Sardar Vallabhbhai National Institute of Technology

Surat-395007

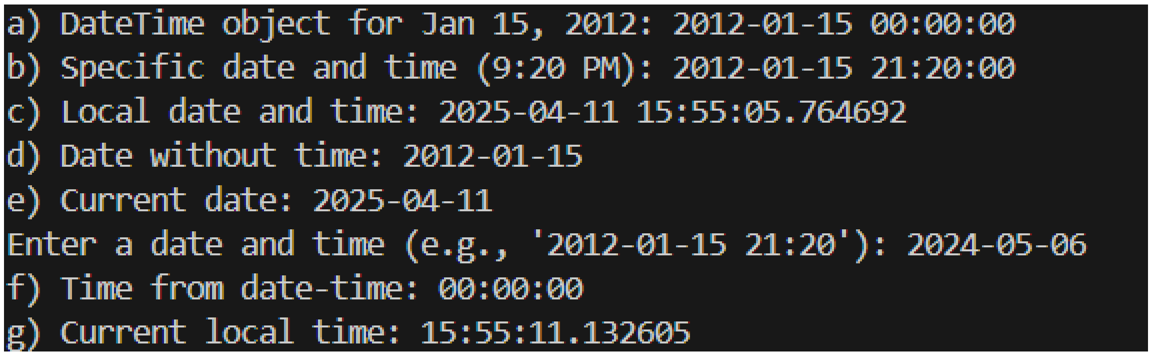
Web Programming and Python (AI104)

Assignment – 11
ROLL NO: I24AI001

1. Write a Pandas program to create
a) Date time object for Jan 15 2012.
b) Specific date and time of 9:20 pm.
c) Local date and time.
d) A date without time.
e) Current date.
t) Time from a date time.
g) Current local time.

```python
import pandas as pd
import datetime

dt_a = pd.to_datetime("2012-01-15")
print("a) DateTime object for Jan 15, 2012:", dt_a)

dt_b = pd.to_datetime("2012-01-15 21:20")
print("b) Specific date and time (9:20 PM):", dt_b)

dt_c = pd.to_datetime("now")
print("c) Local date and time:", dt_c)

dt_d = pd.to_datetime("2012-01-15").date()  # .date() to get the date part only
print("d) Date without time:", dt_d)

current_date = pd.to_datetime("today").date()  # Today's date without time
print("e) Current date:", current_date)

user_input = input("Enter a date and time (e.g., '2012-01-15 21:20'): ")
dt_f = pd.to_datetime(user_input)
time_from_dt = dt_f.time()  # Extract only the time
print("f) Time from date-time:", time_from_dt)
```

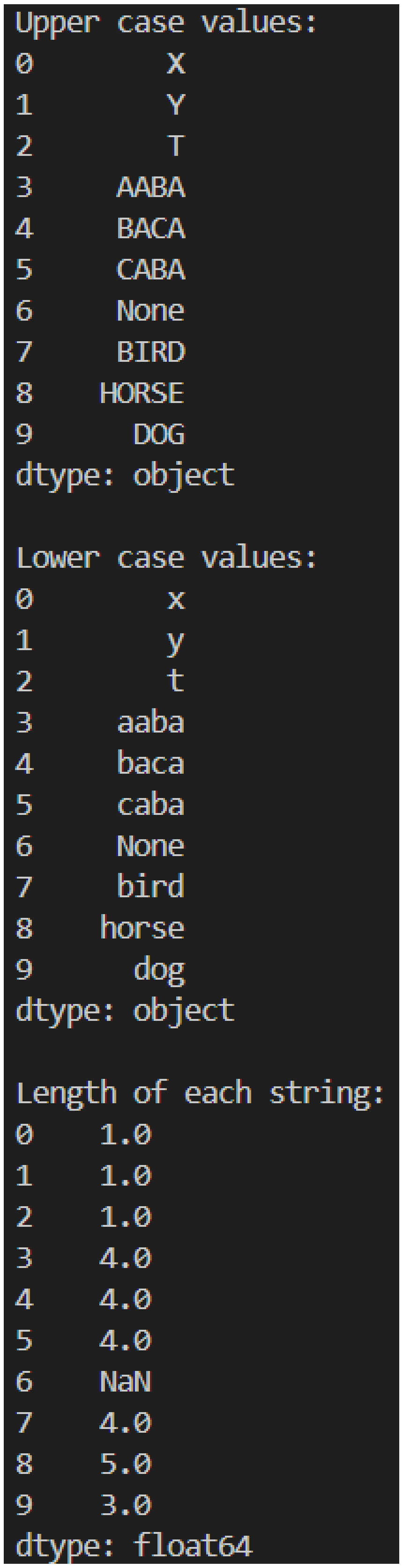
```
current_local_time = pd.to_datetime("now").time()  # Get the time part of the current local time
print("g) Current local time:", current_local_time)
```



```
a) DateTime object for Jan 15, 2012: 2012-01-15 00:00:00
b) Specific date and time (9:20 PM): 2012-01-15 21:20:00
c) Local date and time: 2025-04-11 15:55:05.764692
d) Date without time: 2012-01-15
e) Current date: 2025-04-11
Enter a date and time (e.g., '2012-01-15 21:20'): 2024-05-06
f) Time from date-time: 00:00:00
g) Current local time: 15:55:11.132605
```

2. Write a Pandas program to convert all the string values to upper, lower cases in a given pandas series. Also find the length of the string values.
s = pd.Series ([‘X’, ‘Y’, ‘T’, ‘Aaba’, ‘Baca’, ‘CABA’, None, ‘bird’, ‘horse’, ‘dog’])

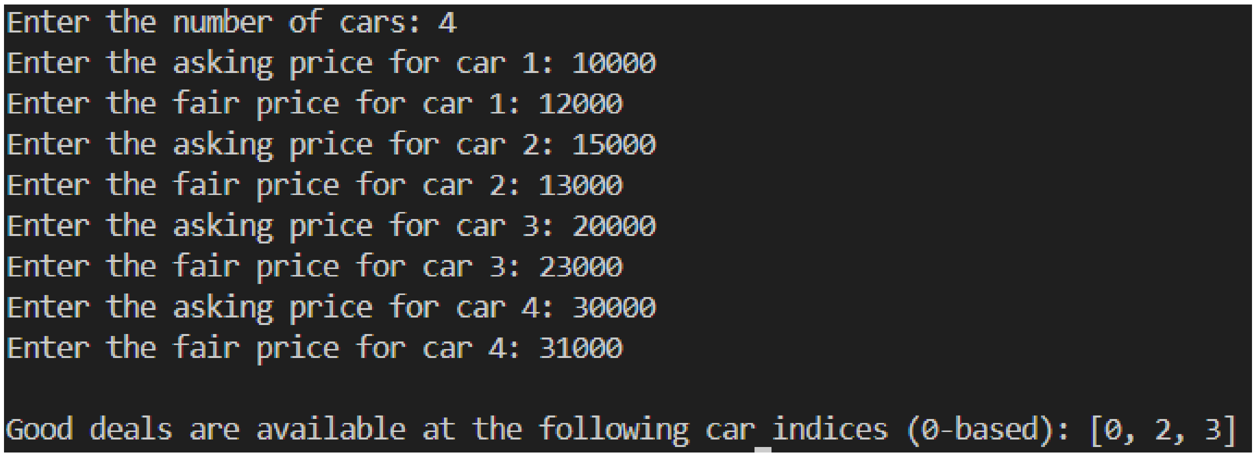
```
import pandas as pd

s = pd.Series(['X', 'Y', 'T', 'Aaba', 'Baca', 'CABA', None, 'bird', 'horse', 'dog'])

s_upper = s.str.upper()
print("Upper case values:")
print(s_upper)

s_lower = s.str.lower()
print("\nLower case values:")
print(s_lower)

s_length = s.str.len()
print("\nLength of each string:")
print(s_length)
```



```
Upper case values:
0        X
1        Y
2        T
3     AABA
4     BACA
5     CABA
6     None
7     BIRD
8    HORSE
9      DOG
dtype: object

Lower case values:
0        x
1        y
2        t
3     aaba
4     baca
5     caba
6     None
7     bird
8    horse
9      dog
dtype: object

Length of each string:
0    1.0
1    1.0
2    1.0
3    4.0
4    4.0
5    4.0
6    NaN
7    4.0
8    5.0
9    3.0
dtype: float64
```

3. After accidentally leaving an ice chest of fish and shrimp in your car for a week while you were on vacation, you're now in the market for a new vehicle. Your insurance didn't cover the loss, so you want to make sure you get a good deal on your new car.
Given a Series of car asking_prices and another Series of car fair_prices, determine which cars for sale are a good deal. In other words, identify cars whose asking price is less than their fair price.
The result should be a list of integer indices corresponding to the good deals in asking_prices.

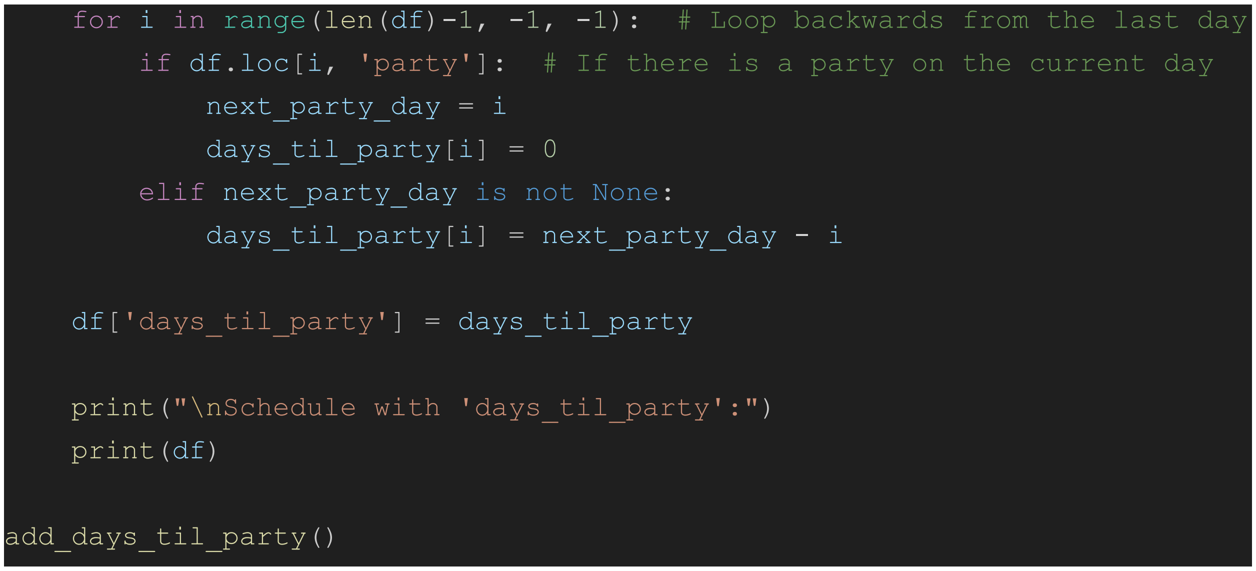
```python
import pandas as pd

def find_good_deals():
    num_cars = int(input("Enter the number of cars: "))

    asking_prices = []
    fair_prices = []

    for i in range(num_cars):
        asking_price = float(input(f"Enter the asking price for car {i+1}: "))
        fair_price = float(input(f"Enter the fair price for car {i+1}: "))
        asking_prices.append(asking_price)
        fair_prices.append(fair_price)

    asking_prices_series = pd.Series(asking_prices)
    fair_prices_series = pd.Series(fair_prices)

    good_deals = asking_prices_series < fair_prices_series

    good_deal_indices = good_deals[good_deals].index.tolist()

    if good_deal_indices:
        print("\nGood deals are available at the following car indices (0-based):", good_deal_indices)
    else:
        print("\nNo good deals found.")

find_good_deals()
```



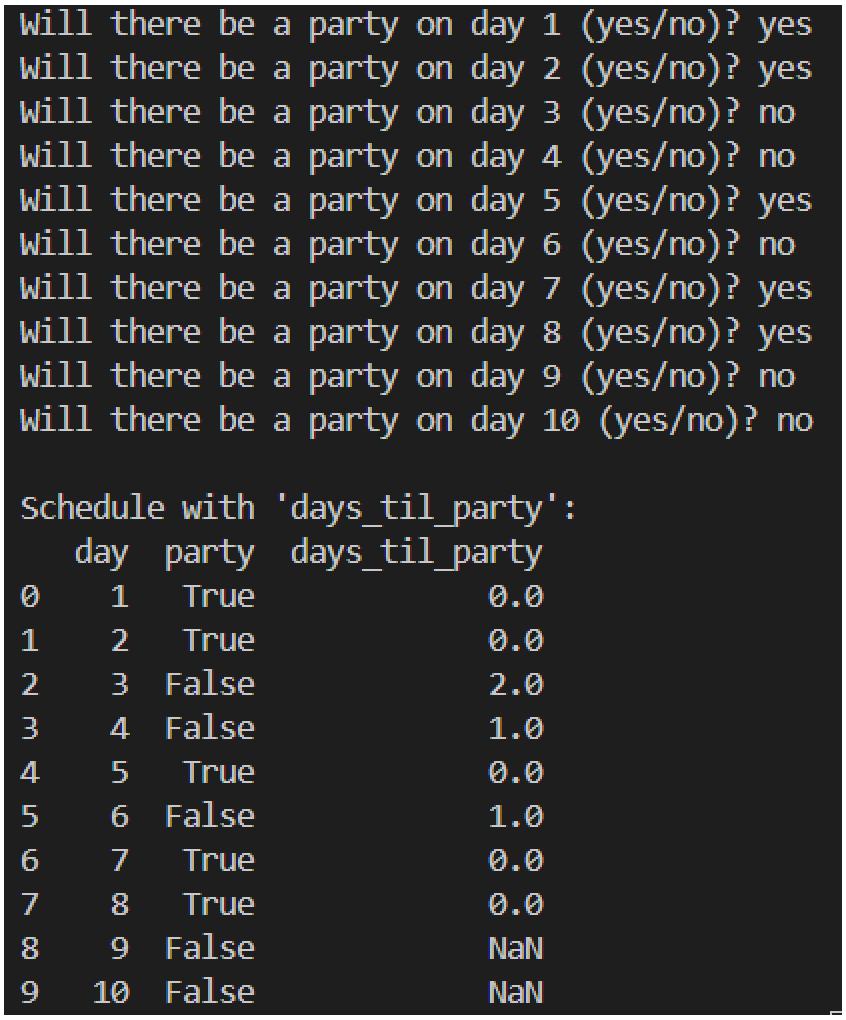
```
Enter the number of cars: 4
Enter the asking price for car 1: 10000
Enter the fair price for car 1: 12000
Enter the asking price for car 2: 15000
Enter the fair price for car 2: 13000
Enter the asking price for car 3: 20000
Enter the fair price for car 3: 23000
Enter the asking price for car 4: 30000
Enter the fair price for car 4: 31000

Good deals are available at the following car indices (0-based): [0, 2, 3]
```

4. Whenever your friends John and Judy visit you together, y'all have a party. Given a DataFrame with 10 rows representing the next 10 days of your schedule and whether John and Judy are scheduled to make an appearance, insert a new column called days_til_party that indicates how many days until the next party. days_til_party should be 0 on days when a party occurs, 1 on days when a party doesn't occur but will occur the next day, etc.

```python
import pandas as pd

def add_days_til_party():
    party_schedule = []

    for i in range(10):
        party = input(f"Will there be a party on day {i + 1} (yes/no)? ").strip().lower()

        if party == 'yes':
            party_schedule.append(True)
        else:
            party_schedule.append(False)

    df = pd.DataFrame({
        'day': range(1, 11),
        'party': party_schedule
    })

    days_til_party = [None] * len(df)

    next_party_day = None
```



```
    for i in range(len(df)-1, -1, -1):  # Loop backwards from the last day
        if df.loc[i, 'party']:  # If there is a party on the current day
            next_party_day = i
            days_til_party[i] = 0
        elif next_party_day is not None:
            days_til_party[i] = next_party_day - i

    df['days_til_party'] = days_til_party

    print("\nSchedule with 'days_til_party':")
    print(df)

add_days_til_party()
```

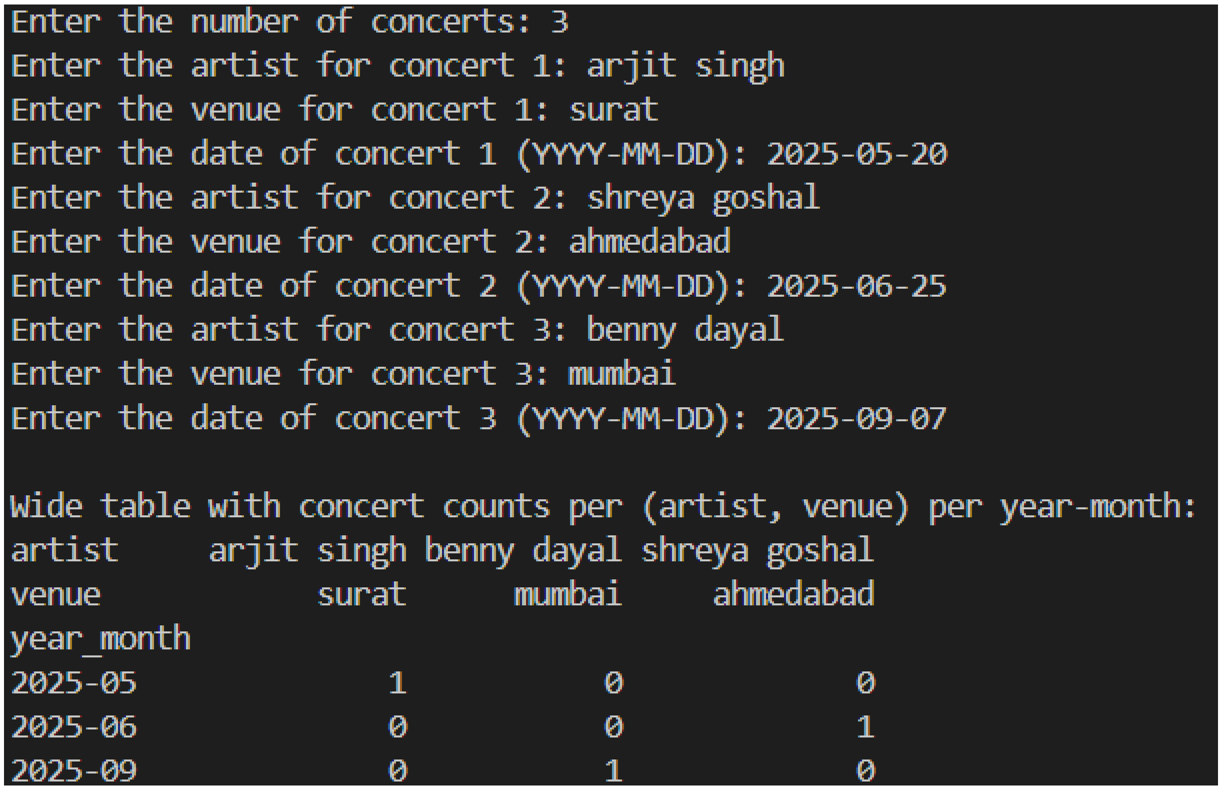


```
Will there be a party on day 1 (yes/no)? yes
Will there be a party on day 2 (yes/no)? yes
Will there be a party on day 3 (yes/no)? no
Will there be a party on day 4 (yes/no)? no
Will there be a party on day 5 (yes/no)? yes
Will there be a party on day 6 (yes/no)? no
Will there be a party on day 7 (yes/no)? yes
Will there be a party on day 8 (yes/no)? yes
Will there be a party on day 9 (yes/no)? no
Will there be a party on day 10 (yes/no)? no

Schedule with 'days_til_party':
   day  party  days_til_party
0    1   True             0.0
1    2   True             0.0
2    3  False             2.0
3    4  False             1.0
4    5   True             0.0
5    6  False             1.0
6    7   True             0.0
7    8   True             0.0
8    9  False             NaN
9   10  False             NaN
```

5. Given a dataset of concerts, count the number of concerts per (artist, venue), per year

month. Make the resulting table be a wide table - one row per year month with a column for each unique (artist, venue) pair. Use the cross product of the artists and venues Series to determine which (artist, venue) pairs to include in the result.

```
import pandas as pd
import numpy as np

def concert_count_per_artist_venue():

    num_concerts = int(input("Enter the number of concerts: "))

    artists = []
    venues = []
    dates = []

    for i in range(num_concerts):
        artist = input(f"Enter the artist for concert {i+1}: ").strip()
        venue = input(f"Enter the venue for concert {i+1}: ").strip()
        date = input(f"Enter the date of concert {i+1} (YYYY-MM-DD):
").strip()

        artists.append(artist)
        venues.append(venue)
        dates.append(date)

    df = pd.DataFrame({
        'artist': artists,
        'venue': venues,
        'date': pd.to_datetime(dates)
    })

    df['year_month'] = df['date'].dt.to_period('M')  # This will give
'YYYY-MM' format

    concert_counts = df.groupby(['year_month', 'artist',
'venue']).size().reset_index(name='concert_count')

    pivot_table = concert_counts.pivot_table(
        index='year_month',
        columns=['artist', 'venue'],
        values='concert_count',
        aggfunc='sum',
        fill_value=0  # Fill missing values with 0
    )

    print("\nWide table with concert counts per (artist, venue) per
year-month:")
    print(pivot_table)

concert_count_per_artist_venue()
```



```
Enter the number of concerts: 3
Enter the artist for concert 1: arjit singh
Enter the venue for concert 1: surat
Enter the date of concert 1 (YYYY-MM-DD): 2025-05-20
Enter the artist for concert 2: shreya goshal
Enter the venue for concert 2: ahmedabad
Enter the date of concert 2 (YYYY-MM-DD): 2025-06-25
Enter the artist for concert 3: benny dayal
Enter the venue for concert 3: mumbai
Enter the date of concert 3 (YYYY-MM-DD): 2025-09-07

Wide table with concert counts per (artist, venue) per year-month:
artist     arjit singh benny dayal shreya goshal
venue            surat      mumbai     ahmedabad
year_month
2025-05              1           0             0
2025-06              0           0             1
2025-09              0           1             0
```