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

In [5]:
```python
data = pd.read_csv(r"C:\Users\Hp\Downloads\train.csv")
```

In [6]:
```python
data.head()
```

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	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

In [7]:
```python
data.shape
```

Out[7]:
```
(2000, 21)
```

In [8]:
```python
data.describe()
```

Out[8]:

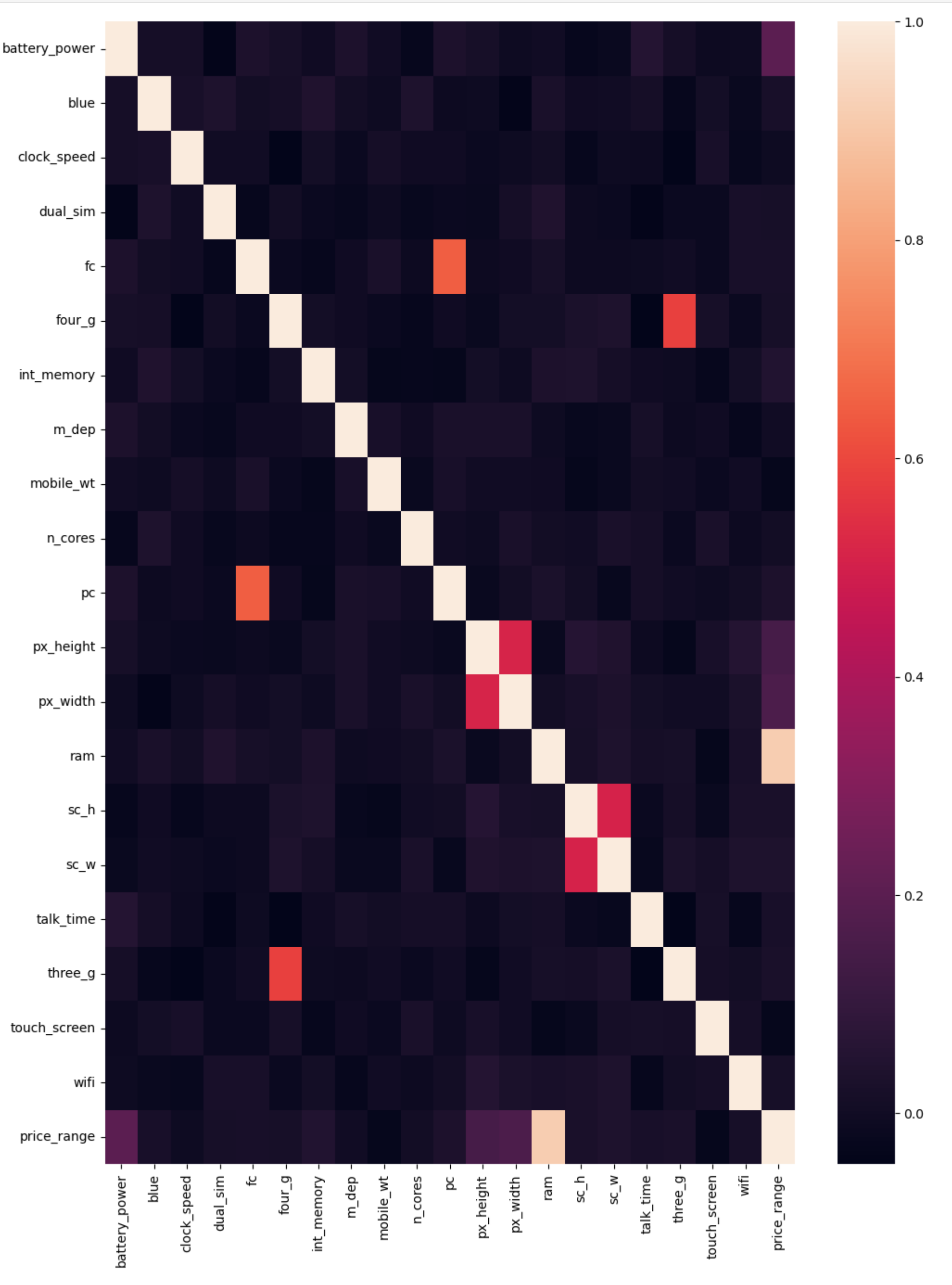
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500
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	4.213245
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	5.000000
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	9.000000
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	12.000000
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	16.000000
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	19.000000

8 rows × 21 columns

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

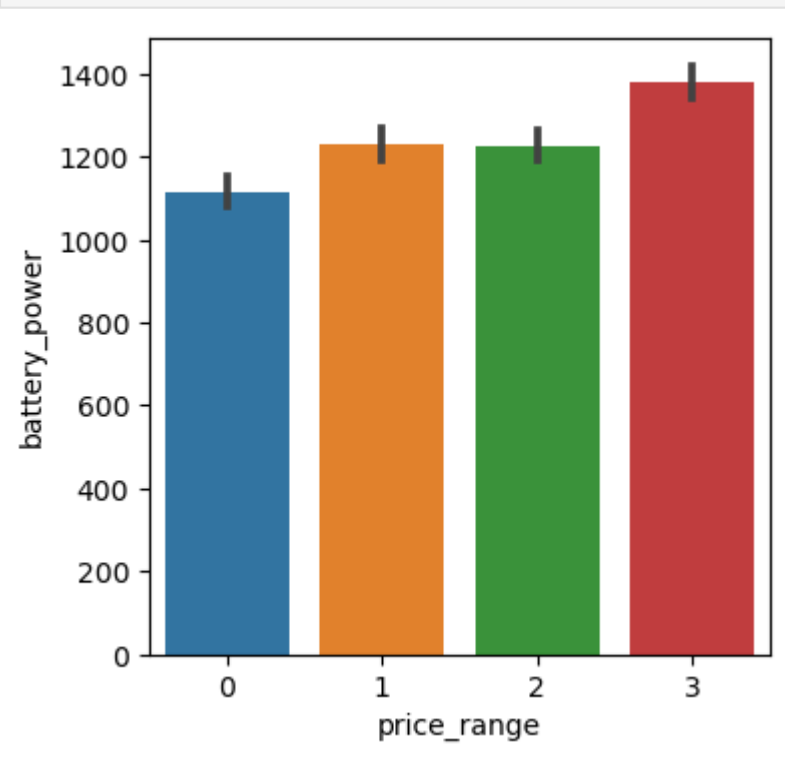
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             1000 non-null   int64
 1   battery_power  1000 non-null   int64
 2   blue           1000 non-null   int64
 3   clock_speed    1000 non-null   float64
 4   dual_sim       1000 non-null   int64
 5   fc             1000 non-null   int64
 6   four_g         1000 non-null   int64
 7   int_memory     1000 non-null   int64
 8   m_dep          1000 non-null   float64
 9   mobile_wt      1000 non-null   int64
 10  n_cores        1000 non-null   int64
 11  pc             1000 non-null   int64
 12  px_height      1000 non-null   int64
 13  px_width       1000 non-null   int64
 14  ram            1000 non-null   int64
 15  sc_h           1000 non-null   int64
 16  sc_w           1000 non-null   int64
 17  talk_time      1000 non-null   int64
 18  three_g        1000 non-null   int64
 19  touch_screen   1000 non-null   int64
 20  wifi           1000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 164.2 KB
```

In [9]:
```python
plt.figure(figsize=(12,16))
sns.heatmap(data.corr())
plt.show()
```



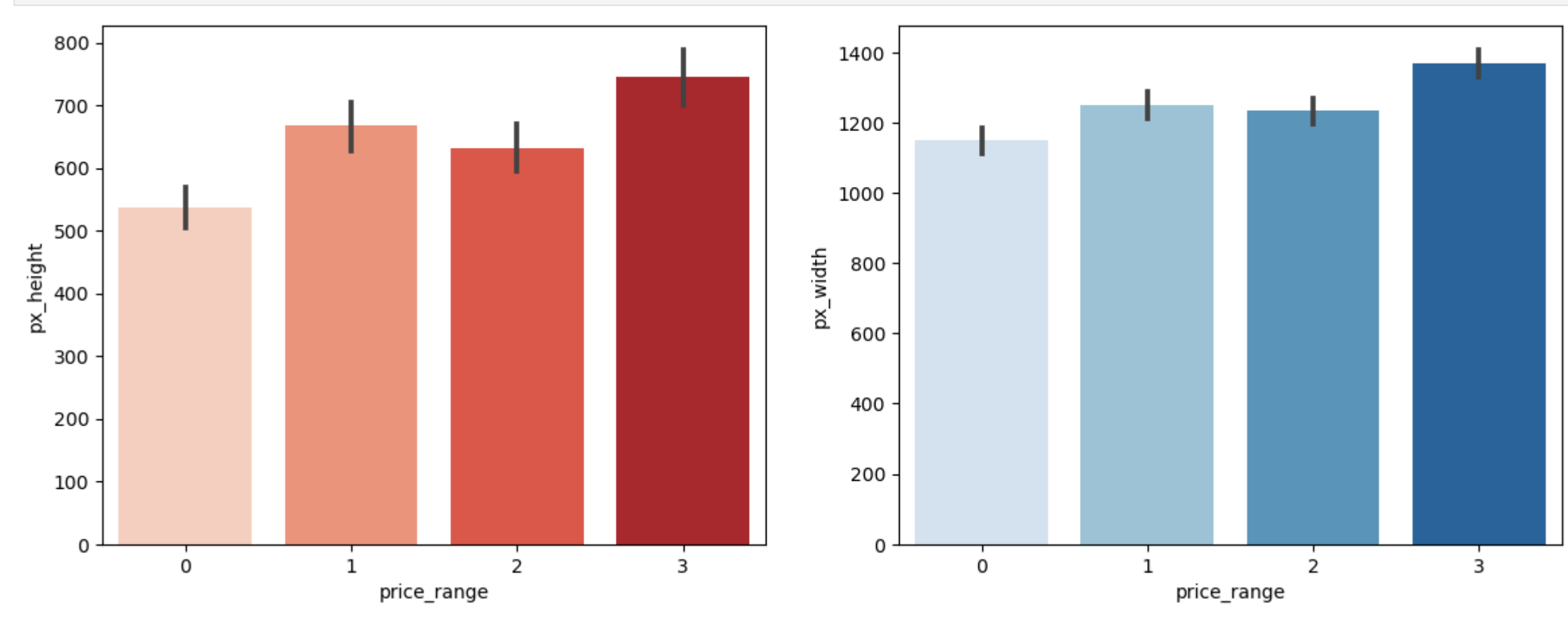
Plotting Relation between Price Range and Battery Power

In [22]:
```python
plt.figure(figsize=(4,4))
sns.barplot(x = 'price_range' , y = 'battery_power' , data=data)
plt.show()
```



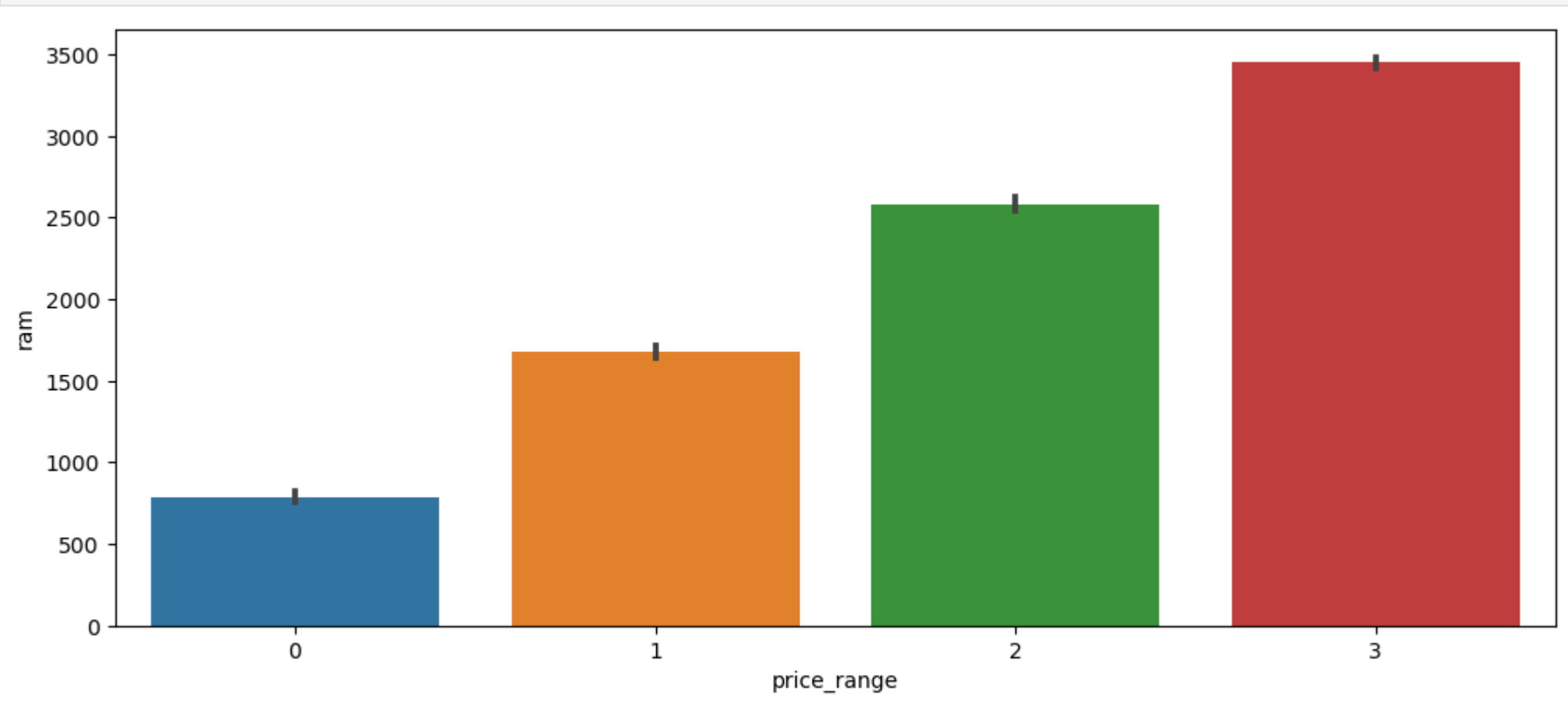
Plotting Relation Between price range and pixel_height/width

In [21]:
```python
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
sns.barplot(x = 'price_range',y='px_height',data=data,palette='Reds')
plt.subplot(1,2,2)
sns.barplot(x = 'price_range',y='px_width',data=data,palette='Blues')
plt.show()
```



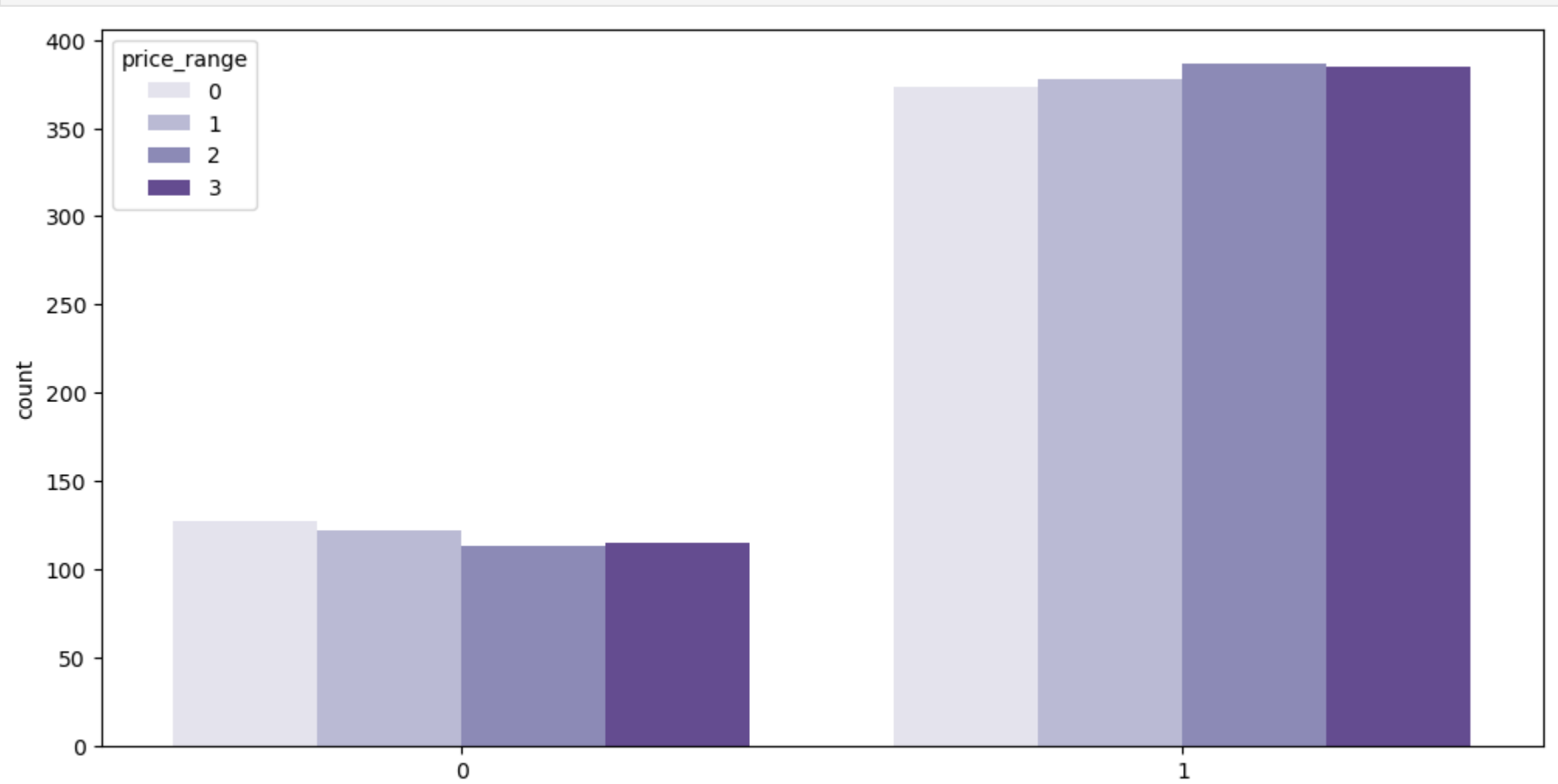
Plotting Relation between Price Range and Ram

In [24]:
```python
plt.figure(figsize=(12,5))
sns.barplot(x = 'price_range', y = "ram",data=data)
plt.show()
```

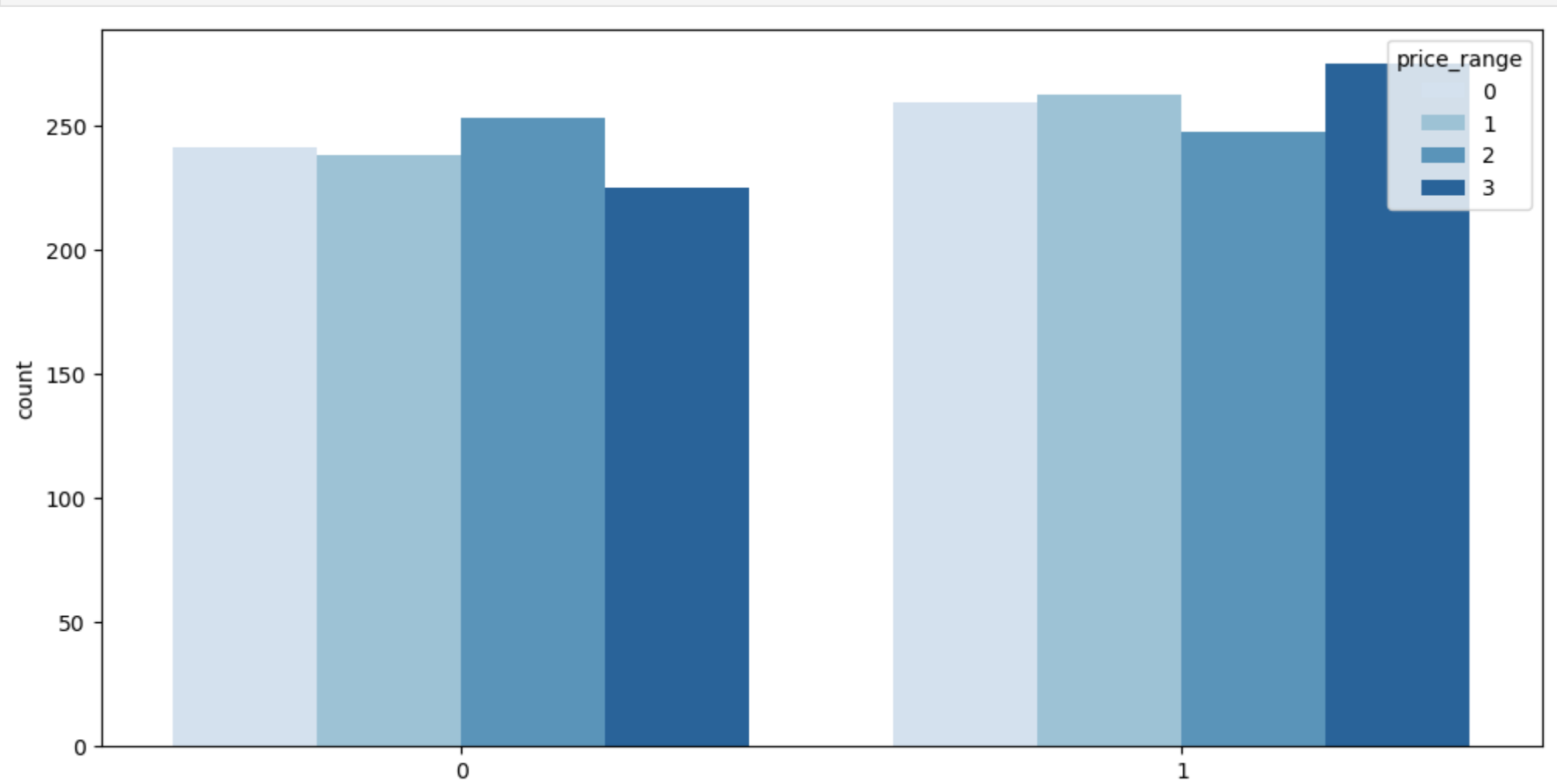


Plotting Relation Between price_range and 3G/4G

In [82]:
```python
plt.figure(figsize=(12,6))
sns.countplot(x='three_g' , data=data,hue='price_range',palette='Purples')
plt.show()
```

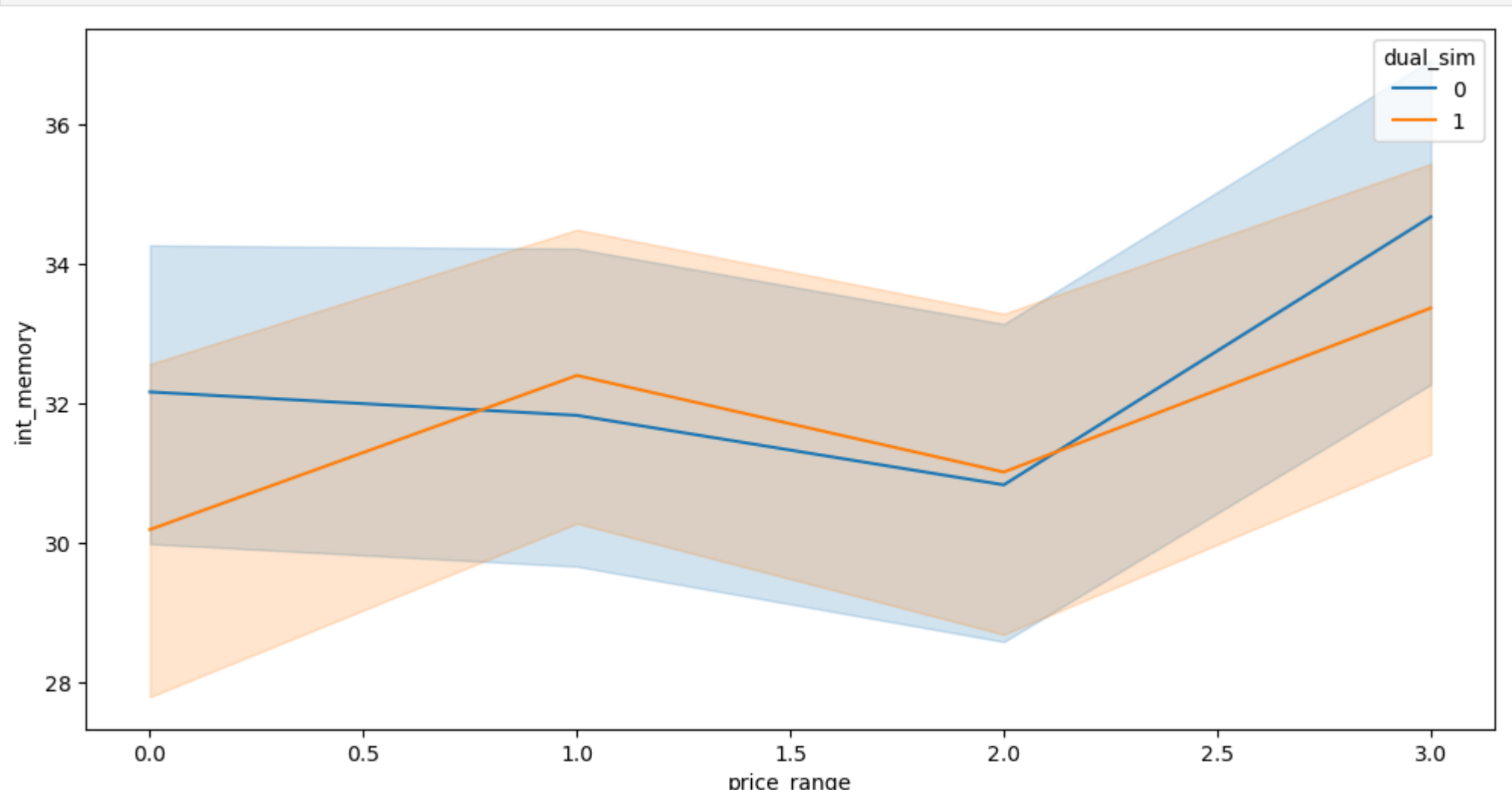


In [79]:
```python
plt.figure(figsize=(12,6))
sns.countplot(x='four_g' , data=data,hue='price_range',palette='Blues')
plt.show()
```



Plotting relation between price_range and Memory

In [41]:
```python
plt.figure(figsize=(12,6))
sns.lineplot(x = 'price_range' , y = 'int_memory',data=data,hue='dual_sim')
plt.show()
```



Data Preprocessing

In [43]:
```python
x = data.drop(['price_range'],axis=1)
y = data['price_range']
```

In [44]:
```python
from sklearn.model_selection import train_test_split
```

In [45]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

KNN

In [46]:
```python
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
```

Out[46]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

In [47]:
```python
knn.score(x_train,y_train)
```

Out[47]:
```
0.9457142857142857
```

In [49]:
```python
predict = knn.predict(x_test)
predict
```

Out[49]:
```
array([3, 0, 2, 2, 2, 0, 0, 2, 3, 1, 0, 3, 0, 2, 3, 0, 3, 2, 2, 1, 0, 0,
       3, 1, 2, 2, 3, 1, 3, 1, 1, 0, 2, 0, 1, 3, 0, 0, 3, 3, 2, 1, 3, 3,
       1, 3, 0, 1, 3, 1, 1, 3, 0, 3, 0, 2, 2, 2, 0, 3, 3, 1, 3, 2, 1, 2,
       3, 2, 1, 2, 3, 2, 1, 0, 1, 3, 2, 1, 1, 2, 3, 3, 3, 0, 0, 0, 2, 0,
       2, 3, 1, 2, 2, 1, 0, 3, 2, 3, 0, 3, 1, 1, 2, 1, 3, 2, 2, 3, 2, 3,
       3, 0, 0, 1, 2, 3, 0, 0, 1, 0, 0, 3, 2, 2, 1, 1, 1, 1, 0, 2, 1, 3,
       2, 3, 3, 3, 3, 2, 0, 1, 1, 2, 1, 3, 0, 3, 0, 0, 2, 0, 1, 1, 1, 1,
       3, 0, 0, 3, 1, 3, 2, 1, 3, 1, 2, 3, 3, 2, 1, 0, 3, 1, 2, 3, 3, 0,
       2, 2, 3, 0, 2, 1, 0, 1, 2, 2, 2, 0, 2, 3, 1, 1, 0, 2, 3, 0, 1, 2,
       2, 0, 3, 3, 3, 1, 2, 3, 3, 3, 0, 0, 0, 2, 3, 3, 0, 0, 1, 3, 1, 3,
       3, 3, 0, 0, 2, 2, 3, 1, 0, 2, 0, 0, 0, 3, 3, 0, 2, 2, 1, 1, 0, 2,
       3, 3, 0, 0, 1, 3, 3, 2, 3, 0, 3, 0, 1, 0, 2, 3, 3, 2, 0, 0, 1, 2,
       3, 2, 2, 3, 1, 1, 0, 3, 3, 2, 1, 3, 3, 2, 2, 1, 0, 2, 2, 1, 0, 0,
       2, 2, 2, 2, 0, 1, 3, 0, 1, 2, 3, 0, 2, 0, 1, 1, 3, 0, 0, 1, 3, 1,
       2, 0, 2, 0, 3, 0, 3, 3, 2, 3, 1, 2, 2, 1, 1, 1, 0, 1, 0, 3, 1, 0,
       3, 0, 0, 0, 2, 0, 3, 1, 2, 0, 1, 3, 0, 2, 2, 1, 2, 1, 1, 0, 2, 0,
       0, 3, 1, 2, 3, 2, 2, 0, 3, 2, 2, 1, 3, 2, 3, 3, 3, 0, 2, 0, 3, 0,
       1, 1, 2, 2, 1, 2, 1, 2, 0, 1, 2, 3, 0, 0, 1, 3, 0, 3, 0, 1, 2, 1,
       1, 0, 3, 1, 0, 1, 3, 0, 3, 3, 0, 2, 1, 3, 1, 1, 3, 2, 0, 3, 3, 2,
       0, 0, 3, 0, 1, 1, 1, 3, 2, 3, 2, 0, 3, 0, 0, 1, 3, 0, 0, 3, 3, 2,
       2, 3, 0, 0, 1, 2, 1, 2, 0, 3, 3, 0, 2, 3, 0, 2, 2, 1, 0, 2, 2, 1,
       3, 2, 2, 0, 2, 0, 3, 3, 2, 1, 0, 3, 0, 2, 0, 0, 1, 3, 0, 3, 0, 0,
       1, 2, 0, 1, 3, 0, 2, 2, 1, 2, 0, 3, 0, 2, 3, 2, 2, 2, 3, 2, 3, 3,
       0, 0, 2, 0, 3, 1, 1, 1, 0, 3, 2, 0, 2, 0, 3, 1, 1, 0, 2, 0, 0, 3,
       2, 3, 0, 0, 2, 0, 3, 1, 0, 0, 2, 3, 0, 0, 0, 0, 2, 3, 1, 3, 2, 2,
       0, 1, 1, 2, 3, 1, 2, 1, 2, 3, 2, 2, 0, 3, 0, 3, 0, 0, 1, 1, 2, 0,
       1, 0, 0, 3, 3, 1, 3, 0, 2, 0, 0, 1, 3, 1, 2, 1, 3, 2, 1, 1, 3, 1,
       0, 1, 2, 0, 2, 2], dtype=int64)
```

In [50]:
```python
from sklearn.metrics import accuracy_score
```

In [52]:
```python
accuracy_score(y_test,predict)*100
```

Out[52]:
```
93.5
```

Predicting Value for test.csv

In [53]:
```python
test_data = pd.read_csv(r"C:\Users\Hp\Downloads\test.csv")
test_data.head()
```

Out[53]:

	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 [54]:
```python
test_data.shape
```

Out[54]:
```
(1000, 21)
```

In [55]:
```python
test_data = test_data.drop(['id'],axis=1)
test_data.shape
```

Out[55]:
```
(1000, 20)
```

In [56]:
```python
test_pred = knn.predict(test_data)
test_pred
```

Out[56]:
```
array([3, 3, 2, 3, 1, 3, 3, 1, 3, 0, 3, 3, 0, 0, 2, 0, 2, 1, 3, 2, 1, 3,
       1, 1, 3, 0, 2, 0, 3, 0, 2, 0, 3, 0, 0, 1, 3, 1, 2, 1, 1, 2, 0, 0,
       0, 1, 0, 3, 1, 2, 1, 0, 3, 0, 3, 0, 3, 1, 1, 3, 3, 2, 0, 2, 0, 1,
       2, 3, 1, 2, 1, 2, 2, 3, 3, 0, 2, 0, 2, 3, 0, 3, 3, 0, 3, 0, 3, 1,
       3, 0, 1, 2, 2, 1, 2, 2, 0, 2, 1, 2, 1, 0, 0, 3, 0, 2, 0, 1, 2, 3,
       3, 3, 1, 3, 3, 3, 3, 1, 3, 0, 0, 3, 2, 1, 2, 0, 3, 2, 3, 1, 0, 2,
       1, 1, 3, 1, 1, 0, 3, 2, 1, 3, 1, 2, 2, 3, 3, 2, 2, 3, 2, 3, 0, 0,
       3, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 1, 0, 3, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 2, 1, 0, 0, 0, 1, 2, 2, 1, 0, 0, 0, 1, 0, 3, 1, 0, 2, 2,
       2, 3, 1, 2, 3, 3, 3, 2, 2, 1, 0, 0, 1, 2, 0, 2, 3, 3, 0, 2, 0, 3,
       2, 2, 3, 1, 0, 1, 0, 3, 0, 1, 0, 2, 2, 1, 2, 0, 3, 0, 3, 1, 2, 0,
       0, 2, 1, 3, 2, 3, 1, 1, 3, 0, 0, 2, 3, 3, 1, 3, 1, 1, 3, 2, 1, 2,
       3, 3, 3, 1, 0, 0, 2, 3, 1, 1, 3, 2, 0, 3, 0, 0, 2, 0, 0, 3, 2, 3,
       3, 2, 1, 3, 3, 2, 3, 2, 2, 1, 2, 0, 2, 3, 1, 0, 0, 3, 0, 3, 0, 1,
       2, 0, 2, 3, 1, 3, 2, 2, 1, 2, 0, 0, 0, 1, 3, 2, 0, 0, 0, 3, 2, 0,
       2, 3, 1, 2, 2, 2, 3, 1, 3, 3, 2, 2, 2, 3, 3, 0, 3, 1, 3, 1, 3, 1,
       3, 3, 0, 1, 0, 3, 1, 3, 1, 3, 0, 0, 0, 0, 2, 0, 0, 2, 2, 1, 2, 2,
       2, 0, 1, 0, 0, 3, 2, 0, 3, 1, 2, 2, 1, 2, 3, 1, 1, 2, 2, 1, 2, 0,
       1, 1, 0, 3, 2, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 2, 3, 2, 3, 0, 3,
       0, 3, 0, 1, 1, 1, 1, 0, 3, 2, 3, 3, 1, 3, 1, 2, 1, 3, 2, 0, 1, 2,
       1, 1, 0, 0, 0, 1, 2, 1, 0, 3, 2, 0, 2, 2, 0, 0, 3, 1, 1, 0, 3, 2,
       3, 0, 3, 0, 2, 3, 3, 3, 0, 2, 0, 1, 3, 0, 1, 1, 0, 0, 1, 1, 1, 3,
       3, 3, 2, 3, 1, 1, 2, 2, 3, 3, 2, 0, 2, 1, 2, 2, 1, 0, 2, 2, 0, 0,
       0, 3, 1, 0, 2, 2, 2, 0, 3, 0, 2, 2, 0, 3, 0, 2, 3, 0, 1, 1, 3, 3,
       1, 1, 2, 3, 2, 0, 2, 0, 2, 0, 3, 3, 1, 2, 2, 2, 3, 0, 1, 2, 3, 1,
       3, 2, 3, 1, 0, 1, 0, 3, 1, 0, 3, 2, 3, 2, 0, 3, 3, 3, 2, 3, 3, 1,
       2, 0, 2, 2, 3, 1, 0, 1, 1, 2, 2, 2, 0, 0, 2, 2, 3, 2, 0, 2, 1, 3,
       3, 0, 1, 3, 0, 2, 1, 1, 0, 0, 2, 1, 0, 1, 1, 2, 2, 0, 2, 2, 1, 0,
       3, 0, 0, 3, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0, 3, 2, 1, 3, 3, 0, 1, 0,
       3, 2, 2, 2, 0, 3, 0, 2, 0, 2, 0, 0, 1, 1, 1, 2, 1, 3, 1, 3, 2, 2,
       1, 3, 2, 0, 1, 2, 0, 3, 3, 0, 2, 1, 1, 2, 0, 3, 2, 0, 3, 2, 3, 0,
       0, 3, 0, 1, 2, 3, 2, 2, 2, 2, 1, 1, 3, 0, 1, 1, 1, 2, 1, 0, 0, 1,
       0, 0, 3, 0, 1, 2, 0, 0, 1, 1, 3, 0, 3, 2, 3, 0, 0, 1, 2, 2, 1, 0,
       1, 2, 0, 1, 1, 0, 0, 3, 3, 0, 3, 1, 2, 3, 0, 1, 0, 2, 2, 0, 3, 1,
       0, 3, 1, 1, 0, 3, 3, 3, 2, 3, 0, 3, 2, 0, 0, 0, 3, 3, 2, 0, 1, 1,
       3, 0, 0, 3, 2, 0, 3, 1, 2, 1, 1, 1, 3, 1, 1, 1, 2, 1, 0, 2, 2, 0,
       2, 0, 0, 0, 0, 2, 3, 3, 3, 0, 1, 2, 2, 1, 0, 0, 2, 1, 0, 2, 0, 2,
       2, 2, 1, 2, 0, 2, 1, 3, 0, 0, 3, 2, 3, 0, 0, 2, 3, 3, 1, 2, 2, 1,
       0, 0, 2, 3, 0, 3, 0, 0, 0, 2, 2, 1, 2, 0, 3, 2, 1, 2, 3, 3, 0, 1,
       1, 2, 1, 2, 2, 0, 1, 3, 1, 1, 3, 0, 2, 3, 2, 1, 1, 1, 3, 2, 0, 2,
       3, 0, 2, 3, 2, 2, 2, 3, 2, 0, 1, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 1,
       1, 1, 3, 1, 0, 1, 2, 3, 1, 0, 0, 2, 2, 2, 3, 0, 3, 3, 2, 1, 3, 0,
       1, 3, 1, 2, 1, 1, 3, 2, 0, 3, 0, 2, 3, 0, 3, 2, 2, 3, 1, 0, 2, 3,
       1, 0, 2, 1, 2, 1, 2, 0, 2, 2, 0, 2, 3, 2, 3, 0, 2, 1, 1, 2, 2, 3,
       3, 0, 2, 1, 2, 1, 3, 1, 0, 3, 0, 1, 0, 0, 3, 3, 2, 0, 0, 0, 0, 3,
       2, 3, 3, 0, 0, 2, 1, 0, 2, 2], dtype=int64)
```

In [57]:
```python
test_data['predicted_price']=test_pred
```

In [58]:
```python
test_data.head()
```

Out[58]:

	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	3
2	1807	1	2.8	0	1	0	27	0.9	186	3	...	1270	1366	2396	17	10	10	0	1	1	2
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	1

5 rows × 21 columns

In [ ]: