الفصل الثاني

التعامل مع السلاسل الزمنية باستخدام بايثون

Time Series Handling with Python

محتويات الفصل الثاني

- مقدمة Introduction
- أنواع البيانات الأساسية للسلاسل الزمنية
- التعبير عن الأزمنة والتواريخ في بايثون
- استخراج معلومات محددة من متغير datetime
- العمل مع النطاقات الزمنية Date Ranges
- الفهرسة في أطر البيانات Indexing in Data frames
- مفاهيم أساسية في التعامل مع السلاسل الزمنية
- تصوير السلاسل الزمنية Time Series Visualization

لتحميل ملحقات هذا الفصل من الأكواد البرمجية والشروحات التوضيحية زيارة المستودع التالي على صفحتي في موقع GitHub :

- السلاسل الزمنية باللغة العربية

مقدمة Introduction

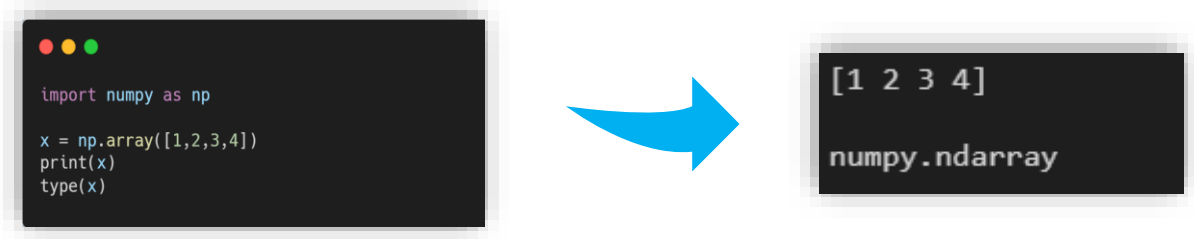
سنغطي في الشرائح التالية تطبيقًا عمليًا لإجراء العمليات الأساسية على السلاسل الزمنية من التعرف على المدخلات والمخرجات وكيفية إنشاء نطاقات زمنية من خلال DateTime DateTimeIndex باستخدام كل من مكتبتي Pandas و NumPy وكيفية إعادة التشكيل والتحويل للسلسلة الزمنية. ستتعلم أيضًا مدى سهولة تصوير بيانات السلاسل الزمنية.

- مقدمة إلى Date Times
- التعامل مع النطاقات التاريخية Date Ranges
- استعراض مفاهيم خاصة بالسلاسل الزمنية كـ Resampling, Shifting, Rolling, and Differencing
- تصوير بيانات السلاسل الزمنية Visualizing Time Series Data

أنواع البيانات الأساسية للسلاسل الزمنية

لفهم كيفية التعامل مع السلاسل الزمنية بشكل أفضل يتوجب علينا فهم أنواع البيانات التي تكون هذه السلاسل وطرق تخزينها:

1- المصفوفة Array : هي عبارة عن هيكل بيانات توفره مكتبة NumPy يستخدم لتخزين بينات من النوع نفسه مثال Integer أو string على هيئة مصفوفات ذات أبعاد محددة و تعتبر اللبنة الأساسية للعديد من خوارزميات ومشاريع التعلم الآلي. (قراءة المزيد)



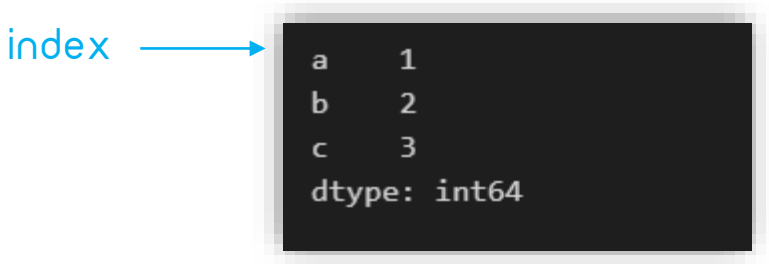
2- datetime : هو تنسيق التاريخ والوقت في مكتبة datetime المدمجة مع لغة بايثون (قراءة المزيد)

3- datetime64 : هو تنسيق التاريخ والوقت في NumPy، حيث تكون كل قيمة طابعًا زمنيًا timestamp تم إنشاؤه لتحسين تنسيق التاريخ والوقت في Python وتخزن هذه الطوابع كأعداد صحيحة 64 بت. غالبًا ما تكون هذه الطوابع بدقة نانو ثانية ns (قراءة المزيد)

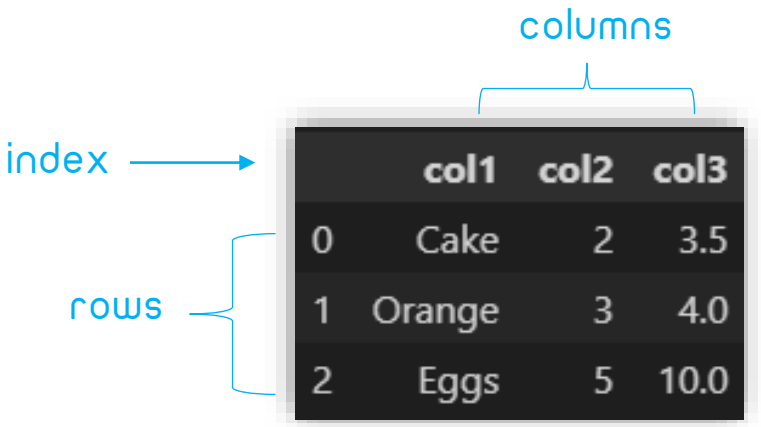
✓ سنتعرف على كل منها بشكل أفضل بالشرائح القادمة.

أنواع البيانات الأساسية للسلاسل الزمنية

Series Example:



Dataframe Example:



4- timedelta64 : هو تنسيق الفاصل الزمني لـ NumPy، والذي يمكن اعتباره فترة زمنية بين قيمتين datetime64 ويستخدم نفس الوحدات مثل datetime64. (قراءة المزيد)

5- Series: عبارة عن مصفوفة ذات بعد واحد تحتفظ بأي نوع بيانات (مثل int، str، float، إلخ) ، لكن كل عنصر يجب أن يكون من نفس النوع يُشار إلى تسميات المحور للسلسلة باسم فهرس index. (قراءة المزيد) .

6- إطار بيانات Dataframe عبارة عن بنية بيانات ثنائية الأبعاد تحتوي على أعمدة قد تتضمن أنواع بيانات مختلف يشبه إلى حد كبير جدول بيانات أو جدول SQL يُشار إلى تسميات المحور الأول لإطار البيانات (الصفوف rows) باسم فهرس السلسلة index ، بينما يُشار إلى تسميات المحور الثاني لإطار البيانات (الأعمدة columns) باسم للسلسلة.

التعبير عن الأزمنة والتواريخ في بايثون

تستخدم مكتبة datetime المضمنة مع لغة بايثون وحدات modules للتعبير عن الأزمنة والتواريخ ومعالجتها وأهم وحدتين هما : datetime و timedelta

الآن لنعبر عن الأمثلة التالية من خلال هذه المكتبة:

- تاريخ عشوائي - لنفترض 20 أبريل 2021
- التاريخ : 15 أبريل 2021 الوقت : 8 ساعات و 22 دقيقة و 43 ثانية
- التاريخ والتوقيت الحالي
- التاريخ والتوقيت في الأمس (الحالي - يوم واحد)

ملاحظة:

- الدالة datetime() : تحويل الأرقام إلى تواريخ ومواقيت قابلة للمعالجة.
- الدالة timedelta() : احتساب الفارق الزمني
- عند استخدام ال method now() يتم استخراج التاريخ والتوقيت بالطريقة التالية:
  - السنة-الشهر-اليوم- الساعة : الدقيقة : الثانية. (أجزاء الثانية micro)

```python
from datetime import datetime, timedelta

# إنشاء تاريخ عشوائي
print('Arbitrary date: ', end=' ')
print(datetime(year=2021, month=4, day=15))

# إنشاء تاريخ وتوقيت عشوائيين
print('Date and time:  ', end=' ')
print(datetime(year=2021, month=4, day=15, hour=8, minute=22, second=43))

# التاريخ والتوقيت الآن
print('Date right now: ', end=' ')
print(datetime.now())

#  استخراج التاريخ التوقيت في الأمس
print('Date yesterday: ', end=' ')
print(datetime.now() - timedelta(days=1))
```

المخرجات O/P :

```
Arbitrary date:  2021-05-15 00:00:00
Date and time:   2021-05-15 07:23:44
Date right now:  2022-05-01 00:09:46.498722
Date yesterday:  2022-04-30 00:09:46.498722
```

استخراج معلومات محددة من متغير datetime

إليك كيفية استخراج معلومات على حدة من التاريخ والوقت مثل السنة ، الشهر ، اليوم ، الساعة ، الدقيقة أو الثانية من خلال متغيرات Datetime كما هو موضح في الأسطر البرمجية جهة اليمين.

```python
from datetime import datetime, timedelta

now = datetime.now()
print(f"Current Time: {now}")
print(now.year, now.month, now.day, now.hour, now.minute, now.second)
```

هذا كل ما تحتاج إلى معرفته حيث أن العمل مع التواريخ الفردية أمر لا بد منه ، ولكنك ستعمل في الغالب خلال النطاقات الزمنية date ranges أثناء تحليل السلاسل الزمنية وهذا ما سنغطيه في الشرح التالي.

المخرجات O/P :

```
Current Time: 2022-05-01 08:37:52.348365
2022 5 1 8 37 52
```

العمل مع النطاقات الزمنية Date Ranges

عندما يتعلق الأمر بنطاقات التواريخ ، يمكنك العمل على إنشاء هذه النطاقات من خلال Python NumPy أو Pandas. الخيار الأخير هو الأبسط سنقوم بإنشاء نطاق تاريخي لمدة ثلاثة أيام

Pure Python

```
from datetime import datetime

date_list = [
    datetime(2022, 2, 13),
    datetime(2022, 2, 14),
    datetime(2022, 2, 15) ]

date_list
```

المخرجات O/P :

```
[datetime.datetime(2022, 2, 13, 0, 0),
 datetime.datetime(2022, 2, 14, 0, 0),
 datetime.datetime(2022, 2, 15, 0, 0)]
```

NumPy

تتيح مكتبة NumPy تحديد نطاقات التاريخ كمصفوفة مكونة من List و متبوعة بمواصفات نوع البيانات - datetime64

```
import numpy as np

date_list = np.array(['2022-02-13', '2022-02-14', '2022-02-15'],
                     dtype='datetime64')

date_list
```

المخرجات O/P :

```
array(['2022-02-13', '2021-02-14', '2021-02-15'], dtype='datetime64[D]')
```

العمل مع النطاقات الزمنية Date Ranges

يمكن إضافة أو حذف اللواحق الزمنية عن طريق المعامل dtype على سبيل المثال إضافة جميع اللواحق إلى التوقيت بالثانية dtype='datetime'[s]

اللواحق	القيم الظاهرة
[Y]	سنة
[M]	شهر
[D]	يوم
[h]	ساعة
[m]	دقيقة
[s]	ثانية
[ns]	النانو ثانية

```
date_list = np.array(['2022-02-13', '2022-02-14', '2022-02-15'],
           dtype='datetime64[s]')
date_list
```

```
array(['2022-02-13T00:00:00', '2022-02-14T00:00:00',
       '2022-02-15T00:00:00'], dtype='datetime64[s]')
```

```
date_list = np.array(['2022-02-13', '2022-02-14', '2022-02-15'],
           dtype='datetime64[M]')
date_list
```

```
array(['2022-02', '2022-02', '2022-02'], dtype='datetime64[M]')
```

العمل مع النطاقات الزمنية Date Ranges

ماذا لو كان لدينا نطاق زمني يتضمن 10 أيام هل سنقوم بكتابة كل تاريخ على حدة ؟

الإجابة تكمن في استخدام الدالة arrange من مكتبة NumPy فكل ما عليك هو تحديد نطاق تاريخ البدء والنهاية :

كما هو موضح تم تحديد الـ arguments التالية :

- تاريخ البدء : 2022-07-01
- تاريخ النهاية : 2022-07-10
- نوع البيانات واللاحقة الزمنية : datetime64[D]

```
date_list = np.arange('2022-07-01', '2022-07-10',
                      dtype='datetime64[D]')
date_list
```

المخرجات O/P :

```
array(['2022-07-01', '2022-07-02', '2022-07-03', '2022-07-04',
       '2022-07-05', '2022-07-06', '2022-07-07', '2022-07-08',
       '2022-07-09'], dtype='datetime64[D]')
```

العمل مع النطاقات الزمنية Date Ranges

الآن سنقوم بالأمر نفسه وهو إنشاء نطاق زمني يتضمن 10 أيام لكن هذه المرة من خلال مكتبة Pandas

لدى مكتبة Pandas دالة `date_range()` والتي تسمح لك بإنشاء نطاقات زمنية بطرق متعددة.

إنشاء نطاق زمني عن طريق تحديد :

- فترة بداية `start`
- فترة نهاية `end`

```
import pandas as pd

date_list = pd.date_range(start='2022-07-01',
                          end='2022-07-10')
date_list
```

```
DatetimeIndex(['2022-07-01', '2022-07-02', '2022-07-03', '2022-07-04',
               '2022-07-05', '2022-07-06', '2022-07-07', '2022-07-08',
               '2022-07-09', '2022-07-10'],
              dtype='datetime64[ns]', freq='D')
```

إنشاء نطاق زمني عن طريق تحديد :

- فترة بداية `start`
- عدد الفترات الزمنية `periods`
- الفاصل الزمني `freq`

عدد الفترات : 10 الفاصل الزمني : يوم

```
import pandas as pd

date_list = pd.date_range(start='2022-07-01',
                          periods=10, freq='D')
date_list
```

```
DatetimeIndex(['2022-07-01', '2022-07-02', '2022-07-03', '2022-07-04',
               '2022-07-05', '2022-07-06', '2022-07-07', '2022-07-08',
               '2022-07-09', '2022-07-10'],
              dtype='datetime64[ns]', freq='D')
```

إنشاء نطاق زمني عن طريق تحديد :

- فترة بداية `start`
- عدد الفترات الزمنية `periods`
- الفاصل الزمني `freq`

عدد الفترات : 10 الفاصل الزمني : سنة

```
import pandas as pd

date_list = pd.date_range(start='2022-07-01',
                          periods=10, freq='Y')
date_list
```

```
DatetimeIndex(['2022-12-31', '2023-12-31', '2024-12-31', '2025-12-31',
               '2026-12-31', '2027-12-31', '2028-12-31', '2029-12-31',
               '2030-12-31', '2031-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

العمل مع النطاقات الزمنية Date Ranges

الشيء الرائع الآخر في مكتبة Pandas هو القدرة على استخدام دوال مثل مثل max () min () للحصول على أقدم وأحدث التواريخ في النطاقات الزمنية على سبيل المثال :

أقدم وأحدث تاريخ في النطاق الزمني

```
date_list = pd.date_range(start='2022-07-01',
periods=10, freq='D')


print(f'Minimum date: {date_list.min()}')
print(f'Maximum date: {date_list.max()}')
```

المخرجات O/P :

```
Minimum date: 2022-07-01 00:00:00
Maximum date: 2022-07-10 00:00:00
```

الفهرسة في أطر البيانات Indexing in Data frames

الفهرس Index: توفر مكتبة Pandas الكثير من دوال الفهارس فكل Dataframe له سمة فهرس index مخصصة والتي تعمل على تصنيف الصفوف والأعمدة بشكل فريد مما يسهل من معالجة البيانات المخزنة بها

Range Index

	Type	Quantity	Price
0	Cake	2	3.5
1	Orange	3	4.0
2	Eggs	5	10.0
3	apple	4	5.0
4	Fish	5	20.0

Datetime Index

Order Date	Category	Sales
2011-01-04	Office Supplies	16.4
2011-01-05	Office Supplies	288.1
2011-01-06	Office Supplies	19.5
2011-01-07	Furniture	2,573.8
2011-01-07	Office Supplies	685.3
...	...	...
2014-12-30	Office Supplies	282.4
2014-12-30	Technology	302.4
2014-12-31	Furniture	323.1
2014-12-31	Office Supplies	299.7
2014-12-31	Technology	90.9

- أبسط أنواع الفهارس (الفهرس الافتراضي في أطر البيانات) هو RangeIndex ويكون عادة عبارة عن قائمة من القيم الصحيحة integers. غالبًا ما يتم إنشاءه تلقائيًا عندما لا يتم تعيين الفهرس بشكل صريح ، أو عند إعادة تعيين الفهرس. (قراءة المزيد) .

- عادةً ما يتم تمثيل بيانات السلاسل الزمنية بشكل أفضل باستخدام DatetimeIndex وهو فهرس لقيم datetime64 ومع ذلك ، من الملائم أحيانًا استخدام الفواصل الزمنية للفهرس باستخدام Timedelta (قراءة المزيد) .

- يوجد أيضًا ما يسمى بالفهرسة المتعددة المستويات multiple-level indexing لكن لن نتطرق إليها في السلاسل الزمنية. (قراءة المزيد)

مفاهيم أساسية في التعامل مع السلاسل الزمنية

في بعض الأحيان ، لن تكون مجموعة بيانات السلاسل الزمنية بالتنسيق الذي تريده. ربما يتم تخزين البيانات في فترات زمنية يومية ، لكنك تحتاج إلى مجاميع شهرية للتحليل. أو أن التسلسل ليس ثابتًا Non-Stationary (المصطلح الذي ستتعلمه لاحقًا) ، وتريد أن تجعله ثابتًا Stationary لإجراء توقعات صحيحة للسلاسل الزمنية أو إيجاد الفرق بين فترتين زمنيتين. كل ذلك وأكثر يمكن القيام به بسهولة مع مكتبة Pandas

في هذا الجزء ، سنستخدم مجددًا مجموعة بيانات مبيعات المركبات خفيفة الوزن إليك كيفية تحميلها باستخدام Pandas

نلاحظ أنه تم استخدام الـ arguments التالية عند قراءة ملف csv الذي يحتوي على البيانات :

- Index_col : لاختيار عمود محدد كفهرس وقد تم اختيار العمود 'Date'
- parse_dates : استنتاج تنسيق سلاسل التاريخ والوقت datetime في العمود

```python
import pandas as pd

df = pd.read_csv('datasets/LTOTALNSA.csv', index_col='DATE',
                 parse_dates=True)

df.head()
```

DATE	LTOTALNSA
1976-01-01	864.6
1976-02-01	973.3
1976-03-01	1216.1
1976-04-01	1163.2
1976-05-01	1176.1

7- إعادة التشكيل (الاختزال) Resampling

ببساطة ، تسمح لك عملية إعادة التشكيل Resampling بتغيير مستوى التجميع الزمني aggregation level لسلسلة زمنية. فعلى سبيل المثال إذا كانت لديك بيانات مجمعة على فترات زمنية ساعية Hourly ولكنك تحتاج إلى أن يكون التجميع بشكل يومي Daily ، لإجراء تحليلاتك فإن عملية إعادة التشكيل Resampling هي الحل لهذه المشكلة وهي على نوعين :

- Downsampling : إعادة التشكيل من فترات أطول إلى فترات أقصر مثال : تحويل من التجميع السنوي إلى تجميع شهري.
- Upsampling : إعادة التشكيل من فترات أقصر إلى فترات أطول مثال : تحويل من تجميع شهري إلى تجميع سنوي.

بالعودة إلى بيانات مبيعات السيارات فإن البيانات قد تم تجميعها على أساس شهري monthly من سنة 1976 إلى 2022 ففي كل شهر يتم احتساب مقدار المبيعات من السيارات كما هو في إطار البيانات الأول ، ولكن هب أنك تريد أن تجري تحليلاً سنويًا بدلاً من ذلك. دالة resample() من مكتبة pandas ستقوم بإجراء هذا التحويل لك (راقب كيف تم تغيير التجميع إلى سنوي وتم جمع المبيعات الشهرية وتحويلها إلى سنوية):

DATE	LTOTALNSA
1976-01-01	864.6
1976-02-01	973.3
1976-03-01	1216.1
1976-04-01	1163.2
1976-05-01	1176.1
1976-06-01	1224.9
1976-07-01	1130.1

أول 7 صفوف في إطار البيانات قبل التحويل

Upsampling

```
yearly_totals = df.resample(rule='Y').sum()
yearly_totals.head(7)
```

DATE	LTOTALNSA
1976-12-31	12969.8
1977-12-31	14481.9
1978-12-31	14981.0
1979-12-31	13758.4
1980-12-31	11197.5
1981-12-31	10563.6
1982-12-31	10357.3

أول 7 صفوف في إطار البيانات بعد التحويل

7- إعادة التشكيل (الاختزال) Resampling

الآن سنقوم بعكس العملية وهو استخدام الـ downsampling لتحويل التجميع السنوي إلى شهري ولكن يختلف الأمر قليلاً هنا حيث سينتج لدينا أشهر بقيم مبيعات صفرية ، لذا سنستخدم بجانب ذلك دالة interpolate() لملئ هذه الفراغات :

DATE	LTOTALNSA
1976-12-31	12969.800
1977-12-31	14481.900
1978-12-31	14981.000
1979-12-31	13758.400
1980-12-31	11197.500
1981-12-31	10563.600
1982-12-31	10357.300
1983-12-31	12107.100
1984-12-31	14205.100
1985-12-31	15425.100

Downsampling

```
monthly_totals = yearly_totals.resample(rule='M')
monthly_totals = monthly_totals.interpolate(method='spline',
                                            order=3)
monthly_totals.head(10)
```

DATE	LTOTALNSA
1976-12-31	12969.800000
1977-01-31	13120.378969
1977-02-28	13254.038620
1977-03-31	13399.044409
1977-04-30	13536.009623
1977-05-31	13673.652110
1977-06-30	13802.696216
1977-07-31	13931.331712
1977-08-31	14054.759973
1977-09-30	14168.848857

2- الإزاحة Shifting

دالة shift () في مكتبة pandas، تستخدم لإزاحة السلسلة بأكملها لأعلى أو لأسفل عن طريق تعيين مقدار الازاحة (فترة الازاحة).

إزاحة من الأعلى

```
df_shift = df.copy()
df_shift['Shift_1'] = df_shift['LTOTALNSA'].shift(1)
df_shift['Shift_2'] = df_shift['LTOTALNSA'].shift(2)
df_shift.head()
```

DATE	LTOTALNSA	Shift_1	Shift_2
1976-01-01	864.6	NaN	NaN
1976-02-01	973.3	864.6	NaN
1976-03-01	1216.1	973.3	864.6
1976-04-01	1163.2	1216.1	973.3
1976-05-01	1176.1	1163.2	1216.1

لاحظ كيف أن أول صف / صفان مفقودان ، نظرًا لعدم توفر أي سجلات قبل الأول من يناير 1976.

إزاحة من الأسفل

```
df_shift = df.copy()
df_shift['Shift_Neg1'] = df_shift['LTOTALNSA'].shift(-1)
df_shift['Shift_Neg2'] = df_shift['LTOTALNSA'].shift(-2)
df_shift.tail()
```

DATE	LTOTALNSA	Shift_Neg1	Shift_Neg2
2021-11-01	1014.411	1203.993	989.560
2021-12-01	1203.993	989.560	1045.307
2022-01-01	989.560	1045.307	1246.336
2022-02-01	1045.307	1246.336	NaN
2022-03-01	1246.336	NaN	NaN

لاحظ كيف أن آخر صف / صفان مفقودان ، نظرًا لعدم توفر أي سجلات في الأول من مارس 2022.

3- الدحرجة Rolling

يمكن أن تكون بيانات السلاسل الزمنية في التنسيق الأصلي متقلبة للغاية ، خاصة على مستويات التجميع الأصغر aggregation levels على سبيل المثال مبيعات شركة ما تتقلب بشكل كبير في الأشهر الأولى فيتم استخدام المتوسط الحسابي على هذه القيم والبدء من تاريخ معين يسمى هذا المفهوم بالدحرجة Rolling أو المتوسط المتحرك Moving Average وهي تقنية مفيدة لتنعيم Smoothing بيانات السلاسل الزمنية والتي سنخصص لها فصل متكامل. يمكنك في الواقع عمل تنبؤات للسلاسل الزمنية من خلال المتوسطات المتحركة .

إليك كيفية حساب المتوسطات المتحركة ربع السنوية والسنوية من السلسلة الأصلية:

```
df_rolling = df.copy()
df_rolling['QuarterRolling'] = df_rolling['LTOTALNSA'].rolling(window=4).mean()
df_rolling['YearRolling'] = df_rolling['LTOTALNSA'].rolling(window=12).mean()
df_rolling.head(15)
```

تم احتساب المتوسط الحسابي لأول أربع أشهر وتعيين القيمة في عمود Quarter Rolling كذلك الأمر في المتوسط المتحرك السنوي يتم احتساب المتوسط الحسابي لأول 12 قيمة لتعيينها في عمود Year Rolling

DATE	LTOTALNSA	QuarterRolling	YearRolling
1976-01-01	864.6	NaN	NaN
1976-02-01	973.3	NaN	NaN
1976-03-01	1216.1	NaN	NaN
1976-04-01	1163.2	1054.300	NaN
1976-05-01	1176.1	1132.175	NaN
1976-06-01	1224.9	1195.075	NaN
1976-07-01	1130.1	1173.575	NaN
1976-08-01	994.9	1131.500	NaN
1976-09-01	1024.9	1093.700	NaN
1976-10-01	1103.7	1063.400	NaN
1976-11-01	1062.0	1046.375	NaN
1976-12-01	1036.0	1056.650	1080.816667
1977-01-01	945.0	1036.675	1087.516667
1977-02-01	1065.4	1027.100	1095.191667
1977-03-01	1414.0	1115.100	1111.683333

4- إيجاد الفرق بين فترتين زمنتين باستخدام مفهوم Differencing

إيجاد الفرق Differencing. هي تقنية أساسية تستخدم عند التعامل مع البيانات المالية مثل أسعار الأسهم Stock prices أو إذا ما أردت احتساب العوائد المالية Revenues. بشكل عام ، يتم استخدام الـ differencing لجعل السلسلة ثابتة stationary. سيتم تغطية هذا المصطلح لاحقًا بمزيد من التوسع ولكنه يشير إلى سلسلة زمنية ذات معدل و تباين ثابتين constant variance and constant mean والذي يعد ضروريًا لبناء نماذج تنبؤية للسلاسل الزمنية.

فيما يلي كيفية إيجاد الفرق بين فترتين زمنيتين :

- فرق الرتبة الأولى First-Order Difference
- فرق الرتبة الثانية Second-Order Difference

```python
df_diff = df.copy()

# First order:   973.3 - 864.6 = 108.7
df_diff['Diff_1'] = df_diff['LTOTALNSA'].diff(periods=1)

# Second order: 1216.1 - 864.6 = 351.5
df_diff['Diff_2'] = df_diff['LTOTALNSA'].diff(periods=2)

df_diff.head()
```

Diff_1
Diff_2

DATE	LTOTALNSA	Diff_1	Diff_2
1976-01-01	864.6	NaN	NaN
1976-02-01	973.3	108.7	NaN
1976-03-01	1216.1	242.8	351.5
1976-04-01	1163.2	-52.9	189.9
1976-05-01	1176.1	12.9	-40.0

تصوير السلاسل الزمنية Time Series Visualization

وأخيرًا ، سنتعرف على عملية تصوير بيانات السلاسل الزمنية. يمكن القيام بذلك مباشرة مع مكتبة Pandas أو مع أي مكتبة أخرى.

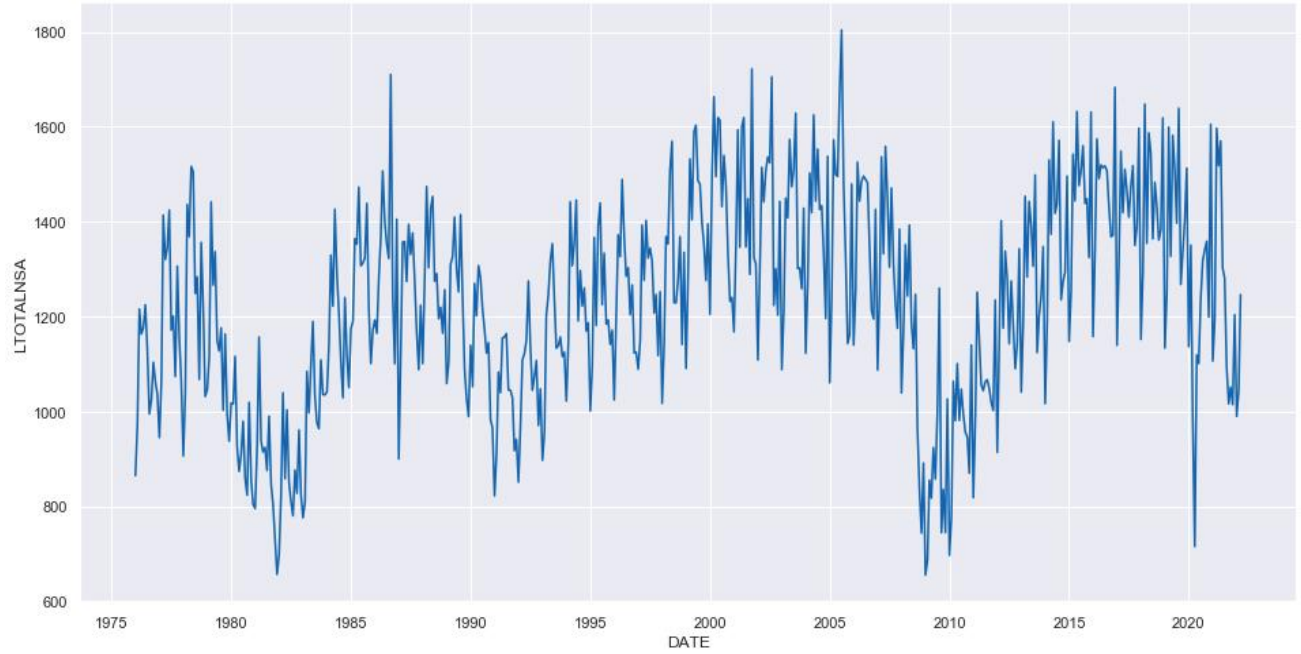
Matplotlib , Seaborn, Plotly أو Altair أو أي أداة أو مكتبة آخرة. سنستخدم في هذا الشرح كل من Seaborn و Matplotlib :

```python
# استيراد المكتبات
import matplotlib.pyplot as plt
import seaborn as sns

# تفضيلات نمط التصوير البياني
sns.set_style("dark")
sns.set(color_codes=True)

# تهيئة التصوير البياني و أبعاده
ax, fig = plt.subplots(figsize=(16,8))

# تصوير السلسلة الزمنية
sns.lineplot(data=df, x="DATE", y="LTOTALNSA",
             color='#0063B1', )
plt.show()
```



تصوير السلاسل الزمنية Time Series Visualization

رائع! ولكن ماذا لو كنت تريد تصوير جزء معين فقط من السلسلة الزمنية؟ حسنًا ، نظرًا لأنك تتعامل مع فهرس datetime، يمكنك استخدام التقطيع slicing لاجتزاء الفترة التي تريد ، و لتكن تصوير المبيعات من عام 2000 إلى عام 2020:

```python
import datetime as dt

# تحديد تاريخ البداية والنهاية
start = dt.datetime(2000, 1, 1)
end = dt.datetime(2021, 1, 1)

# اجتزاء الفترة المحددة
df_2000_2020 = df.loc[start:end]
df_2000_2020
```

DATE	LTOTALNSA
2000-01-01	1204.800
2000-02-01	1499.800
2000-03-01	1663.500
2000-04-01	1495.400
2000-05-01	1619.700
...	...
2020-09-01	1341.099
2020-10-01	1358.922
2020-11-01	1199.137
2020-12-01	1605.497
2021-01-01	1106.286

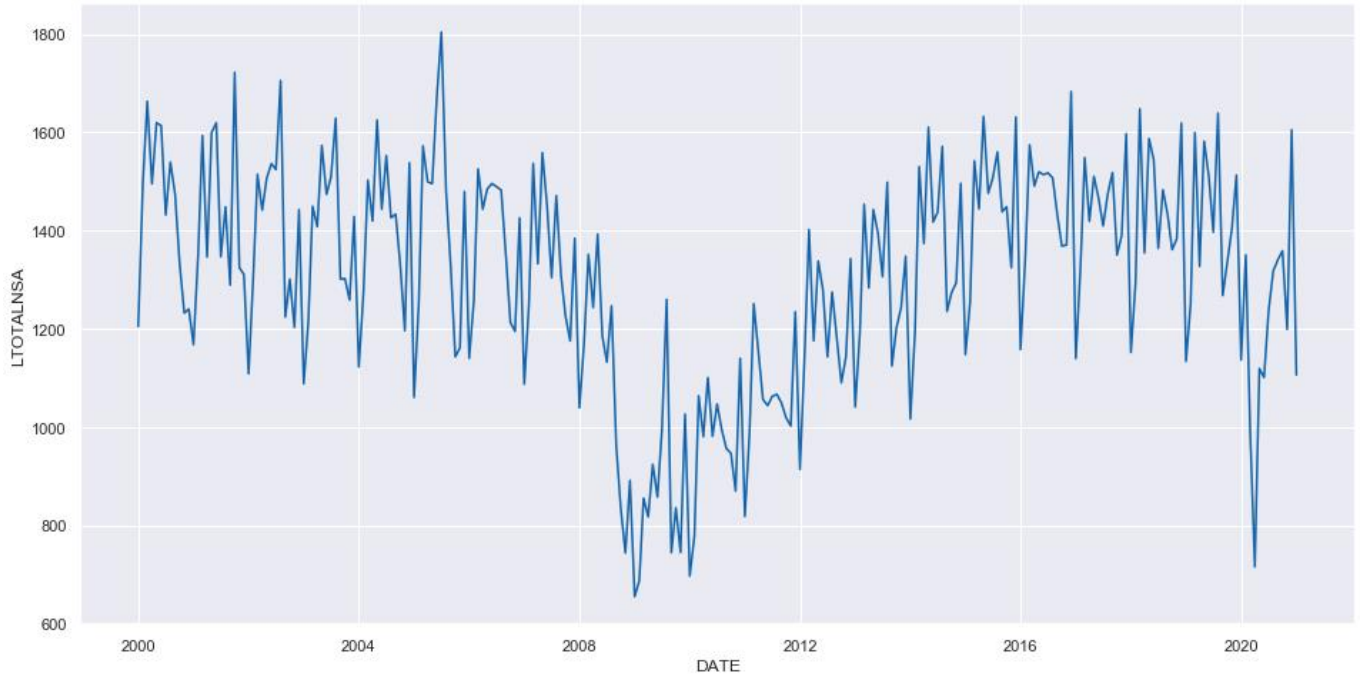
253 rows × 1 columns

تصوير السلاسل الزمنية Time Series Visualization

تصوير المبيعات من عام 2000 إلى عام 2020:

```
# تهيئة التصوير البياني و أبعاده
ax, fig = plt.subplots(figsize=(16,8))

# تصوير السلسلة الزمنية
sns.lineplot(data=df_2000_2020, x="DATE",
y="LTOTALNSA",
             color='#0063B1', )
plt.show()
```



كما أشرنا سابقًا يوجد العديد من المكتبات لتصوير بيانات السلاسل الزمنية والعديد من الطرق لاجتزاء الفترات الزمنية بإمكانك استكشافها بنفسك أثناء عملك على السلاسل الزمنية !