11	0.09	0.09
13760	Thaïlande	Agrumes, Autres	vegetale	0	0	0	0.09	0.0	0.0
13761	Thaïlande	Alcool, non Comestible	vegetale	0	358000000	0	0.0	0.0	0.0
13762	Thaïlande	Aliments pour enfants	vegetale	0	0	2	0.18	0.01	
13763	Thaïlande	Ananas	vegetale	0	0	10	10.02	0.04	

```
#dispo_alimentaire_Thailande['Produit'].tolist()
```

```
dispo_Manioc_Thailande_2 = dispo_alimentaire_Thailande.loc[dispo_alimentaire_Thailande.Produit=='Manioc', ['Produit', 'Exportations - Quantité(kg)', 'Importations - Quantité(kg)', 'Production(kg)', 'Variation de stock(kg)']]
```

```
dispo_Manioc_Thailande_2.head()
```

	Produit	Exportations - Quantité(kg)	Importations - Quantité(kg)	Production(kg)	Variation de stock(kg)
13809	Manioc	25214000000	1250000000	30228000000	0

```
dispo_Manioc_Thailande_2['Production pour Interieur(kg)'] = dispo_Manioc_Thailande_2['Production(kg)']-dispo_Manioc_Thailande_2['Exportations - Quantité(kg)']
display(dispo_Manioc_Thailande_2)
```

	Produit	Exportations - Quantité(kg)	Importations - Quantité(kg)	Production(kg)	Variation de stock(kg)	Production pour Interieur(kg)
13809	Manioc	25214000000	1250000000	30228000000	0	5014000000

```
dispo_Manioc_Thailande_2.columns.tolist()
```

```
['Produit', 'Exportations - Quantité(kg)', 'Importations - Quantité(kg)', 'Production(kg)', 'Variation de stock(kg)', 'Production pour Interieur(kg)']
```

```
dispo_Manioc_Thailande_3 = dispo_Manioc_Thailande_2[['Exportations - Quantité(kg)', 'Production pour Interieur(kg)']]
dispo_Manioc_Thailande_3.head()
```

	Exportations - Quantité(kg)	Production pour Interieur(kg)
13809	25214000000	5014000000

```
dispo_Manioc_Thailande_3.rename(index={13809 : 'Quantité(kg)'}, inplace=True)
dispo_Manioc_Thailande_3.head()
```

```
C:\Users\matth\AppData\Local\Temp\ipykernel_21284\116079948.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus
dispo_Manioc_Thailande_3.rename(index={13809 : 'Quantité(kg)'}, inplace=True)
```

	Exportations - Quantité(kg)	Production pour Interieur(kg)
Quantité(kg)	25214000000	5014000000

```
dispo_Manioc_Thailande_4 = dispo_Manioc_Thailande_3.transpose()
dispo_Manioc_Thailande_4.head()
```

	Quantité(kg)
Exportations - Quantité(kg)	25214000000
Production pour Interieur(kg)	5014000000

```
dispo_Manioc_Thailande_4.reset_index(inplace=True)
dispo_Manioc_Thailande_4.head()
```

	index	Quantité(kg)
0	Exportations - Quantité(kg)	25214000000
1	Production pour Interieur(kg)	5014000000

```
dispo_Manioc_Thailande_4.rename(columns={"index" : 'Utilisation'},inplace=True)
dispo_Manioc_Thailande_4.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	25214000000
1	Production pour Interieur(kg)	5014000000

```
dispo_Manioc_Thailande_4.loc[0,'Utilisation']='Exportations - Quantité'
dispo_Manioc_Thailande_4.loc[1,'Utilisation']='Production pour Interieur'
dispo_Manioc_Thailande_4.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité	25214000000
1	Production pour Interieur	5014000000

```
Proportion_Exportation_Manioc_Thailande = ((dispo_Manioc_Thailande_4.loc[0,'Quantité(kg)'] *100)
                                             / (dispo_Manioc_Thailande_4.loc[0,'Quantité(kg)']+dispo_Manioc_Thailande_4.loc[1,'Quantité(kg)']))
print(Proportion_Exportation_Manioc_Thailande)
```

```
83.41272991928014
```

