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

In [2]:
```
df=pd.read_csv('UpdatedStudentsPerformance.csv')
```

In [8]:
```
df.isnull().any().sum
```

Out[8]:
```
<bound method NDFrame._add_numeric_operations.<locals>.sum of gender
False
race/ethnicity                 False
parental level of education    False
lunch                          False
test preparation course        False
math score                      True
reading score                   True
writing score                   True
dtype: bool>
```

In [9]:
```
rsm=df['reading score'].mean()
msm=df['math score'].mean()
wsm=df['writing score'].mean()
```

In [10]:
```
df.head()
```

Out[10]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72.0	72.0	74.0
1	female	group C	some college	standard	completed	69.0	90.0	88.0
2	female	group B	master's degree	standard	none	90.0	95.0	93.0
3	male	group A	associate's degree	free/reduced	none	47.0	57.0	44.0
4	male	group C	some college	standard	none	76.0	78.0	75.0

In [15]:
```
##check syntax
df['reading score'].fillna(rsm,inplace=True)
df['math score'].fillna(msm,inplace=True)
df['writing score'].fillna(wsm,inplace=True)
```

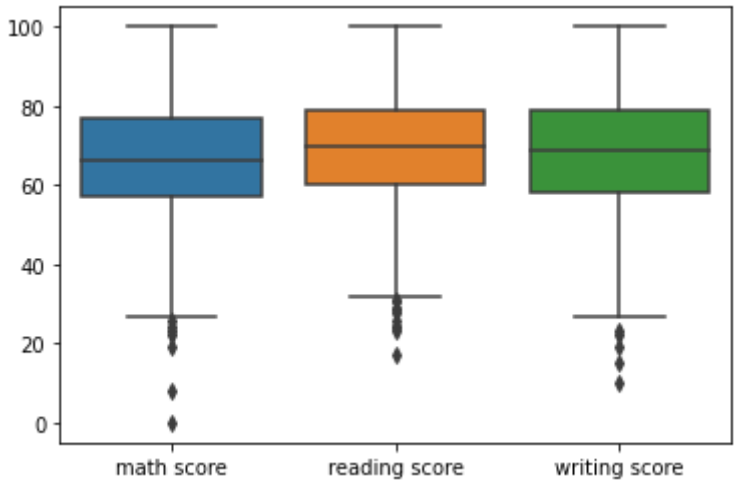
In [16]:
```
df.isnull().any().sum()
```

Out[16]:
```
0
```

In [17]:
```
import seaborn as sns
```

In [18]:
```
sns.boxplot(data=df)
```

Out[18]:
```
<AxesSubplot:>
```



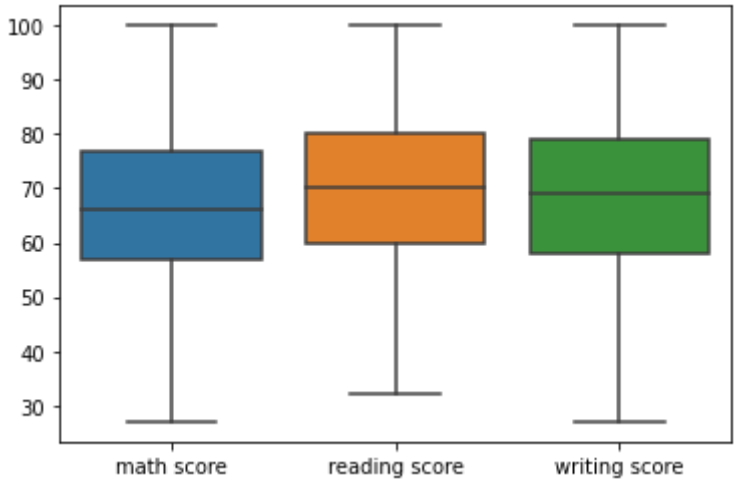
In [23]:
```
q1=df.quantile(0.25)
q2=df.quantile(0.75)
iqr=q2-q1
upper=q2+1.5*iqr
low=q1-1.5*iqr
```

In [27]:
```
#check
df2=df[-((df<low) | (df>upper))]
```

```
C:\Users\brima\AppData\Local\Temp\ipykernel_17496\3611337015.py:1: FutureWarning:
Automatic reindexing on DataFrame vs Series comparisons is deprecated and will rai
se ValueError in a future version. Do `left, right = left.align(right, axis=1, cop
y=False)` before e.g. `left == right`
  df2=df[-((df<low) | (df>upper))]
```

In [28]:
```
sns.boxplot(data=df2)
```

Out[28]:
```
<AxesSubplot:>
```



In [29]:
```
col=['math score','reading score','writing score']
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scaled=sc.fit_transform(df[col].to_numpy())
scaled=pd.DataFrame(scaled,columns=col)
```

In [33]:
```
scaled
```

Out[33]:

	math score	reading score	writing score
0	0.385603	0.188616	0.387696
1	0.185875	1.428574	1.314339
2	1.583971	1.773006	1.645283
3	-1.278798	-0.844682	-1.597967
4	0.651907	0.601935	0.453885
...	...	...	...
995	1.450819	2.048553	1.777660
996	-0.280157	-0.982455	-0.869890
997	-0.479885	0.119730	-0.208003
998	0.119299	0.601935	0.586262
999	0.718483	1.153028	1.181961

1000 rows × 3 columns

In [34]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
maxm=mms.fit_transform(df[col].to_numpy())
maxm=pd.DataFrame(maxm,columns=col)
```

In [35]:
```
maxm
```

Out[35]:

	math score	reading score	writing score
0	0.72	0.662651	0.711111
1	0.69	0.879518	0.866667
2	0.90	0.939759	0.922222
3	0.47	0.481928	0.377778
4	0.76	0.734940	0.722222
...	...	...	...
995	0.88	0.987952	0.944444
996	0.62	0.457831	0.500000
997	0.59	0.650602	0.611111
998	0.68	0.734940	0.744444
999	0.77	0.831325	0.844444

1000 rows × 3 columns

In [36]:
```
maxm.min()
```

Out[36]:
```
math score       0.0
reading score    0.0
writing score    0.0
dtype: float64
```

In [37]:
```
maxm.max()
```

Out[37]:
```
math score       1.0
reading score    1.0
writing score    1.0
dtype: float64
```

In [38]:
```
scaled.min()
```

Out[38]:
```
math score      -4.407870
reading score   -3.600143
writing score   -3.848384
dtype: float64
```

In [39]:
```
scaled.max()
```

Out[39]:
```
math score       2.249731
reading score    2.117439
writing score    2.108604
dtype: float64
```

In [ ]: