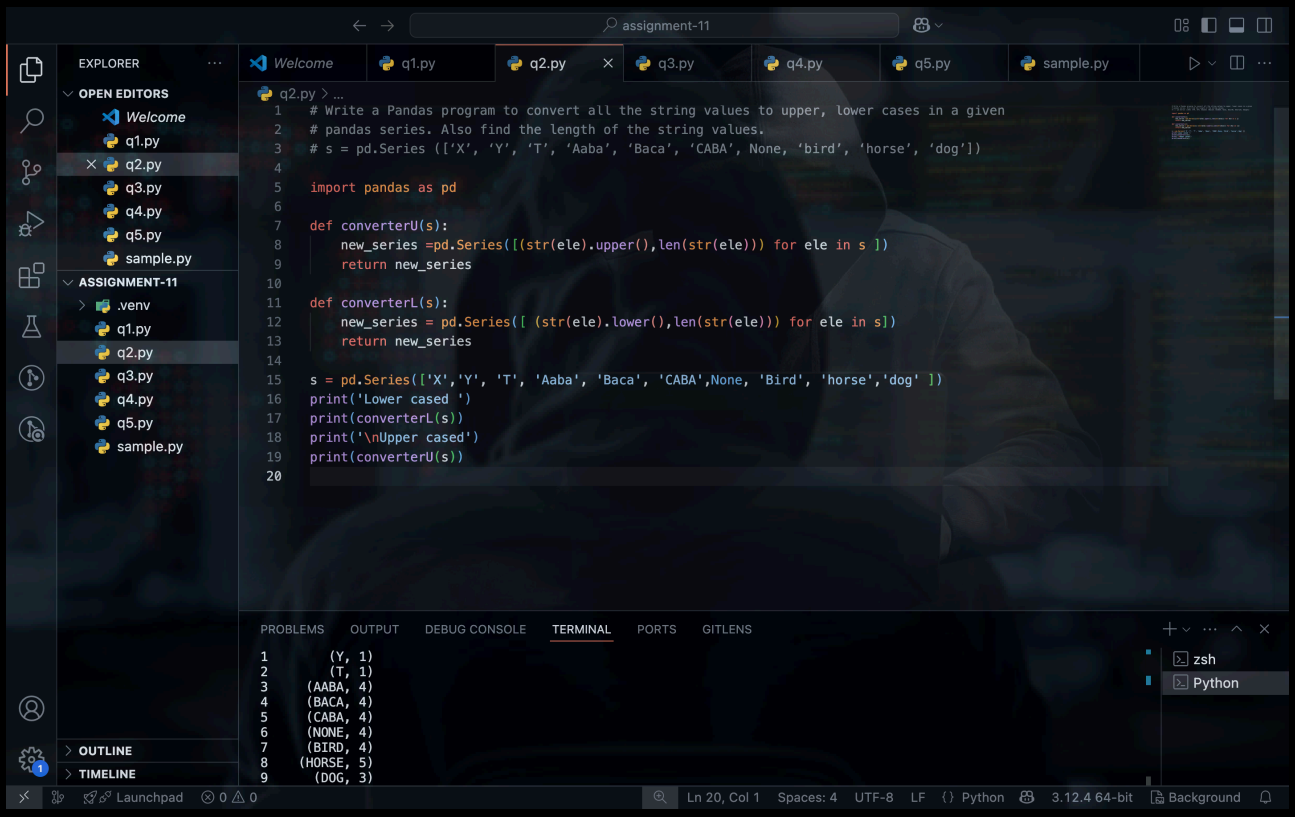
Assignment-11: U24AI038

