In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\v mouli\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0
is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [3]:
```
data=pd.read_csv('Dataset11-Weather-Data.csv')
data.head(5)
```

Out[3]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2	1/1/2012 2:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
3	1/1/2012 3:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog

In [4]:
```
data.shape
```

Out[4]:
```
(8784, 8)
```

In [5]:
```
data.columns
```

Out[5]:
```
Index(['Date/Time', 'Temp_C', 'Dew Point Temp_C', 'Rel Hum_%',
       'Wind Speed_km/h', 'Visibility_km', 'Press_kPa', 'Weather'],
      dtype='object')
```

In [6]:
```
data.dtypes
```

Out[6]:
```
Date/Time            object
Temp_C              float64
Dew Point Temp_C    float64
Rel Hum_%             int64
Wind Speed_km/h       int64
Visibility_km       float64
Press_kPa           float64
Weather              object
dtype: object
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8784 entries, 0 to 8783
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Date/Time         8784 non-null   object 
 1   Temp_C            8784 non-null   float64
 2   Dew Point Temp_C  8784 non-null   float64
 3   Rel Hum_%         8784 non-null   int64  
 4   Wind Speed_km/h   8784 non-null   int64  
 5   Visibility_km     8784 non-null   float64
 6   Press_kPa         8784 non-null   float64
 7   Weather           8784 non-null   object 
dtypes: float64(4), int64(2), object(2)
memory usage: 549.1+ KB
```

In [8]:
```
#type of weather
data.Weather.value_counts()
```

Out[8]:

Mainly Clear	2106
Mostly Cloudy	2069
Cloudy	1728
Clear	1326
Snow	390
Rain	306
Rain Showers	188
Fog	150
Rain,Fog	116
Drizzle,Fog	80
Snow Showers	60
Drizzle	41
Snow,Fog	37
Snow,Blowing Snow	19
Rain,Snow	18
Thunderstorms,Rain Showers	16
Haze	16
Drizzle,Snow,Fog	15
Freezing Rain	14
Freezing Drizzle,Snow	11
Freezing Drizzle	7
Snow,Ice Pellets	6
Freezing Drizzle,Fog	6
Snow,Haze	5
Freezing Fog	4
Snow Showers,Fog	4
Moderate Snow	4
Rain,Snow,Ice Pellets	4
Freezing Rain,Fog	4
Freezing Drizzle,Haze	3
Rain,Haze	3
Thunderstorms,Rain	3
Thunderstorms,Rain Showers,Fog	3
Freezing Rain,Haze	2
Drizzle,Snow	2
Rain Showers,Snow Showers	2
Thunderstorms	2
Moderate Snow,Blowing Snow	2
Rain Showers,Fog	1
Thunderstorms,Moderate Rain Showers,Fog	1
Snow Pellets	1

```
Rain,Snow,Fog                              1
Moderate Rain,Fog                          1
Freezing Rain,Ice Pellets,Fog              1
Drizzle,Ice Pellets,Fog                    1
Thunderstorms,Rain,Fog                     1
Rain,Ice Pellets                           1
Rain,Snow Grains                           1
Thunderstorms,Heavy Rain Showers           1
Freezing Rain,Snow Grains                  1
Name: Weather, dtype: int64
```

In [9]:
```
data.Weather.unique()
```

Out[9]:
```
array(['Fog', 'Freezing Drizzle,Fog', 'Mostly Cloudy', 'Cloudy', 'Rain',
       'Rain Showers', 'Mainly Clear', 'Snow Showers', 'Snow', 'Clear',
       'Freezing Rain,Fog', 'Freezing Rain', 'Freezing Drizzle',
       'Rain,Snow', 'Moderate Snow', 'Freezing Drizzle,Snow',
       'Freezing Rain,Snow Grains', 'Snow,Blowing Snow', 'Freezing Fog',
       'Haze', 'Rain,Fog', 'Drizzle,Fog', 'Drizzle',
       'Freezing Drizzle,Haze', 'Freezing Rain,Haze', 'Snow,Haze',
       'Snow,Fog', 'Snow,Ice Pellets', 'Rain,Haze', 'Thunderstorms,Rain',
       'Thunderstorms,Rain Showers', 'Thunderstorms,Heavy Rain Showers',
       'Thunderstorms,Rain Showers,Fog', 'Thunderstorms',
       'Thunderstorms,Rain,Fog',
       'Thunderstorms,Moderate Rain Showers,Fog', 'Rain Showers,Fog',
       'Rain Showers,Snow Showers', 'Snow Pellets', 'Rain,Snow,Fog',
       'Moderate Rain,Fog', 'Freezing Rain,Ice Pellets,Fog',
       'Drizzle,Ice Pellets,Fog', 'Drizzle,Snow', 'Rain,Ice Pellets',
       'Drizzle,Snow,Fog', 'Rain,Snow Grains', 'Rain,Snow,Ice Pellets',
       'Snow Showers,Fog', 'Moderate Snow,Blowing Snow'], dtype=object)
```

In [10]:
```
data.Weather.nunique()
```

Out[10]:
```
50
```

In [ ]:
```
#converting the weather categories into Standard categories
```

In [11]:
```
x='Thunderstorms,Moderate Rain Showers,Fog'
```

In [12]:
```
list_of_lists=[w.split() for w in x.split(',')]
list_of_lists
```

Out[12]:
```
[['Thunderstorms'], ['Moderate', 'Rain', 'Showers'], ['Fog']]
```

In [14]:
```
from itertools import chain
flat_list=list(chain(*list_of_lists))
flat_list
```

Out[14]:
```
['Thunderstorms', 'Moderate', 'Rain', 'Showers', 'Fog']
```

In [15]:
```
def Create_list(x):
    list_of_lists=[w.split() for w in x.split(',')]
    flat_list=list(chain(*list_of_lists))
    return flat_list
def Get_Weather(list1):
    if 'Fog' in list1 and 'Rain' in list1:
        return 'RAIN+FOG'
    elif 'Snow' in list1 and 'Rain' in list1:
        return 'SNOW+RAIN'
    elif 'Snow' in list1:
        return 'Snow'
    elif 'Rain' in list1:
        return 'RAIN'
    elif 'Fog' in list1:
        return 'FOG'
    elif 'Clear' in list1:
        return 'Clear'
    elif 'Cloudy' in list1:
        return 'Cloudy'
    else:
        return 'RAIN'
```

In [16]:
```
Create_list(x)
```

Out[16]:
```
['Thunderstorms', 'Moderate', 'Rain', 'Showers', 'Fog']
```

In [17]:
```
Get_Weather(Create_list(x))
```

Out[17]:
```
'RAIN+FOG'
```

In [18]:
```
data['Std_Weather']=data['Weather'].apply(lambda x:Get_Weather(Create_list(x)))
```

In [19]:
```
data.head(10)
```

Out[19]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather	Std_Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog	FOG
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog	FOG
2	1/1/2012 2:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog	FOG
3	1/1/2012 3:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog	FOG
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog	FOG
5	1/1/2012 5:00	-1.4	-3.3	87	9	6.4	101.27	Fog	FOG
6	1/1/2012 6:00	-1.5	-3.1	89	7	6.4	101.29	Fog	FOG
7	1/1/2012 7:00	-1.4	-3.6	85	7	8.0	101.26	Fog	FOG
8	1/1/2012 8:00	-1.4	-3.6	85	9	8.0	101.23	Fog	FOG
9	1/1/2012 9:00	-1.3	-3.1	88	15	4.0	101.20	Fog	FOG

In [20]:
```
data.Std_Weather.value_counts()
```

Out[20]:
```
Cloudy       3797
Clear        3432
RAIN          603
Snow          556
FOG           241
RAIN+FOG      129
SNOW+RAIN      26
Name: Std_Weather, dtype: int64
```

In [ ]:
```
#sampling Selection and Data Balancing
```

In [22]:
```
cloudy_df=data[data['Std_Weather']=='Cloudy']
cloudy_df_sample=cloudy_df.sample(600)
cloudy_df_sample.size
```

Out[22]:
```
5400
```

In [23]:
```
clear_df=data[data['Std_Weather']=='Cloudy'].sample(600)
clear_df.shape
```

Out[23]:
```
(600, 9)
```

In [ ]:
```
#Dataset Balancing
```

In [24]:
```
rain_df=data[data['Std_Weather']=='RAIN']
snow_df=data[data['Std_Weather']=='SNOW']
```

In [25]:
```
rain_df.shape
```

Out[25]:
```
(603, 9)
```

In [26]:
```
snow_df.shape
```

Out[26]:
```
(0, 9)
```

In [ ]:
```
#Create new weather dataset
```

In [27]:
```
weather_df=pd.concat([cloudy_df,clear_df,rain_df,snow_df],axis=0)
weather_df.head()
```

Out[27]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather	Std_Weather
16	1/1/2012 16:00	2.6	-0.2	82	13	12.9	99.93	Mostly Cloudy	Cloudy
17	1/1/2012 17:00	3.0	0.0	81	13	16.1	99.81	Cloudy	Cloudy
20	1/1/2012 20:00	3.2	1.3	87	19	25.0	99.50	Cloudy	Cloudy
21	1/1/2012 21:00	4.0	1.7	85	20	25.0	99.39	Cloudy	Cloudy
23	1/1/2012 23:00	5.3	2.0	79	30	25.0	99.31	Cloudy	Cloudy

In [28]:
```
weather_df.shape
```

Out[28]:
```
(5000, 9)
```

In [29]:
```
weather_df.Std_Weather.value_counts()
```

Out[29]:
```
Cloudy    4397
RAIN       603
Name: Std_Weather, dtype: int64
```

In [ ]:
```
#Drop Columns date and weather
```

In [32]:
```
weather_df.drop(columns=['Date/Time','Weather'],axis=1,inplace=True)
```

In [33]:
```
weather_df.head()
```

Out[33]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Std_Weather
16	2.6	-0.2	82	13	12.9	99.93	Cloudy
17	3.0	0.0	81	13	16.1	99.81	Cloudy
20	3.2	1.3	87	19	25.0	99.50	Cloudy
21	4.0	1.7	85	20	25.0	99.39	Cloudy
23	5.3	2.0	79	30	25.0	99.31	Cloudy

In [ ]:
```
#Duplicate Records
```

In [34]:
```
weather_df[weather_df.duplicated()]
```

Out[34]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Std_Weather
3660	19.3	3.3	35	20	48.3	101.32	Cloudy
228	1.4	-3.1	72	30	24.1	100.32	Cloudy
8488	1.3	-0.4	88	7	24.1	100.96	Cloudy
4025	26.5	11.3	39	4	48.3	101.73	Cloudy
2284	1.5	-5.2	61	17	25.0	100.47	Cloudy
...	...	...	...	...	...	...	...
2914	9.2	7.0	86	6	16.1	101.12	Cloudy
7005	13.8	6.0	59	17	25.0	100.95	Cloudy
664	1.4	-3.7	69	22	48.3	100.14	Cloudy
7217	13.9	9.3	74	9	24.1	101.09	Cloudy
6533	10.3	8.9	91	19	25.0	101.23	Cloudy

600 rows × 7 columns

In [ ]:
```
#Null/Missing Values
```

In [35]:
```
weather_df.isnull().sum()
```

Out[35]:
```
Temp_C              0
Dew Point Temp_C    0
Rel Hum_%           0
Wind Speed_km/h     0
Visibility_km       0
Press_kPa           0
Std_Weather         0
dtype: int64
```

In [36]:
```
weather_df.dtypes
```

Out[36]:
```
Temp_C              float64
Dew Point Temp_C    float64
Rel Hum_%             int64
Wind Speed_km/h       int64
Visibility_km       float64
Press_kPa           float64
Std_Weather          object
dtype: object
```

In [ ]:
```
#Data Visualization
```

In [37]:
```
weather_df.describe()
```

Out[37]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.0000	5000.000000
mean	9.446160	3.255260	67.391000	16.189200	27.9790	100.897204
std	10.871937	10.534356	15.945274	8.609682	10.5129	0.831018
min	-23.200000	-28.500000	18.000000	0.000000	2.0000	97.520000
25%	1.700000	-4.400000	57.000000	9.000000	24.1000	100.390000
50%	9.900000	3.600000	69.000000	15.000000	25.0000	100.900000
75%	18.600000	12.200000	80.000000	20.000000	25.0000	101.410000
max	32.400000	24.400000	100.000000	83.000000	48.3000	103.650000

In [ ]:
```
#Correlation among the features
```

In [40]:
```
cols=['Temp_C','Dew Point Temp_C','Rel Hum_%','Wind Speed_km/h','Visibility_km','Press_kPa']
```

In [42]:
```
cor_matrix=weather_df[cols].corr()
cor_matrix
```

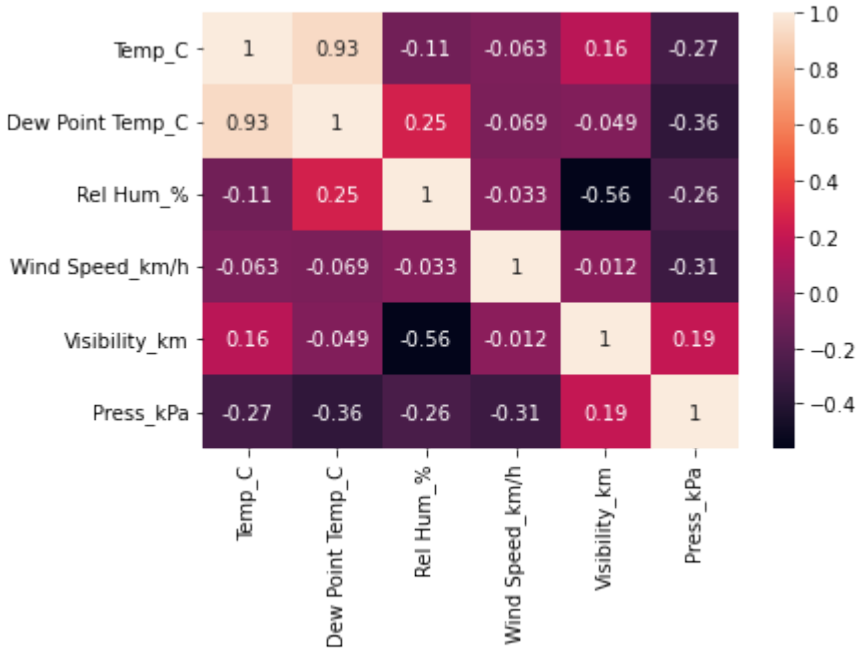
Out[42]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
Temp_C	1.000000	0.932273	-0.107511	-0.063105	0.157767	-0.274974
Dew Point Temp_C	0.932273	1.000000	0.253368	-0.068527	-0.049390	-0.359343
Rel Hum_%	-0.107511	0.253368	1.000000	-0.033061	-0.559318	-0.255950
Wind Speed_km/h	-0.063105	-0.068527	-0.033061	1.000000	-0.012268	-0.314617
Visibility_km	0.157767	-0.049390	-0.559318	-0.012268	1.000000	0.191838
Press_kPa	-0.274974	-0.359343	-0.255950	-0.314617	0.191838	1.000000

In [ ]:
```
#Heat Map
```

In [44]:
```
sns.heatmap(cor_matrix,annot=True)
```

Out[44]: <AxesSubplot:>

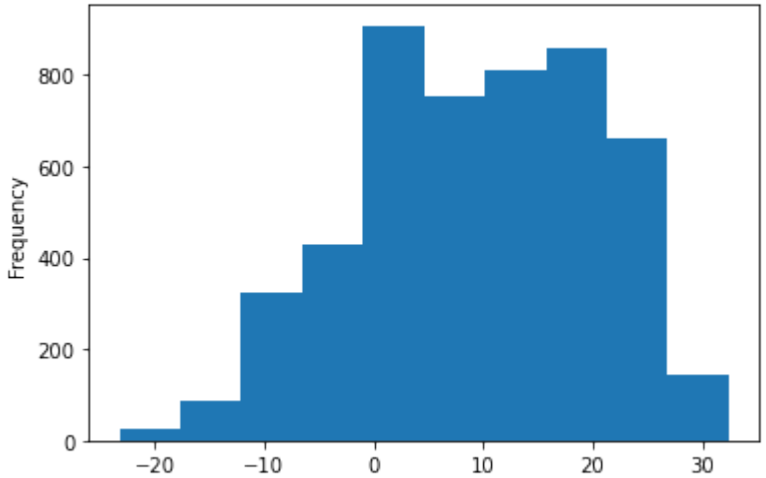


In [45]:
```
weather_df.columns
```

Out[45]:
```
Index(['Temp_C', 'Dew Point Temp_C', 'Rel Hum_%', 'Wind Speed_km/h',
       'Visibility_km', 'Press_kPa', 'Std_Weather'],
      dtype='object')
```

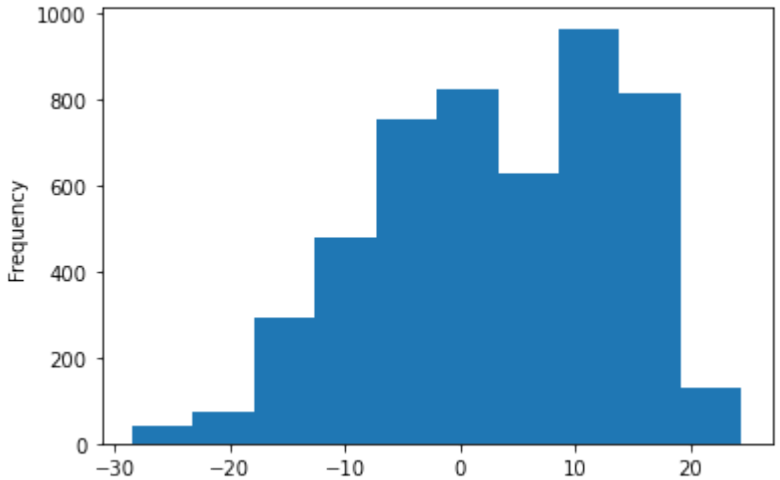
In [46]:
```
weather_df['Temp_C'].plot(kind='hist')
```

Out[46]: <AxesSubplot:ylabel='Frequency'>



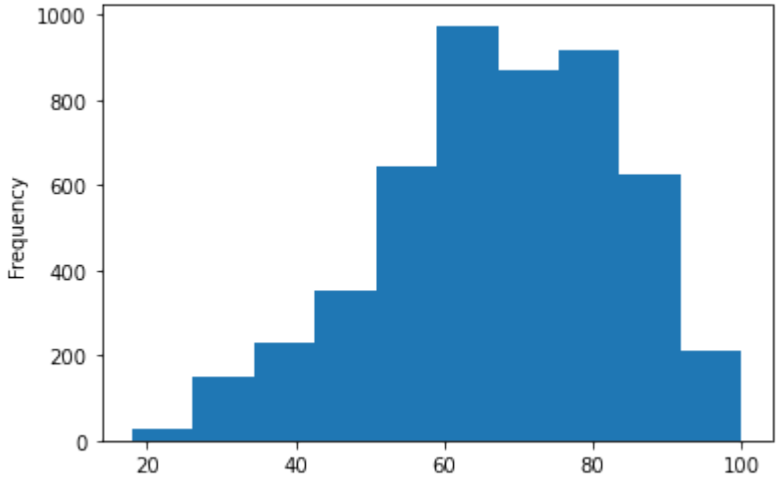
In [47]:
```
weather_df['Dew Point Temp_C'].plot(kind='hist')
```

Out[47]: <AxesSubplot:ylabel='Frequency'>



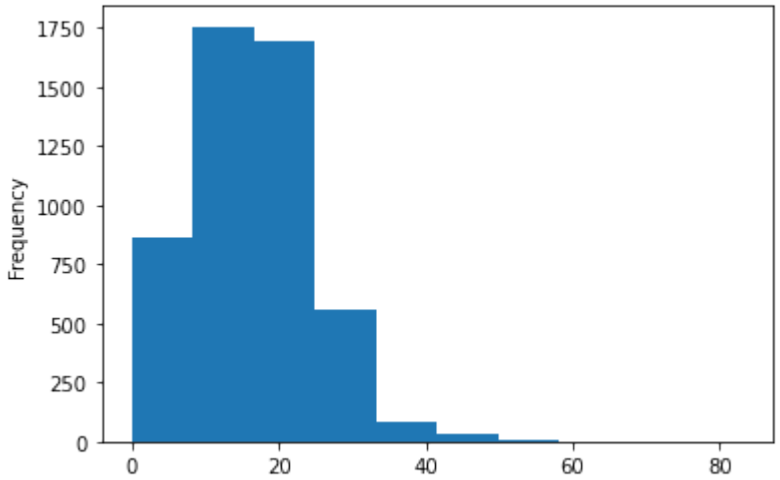
In [48]:
```
weather_df['Rel Hum_%'].plot(kind='hist')
```

Out[48]: <AxesSubplot:ylabel='Frequency'>



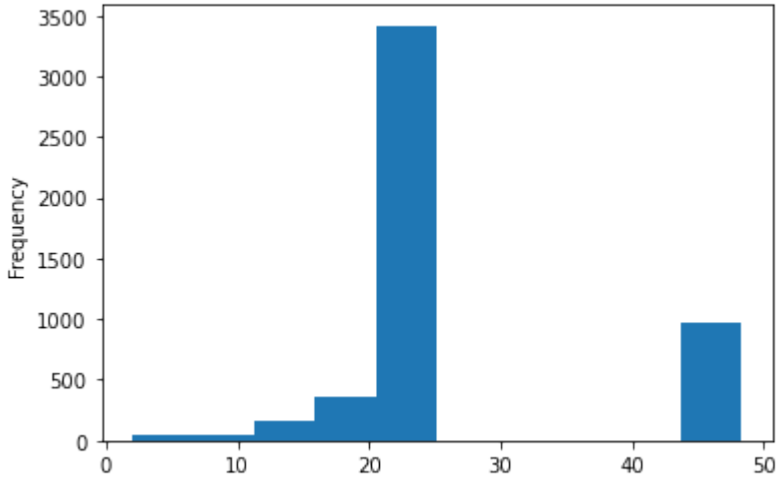
In [49]:
```
weather_df['Wind Speed_km/h'].plot(kind='hist')
```

Out[49]: <AxesSubplot:ylabel='Frequency'>



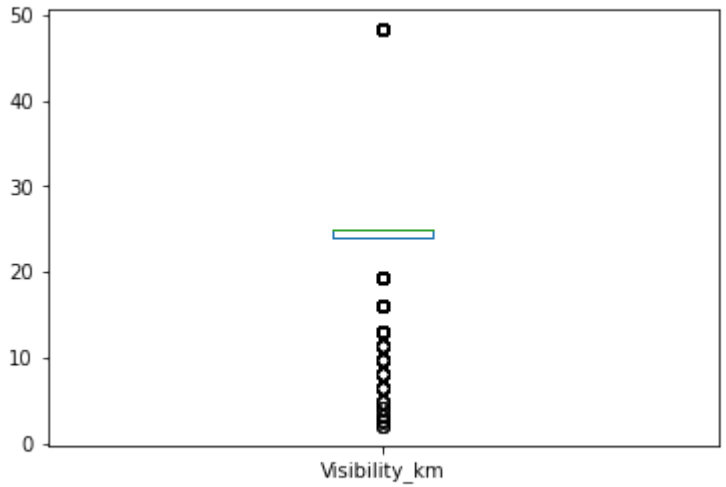
In [50]:
```
weather_df['Visibility_km'].plot(kind='hist')
```

Out[50]: <AxesSubplot:ylabel='Frequency'>



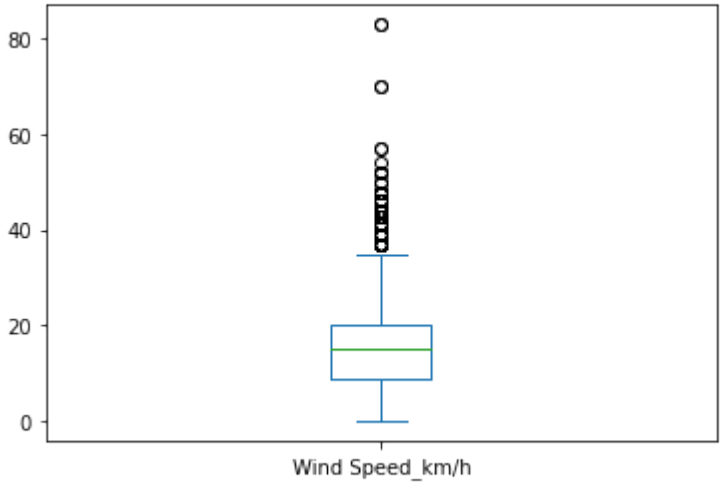
In [51]:
```
weather_df['Visibility_km'].plot(kind='box')
```

Out[51]: <AxesSubplot:>



In [53]:
```
weather_df['Wind Speed_km/h'].plot(kind='box')
```

Out[53]: <AxesSubplot:>



In [54]:
```
weather_df.head()
```

Out[54]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Std_Weather
16	2.6	-0.2	82	13	12.9	99.93	Cloudy
17	3.0	0.0	81	13	16.1	99.81	Cloudy
20	3.2	1.3	87	19	25.0	99.50	Cloudy
21	4.0	1.7	85	20	25.0	99.39	Cloudy
23	5.3	2.0	79	30	25.0	99.31	Cloudy

In [ ]:
```
#Label Encoding
```

In [69]:
```
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

In [56]:
```
label=LabelEncoder()
```

In [57]:
```
weather_df['Std_Weather']=label.fit_transform(weather_df['Std_Weather'])
```

In [58]:
```
label.classes_
```

Out[58]:
```
array(['Cloudy', 'RAIN'], dtype=object)
```

In [59]:
```
weather_df.head()
```

Out[59]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Std_Weather
16	2.6	-0.2	82	13	12.9	99.93	0
17	3.0	0.0	81	13	16.1	99.81	0
20	3.2	1.3	87	19	25.0	99.50	0
21	4.0	1.7	85	20	25.0	99.39	0
23	5.3	2.0	79	30	25.0	99.31	0

In [60]:
```
weather_df.Std_Weather.value_counts()
```

Out[60]:
```
0    4397
1     603
Name: Std_Weather, dtype: int64
```

In [ ]:
```
#x,y Variables
```

In [61]:
```
X=weather_df.drop(['Std_Weather'],axis=1)
X
#independent variable
```

Out[61]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
16	2.6	-0.2	82	13	12.9	99.93
17	3.0	0.0	81	13	16.1	99.81
20	3.2	1.3	87	19	25.0	99.50
21	4.0	1.7	85	20	25.0	99.39
23	5.3	2.0	79	30	25.0	99.31
...	...	...	...	...	...	...
8538	1.4	0.3	92	46	25.0	97.56
8539	2.3	1.1	92	37	11.3	97.52
8540	2.4	1.7	95	15	9.7	97.58
8541	1.3	0.4	94	22	9.7	97.64
8545	1.2	0.1	92	30	12.9	97.84

5000 rows × 6 columns

In [63]:
```
#Target Variable
y=weather_df['Std_Weather']
y
```

Out[63]:
```
16      0
17      0
20      0
21      0
23      0
       ..
8538    1
8539    1
8540    1
8541    1
8545    1
Name: Std_Weather, Length: 5000, dtype: int32
```

In [ ]:
```
#Feature Scaling
```

In [70]:
```
from sklearn.model_selection import train_test_split
std_scalar=StandardScaler()
X_std=std_scalar.fit_transform(X)
X_std
```

Out[70]:
```
array([[-0.62977231, -0.32803197,  0.91628786, -0.37045722, -1.43447648,
        -1.16399561],
       [-0.59297666, -0.30904458,  0.85356708, -0.37045722, -1.1300581 ,
        -1.30841134],
       [-0.57457883, -0.18562649,  1.22989175,  0.32650231, -0.28339448,
        -1.68148532],
       ...,
       [-0.64817014, -0.1476517 ,  1.73165798, -0.13813738, -1.73889486,
        -3.99213703],
       [-0.74935818, -0.27106978,  1.6689372 ,  0.67498207, -1.73889486,
        -3.91992916],
       [-0.75855709, -0.29955088,  1.54349564,  1.60426144, -1.43447648,
        -3.67923628]])
```

In [73]:
```
x_train, x_test, y_train, y_test = train_test_split(X_std, y, test_size=0.2, random_state=42)
```

In [74]:
```
x_train.shape,x_test.shape
```

Out[74]:
```
((4000, 6), (1000, 6))
```

In [ ]:
```
#Model Building
```

In [75]:
```
from sklearn.tree import DecisionTreeClassifier
decision=DecisionTreeClassifier()
```

In [76]:
```
#Model Training
decision.fit(x_train,y_train)
```

Out[76]:
```
DecisionTreeClassifier()
```

In [ ]:
```
#Model Predictions
```

In [77]:
```
y_pred=decision.predict(x_test)
```

In [ ]:
```
#Model Evaluation
```

In [78]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [ ]:
```
#Accuracy
```

In [79]:
```
accuracy_score(y_test,y_pred)
```

Out[79]:
```
0.88
```

In [80]:
```
#classfication report
print(classification_report(y_test,y_pred))
```

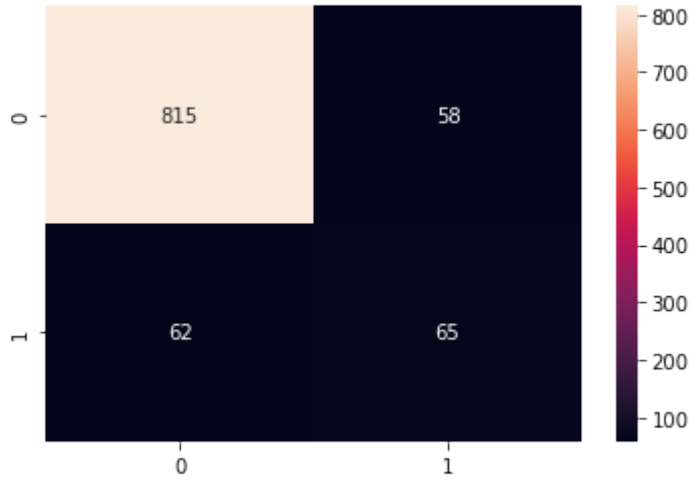
```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       873
           1       0.53      0.51      0.52       127

    accuracy                           0.88      1000
   macro avg       0.73      0.72      0.73      1000
weighted avg       0.88      0.88      0.88      1000
```

In [81]:
```
#confusion matrix
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt='d')
```

Out[81]: <AxesSubplot:>



In [83]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [84]:
```
rf=RandomForestClassifier()
```

In [85]:
```
rf.fit(x_train,y_train)
```

Out[85]:
```
RandomForestClassifier()
```

In [86]:
```
y_pred_rf=rf.predict(x_test)
```

In [87]:
```
accuracy_score(y_test,y_pred_rf)
```

Out[87]:
```
0.913
```

In [88]:
```
#Hyperparameter tuning
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
```

In [89]:
```
parameters={
    'n_estimators':[50,100],
    'max_features':['sqrt','log2',None]
}
```

In [90]:
```
grid_search=GridSearchCV(estimator=rf,param_grid=parameters)
```

In [91]:
```
grid_search.fit(x_train,y_train)
```

Out[91]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'max_features': ['sqrt', 'log2', None],
                         'n_estimators': [50, 100]})
```

In [92]:
```
#Best Hyper Parameters
grid_search.best_params_
```

Out[92]:
```
{'max_features': 'sqrt', 'n_estimators': 50}
```

In [94]:
```
Random_forest_model=RandomForestClassifier(max_features='log2',n_estimators=50)
```

In [95]:
```
Random_forest_model.fit(x_train,y_train)
```

Out[95]:
```
RandomForestClassifier(max_features='log2', n_estimators=50)
```

In [96]:
```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(Random_forest_model,X_std,y,cv=5,scoring='accuracy')
print('Cross-Validation scores=',scores)
```

```
Cross-Validation scores= [0.935 0.883 0.851 0.85  0.906]
```

In [97]:
```
scores.mean()
```

Out[97]:
```
0.885
```

In [ ]: