TD : Manipuler des données avec pandas

Objectifs

Découvrir la librairie pandas et ses fonctionnalités

Prérequis

Avoir suivi le cours sur les bases de Python

Introduction

Pandas est une librairie Python qui permet de manipuler des données de manière simple et efficace. Elle est particulièrement adaptée pour le traitement de données tabulaires, comme des fichiers CSV ou des bases de données.

Documentation utile

- Documentation officielle
- Tutoriel

Installation

Pour installer pandas, il suffit d'utiliser la commande pip :

```
pip install pandas
```

Exercices TD

Exemple d'utilisation de pandas et basés sur un fichier CSV avec les colonnes suivantes : id, firstname, lastname, email, profession, country, city.

Exercice 1 : Charger le fichier CSV

```
import pandas as p

file_path = "your_file.csv"
data = p.read_csv(file_path)
print(data.head())
```

Exercice 2 : Afficher les informations générales du DataFrame

```
data.info()
```

Exercice 3 : Afficher le nombre de personnes par profession

```
professions = data['profession'].value_counts()
print(professions)
```

Exercice 4 : Afficher le nombre de personnes par pays

```
countries = data['country'].value_counts()
print(countries)
```

Exercice 5 : Afficher les 10 premières personnes dont le nom de famille commence par la lettre 'D'

```
people_d = data[data['lastname'].str.startswith('D')].head(10)
print(people_d)
```

Exercice 6 : Créer un nouveau DataFrame contenant uniquement les colonnes firstname, lastname et email

```
contact_info = data[['firstname', 'lastname', 'email']]
print(contact_info.head())
```

Exercice 7 : Trier le DataFrame par ordre alphabétique croissant des noms de famille

```
sorted_data = data.sort_values(by=['lastname'])
print(sorted_data.head())
```

Exercice 8 : Sauvegarder les données triées dans un nouveau fichier CSV

```
sorted_data.to_csv('sorted_data.csv', index=False)
```

Exercice 9 : Trouver la ville la plus fréquente dans le fichier

```
most_common_city = data['city'].mode().iloc[0]
print("La ville la plus fréquente est :", most_common_city)
```

Exercice 10 : Filtrer les personnes travaillant dans une profession spécifique, par exemple "Data Scientist"

```
data_scientists = data[data['profession'] == "Data Scientist"]
print(data_scientists)
```

Exercice 11 : Calculer l'age moyen par profession

```
data['birthdate'] = p.to_datetime(data['birthdate'], format="%m-%d-%Y")

def calcule_age(birthdate):
    today = datetime.datetime.now()
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.
    return age

data['age'] = data['birthdate'].apply(calcule_age)
print(data['age'])
age_par_profession = data.groupby('profession')['age'].mean()
print(age_par_profession)
```

Exercice 12 : Calculer le salaire moyen par profession

```
average_salary_by_profession = data.groupby('profession')['salary'].mean()
print(average_salary_by_profession)
```

Exercice 13 : Afficher les personnes ayant un salaire supérieur à un montant donné, par exemple 5000

```
high_salary = data[data['salary'] > 5000]
print(high_salary)
```

Exercice 14 : Afficher le pourcentage de personnes par pays

```
percentage_by_country = data['country'].value_counts(normalize=True) * 100
print(percentage_by_country)
```

Exercice 15 : Trouver le salaire le plus élevé et le plus bas par pays

```
max_salary_by_country = data.groupby('country')['salary'].max()
min_salary_by_country = data.groupby('country')['salary'].min()
print("Salaire maximum par pays :\n", max_salary_by_country)
print("Salaire minimum par pays :\n", min_salary_by_country)
```

Exercice 16 : Créer un nouveau DataFrame en ne gardant que les personnes ayant un certain âge, par exemple 30 ans

```
data['birthdate'] = p.to_datetime(data['birthdate'], format="%m-%d-%Y")

def calculate_age(birthdate):
    today = datetime.datetime.now()
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.
    return age

data['age'] = data['birthdate'].apply(calculate_age)
age_filtre = data[data['age'] == 30]
print(age_filtre)
```

Exercice 17 : Afficher le nombre de personnes par tranche d'âge (par exemple, tous les 10 ans) en utilisant la fonction utilisée dans l'exercice 16

```
age_bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
age_labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90
data['age_group'] = p.cut(data['age'], bins=age_bins, labels=age_labels, include_lowest=True)
people_by_age_group = data['age_group'].value_counts()
print(people_by_age_group)
```

Exercice 18 : Afficher la répartition des salaires en utilisant des histogrammes

```
import matplotlib.pyplot as plt

salary_bins = range(0, int(data['salary'].max()) + 10000, 10000)
plt.hist(data['salary'], bins=salary_bins, edgecolor='black')
plt.title('Répartition des salaires')
plt.xlabel('Salaire')
plt.ylabel('Nombre de personnes')
plt.show()
```

Exercice 19 : Trouver les personnes avec le même nom de famille

```
nom_doublon = data[data['lastname'].duplicated(keep=False)].sort_values(by='lastname')
print(nom_doublon)
```

Exercice 20 : Afficher la répartition des professions par pays

```
profession_par_pays = data.groupby(['country', 'profession']).size().unstack()
print(profession_par_pays)
```