```
# 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.

import pandas as pd

date1= pd.Timestamp("2012-01-15")
print(date1)
date1_with_time = pd.Timestamp('15-01-2012 21:20:00')
print(date1_with_time)
local_dt = pd.Timestamp.now()
print('Local date and time:',local_dt)
date_without_time = local_dt.date()
print('Date without time:',date_without_time)
current_date = pd.Timestamp.today().date()
print("Today's data:",current_date)
time = date1_with_time.time()
print(time)
current_time = local_dt.time()
print(current_time)
```

Output:

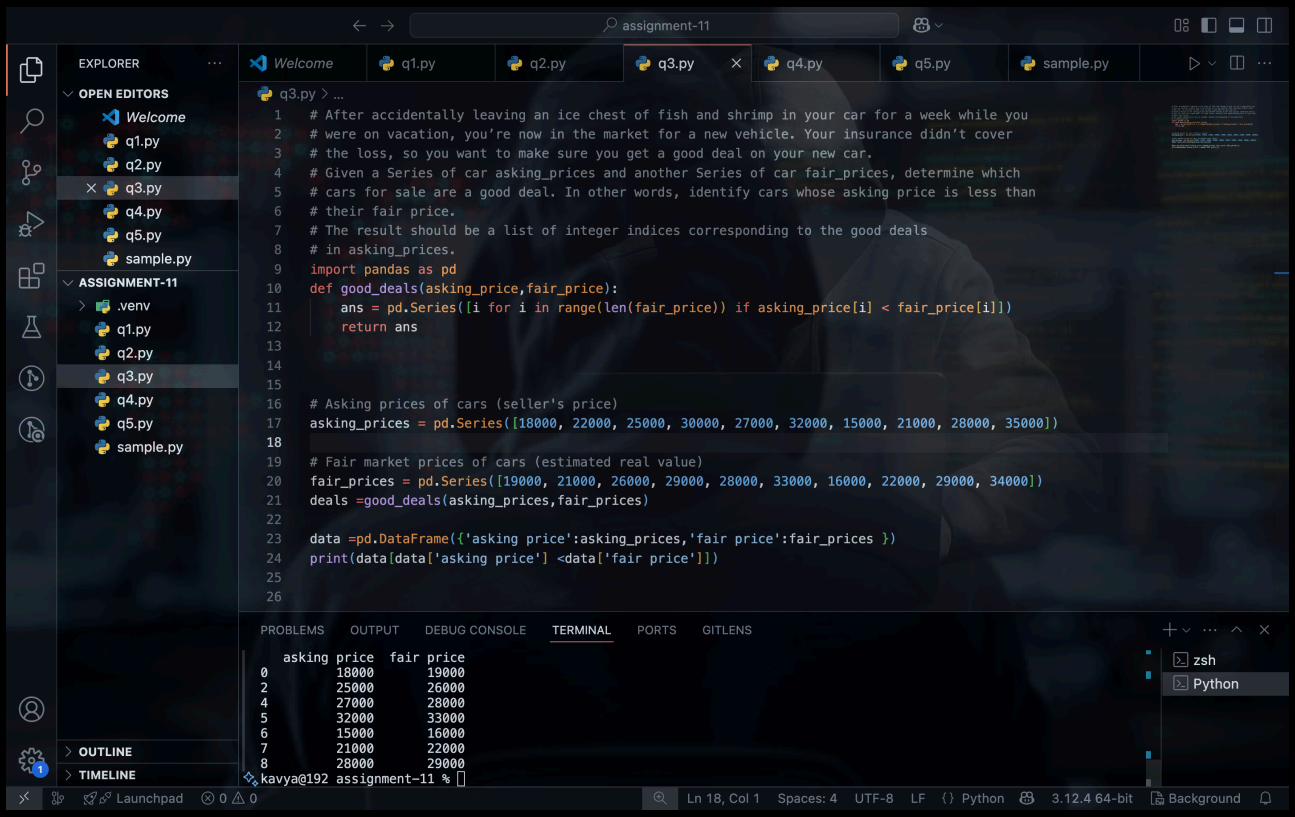
```
# Write a Pandas program to convert all the string values to
upper, lower cases in a given
# pandas series. Also find the length of the string values.
# s = pd.Series (['X', 'Y', 'T', 'Aaba', 'Baca', 'CABA', None,
'bird', 'horse', 'dog'])

import pandas as pd

def converterU(s):
    new_series =pd.Series([(str(ele).upper(),len(str(ele))) for
ele in s ])
    return new_series

def converterL(s):
    new_series = pd.Series([ (str(ele).lower(),len(str(ele))) for
ele in s])
    return new_series

s = pd.Series(['X','Y', 'T', 'Aaba', 'Baca', 'CABA',None, 'Bird',
'horse','dog' ])
print('Lower cased ')
print(converterL(s))
print('\nUpper cased')
print(converterU(s))
```

Output:



```
# 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.
import pandas as pd
def good_deals(asking_price,fair_price):
    ans = pd.Series([i for i in range(len(fair_price)) if
asking_price[i] < fair_price[i]])
    return ans



# Asking prices of cars (seller's price)
asking_prices = pd.Series([18000, 22000, 25000, 30000, 27000,
32000, 15000, 21000, 28000, 35000])

# Fair market prices of cars (estimated real value)
fair_prices = pd.Series([19000, 21000, 26000, 29000, 28000, 33000,
16000, 22000, 29000, 34000])
deals =good_deals(asking_prices,fair_prices)

data =pd.DataFrame({'asking price':asking_prices,'fair
price':fair_prices })
print(data[data['asking price'] <data['fair price’]])
```

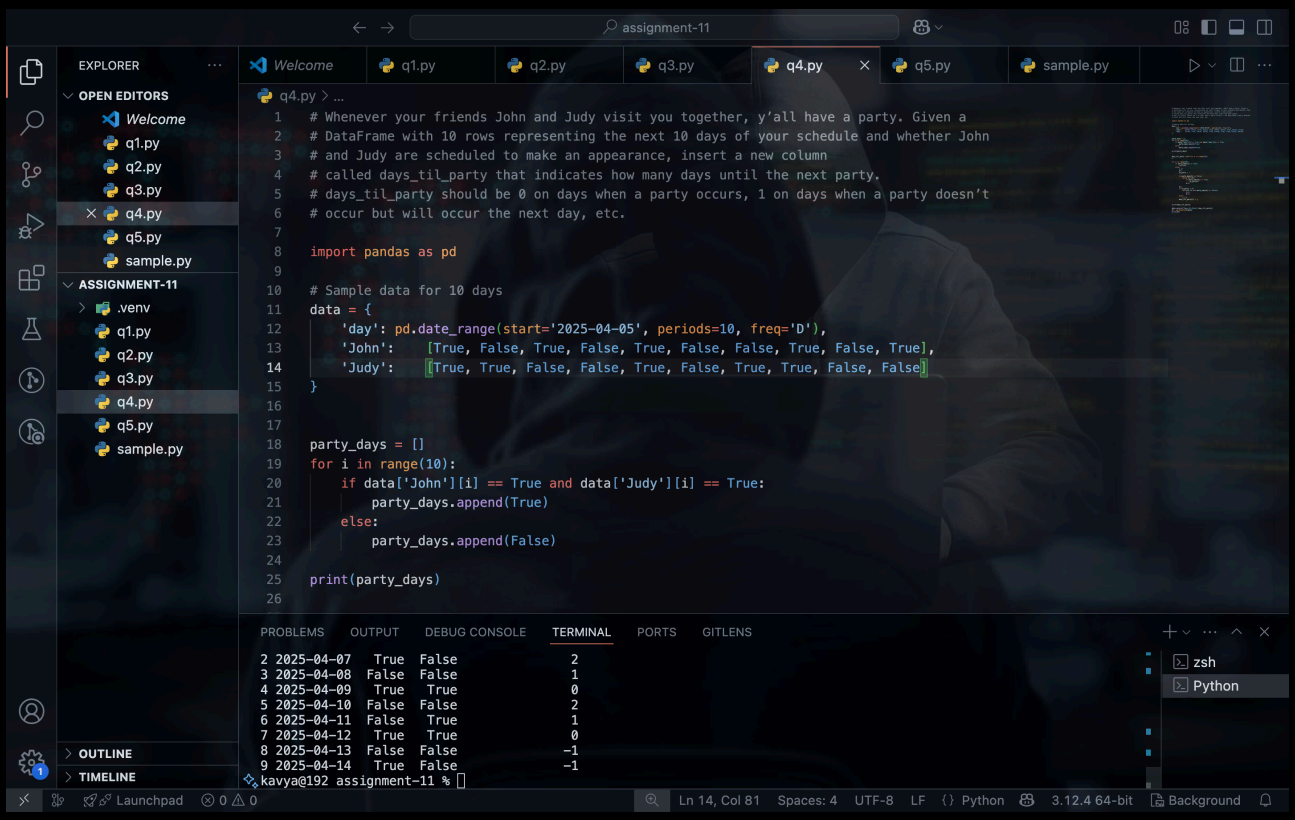
Output:




```
# 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.

import pandas as pd

# Sample data for 10 days
data = {
    'day': pd.date_range(start='2025-04-05', periods=10, freq='D'),
    'John':    [True, False, True, False, True, False, False, True, False, True],
    'Judy':    [True, True, False, False, True, False, True, True, False, False]
}
```

```
party_days = []
for i in range(10):
    if data['John'][i] == True and data['Judy'][i] == True:
        party_days.append(True)
    else:
        party_days.append(False)

print(party_days)


days_til_party =[x*0 for x in range(10)]
#

for i in range(10):
    if party_days[i] == True:
        continue
    else:
        j =0
        k=i
        no_party = 1

        if party_days[k] == False:
            while (k <10):
                if party_days[k] == True:
                    no_party=0
                k+=1

        k=i
        if no_party == 0:
            while(k <10 and party_days[k] == False):
                j+=1
                k+=1
        else:
            j =-1
        days_til_party[i] = j


print(days_til_party)

data.update({"days_til_Party":days_til_party})
df = pd.DataFrame(data)
print(df)
```

Output:




```
# 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

# Sample concert data
concerts_data = {
    'date': [
        '2024-01-05', '2024-01-10', '2024-01-15',
        '2024-02-03', '2024-02-10', '2024-02-20',
        '2024-03-01'
    ],
    'artist': [
        'Taylor Swift', 'Ed Sheeran', 'Taylor Swift',
        'Taylor Swift', 'Ed Sheeran', 'BTS',
        'BTS'
    ],
    'venue': [
```

```
        'MSG', 'MSG', 'MSG',
        'Staples Center', 'MSG', 'Staples Center',
        'MSG'
    ]
}

concerts_df = pd.DataFrame(concerts_data)

# Convert date column to datetime
concerts_df['date'] = pd.to_datetime(concerts_df['date'])

# Sample artists and venues as Series
artists = pd.Series(['Taylor Swift', 'Ed Sheeran', 'BTS'],
name='artist')
venues = pd.Series(['MSG', 'Staples Center'], name='venue')

# print(concerts_df)

year_month = []
for date in concerts_df['date']:
    month = date.month
    year = date.year
    ym = str(year) + '-' + str(month)
    year_month.append(ym)

year_month1 = pd.to_datetime(year_month)
# print(year_month1)

artist_venue_pairs = pd.MultiIndex.from_product(
    [artists, venues], names=['artist', 'venue']
).to_frame(index=False)

# print(artist_venue_pairs)

# Add year_month column to concerts_df
concerts_df['year_month'] = year_month

# Group by year_month, artist, and venue, and count concerts
grouped = concerts_df.groupby(['year_month', 'artist',
'venue']).size().reset_index(name='concert_count')

# Pivot the grouped data into a wide format
wide_table = grouped.pivot(index='year_month', columns=['artist',
'venue'], values='concert_count')

# Ensure all (artist, venue) pairs are included
wide_table =
wide_table.reindex(columns=pd.MultiIndex.from_frame(artist_venue_p
airs), fill_value=0)

# Fill missing values with zeros
wide_table = wide_table.fillna(0)
```

```
print(wide_table)
```

Output: