

Intitulé de la formation préparant à la certification :

« Manipuler, analyser et visualiser des données grâce aux modules Python de Data Science, certification API Society »

Durée : 4 jours ou plus - en présentiel ou distanciel - formateur certifié mention Expert ou Instructeur

Pré-requis : Connaître les bases du langage de programmation Python

Programme

Le formateur a la liberté pédagogique d'aborder les points du programme dans l'ordre qui lui paraît le plus pertinent, dès lors qu'il traite l'intégralité du programme.

L'écosystème Python scientifique

Tour d'horizon de packages Python de Data Science
Installation de bibliothèques scientifiques dans un environnement virtuel : pip et le module venv, miniconda, mamba, miniforge, WinPython
Environnement de développement : IPython, Jupyter Notebook, JupyterLab, IDE : l'exemple de Spyder, Editeur de texte : l'exemple de VS Code

La bibliothèque NumPy

Présentation de la librairie

Intérêts de travailler avec les tableaux : performance, représentation des données (matrices, séries de temps, images, données géographiques, données génériques) et facilité dans la manipulation des données

Création de tableaux : fonctions `array()`, `zeros()`, `ones()`, `full()`, `arange()`, `linspace()`, `logspace()`

La multiplication matricielle avec `np.dot` et l'opérateur `@`

Créer une matrice identité avec les fonctions `identity()` et `eye()`

Construire une matrice diagonale avec la fonction `diag()`

Initialisation avec des données aléatoires (fonctions du module `random` de NumPy)

Les types de données (`bool`, `int`, `uint`, `float`, `complex`, `unicode`) et changer le type de données avec `astype()`

Les attributs `ndim`, `shape`, `size`, `dtype`, `itemsize`, `nbytes`

Manipulation de tableaux : indexation, slicing, indexation avancée

Broadcasting pour l'indexation et les opérations sur les données

Copie et vue d'un tableau

Transposer un tableau avec la méthode `transpose()` ou l'attribut `T`

Changer les dimensions d'un tableau : fonctions `reshape()` et `newaxis()`

Concaténer des tableaux : fonctions `concatenate()`, `vstack()`, `hstack()` et `stack()`

Découper des tableaux : fonctions `split()`, `hsplit()` et `vsplit()`

Pourquoi éviter les boucles for ?

Fonctions classiques (addition, soustraction...), fonctions trigonométriques, exposants et logarithmes

Fonctions `sum()`, `min()`, `max()`, `median()`, `percentile()`, `prod()`, `cumsum()`, `var()`, `ravel()`, `argmin()`, `argmax()`, `any()`, `all()` et `where()`

Apprendre à utiliser l'option `axis`

Fonctions de comparaisons

Extraire des informations de vos données avec des masques booléens

Charger et sauvegarder les tableaux : les fonctions `loadtxt()` (et ses options `usecols` et `skiprows`), `save()` et `load()`

La bibliothèque Pandas

Présentation de la librairie

Créer une série avec la classe `Series`

Créer un tableau 2D ou dataframe avec la classe `DataFrame`

Extraire les indices de ligne et de colonnes (attributs `index` et `columns`)

Lire et exporter des données dans différents formats (`csv`, `excel`...)

Les méthodes `head()` et `tail()`

Indexation et slicing : Indexation implicite et explicite, Utilisation des indexeurs `loc` et `iloc`, Sélectionner des données avec la sélection avancée

Sélectionner des données : avec des expressions booléennes, avec la méthode `query()`

Insérer et modifier des données

Renommer une colonne avec la fonction `rename()`

Concaténer des données avec la fonction `concat()`

Fusion et jointure de données avec les fonctions `merge()` et `join()`

Copier des données : copie superficielle ou profonde (fonction `copy()`)

Traiter les données manquantes avec les fonctions `isna()`, `isnull()`, `notna()`, `notnull()`, `dropna()`, `fillna()` et `interpolate()`

Mettre une colonne en indice avec la fonction `set_index()`

Trier les indices avec `sort_index()`

Trier les valeurs avec `sort_values()`

Transposer des données avec la fonction `transpose()`

Supprimer des données avec la fonction `drop()`

Supprimer les données en double avec la fonction `drop_duplicates()`

Aggréger des données avec les fonctions `sum()`, `cumsum()`, `min()`, `max()`, `count()`, `mean()`, `median()`, `var()`, `std()`, `median()`, `quantile()` et `describe()`

Grouper et analyser des données avec la fonction `groupby()`

Analyser les données avec les fonctions `aggregate()`, `apply()`, `filter()` et `transform()`

Créer des tableaux croisés dynamiques avec la fonction `pivot_table()`

Segmenter les données avec les fonctions `qcut()` et `cut()`

Calculer des moyennes glissantes avec les méthodes `rolling()`, `expanding()` et `ewm()`

Multi-indices : Créer des multi-indices avec `pd.MultiIndex.from_product()`, `pd.MultiIndex.from_tuple()`, `pd.MultiIndex.from_arrays()`, Ajouter des dimensions aux `Series` et `DataFrames` avec l'objet `MultiIndex`, Indexer les `Series` et `DataFrames` multi-indexés et l'objet `pd.IndexSlice`

Chaînes de caractères : Indexation et slicing sur les chaînes de caractères, Traiter les données avec les méthodes de chaînes de caractères implémentées dans `pandas`, Enchaîner plusieurs méthodes de chaînes de caractères, Utiliser des expressions régulières avec `pandas` avec les méthodes `match()`, `extract()`, `findall()`, `replace()`, `contains()`, `count()`, `split()`, `rsplit()`

Traiter les données temporelles : Créer des dates, des durées et des périodes (fonctions `to_datetime()`, `to_timedelta()`, `date_range()`, `period_range()`, `timedelta_range()`), Indexation et slicing des données temporelles, Echantillonnage avec les fonctions `asfreq()` et `resample()`

La bibliothèque Matplotlib

Présentation de la librairie

Afficher les graphiques depuis un script Python (`plt.show()`) ou depuis un notebook

Afficher les graphiques en utilisant le style MATLAB ou le style orienté objet

Modifier le style du graphique

Les objets Figure et Axes

Tracer des courbes avec la méthode `plot()` : modifier la couleur, modifier le style du tracé, modifier la largeur du tracé, ajuster la longueur des axes, ajouter un titre, nommer les axes, changer les graduations, ajouter une légende

Afficher des nuages de points avec la méthode `scatter()`

Afficher des barres d'erreurs avec la méthode `error_bar()`

Remplir la surface entre 2 lignes avec la méthode `fill_between()`

Tracer un histogramme avec la méthode `hist()`

Tracer plusieurs graphiques : placer un graphique à l'intérieur d'un autre avec la méthode `add_axes()`, créer une grille avec la fonction `subplots()`

Tracer des graphiques en 3 dimensions avec `mplot3d`

Interagir avec les graphiques dans le Jupyter notebook avec le widget interact

Utiliser `pandas plot` pour réaliser rapidement des tracés depuis un objet Series ou DataFrame : tour d'horizon des méthodes `plot()`, `bar()`, `barh()`, `hist()`, `box()`, `scatter()`, `pie()`

Conseils pour améliorer la lisibilité des graphiques pour les personnes en situation de handicap visuel

La bibliothèque Seaborn

Présentation de la librairie

Fonctionnement de l'API Seaborn : Figure-level et Axes-level

Les "relational plots" : les fonctions `relplot()`, `lineplot()` et `scatterplot()`

Tracer des distributions avec les fonctions `displot()`, `histplot()`, `jointplot()` et `pairplot()`

Tracer des données qualitatives (categorical data) avec les fonctions `catplot()`, `barplot()`, `countplot()`, `boxplot()`, `violinplot()`, `stripplot()`, `swarmplot()`

Tracer des cartes thermiques avec la fonction `heatmap()`

Tracer des modèles de régression linéaire avec la fonction `lmplot()`

Changer le rendu de la figure : ajouter un titre, changer les couleurs, les fonctions `set_theme()`, `set_style()`, `set_context()` et `despine()`

La bibliothèque Plotly

Présentation de la librairie `plotly` et de `Kaleido`, exploration de `plotly.express`

Tracer des courbes avec la fonction `line()` : modifier la figure avec les options `title`, `width`, `height`, `marker` et `labels`, sauvegarder la figure, tracer plusieurs courbes grâce à l'option `color`, ajouter des informations avec les options `hover_data`, `hover_name` et `text`, tracer plusieurs graphiques avec `facet_row` et `facet_col`, ajouter des barres d'erreurs avec les options `error_x` et `error_y`, modifier le style de la figure (option `template`, changer le thème par défaut avec `plotly.io.templates.default`)

Tracer des graphiques en aires avec la fonction `area()` : ajouter des motifs avec l'option `pattern_shape`

Afficher des nuages de points avec la fonction `scatter()` : utiliser les options `size` et `size_max` pour définir la taille des points, utiliser le paramètre `opacity` pour générer de la transparence, les options `symbol` et `symbol_sequence`, modifier la barre de couleur avec l'option `color_continuous_scale`, la méthode `update_layout()` pour modifier la position de la légende, la méthode `update_coloraxes()` pour modifier la position de la barre de couleurs

Réaliser des graphiques en 3D avec `scatter_3d()` et `line_3d()`

Tracer des diagrammes en barres avec la fonction `bar()`

Tracer des histogrammes avec la fonction `histogram()`

Tracer des cartes : les fonctions `line_map()` et `scatter_map()` avec les options `zoom` et `center`, les fonctions `line_geo()` et `scatter_geo()` avec les options `scope` et `projection`, la fonction `choropleth()`