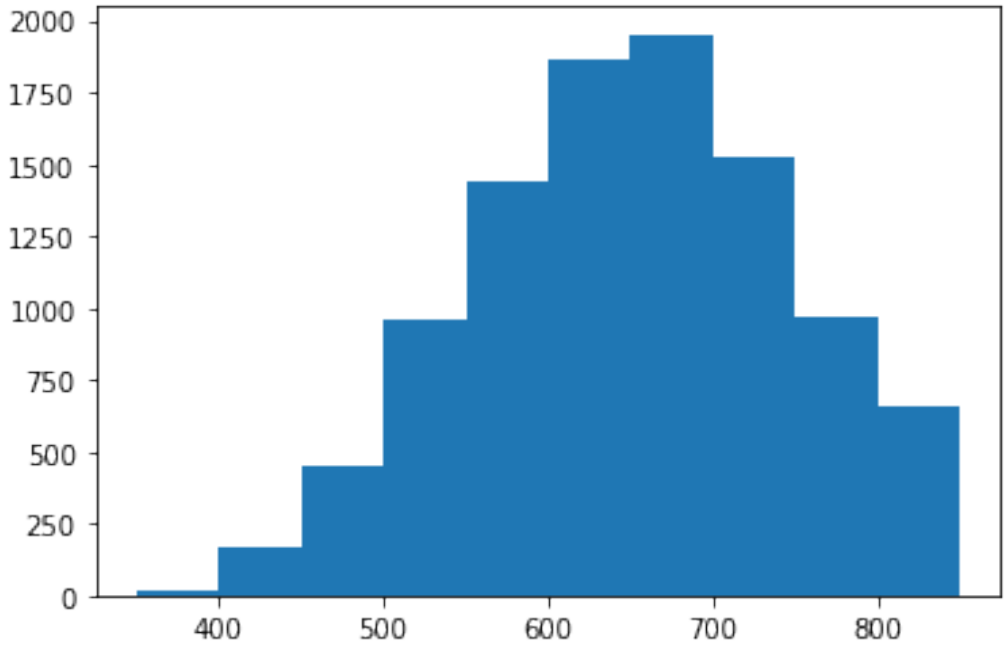
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

df = pd.read_csv('/content/Churn_Modelling.csv')

df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

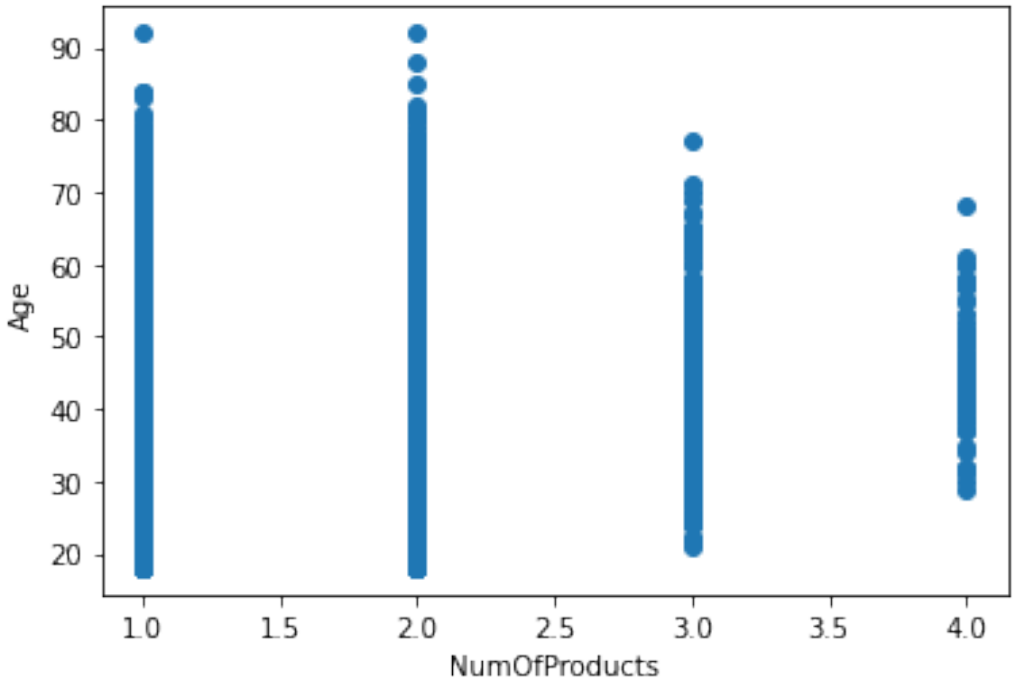
```
plt.hist(df['CreditScore'])
```

```
(array([  19.,  166.,  447.,  958., 1444., 1866., 1952., 1525.,  968.,
         655.]),
 array([350., 400., 450., 500., 550., 600., 650., 700., 750., 800.,
850.]),
 <a list of 10 Patch objects>)
```



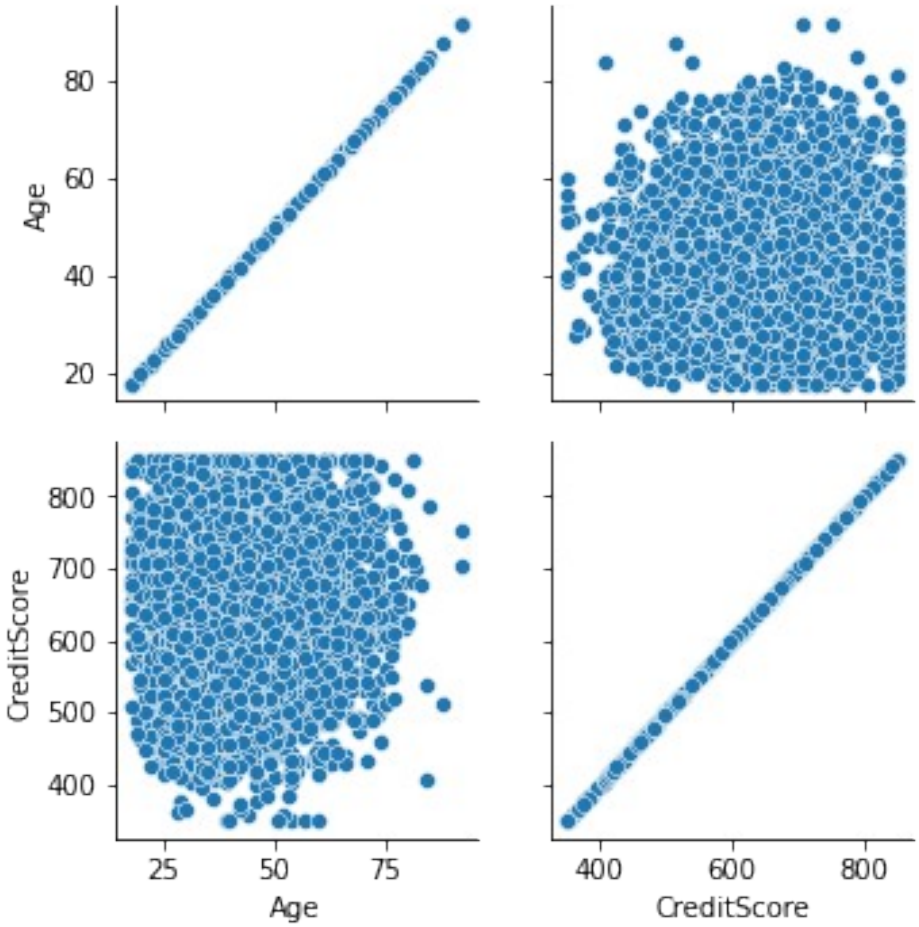

```
plt.scatter(df.NumOfProducts, df.Age)
plt.xlabel('NumOfProducts')
plt.ylabel('Age')
```

```
Text(0, 0.5, 'Age')
```




```
g = sns.PairGrid(df, vars=["Age", "CreditScore"], )
g.map(sns.scatterplot)
```

```
<seaborn.axisgrid.PairGrid at 0x7f37398fa290>
```




```
df.describe()
```

```
         RowNumber    CustomerId   CreditScore           Age
Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000
10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800
5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806
2.892174
min        1.00000  1.556570e+07    350.000000     18.000000
0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000
```

```
             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

```
df.info()
```

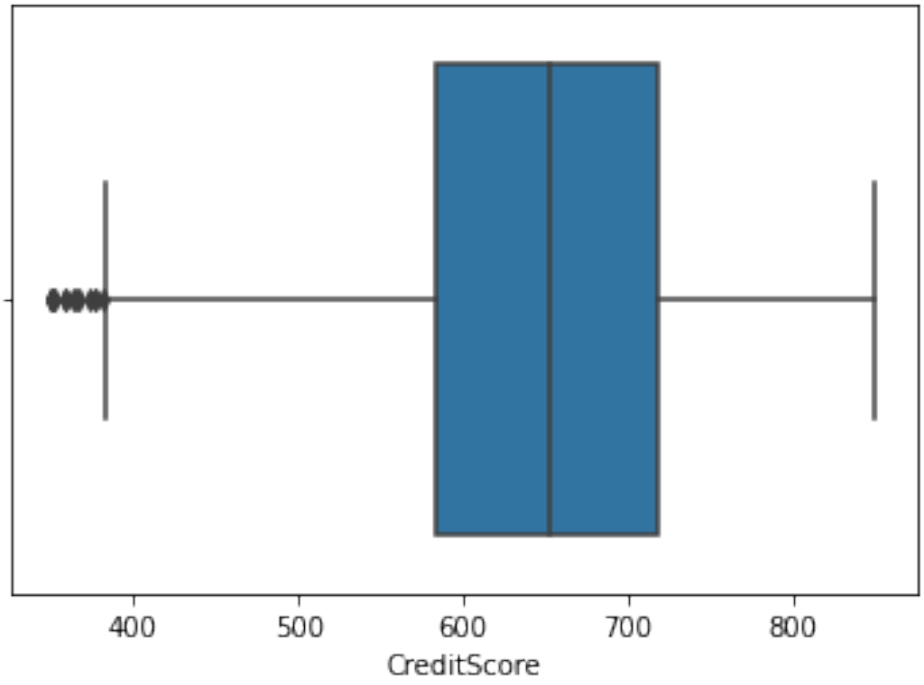
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
missing_values=(df.isnull().sum())
missing_values[missing_values>0]/len(df)*100
```

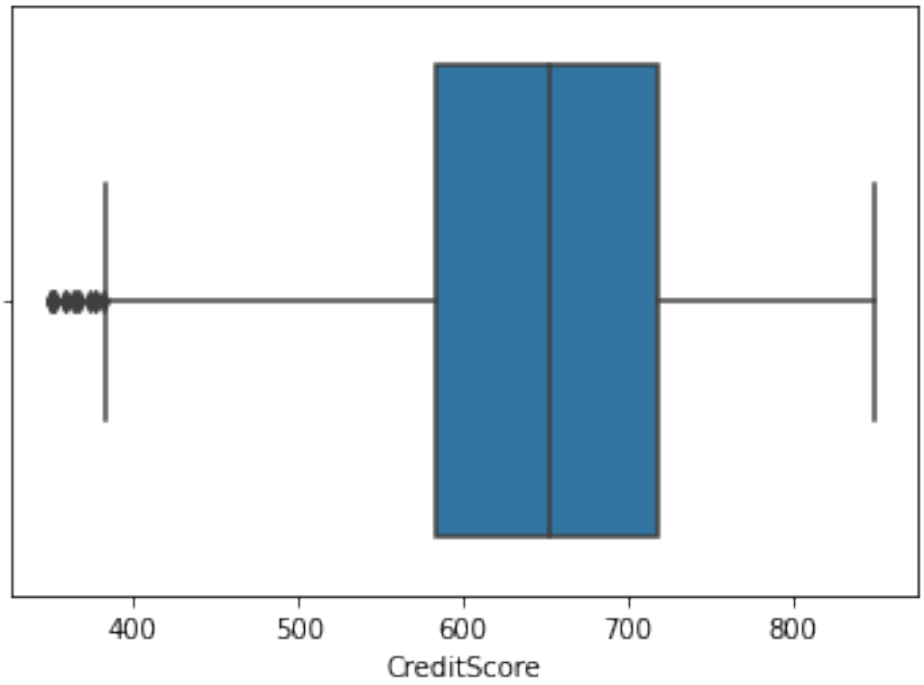
```
Series([], dtype: float64)
```

```
sns.boxplot(df['CreditScore'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7f3734714e10>
```



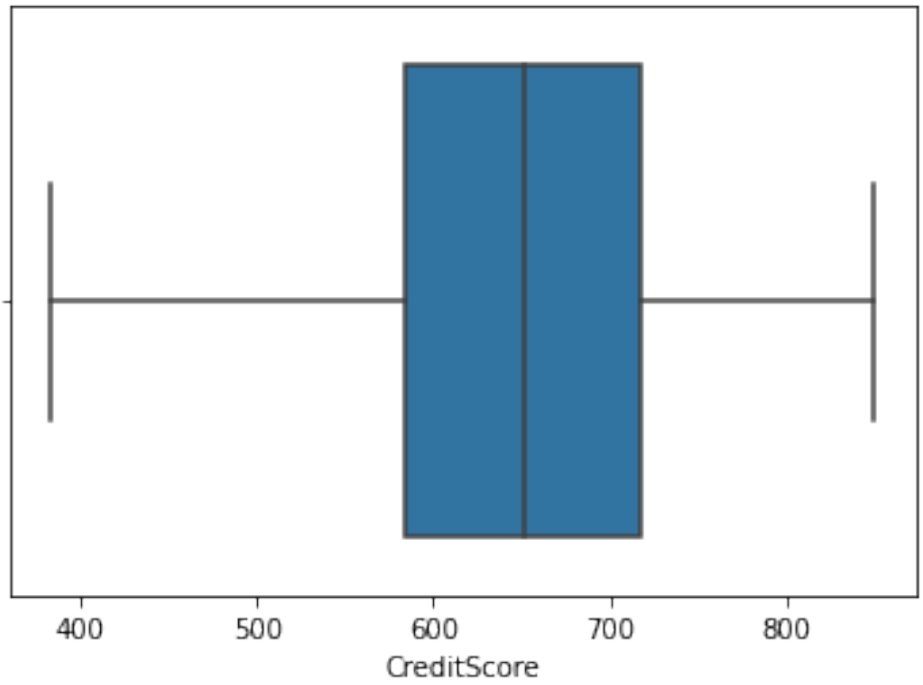

```
sns.boxplot(df['CreditScore'],data=df)

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7f37346a09d0>
```




```
Q1 = df['CreditScore'].quantile(0.25)
Q3 = df['CreditScore'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 +(whisker_width*IQR)
df['CreditScore']=np.where(df['CreditScore']>upper_whisker,upper_whisk
er,np.where(df['CreditScore']<lower_whisker,lower_whisker,df['CreditSc
ore']))

sns.boxplot(df['CreditScore'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3739311550>
```




```
new_df=df.copy()
new_df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
0	1	15634602	Hargrave	619.0	France	Female	42
1	2	15647311	Hill	608.0	Spain	Female	41
2	3	15619304	Onio	502.0	France	Female	42
3	4	15701354	Boni	699.0	France	Female	39
4	5	15737888	Mitchell	850.0	Spain	Female	43

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1

```
3         93826.63       0
4         79084.10       0
```

```
categorical = df.select_dtypes(include=['object']).copy()
categorical.head()
```

```
    Surname Geography  Gender
0  Hargrave    France  Female
1      Hill     Spain  Female
2      Onio    France  Female
3      Boni    France  Female
4  Mitchell     Spain  Female
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for feat in categorical:
    new_df[feat] = le.fit_transform(new_df[feat].astype(str))

print (new_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  int64
 3   CreditScore      10000 non-null  float64
 4   Geography        10000 non-null  int64
 5   Gender           10000 non-null  int64
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(3), int64(11)
memory usage: 1.1 MB
None
```

```
new_df.head()
```

```
   RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age
\
0          1    15634602     1115        619.0          0       0   42

1          2    15647311     1177        608.0          2       0   41
```

```
2          3    15619304     2040        502.0          0       0   42

3          4    15701354      289        699.0          0       0   39

4          5    15737888     1822        850.0          2       0   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0

new_df.tail()

      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender
Age  \
9995       9996    15606229     1999        771.0          0       1
39
9996       9997    15569892     1336        516.0          0       1
35
9997       9998    15584532     1570        709.0          0       0
36
9998       9999    15682355     2345        772.0          1       1
42
9999      10000    15628319     2751        792.0          0       0
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0
```

```
X = df.iloc[:, :-1].values
print(X)
```

```
[[1 15634602 'Hargrave' ... 1 1 101348.88]
 [2 15647311 'Hill' ... 0 1 112542.58]
 [3 15619304 'Onio' ... 1 0 113931.57]
 ...
 [9998 15584532 'Liu' ... 0 1 42085.58]
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]
 [10000 15628319 'Walker' ... 1 0 38190.78]]
```

```
y= df.iloc[:,3].values
print(y)
```

```
[619. 608. 502. ... 709. 772. 792.]
```

```
from sklearn.preprocessing import StandardScaler

object = StandardScaler()
object.fit_transform(new_df)
```

```
array([[-1.73187761, -0.78321342, -0.46418322, ...,  0.97024255,
         0.02188649,  1.97716468],
       [-1.7315312 , -0.60653412, -0.3909112 , ...,  0.97024255,
         0.21653375, -0.50577476],
       [-1.73118479, -0.99588476,  0.62898807, ..., -1.03067011,
         0.2406869 ,  1.97716468],
       ...,
       [ 1.73118479, -1.47928179,  0.07353887, ...,  0.97024255,
        -1.00864308,  1.97716468],
       [ 1.7315312 , -0.11935577,  0.98943914, ..., -1.03067011,
        -0.12523071,  1.97716468],
       [ 1.73187761, -0.87055909,  1.4692527 , ..., -1.03067011,
        -1.07636976, -0.50577476]])
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0, train_size = .75)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(7500, 13) (2500, 13) (7500,) (2500,)
```