Task 2-Diminos Case Study

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df=pd.read_csv("C:/Users/priya/Downloads/diminos_data.csv")
```

In [3]:

```
df.shape
```

Out[3]:

```
(15000, 3)
```

In [4]:

```
df.size
```

Out[4]:

```
45000
```

In [5]:

```
df.head()
```

Out[5]:

	order_id	order_placed_at	order_delivered_at
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056

In [6]:

```
df.tail()
```

Out[6]:

	order_id	order_placed_at	order_delivered_at
14995	1538106	2023-03-27 23:37:05	2023-03-27 23:52:37.409378
14996	1538107	2023-03-27 23:47:38	2023-03-28 00:04:22.672912
14997	1538108	2023-03-27 23:50:16	2023-03-28 00:05:40.676238
14998	1538109	2023-03-27 23:52:44	2023-03-28 00:08:41.810358
14999	1538110	2023-03-27 23:58:20	2023-03-28 00:13:42.499311

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 3 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            15000 non-null  int64 
 1   order_placed_at     15000 non-null  object
 2   order_delivered_at  15000 non-null  object
dtypes: int64(1), object(2)
memory usage: 351.7+ KB
```

In [8]:

```
df.describe()
```

Out[8]:

	order_id
count	1.500000e+04
mean	1.530610e+06
std	4.330271e+03
min	1.523111e+06
25%	1.526861e+06
50%	1.530610e+06
75%	1.534360e+06
max	1.538110e+06

Type Casting from object to datetime

In [9]:

```
df['order_placed_at']=pd.to_datetime(df['order_placed_at'])
df['order_delivered_at']=pd.to_datetime(df['order_delivered_at'])
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 3 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   order_id            15000 non-null  int64         
 1   order_placed_at     15000 non-null  datetime64[ns]
 2   order_delivered_at  15000 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(1)
memory usage: 351.7 KB
```

Number of day of month

In [11]:

```
df['order_placed_at'].dt.day
```

Out[11]:

```
0         1
1         1
2         1
3         1
4         1
         ..
14995    27
14996    27
14997    27
14998    27
14999    27
Name: order_placed_at, Length: 15000, dtype: int64
```

NUmber of month of year

In [12]:

```
df['order_placed_at'].dt.month
```

Out[12]:

```
0        3
1        3
2        3
3        3
4        3
        ..
14995    3
14996    3
14997    3
14998    3
14999    3
Name: order_placed_at, Length: 15000, dtype: int64
```

Number of week of year(for week in month(-8))

In [13]:

```
df['order_placed_at'].dt.week
```

```
C:\Users\priya\AppData\Local\Temp\ipykernel_14652\1912616968.py:1: FutureW
arning: Series.dt.weekofyear and Series.dt.week have been deprecated. Plea
se use Series.dt.isocalendar().week instead.
  df['order_placed_at'].dt.week
```

Out[13]:

```
0         9
1         9
2         9
3         9
4         9
         ..
14995    13
14996    13
14997    13
14998    13
14999    13
Name: order_placed_at, Length: 15000, dtype: int64
```

Number of day of week on which order is placed

In [14]:

```
df['order_placed_at'].dt.dayofweek
```

Out[14]:

```
0        2
1        2
2        2
3        2
4        2
        ..
14995    0
14996    0
14997    0
14998    0
14999    0
Name: order_placed_at, Length: 15000, dtype: int64
```

Number of day of month on which order is placed

In [15]:

```
df['day_of_month']=df['order_placed_at'].dt.day
```

In [16]:

```
df.head()
```

Out[16]:

	order_id	order_placed_at	order_delivered_at	day_of_month
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132	1
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241	1
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385	1
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399	1
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056	1

In [17]:

```
df.tail()
```

Out[17]:

	order_id	order_placed_at	order_delivered_at	day_of_month
14995	1538106	2023-03-27 23:37:05	2023-03-27 23:52:37.409378	27
14996	1538107	2023-03-27 23:47:38	2023-03-28 00:04:22.672912	27
14997	1538108	2023-03-27 23:50:16	2023-03-28 00:05:40.676238	27
14998	1538109	2023-03-27 23:52:44	2023-03-28 00:08:41.810358	27
14999	1538110	2023-03-27 23:58:20	2023-03-28 00:13:42.499311	27

Name of the day on which order is placed

In [18]:

```
df['order_placed_at'].dt.day_name()
```

Out[18]:

```
0        Wednesday
1        Wednesday
2        Wednesday
3        Wednesday
4        Wednesday
           ...
14995       Monday
14996       Monday
14997       Monday
14998       Monday
14999       Monday
Name: order_placed_at, Length: 15000, dtype: object
```

In [19]:

```
df['day_of_week']=df['order_placed_at'].dt.day_name()
```

In [20]:

```
df.head()
```

Out[20]:

	order_id	order_placed_at	order_delivered_at	day_of_month	day_of_week
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132	1	Wednesday
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241	1	Wednesday
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385	1	Wednesday
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399	1	Wednesday
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056	1	Wednesday

In [21]:

```
df.tail()
```

Out[21]:

	order_id	order_placed_at	order_delivered_at	day_of_month	day_of_week
14995	1538106	2023-03-27 23:37:05	2023-03-27 23:52:37.409378	27	Monday
14996	1538107	2023-03-27 23:47:38	2023-03-28 00:04:22.672912	27	Monday
14997	1538108	2023-03-27 23:50:16	2023-03-28 00:05:40.676238	27	Monday
14998	1538109	2023-03-27 23:52:44	2023-03-28 00:08:41.810358	27	Monday
14999	1538110	2023-03-27 23:58:20	2023-03-28 00:13:42.499311	27	Monday

In [22]:

```
df['day_of_week'].value_counts()
```

Out[22]:

```
Thursday     2280
Wednesday    2259
Monday       2256
Friday       2223
Saturday     2209
Sunday       2191
Tuesday      1582
Name: day_of_week, dtype: int64
```

In [23]:

```
df['day_of_month'].value_counts()
```

Out[23]:

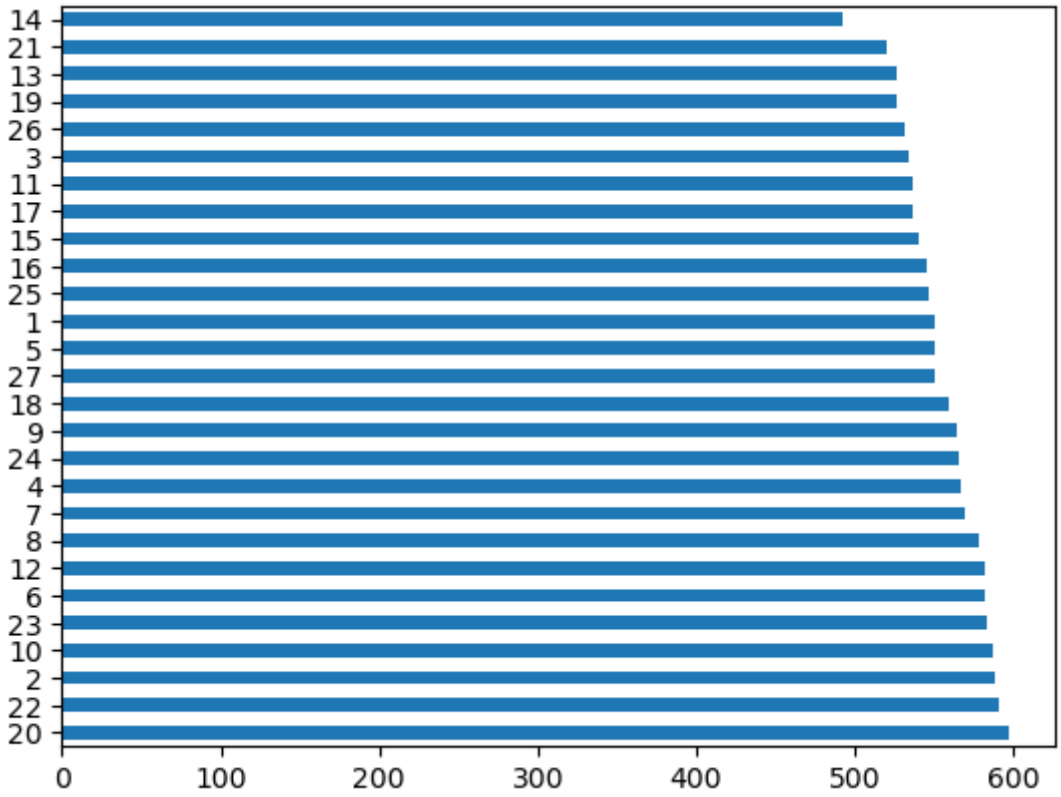
```
20    597
22    591
2     588
10    587
23    583
6     582
12    582
8     578
7     570
4     567
24    565
9     564
18    559
27    551
5     551
1     550
25    547
16    545
15    540
17    537
11    536
3     534
26    532
19    526
13    526
21    520
14    492
Name: day_of_month, dtype: int64
```

In [24]:

```
df['day_of_month'].value_counts().plot(kind='barh')
```

Out[24]:

<AxesSubplot:>

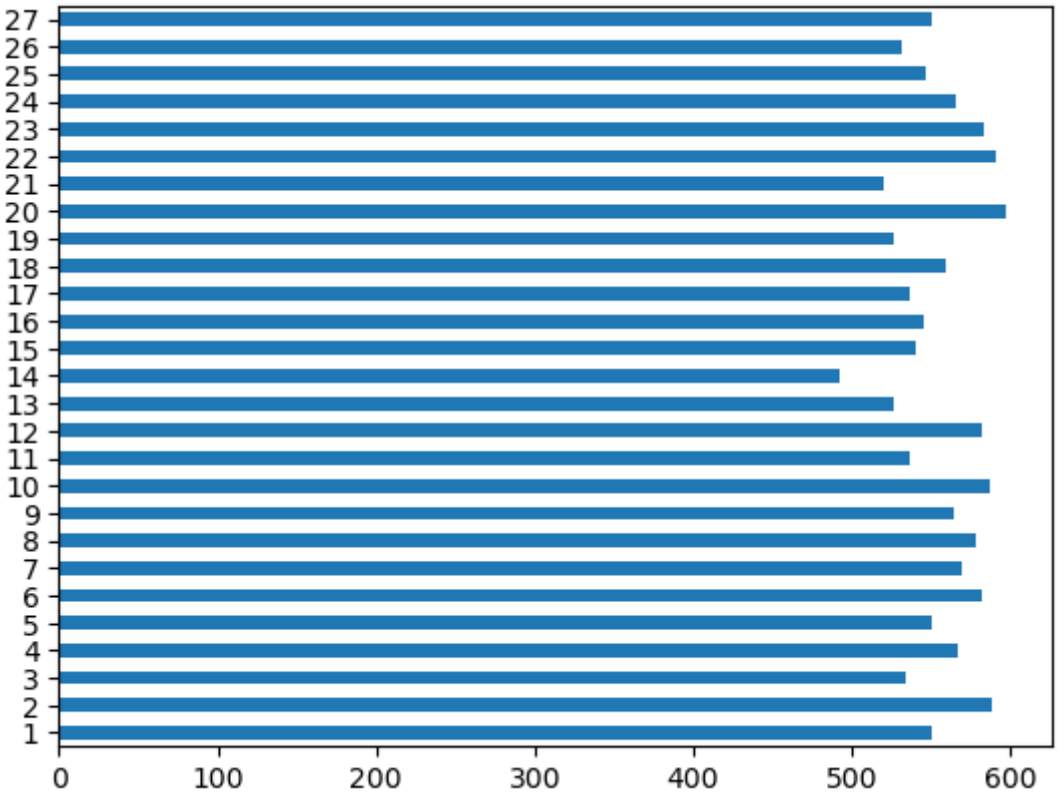


In [25]:

```
df['day_of_month'].value_counts().sort_index().plot(kind='barh')
```

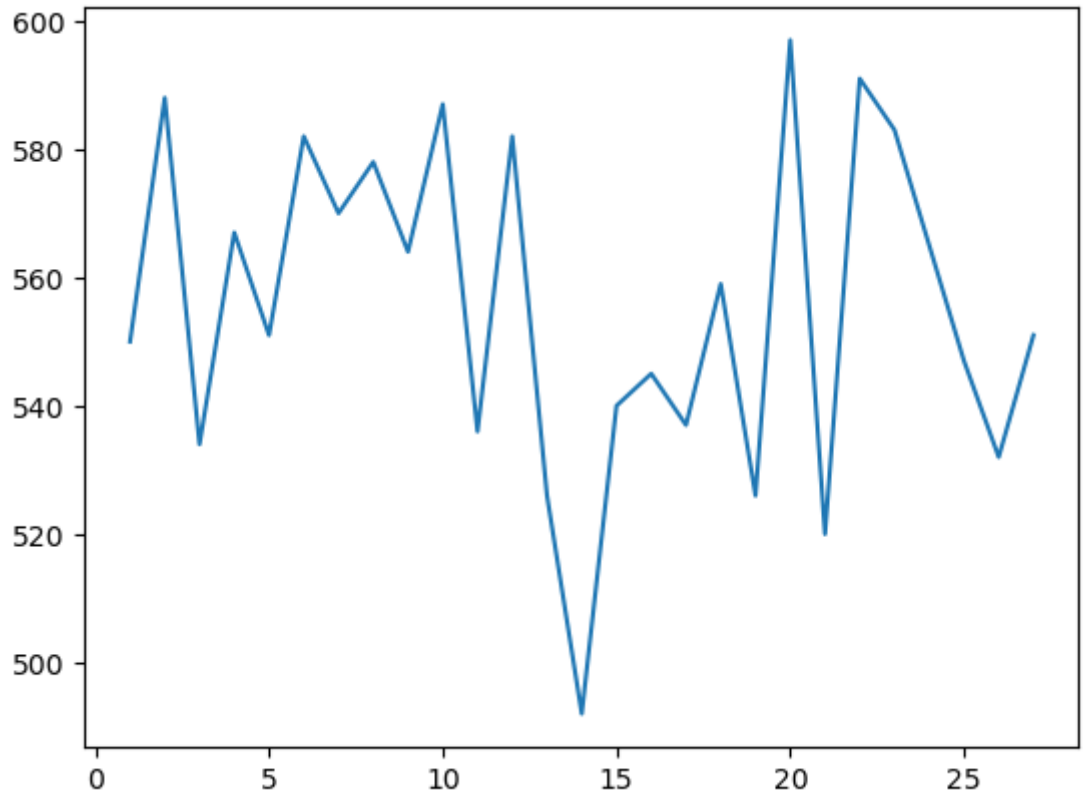
Out[25]:

<AxesSubplot:>



In [26]:

```
df['day_of_month'].value_counts().sort_index().plot(kind='line');
```

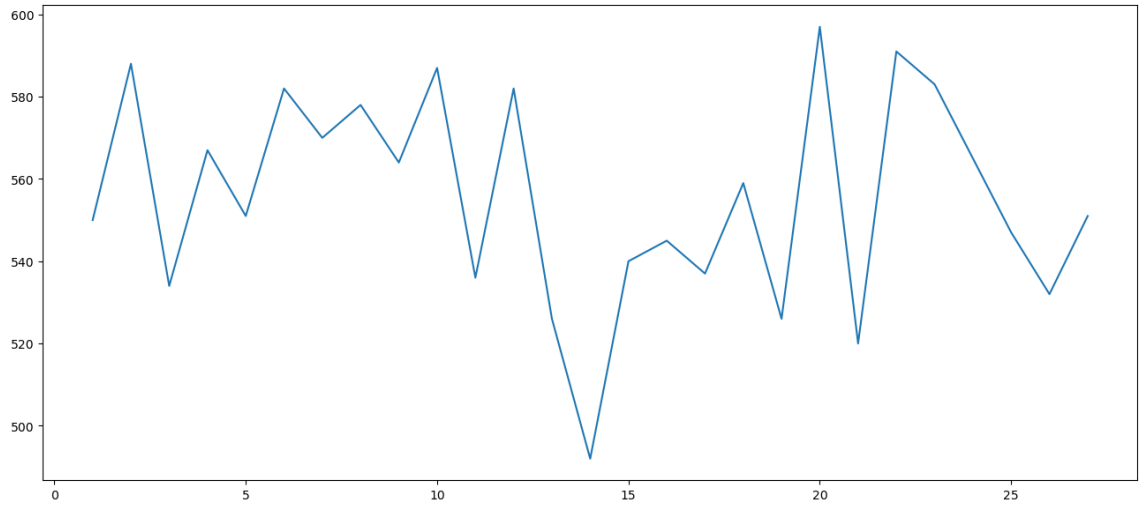


In [27]:

```
df['day_of_month'].value_counts().sort_index().plot(kind='line',figsize=(16,7))
```

Out[27]:

<AxesSubplot:>

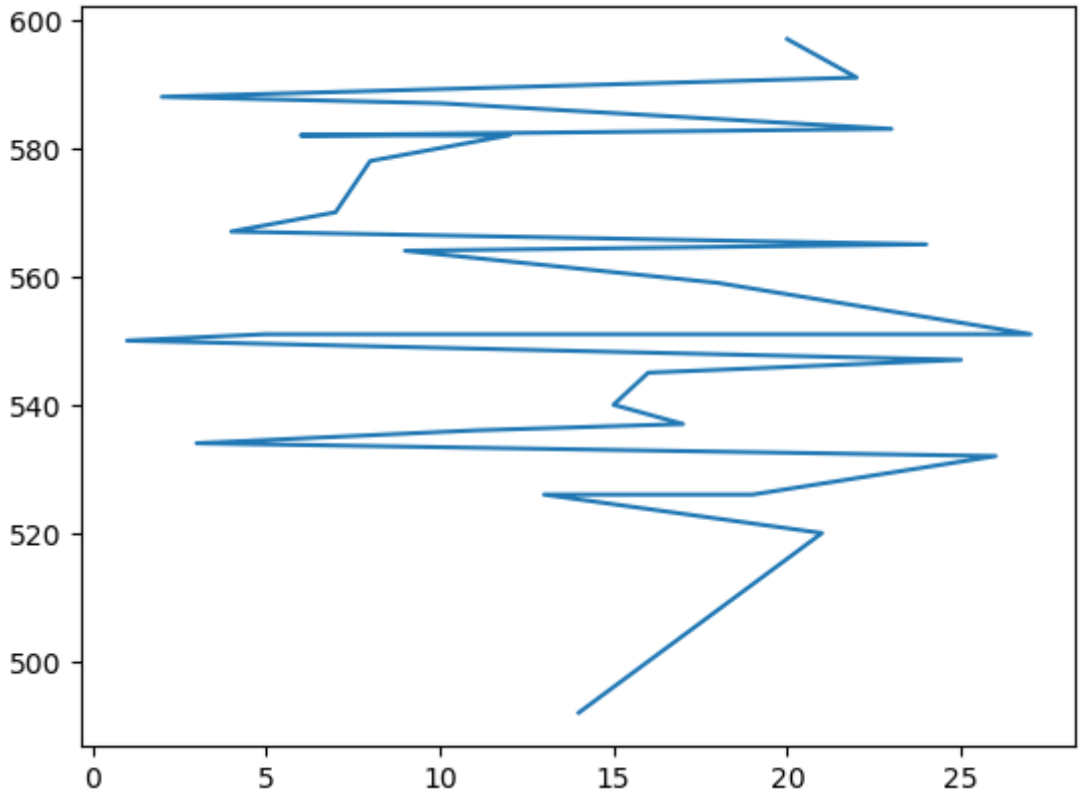


In [28]:

```
df['day_of_month'].value_counts().plot(kind='line')
```

Out[28]:

<AxesSubplot:>

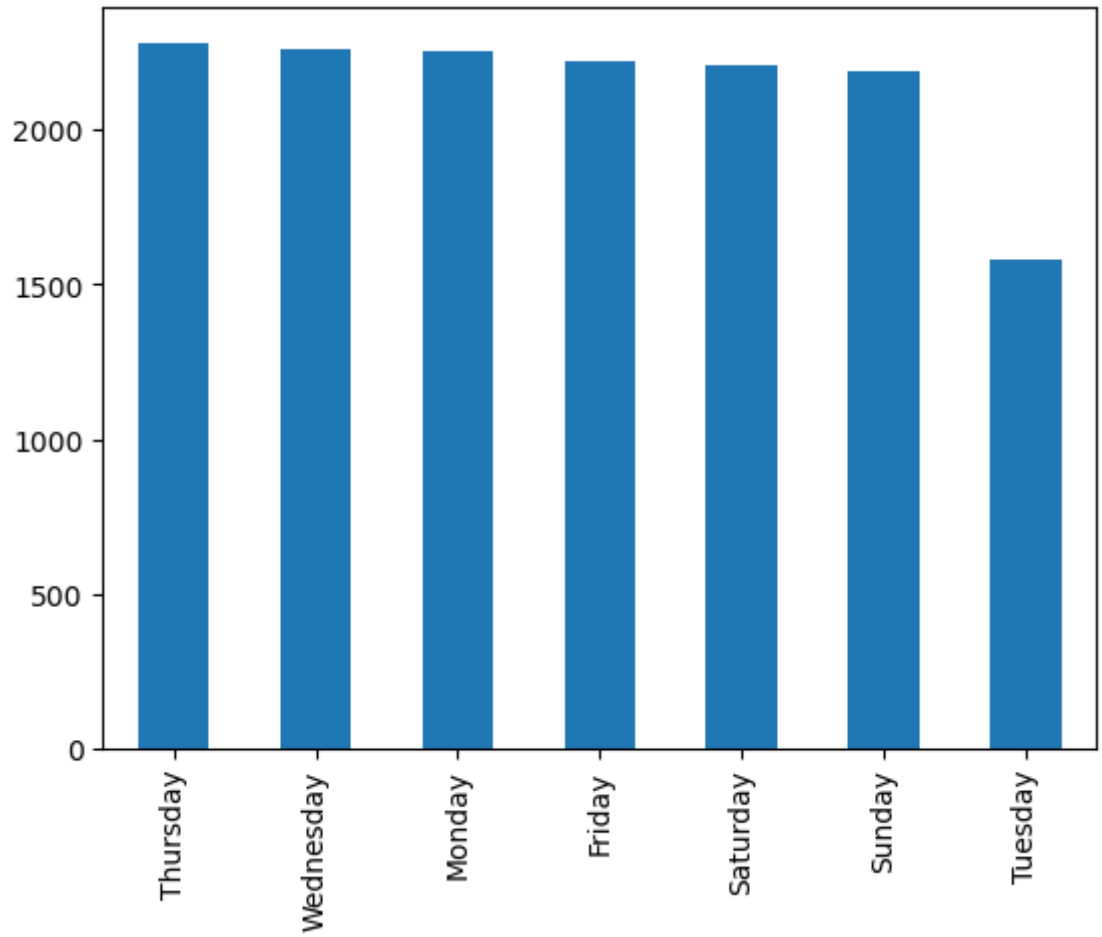


In [29]:

```
df['day_of_week'].value_counts().plot(kind='bar')
```

Out[29]:

<AxesSubplot:>



In [30]:

```
df.head()
```

Out[30]:

	order_id	order_placed_at	order_delivered_at	day_of_month	day_of_week
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132	1	Wednesday
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241	1	Wednesday
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385	1	Wednesday
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399	1	Wednesday
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056	1	Wednesday

Calculation of Delivery time

In [31]:

```
df['delivery_time']=df['order_delivered_at']-df['order_placed_at']
df.head()
```

Out[31]:

	order_id	order_placed_at	order_delivered_at	day_of_month	day_of_week	delivery_time
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132	1	Wednesday	0 days 00:17:08.443132
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241	1	Wednesday	0 days 00:15:35.925241
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385	1	Wednesday	0 days 00:15:06.291385
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399	1	Wednesday	0 days 00:38:32.019399
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056	1	Wednesday	0 days 00:16:10.619056

Calculating it in seconds

In [32]:

```
df['delivery_time'].dt.total_seconds()
```

Out[32]:

```
0        1028.443132
1         935.925241
2         906.291385
3        2312.019399
4         970.619056
            ...
14995     932.409378
14996    1004.672912
14997     924.676238
14998     957.810358
14999     922.499311
Name: delivery_time, Length: 15000, dtype: float64
```

Calculating time in minutes

In [33]:

```
df['delivery_time']=df['delivery_time'].dt.total_seconds()/60
```

In [34]:

```
df.head()
```

Out[34]:

	order_id	order_placed_at	order_delivered_at	day_of_month	day_of_week	delivery_time
0	1523111	2023-03-01 00:00:59	2023-03-01 00:18:07.443132	1	Wednesday	17.140719
1	1523112	2023-03-01 00:03:59	2023-03-01 00:19:34.925241	1	Wednesday	15.598754
2	1523113	2023-03-01 00:07:22	2023-03-01 00:22:28.291385	1	Wednesday	15.104856
3	1523114	2023-03-01 00:07:47	2023-03-01 00:46:19.019399	1	Wednesday	38.533657
4	1523115	2023-03-01 00:09:03	2023-03-01 00:25:13.619056	1	Wednesday	16.176984

In [35]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   order_id            15000 non-null  int64         
 1   order_placed_at     15000 non-null  datetime64[ns]
 2   order_delivered_at  15000 non-null  datetime64[ns]
 3   day_of_month        15000 non-null  int64         
 4   day_of_week         15000 non-null  object        
 5   delivery_time       15000 non-null  float64       
dtypes: datetime64[ns](2), float64(1), int64(2), object(1)
memory usage: 703.2+ KB
```

In [36]:

```
df['delivery_time'].describe()
```

Out[36]:

```
count    15000.000000
mean        20.499389
std         96.160362
min         15.000010
25%         15.274826
50%         15.797986
75%         17.279661
max       7299.831375
Name: delivery_time, dtype: float64
```

In [37]:

```
df['delivery_time'].median()
```

Out[37]:

```
15.797986066666668
```

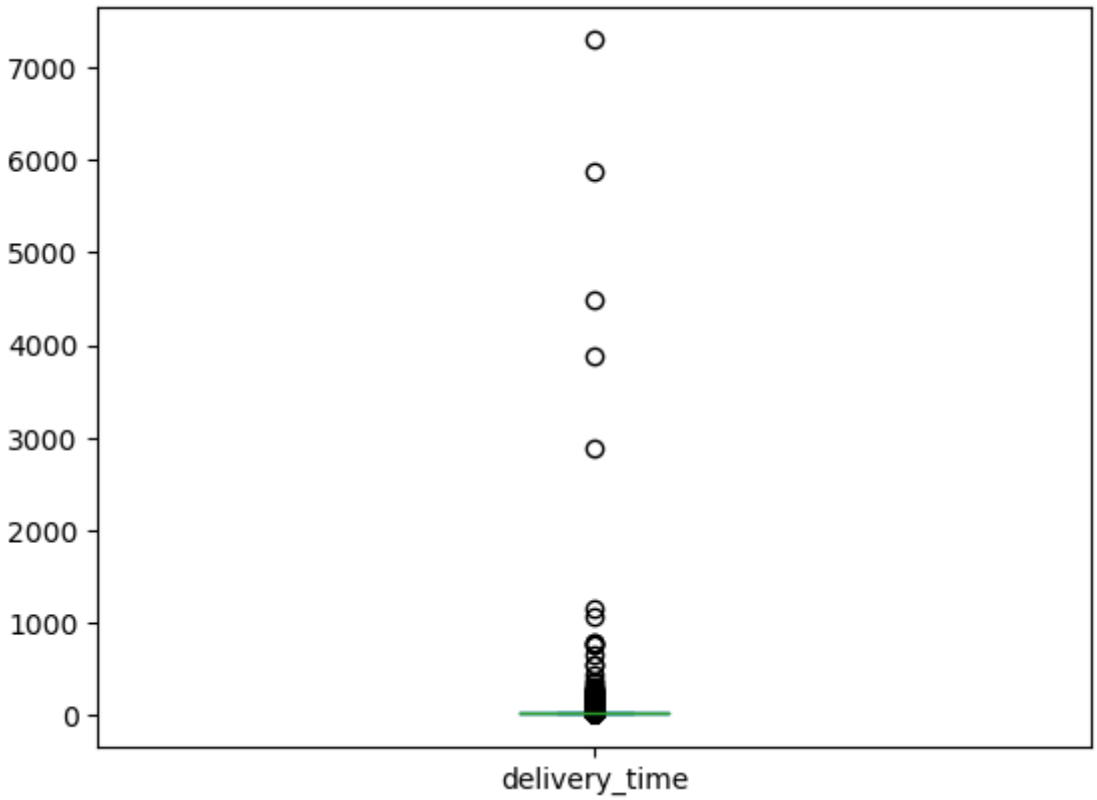
Checking for outliers

In [38]:

```
df['delivery_time'].plot(kind='box')
```

Out[38]:

```
<AxesSubplot:>
```



Removing the outliers

In [39]:

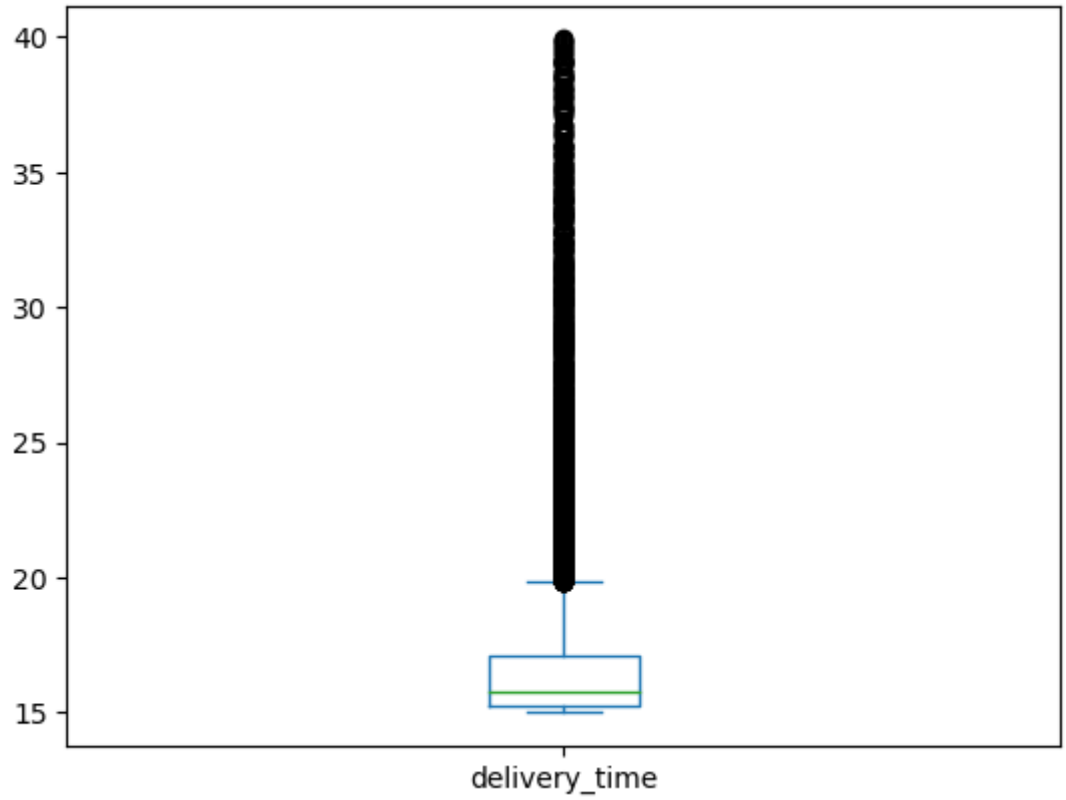
```
df=df[df['delivery_time']<40]
```

In [40]:

```
df['delivery_time'].plot(kind='box')
```

Out[40]:

```
<AxesSubplot:>
```



In [41]:

```
df['delivery_time'].quantile(0.95)
```

Out[41]:

```
23.596276583333314
```

In [42]:

```
df.shape
```

Out[42]:

```
(14669, 6)
```

In [43]:

```
df[df['delivery_time']>31].shape
```

Out[43]:

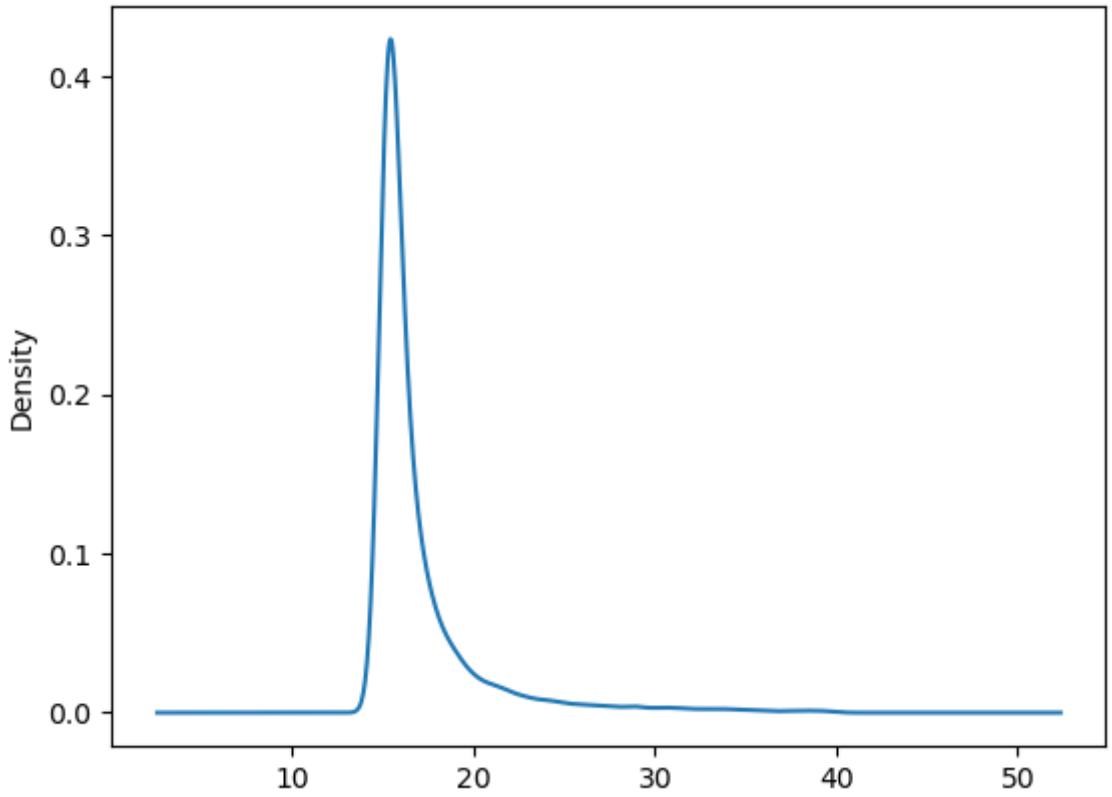
```
(226, 6)
```

In [44]:

```
df['delivery_time'].plot(kind='kde')
```

Out[44]:

```
<AxesSubplot:ylabel='Density'>
```



Now,it is ok as 95 percentile deliveries are in range but around 500 pizzas are given for free that is loss of around 5lakh in a month. It is loss in bussiness. Suggetions I can give:- 1.Improvement should be in delivery time. 2.More hubs should be set in areas so that delivery take less time.