Ex.No-8

Time Series 1

Aim:

To implement and check Time Series functions in Python

Description:

1. Get today's, previous day and next day dates.
2. Display the data format from the raw data.
3. Replacing the day/month/year with replace function.
4. Display day/month/year from current date.

Program:

```
import datetime
res=datetime.datetime.now()
print("today-now function: ",res)

today=datetime.datetime.today()
print("\ntoday: ",today)

tomorrow=today+datetime.timedelta(days=2)
print("\ntomorrow: ",tomorrow)

yesterday=today-datetime.timedelta(days=2)
print("\nyesterday: ",yesterday)

res=datetime.datetime.now()
res1=datetime.datetime(2020,6,8,23,10,25,404040)
print("\ndate format[2020,6,8,23,10,25,404040] : ",res1)

from datetime import date
d = date(2002, 12, 31)
print("\nReplacing date:",d,d.replace(day=26))

t=date(2022, 10, 13)
```

```
d=date(2022, 10, 13).ctime()
print("\nDay and Month:",t,d)

# date object of today's date
today = date.today()
print("\nToday:",today)
print("Current year:", today.year)
print("Current month:", today.month)
print("Current day:", today.day)
```

Output:

```
today-now function:  2022-10-13 19:36:00.627771

today:  2022-10-13 19:36:00.643792

tomorrow:  2022-10-15 19:36:00.643792

yesterday:  2022-10-11 19:36:00.643792

date format[2020,6,8,23,10,25,404040] :  2020-06-08 23:10:25.404040

Replacing date: 2002-12-31 2002-12-26

Day and Month: 2022-10-13 Thu Oct 13 00:00:00 2022

Today: 2022-10-13
Current year: 2022
Current month: 10
Current day: 13
```

Time Series 2

Aim:

To implement and check Time Series functions in Python

Description:

1. Different date string formats are checked

Program:

```
import datetime
cd=datetime.datetime.now()
print(cd)

# Two-digit year
res=cd.strftime("%y")
print("\nTwo-digit year:",res)

# Four-digit year
res1=cd.strftime("%Y")
print("\nFour-digit year:",res1)

# Two-digit month [01, 12]
res=cd.strftime("%m")
print("\nTwo-digit month [01, 12]:",res)


# Short version of month
res=cd.strftime("%b")
print("\nShort version of month:",res)

# Full name of month
res=cd.strftime("%B")
print("\nFull name of month:",res)
```

```
# Days of the year
res=cd.strftime("%j")
print("\nDays of the year:",res)

# Shortcut for %m/%d/%y (e.g., 04/18/12)
res=cd.strftime("%D")
print("\nShortcut for %m/%d/%y (e.g., 04/18/12):",res)

# Two-digit day [01, 31]
res=cd.strftime("%d")
print("\nTwo-digit day [01, 31]:",res)

# Short version of day
res=cd.strftime("%a")
print("\nShort version of day:",res)

# Full name of a day
res=cd.strftime("%A")
print("\nFull name of a day:",res)


# Hour (24-hour clock) [00, 23]
res=cd.strftime("%H")
print("\nHour (24-hour clock) [00, 23]:",res)


# Hour (12-hour clock) [01, 12]
res=cd.strftime("%I")
print("\nHour (12-hour clock) [01, 12]:",res)
```

```
# Two-digit minute [00, 59]
res=cd.strftime("%M")
print("\nTwo-digit minute [00, 59]:",res)

# Second [00, 61] (seconds 60, 61 account for leap seconds)
res=cd.strftime("%S")
print("\nSecond [00, 61] (seconds 60, 61 account for leap seconds):",res)

# Shortcut for %Y-%m-%d (e.g., 2012-4-18)
res=cd.strftime("%F")
print("\nShortcut for %Y-%m-%d (e.g., 2012-4-18):",res)

# Microsecond as an integer, zero-padded (from 000000 to 999999)
res=cd.strftime("%f")
print("\nMicrosecond as an integer, zero-padded (from 000000 to 999999):",res)

# Locale equivalent of AM or PM
res=cd.strftime("%p")
print("\nLocale equivalent of AM or PM:",res)

# Locale-appropriate formatted date (e.g., in the United States, May 1, 2012 yields '05/01/2012')
res=cd.strftime("%x")
print("\nLocale-appropriate formatted date (e.g., in the United States, May 1, 2012 yields
'05/01/2012'):",res)

# Locale-appropriate time (e.g., '04:24:12 PM')
res=cd.strftime("%X")
```

```
print("\nLocale-appropriate time (e.g., '04:24:12 PM'):",res)
```

Output:

```
2022-10-13 20:05:39.398347

Two-digit year: 22

Four-digit year: 2022

Two-digit month [01, 12]: 10

Short version of month: Oct

Full name of month: October

Days of the year: 286

Shortcut for %m/%d/%y (e.g., 04/18/12): 10/13/22

Two-digit day [01, 31]: 13

Short version of day: Thu

Full name of a day: Thursday

Hour (24-hour clock) [00, 23]: 20

Hour (12-hour clock) [01, 12]: 08

Two-digit minute [00, 59]: 05
```

Second [00, 61] (seconds 60, 61 account for leap seconds): 39

Shortcut for %Y-%m-%d (e.g., 2012-4-18): 2022-10-13

Microsecond as an integer, zero-padded (from 000000 to 999999): 398347

Locale equivalent of AM or PM: PM

Locale-appropriate formatted date (e.g., in the United States, May 1, 2012 yields '05/01/2012'): 10/13/22

Locale-appropriate time (e.g., '04:24:12 PM'): 20:05:39

Time Series -3

Aim:

To implement and check Time Series functions in Python.

Description :

Implements different date time functions through DataFrame using CSV

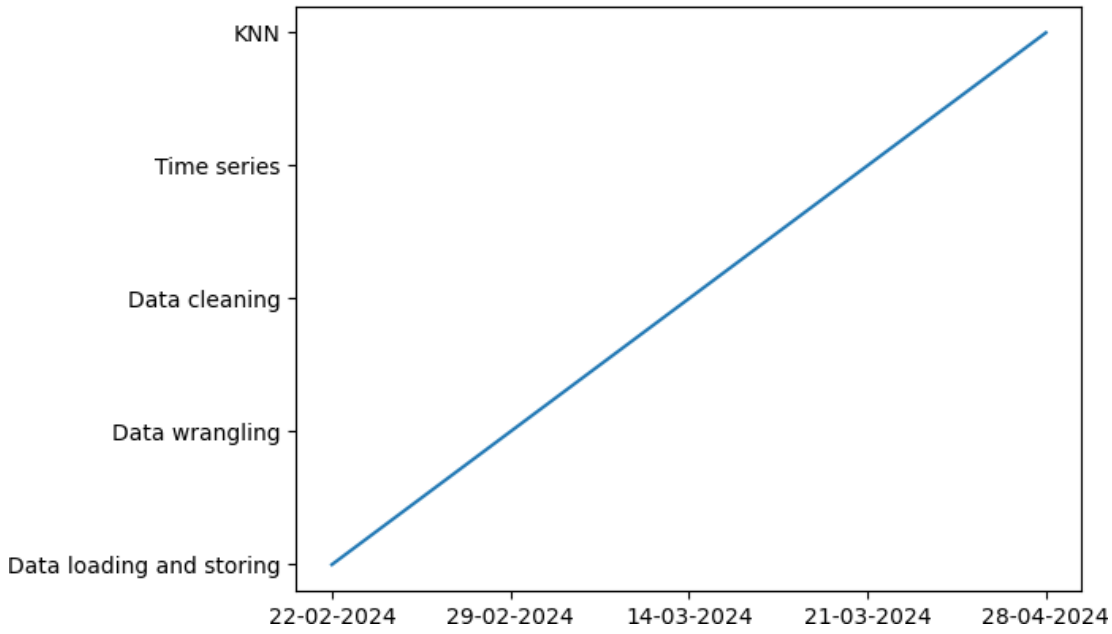
Program :

```
import pandas as pd
import datetime
import matplotlib.pyplot as plt
df = pd.read_csv("Datas.csv")
print("Original Dataframe:\n",df)
xpoints = df["Dates"]
ypoints = df["Experiment"]
plt.plot(xpoints,ypoints)
plt.show()
```

Output :

Original Dataframe:

	Dates	Experiment
0	22-02-2024	Data loading and storing
1	29-02-2024	Data wrangling
2	14-03-2024	Data cleaning
3	21-03-2024	Time series
4	28-04-2024	KNN



Result:

Hence the programs were run successfully.