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

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
df = pd.read_csv('C:\Users\Lenovo\Downloads\train.csv')
df.head()
```

Out[3]:

| | battery_power | blue | clock_speed | dual_sim | fc | four_g | int_memory | m_dep | mobile_wt | n_cores | ... | px_height | px_width | ram | sc_h | sc_w | talk_time | three_g | touch_screen | wifi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 842 | 0 | 2.2 | 0 | 1 | 0 | 7 | 0.6 | 188 | 2 | ... | 20 | 756 | 2549 | 9 | 7 | 19 | 0 | 0 | 1 |
| **1** | 1021 | 1 | 0.5 | 1 | 0 | 1 | 53 | 0.7 | 136 | 3 | ... | 905 | 1988 | 2631 | 17 | 3 | 7 | 1 | 1 | 0 |
| **2** | 563 | 1 | 0.5 | 1 | 2 | 1 | 41 | 0.9 | 145 | 5 | ... | 1263 | 1716 | 2603 | 11 | 2 | 9 | 1 | 1 | 0 |
| **3** | 615 | 1 | 2.5 | 0 | 0 | 0 | 10 | 0.8 | 131 | 6 | ... | 1216 | 1786 | 2769 | 16 | 8 | 11 | 1 | 0 | 0 |
| **4** | 1821 | 1 | 1.2 | 0 | 13 | 1 | 44 | 0.6 | 141 | 2 | ... | 1208 | 1212 | 1411 | 8 | 2 | 15 | 1 | 1 | 0 |

5 rows × 21 columns

In [35]:
```
df.shape
```

Out[35]:
```
(2000, 21)
```

In [6]:
```
df.describe()
```

Out[6]:

| | battery_power | blue | clock_speed | dual_sim | fc | four_g | int_memory | m_dep | mobile_wt | n_cores | ... | px_height | px_width | ram |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | ... | 2000.000000 | 2000.000000 | 2000.000000 |
| **mean** | 1238.518500 | 0.495000 | 1.522250 | 0.509500 | 4.309500 | 0.521500 | 32.046500 | 0.501750 | 140.249000 | 4.520500 | ... | 645.108000 | 1251.515500 | 2124.213000 |
| **std** | 439.418206 | 0.5001 | 0.816004 | 0.500035 | 4.341444 | 0.499662 | 18.145715 | 0.288416 | 35.399655 | 2.287837 | ... | 443.780811 | 432.199447 | 1084.732044 |
| **min** | 501.000000 | 0.0000 | 0.500000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 | 0.100000 | 80.000000 | 1.000000 | ... | 0.000000 | 500.000000 | 256.000000 |
| **25%** | 851.750000 | 0.0000 | 0.700000 | 0.000000 | 1.000000 | 0.000000 | 16.000000 | 0.200000 | 109.000000 | 3.000000 | ... | 282.750000 | 874.750000 | 1207.500000 |
| **50%** | 1226.000000 | 0.0000 | 1.500000 | 1.000000 | 3.000000 | 1.000000 | 32.000000 | 0.500000 | 141.000000 | 4.000000 | ... | 564.000000 | 1247.000000 | 2146.500000 |
| **75%** | 1615.250000 | 1.0000 | 2.200000 | 1.000000 | 7.000000 | 1.000000 | 48.000000 | 0.800000 | 170.000000 | 7.000000 | ... | 947.250000 | 1633.000000 | 3064.500000 |
| **max** | 1998.000000 | 1.0000 | 3.000000 | 1.000000 | 19.000000 | 1.000000 | 64.000000 | 1.000000 | 200.000000 | 8.000000 | ... | 1960.000000 | 1998.000000 | 3998.000000 |

8 rows × 21 columns

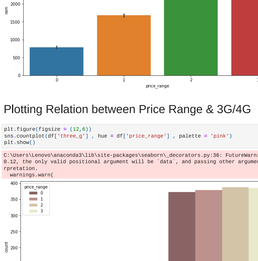
In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

In [9]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr())
plt.show()
```


## Plotting Relation between Price Range & Battery Power

In [10]:
```
plt.figure(figsize = (12,6))
sns.barplot(x = 'price_range', y = 'battery_power', data = df)
plt.show()
```

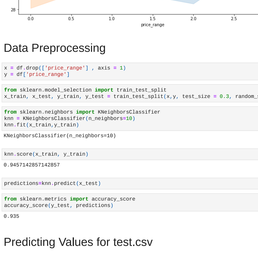



## Plotting Relation between Price Range & Pixel Height/Width

In [11]:
```
plt.figure(figsize(14,6))
plt.subplot(1,2,1)
sns.barplot(x = 'price_range', y = 'px_height', data = df, palette = 'Reds')
plt.subplot(1,2,2)
sns.barplot(x = 'price_range', y = 'px_width', data=df, palette = 'Blues')
plt.show()
```


## Plotting Relation between Price Range & Ram

In [12]:
```
plt.figure(figsize = (12,6))
sns.barplot(x = 'price_range', y = 'ram', data = df)
plt.show()
```



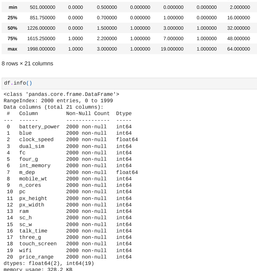

## Plotting Relation between Price Range & 3G/4G

In [14]:
```
plt.figure(figsize = (12,6))
sns.countplot(df['three_g'], hue = df['price_range'], palette = 'pink')
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: 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.
  warnings.warn(
```

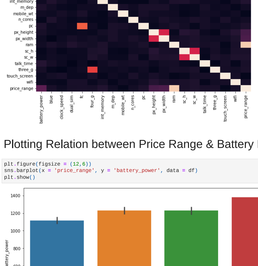

In [15]:
```
plt.figure(figsize = (12,6))
sns.countplot(df['four_g'] , hue = df['price_range'] , palette = 'ocean')
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: 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.
  warnings.warn(
```




## Plotting Relation between Price Range & Memory

In [16]:
```
plt.figure(figsize = (12, 6))
sns.lineplot(x = 'price_range', y = 'int_memory' , data = df , hue = 'dual_sim')
plt.show()
```




## Data Preprocessing

In [17]:
```
x = df.drop(['price_range'] , axis = 1)
y = df['price_range']
```

In [18]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 0)
```

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
```

Out[19]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [20]:
```
knn.score(x_train, y_train)
```

Out[20]:
```
0.9457142857142857
```

In [22]:
```
predictions=knn.predict(x_test)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

Out[23]:
```
0.935
```

## Predicting Values for test.csv

In [39]:
```
test_df = pd.read_csv(r'C:\Users\Lenovo\Downloads\test.csv')
test_df.head()
```

Out[39]:

| | id | battery_power | blue | clock_speed | dual_sim | fc | four_g | int_memory | m_dep | mobile_wt | ... | pc | px_height | px_width | ram | sc_h | sc_w | talk_time | three_g | touch_screen | wifi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1043 | 1 | 1.8 | 1 | 14 | 0 | 5 | 0.1 | 193 | ... | 16 | 226 | 1412 | 3476 | 12 | 7 | 2 | 0 | 1 | 0 |
| **1** | 2 | 841 | 1 | 0.5 | 1 | 4 | 1 | 61 | 0.8 | 191 | ... | 12 | 746 | 857 | 3895 | 6 | 0 | 7 | 1 | 0 | 0 |
| **2** | 3 | 1807 | 1 | 2.8 | 0 | 1 | 0 | 27 | 0.9 | 186 | ... | 4 | 1270 | 1366 | 2396 | 17 | 10 | 10 | 0 | 1 | 1 |
| **3** | 4 | 1546 | 0 | 0.5 | 1 | 18 | 1 | 25 | 0.5 | 96 | ... | 20 | 295 | 1752 | 3893 | 10 | 0 | 7 | 1 | 1 | 0 |
| **4** | 5 | 1434 | 0 | 1.4 | 0 | 11 | 1 | 49 | 0.5 | 108 | ... | 18 | 749 | 810 | 1773 | 15 | 8 | 7 | 1 | 0 | 1 |

5 rows × 21 columns

In [40]:
```
test_df.shape
```

Out[40]:
```
(1000, 21)
```

In [41]:
```
test_df = test_df.drop(['id'] , axis = 1)
test_df.shape
```

Out[41]:
```
(1000, 20)
```

In [42]:
```
test_pred = knn.predict(test_df)
```

In [43]:
```
test_df['predicted_price'] = test_pred
```

In [44]:
```
test_df.head()
```

Out[44]:

| | battery_power | blue | clock_speed | dual_sim | fc | four_g | int_memory | m_dep | mobile_wt | n_cores | ... | px_height | px_width | ram | sc_h | sc_w | talk_time | three_g | touch_screen | wifi | predicted_price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1043 | 1 | 1.8 | 1 | 14 | 0 | 5 | 0.1 | 193 | 3 | ... | 226 | 1412 | 3476 | 12 | 7 | 2 | 0 | 1 | 0 | 3 |
| **1** | 841 | 1 | 0.5 | 1 | 4 | 1 | 61 | 0.8 | 191 | 5 | ... | 746 | 857 | 3895 | 6 | 0 | 7 | 1 | 0 | 0 | 2 |
| **2** | 1807 | 1 | 2.8 | 0 | 1 | 0 | 27 | 0.9 | 186 | 3 | ... | 1270 | 1366 | 2396 | 17 | 10 | 10 | 0 | 1 | 1 | 1 |
| **3** | 1546 | 0 | 0.5 | 1 | 18 | 1 | 25 | 0.5 | 96 | 8 | ... | 295 | 1752 | 3893 | 10 | 0 | 7 | 1 | 1 | 0 | 3 |
| **4** | 1434 | 0 | 1.4 | 0 | 11 | 1 | 49 | 0.5 | 108 | 6 | ... | 749 | 810 | 1773 | 15 | 8 | 7 | 1 | 0 | 1 | 2 |

5 rows × 21 columns

In [ ]:
```

```