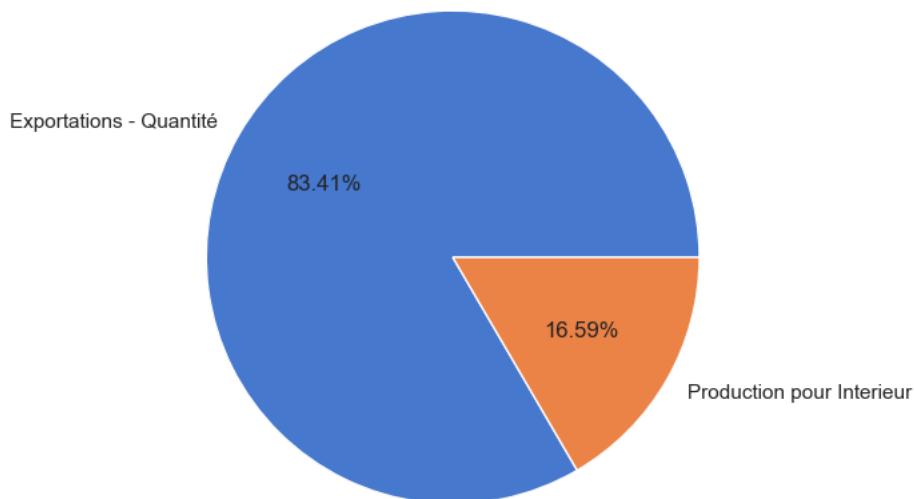
```
print("En 2017, en Thaïlande, la proportion de la production de Manioc qui est exportée est de",
      round(Proportion_Exportation_Manioc_Thailande , 2),
      "% .")
```

```
En 2017, en Thaïlande, la proportion de la production de Manioc qui est exportée est de 83.41 % .
```

```
plt.figure(figsize=(13,6))
plt.pie(x= dispo_Manioc_Thailande_4['Quantité(kg)'], labels=dispo_Manioc_Thailande_4['Utilisation'], autopct='%.2f%%')
```

```
plt.title("Utilisation de la production de Manioc en Thaïlande en 2017",
          fontdict = {'fontsize' : 18, 'color':'green', 'verticalalignment': 'baseline'})
plt.show()
```

Utilisation de la production de Manioc en Thaïlande en 2017



4. Calcul la proportion exportée en fonction de la production tous produits, pour la Thaïlande

✓ POUR L'ENSEMBLE DES PRODUITS :

✓ RECUPERONS LES INFOS UTILES :

```
dispo_alimentaire_Thailande.head()
```

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disp (g/pe)
13759	Thaïlande	Abats Comestible	animale	0	0	3	1.11	0.09	
13760	Thaïlande	Agrumes, Autres	vegetale	0	0	0	0.09	0.0	
13761	Thaïlande	Alcool, non Comestible	vegetale	0	358000000	0	0.0	0.0	
13762	Thaïlande	Aliments pour enfants	vegetale	0	0	2	0.18	0.01	
13763	Thaïlande	Ananas	vegetale	0	0	10	10.02	0.04	

```
dispo_alimentaire_Thailande_2 = dispo_alimentaire_Thailande[ ['Exportations - Quantité(kg)', 'Importations - Quantité(kg)', 'Production(kg)', 'Variation de stock(kg)'] ].sum()
```

```
print(dispo_alimentaire_Thailande_2)
```

```
Exportations - Quantité(kg)      50430000000
Importations - Quantité(kg)      11335000000
Production(kg)                  201764000000
Variation de stock(kg)          -4534000000
dtype: Int64
```

```
type (dispo_alimentaire_Thailande_2)
```

```
pandas.core.series.Series
```

```
Total_dispo_alim_Thailande = dispo_alimentaire_Thailande_2.sum()
print(Total_dispo_alim_Thailande)
```

```
258995000000
```

✓ REFORMATONS LES DONNEES POUR NE GARDER QUE CE QUI EST NECESSAIRE :

```
dispo_alimentaire_Thailande_3 = pd.DataFrame(dispo_alimentaire_Thailande_2)
dispo_alimentaire_Thailande_3.head()
```

	0
Exportations - Quantité(kg)	50430000000
Importations - Quantité(kg)	11335000000
Production(kg)	201764000000
Variation de stock(kg)	-4534000000

```
dispo_alimentaire_Thailande_3.reset_index(inplace=True)
dispo_alimentaire_Thailande_3.head()
```

	index	0
0	Exportations - Quantité(kg)	50430000000
1	Importations - Quantité(kg)	11335000000
2	Production(kg)	201764000000
3	Variation de stock(kg)	-4534000000

```
dispo_alimentaire_Thailande_3.rename(columns={'0': 'Quantité(kg)', 'index': 'Utilisation'}, inplace=True)
dispo_alimentaire_Thailande_3.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	50430000000
1	Importations - Quantité(kg)	11335000000
2	Production(kg)	201764000000
3	Variation de stock(kg)	-4534000000

```
dispo_alimentaire_Thailande_3.loc[4,['Utilisation']] = 'Production pour Interieur(kg)'
dispo_alimentaire_Thailande_3.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	50430000000
1	Importations - Quantité(kg)	11335000000
2	Production(kg)	201764000000
3	Variation de stock(kg)	-4534000000
4	Production pour Interieur(kg)	<NA>

```
Production_Th_2 = dispo_alimentaire_Thailande_3.iloc[2,1]
print(Production_Th_2)
```

```
201764000000
```

```
Exportation_Th_2 = dispo_alimentaire_Thailande_3.iloc[0,1]
print(Exportation_Th_2)
```

```
50430000000
```

```
dispo_alimentaire_Thailande_3.loc[4,['Quantité(kg)']] = Production_Th_2 - Exportation_Th_2
dispo_alimentaire_Thailande_3.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	50430000000
1	Importations - Quantité(kg)	11335000000
2	Production(kg)	201764000000
3	Variation de stock(kg)	-4534000000
4	Production pour Interieur(kg)	151334000000

```
dispo_alimentaire_Thailande_4 = dispo_alimentaire_Thailande_3.drop(index=[1,2,3])
dispo_alimentaire_Thailande_4.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	50430000000
4	Production pour Interieur(kg)	151334000000

```
dispo_alimentaire_Thailande_4.reset_index(inplace=True)
del dispo_alimentaire_Thailande_4['index']
dispo_alimentaire_Thailande_4.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité(kg)	50430000000
1	Production pour Interieur(kg)	151334000000

```
dispo_alimentaire_Thailande_4.loc[0,'Utilisation']='Exportations - Quantité'  
dispo_alimentaire_Thailande_4.loc[1,'Utilisation']='Production pour Interieur'  
dispo_alimentaire_Thailande_4.head()
```

	Utilisation	Quantité(kg)
0	Exportations - Quantité	50430000000
1	Production pour Interieur	151334000000

▼ LECTURE :

```
Proportion_Exportation_Thailande = ((dispo_alimentaire_Thailande_4.loc[0,'Quantité(kg)'] *100)  
                                    / (dispo_alimentaire_Thailande_4.loc[0,'Quantité(kg)']+dispo_alimentaire_Thailande_4.loc[1,'Quantité(kg)']))  
print(Proportion_Exportation_Thailande)
```

→ 24.994548085882517

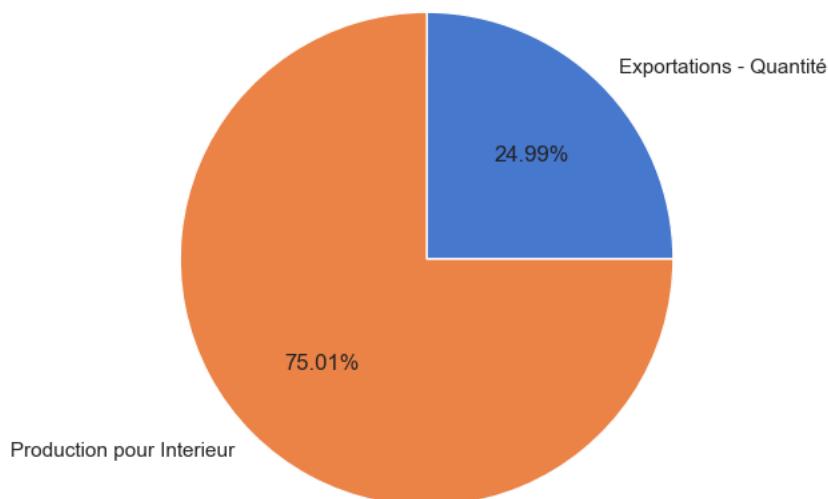
```
print("En 2017, en Thaïlande, la proportion de la production qui est exportée est de",  
      round(Proportion_Exportation_Thailande , 2),  
      "% .")
```

→ En 2017, en Thaïlande, la proportion de la production qui est exportée est de 24.99 % .

▼ VISUALISATION :

```
plt.figure(figsize=(13,6))  
plt.pie(x= dispo_alimentaire_Thailande_4['Quantité(kg)'], labels=dispo_alimentaire_Thailande_4['Utilisation'], autopct='%.2f%%')  
  
plt.title("Utilisation de la production en Thaïlande en 2017",  
          fontdict = {'fontsize' : 16, 'color':'green', 'verticalalignment': 'baseline'})  
plt.show()
```

→ Utilisation de la production en Thaïlande en 2017



CONCLUSION :

Au total la Thaïlande exporte 25% de sa production. Cela vient nuancer l'analyse sur le Manioc. Mais exporter moins resterait une solution pour améliorer son Taux de nutrition.

Commencez à coder ou à générer avec l'IA.

Commencez à coder ou à générer avec l'IA.

▼ 5. Etude complémentaire, pour la Thaïlande : les 10 plus grosses productions Thaïlandaise

dispo_alimentaire_Thailande

	Zone	Produit	Origine	Aliments pour animaux(kg)	Autres Utilisations(kg)	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité matière grasse en quantité (g/personne/jour)	Disp
13759	Thaïlande	Abats Comestible	animale	0	0	3	1.11	0.09	0.09
13760	Thaïlande	Agrumes, Autres	vegetale	0	0	0	0.09	0.0	0.0
13761	Thaïlande	Alcool, non Comestible	vegetale	0	358000000	0	0.0	0.0	0.0
13762	Thaïlande	Aliments pour enfants	vegetale	0	0	2	0.18	0.01	0.01
13763	Thaïlande	Ananas	vegetale	0	0	10	10.02	0.04	0.04
...
13849	Thaïlande	Viande de Suides	animale	0	0	124	13.0	11.83	11.83
13850	Thaïlande	Viande de Volailles	animale	0	0	52	13.69	3.62	3.62
13851	Thaïlande	Viande, Autre	animale	0	0	0	0.03	0.01	0.01
13852	Thaïlande	Vin	vegetale	0	0	0	0.12	0.0	0.0
13853	Thaïlande	Épices, Autres	vegetale	0	0	16	1.7	0.3	0.3

95 rows × 18 columns

dispo_alimentaire_Thailande.columns

```
Index(['Zone', 'Produit', 'Origine', 'Aliments pour animaux(kg)', 'Autres Utilisations(kg)', 'Disponibilité alimentaire (Kcal/personne/jour)', 'Disponibilité alimentaire en quantité (kg/personne/an)', 'Disponibilité matière grasse en quantité (g/personne/jour)', 'Disponibilité de protéines en quantité (g/personne/jour)', 'Disponibilité intérieure(kg)', 'Exportations - Quantité(kg)', 'Importations - Quantité(kg)', 'Nourriture(kg)', 'Pertes(kg)', 'Production(kg)', 'Semences(kg)', 'Traitement(kg)', 'Variation de stock(kg)'],  
      dtype='object')
```

```
dispo_alimentaire_Thailande_Complement_1 = dispo_alimentaire_Thailande[['Produit', 'Production(kg)']].groupby('Produit').sum()  
dispo_alimentaire_Thailande_Complement_1
```

Production(kg)

Produit	
Abats Comestible	45000000
Agrumes, Autres	12000000
Alcool, non Comestible	447000000
Aliments pour enfants	0
Ananas	2209000000
...	...
Viande de Suides	891000000
Viande de Volailles	1470000000
Viande, Autre	0
Vin	0
Épices, Autres	143000000

95 rows × 1 columns

```
dispo_alimentaire_Thailande_Complement_1.reset_index(inplace=True)  
dispo_alimentaire_Thailande_Complement_1
```

	Produit	Production(kg)
0	Abats Comestible	45000000
1	Agrumes, Autres	12000000
2	Alcool, non Comestible	447000000
3	Aliments pour enfants	0
4	Ananas	2209000000
...
90	Viande de Suides	891000000
91	Viande de Volailles	1470000000
92	Viande, Autre	0
93	Vin	0
94	Épices, Autres	143000000

95 rows × 2 columns

```
dispo_alimentaire_Thailande_Complement_1['Production(Kt)']=dispo_alimentaire_Thailande_Complement_1['Production(kg)']/1_000_000
dispo_alimentaire_Thailande_Complement_1
```

	Produit	Production(kg)	Production(Kt)
0	Abats Comestible	45000000	45.0
1	Agrumes, Autres	12000000	12.0
2	Alcool, non Comestible	447000000	447.0
3	Aliments pour enfants	0	0.0
4	Ananas	2209000000	2209.0
...
90	Viande de Suides	891000000	891.0
91	Viande de Volailles	1470000000	1470.0
92	Viande, Autre	0	0.0
93	Vin	0	0.0
94	Épices, Autres	143000000	143.0

95 rows × 3 columns

```
del dispo_alimentaire_Thailande_Complement_1['Production(kg)']
dispo_alimentaire_Thailande_Complement_1
```

	Produit	Production(Kt)
0	Abats Comestible	45.0
1	Agrumes, Autres	12.0
2	Alcool, non Comestible	447.0
3	Aliments pour enfants	0.0
4	Ananas	2209.0
...
90	Viande de Suides	891.0
91	Viande de Volailles	1470.0
92	Viande, Autre	0.0
93	Vin	0.0
94	Épices, Autres	143.0

95 rows × 2 columns

```
dispo_alimentaire_Thailande_Complement_2=dispo_alimentaire_Thailande_Complement_1.sort_values('Production(Kt)', ascending=False)
dispo_alimentaire_Thailande_Complement_2
```

	Produit	Production(Kt)
84	Sucre, canne	100096.0
50	Manioc	30228.0
78	Riz (Eq Blanchi)	24054.0
82	Sucre Eq Brut	10024.0
24	Fruits, Autres	6141.0
...
64	Patates douces	0.0
37	Huile de Germe de Maïs	0.0
74	Pommes	0.0
67	Plantes Aquatiques	0.0

dispo_alimentaire_Thailande_Complement_3=dispo_alimentaire_Thailande_Complement_2.head(10)

05 rows x 2 columns

dispo_alimentaire_Thailande_Complement_3

	Produit	Production(Kt)
84	Sucre, canne	100096.0
50	Manioc	30228.0
78	Riz (Eq Blanchi)	24054.0