In [1]:
```
import pandas as pd
```

In [8]:
```
data=pd.read_csv('student_dataset.csv')
df=pd.DataFrame(data)
print(df)
```

```
        ID          Name  Age  Marks  Gender
0        1     Student_1   24     94    Male
1        2     Student_2   21     81  Female
2        3     Student_3   22     90    Male
3        4     Student_4   24     56  Female
4        5     Student_5   20     44  Female
..     ...           ...  ...    ...     ...
995    996   Student_996   23     82  Female
996    997   Student_997   18     73  Female
997    998   Student_998   24     96  Female
998    999   Student_999   21     65    Male
999   1000  Student_1000   23     80    Male

[1000 rows x 5 columns]
```

In [9]:
```
df.isnull().sum()    #check mising values
```

Out[9]:
```
ID        0
Name      0
Age       0
Marks     0
Gender    0
dtype: int64
```

In [13]:
```
df.fillna(df[['Age', 'Marks']].mean(), inplace=True) # fill NaN with mean
print(df)
```

```
        ID          Name  Age  Marks  Gender
0        1     Student_1   24     94    Male
1        2     Student_2   21     81  Female
2        3     Student_3   22     90    Male
3        4     Student_4   24     56  Female
4        5     Student_5   20     44  Female
..     ...           ...  ...    ...     ...
995    996   Student_996   23     82  Female
996    997   Student_997   18     73  Female
997    998   Student_998   24     96  Female
998    999   Student_999   21     65    Male
999   1000  Student_1000   23     80    Male

[1000 rows x 5 columns]
```

In [14]:
```
df.dropna()                    # remove rows with NaN
df.duplicated().sum()          # check duplicates
df.drop_duplicates(inplace=True)  # remove duplicates
```

In [20]:
```
df['Marks'].fillna(df['Marks'].mean(), inplace=True)
print("\nAfter Filling NaN:\n", df)
print(df.head())
```

```
After Filling NaN:
        ID          Name  Age  Marks  Gender
0        1     Student_1   24     94    Male
1        2     Student_2   21     81  Female
2        3     Student_3   22     90    Male
3        4     Student_4   24     56  Female
4        5     Student_5   20     44  Female
..     ...           ...  ...    ...     ...
995    996   Student_996   23     82  Female
996    997   Student_997   18     73  Female
997    998   Student_998   24     96  Female
998    999   Student_999   21     65    Male
999   1000  Student_1000   23     80    Male

[1000 rows x 5 columns]
   ID       Name  Age  Marks  Gender
0   1  Student_1   24     94    Male
1   2  Student_2   21     81  Female
2   3  Student_3   22     90    Male
3   4  Student_4   24     56  Female
4   5  Student_5   20     44  Female
```

```
C:\Users\kalen\AppData\Local\Temp\ipykernel_4620\1802312897.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained as
signment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['Marks'].fillna(df['Marks'].mean(), inplace=True)
```

In [21]:
```
print(df.groupby('Gender')['Marks'].mean())
```

```
Gender
Female    70.689861
Male      69.768612
Name: Marks, dtype: float64
```

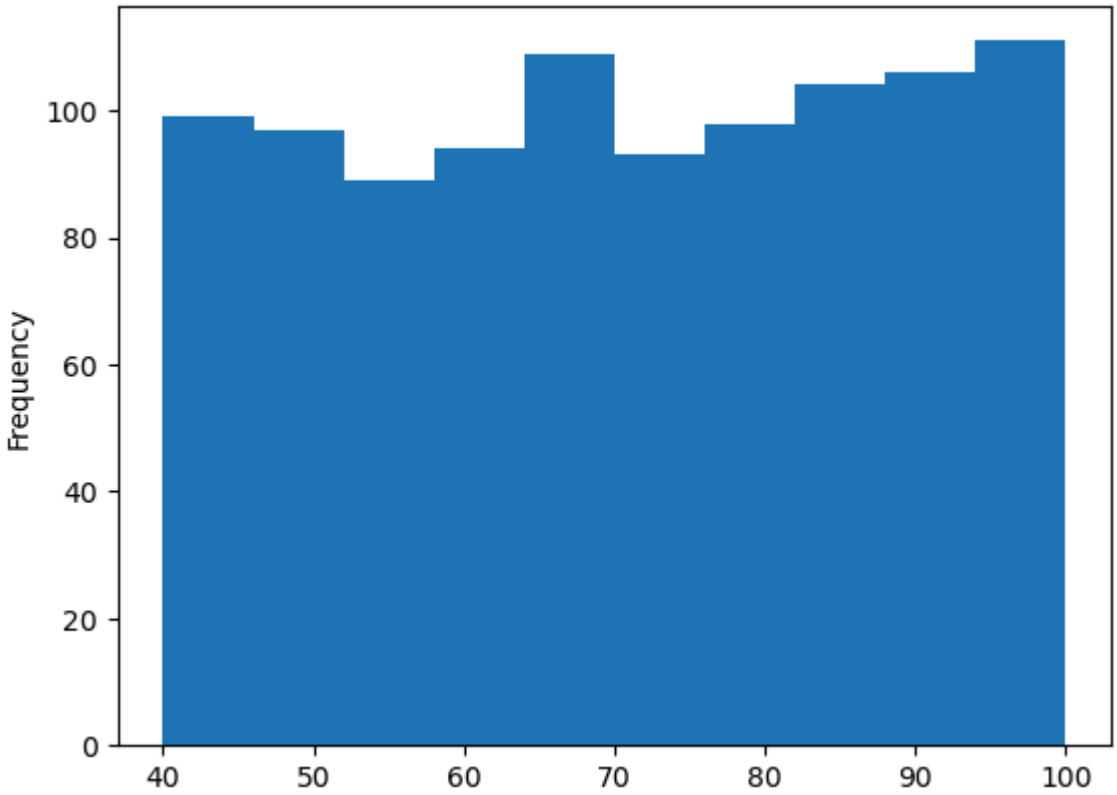
In [22]:
```
df.groupby('Age')['Marks'].agg(['mean','max','min','count'])
```

Out[22]:

Age	mean	max	min	count
18	67.641026	100	40	156
19	69.328467	100	40	137
20	71.284615	99	40	130
21	73.096154	100	40	156
22	69.824324	100	40	148
23	69.118519	100	40	135
24	71.355072	100	41	138

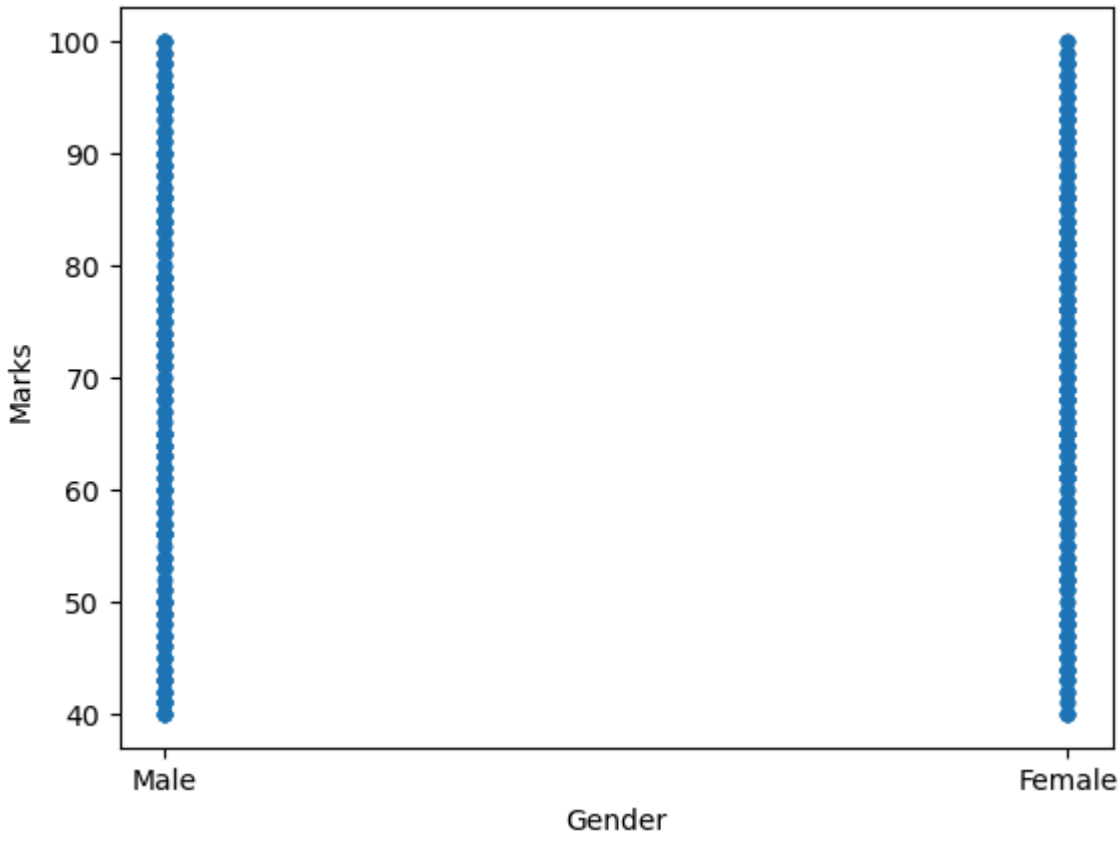
In [23]:
```
df['Marks'].plot(kind='hist') # histogram
```

Out[23]:
```
<Axes: ylabel='Frequency'>
```



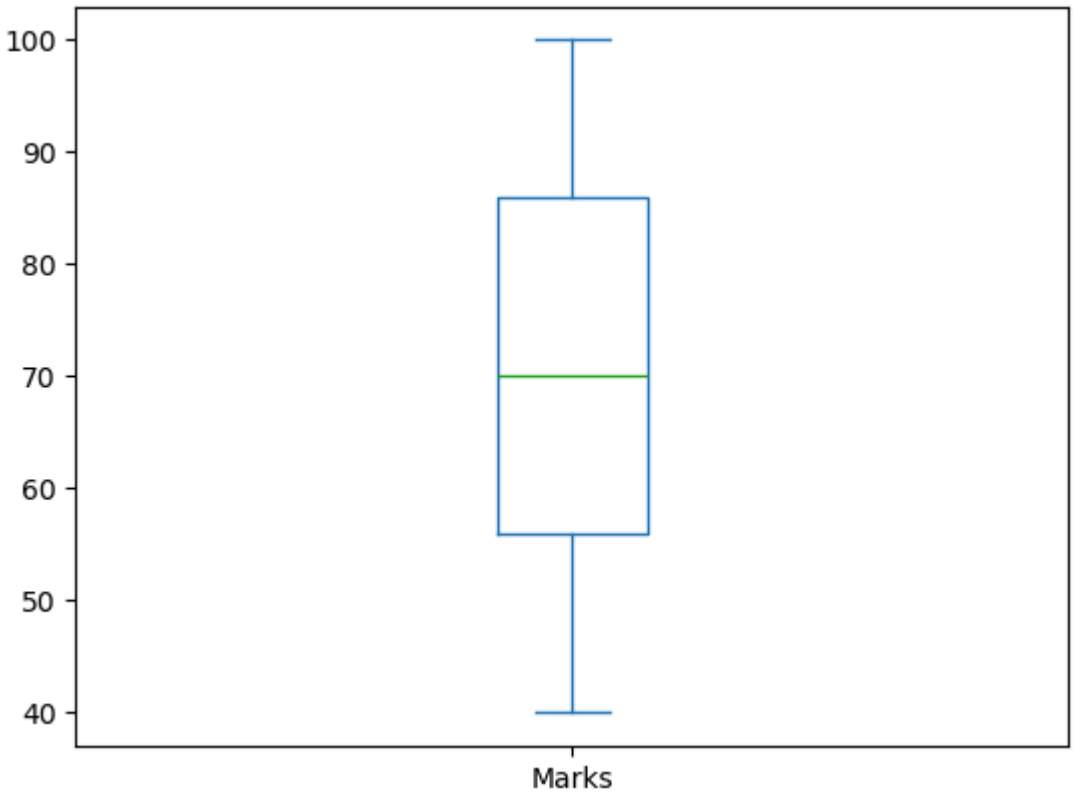
In [25]:
```
df.plot(x='Gender', y='Marks', kind='scatter')  # scatter plot
```

Out[25]:
```
<Axes: xlabel='Gender', ylabel='Marks'>
```



In [26]:
```
df['Marks'].plot(kind='box')  # box plot
```

Out[26]: <Axes: >



In [28]:
```
data = {
    'Name': ['A', 'B', 'C', 'D', 'E'],
    'Class': ['X', 'X', 'Y', 'Y', 'X'],
    'Marks': [85, 90, 78, 88, 95]
}
```

```
df1 = pd.DataFrame(data)
print(df1.groupby('Class')['Marks'].mean())
```

```
Class
X    90.0
Y    83.0
Name: Marks, dtype: float64
```

In [ ]:

```
data = {
    'Department': ['IT','IT','CS','CS','EC','EC','IT','CS'],
    'Student': ['Amit','Riya','Karan','Meena','Ankit','Neha','Rahul','Simran'],
    'Marks': [88,92,76,85,95,90,88,72],
    'Age': [21,20,23,22,21,20,22,24]
}

df2 = pd.DataFrame(data)
print(df2)
```

```
  Department Student  Marks  Age
0         IT    Amit     88   21
1         IT    Riya     92   20
2         CS   Karan     76   23
3         CS   Meena     85   22
4         EC   Ankit     95   21
5         EC    Neha     90   20
6         IT   Rahul     88   22
7         CS  Simran     72   24
```