In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
df=pd.read_csv('Downloads/Heart_Disease_Prediction.csv')
```

In [5]:

```
df.head()
```

Out[5]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Numbe o flur vesse
0	701	4	130	322	0	2	109	0	2.4	2		
1	670	3	115	564	0	2	160	0	1.6	2		
2	571	2	124	261	0	0	141	0	0.3	1		
3	641	4	128	263	0	0	105	1	0.2	2		
4	740	2	120	269	0	2	121	1	0.2	1		

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Age                     0
Sex                     0
Chest pain type         0
BP                      0
Cholesterol             0
FBS over 120            0
EKG results             0
Max HR                  0
Exercise angina         0
ST depression           0
Slope of ST             0
Number of vessels fluro 0
```

In
Thallium 0 Heart
Disease 0 dtype:
int64
[7]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
  0   Age                       270 non-null    int64
  1   Sex                       270 non-null    int64
  2   Chest pain type           270 non-null    int64
  3   BP                        270 non-null    int64
  4   Cholesterol               270 non-null    int64
  5   FBS over 120              270 non-null    int64
  6   EKG results               270 non-null    int64
  7   Max HR                    270 non-null    int64
  8   Exercise angina           270 non-null    int64
  9   ST depression             270 non-null    float64
  10  Slope of ST               270 non-null    int64
  11  Number of vessels fluro   270 non-null    int64
  12  Thallium                  270 non-null    int64
  13  Heart Disease             270 non-null    object  dtypes: float64(1), int64(12),
 object(1) memory usage: 29.7+ KB None
```

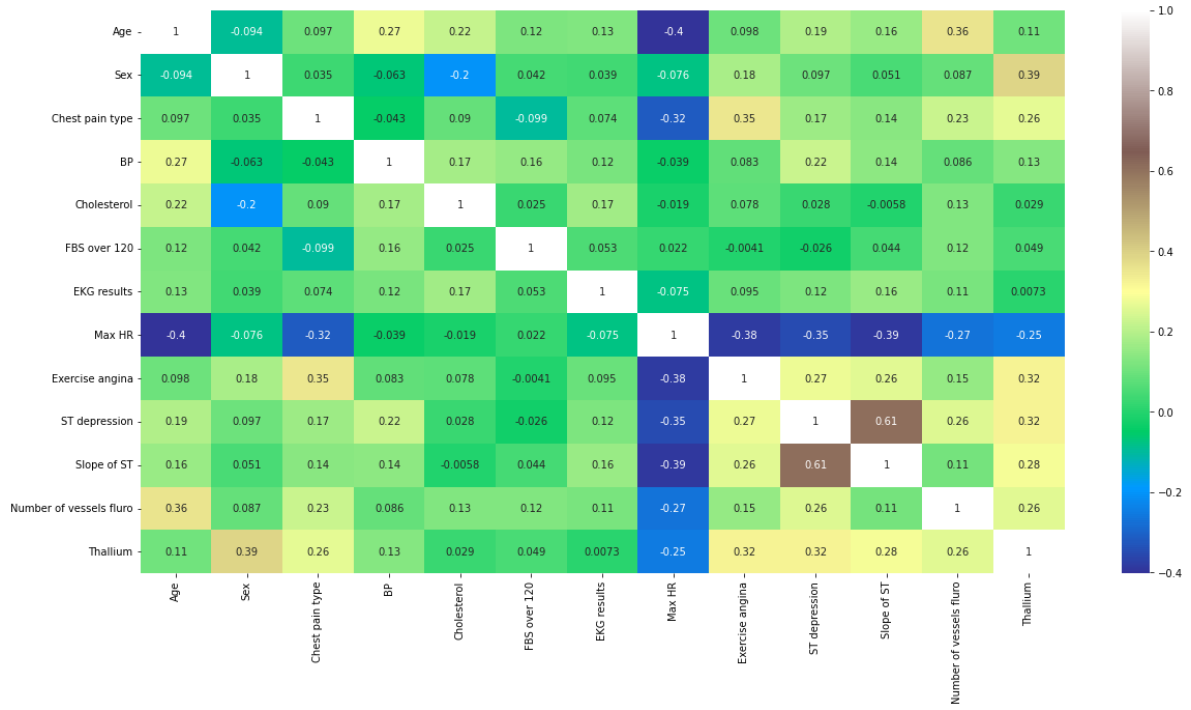
In [9]:

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(), annot=True, cmap='terrain')
```

Out[9]:

```
<AxesSubplot:>
```

In



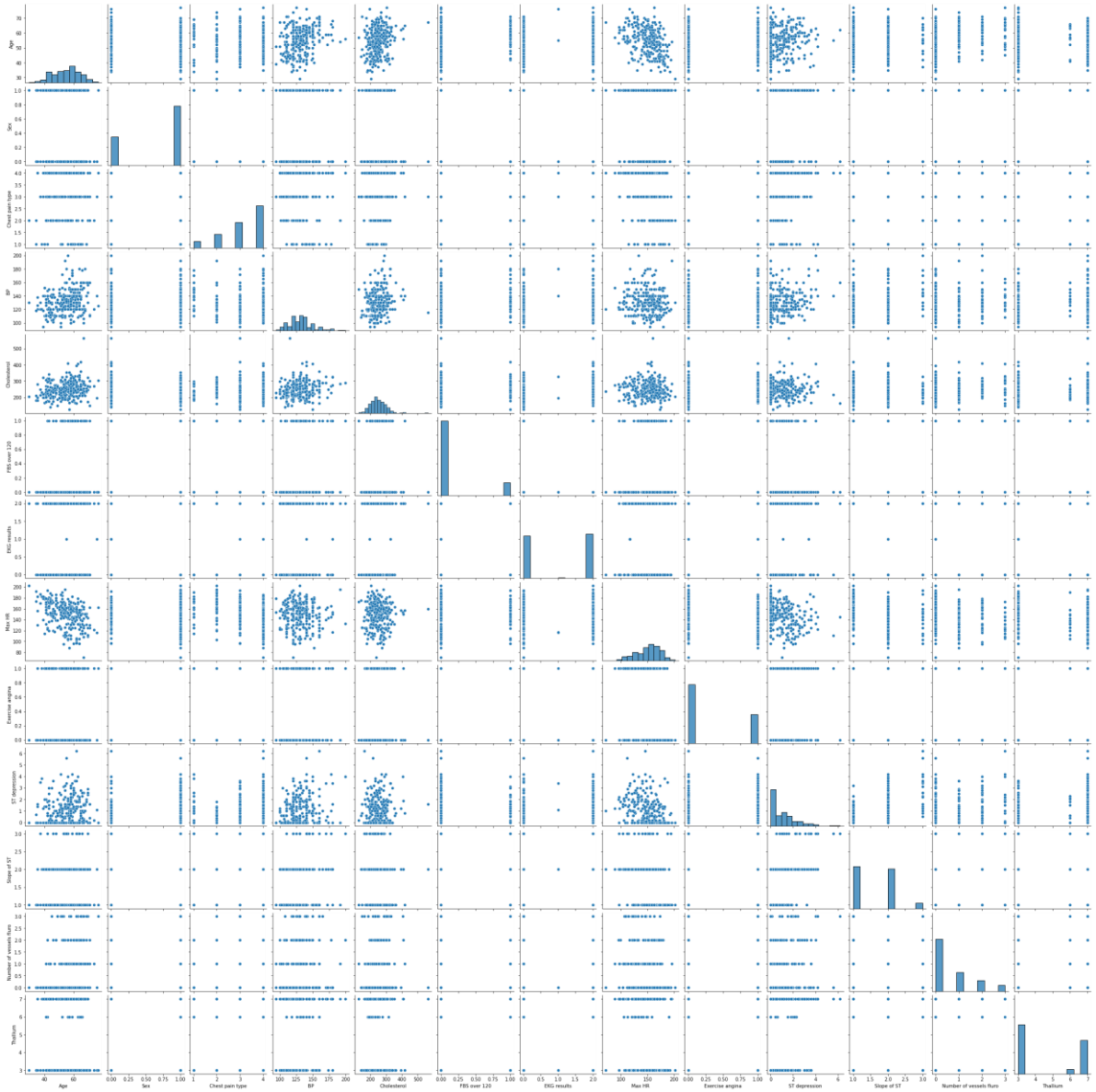
[10]:

```
sns.pairplot(data=df)
```

Out[10]:

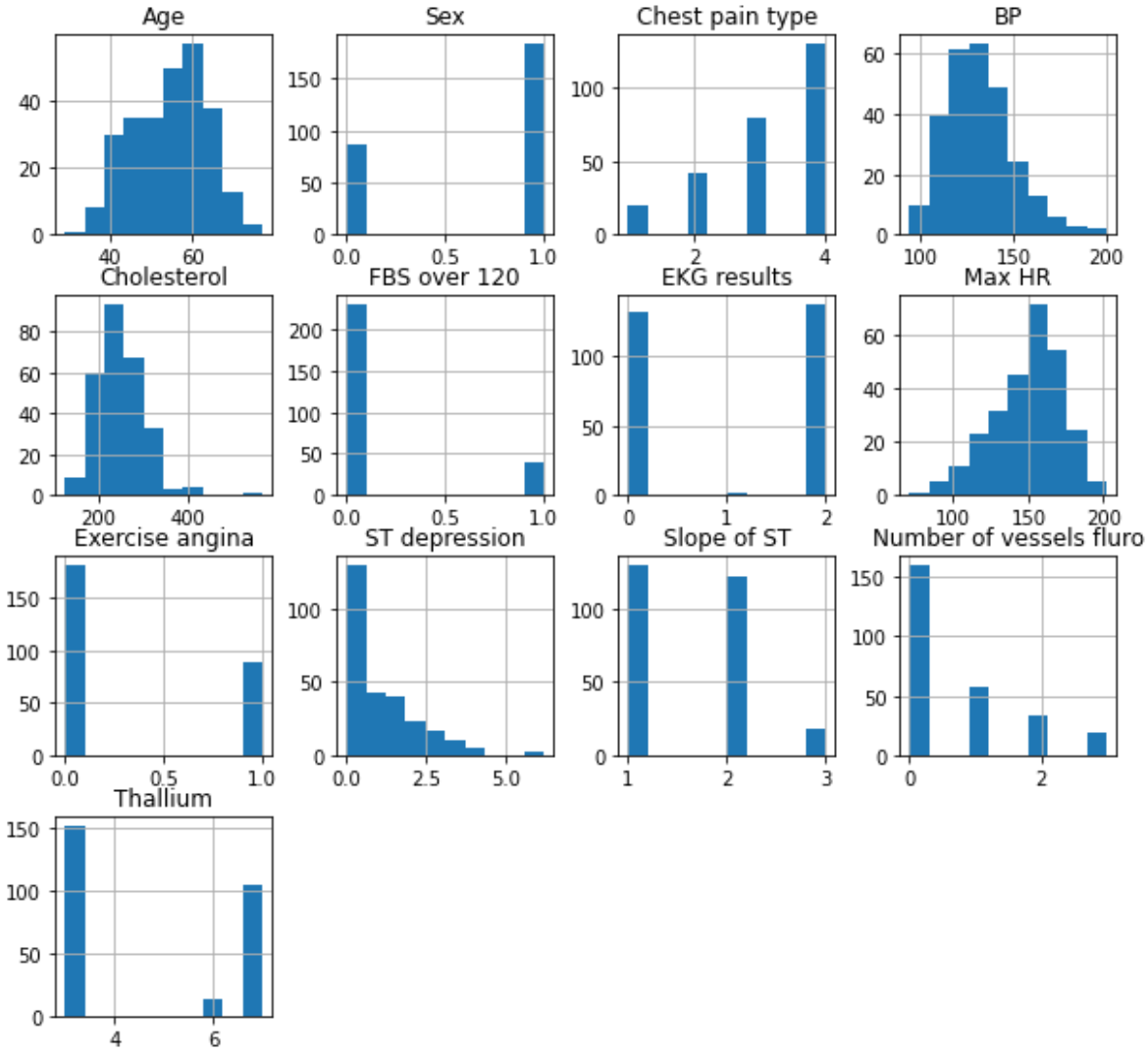
```
<seaborn.axisgrid.PairGrid at 0x2059aec2448>
```

In


Age
Sex
Chest pain type
BP
Cholesterol
FBS over 120
EKG results
Max HR
Exercise angina
ST depression
Slope of ST
Number of vessels fluro
Thallium

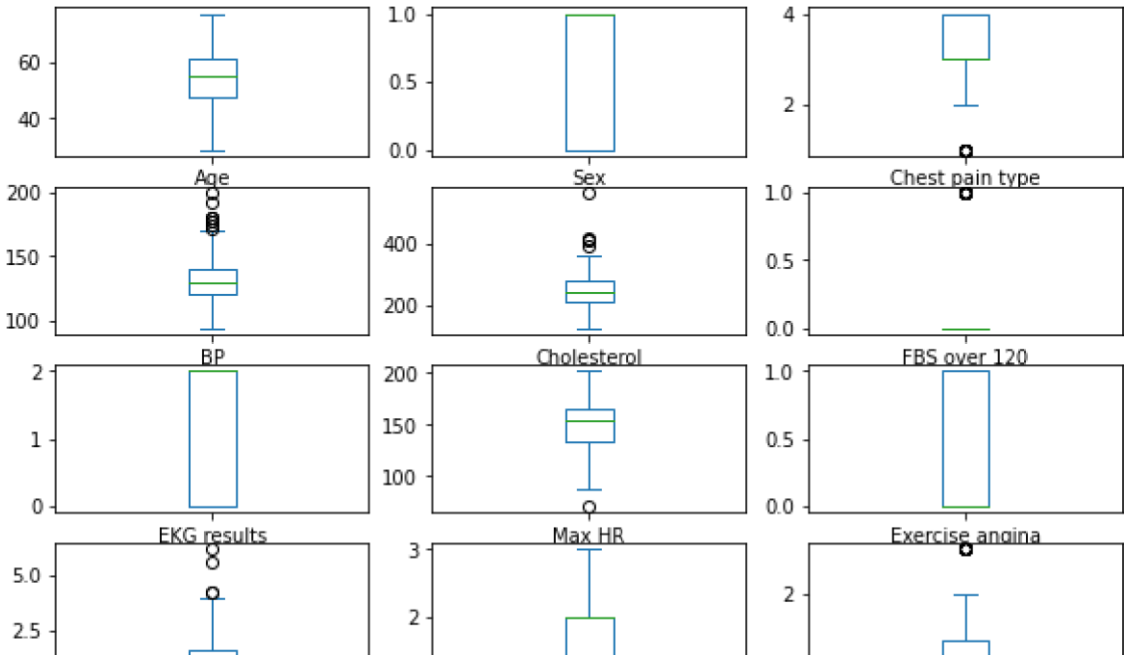
In [11]:

```
df.hist(figsize=(10,12), layout=(5,4));
```



In [13]:

```
df.plot(kind='box', subplots=True, layout=(6,3), figsize=(10,10))
plt.show()
```

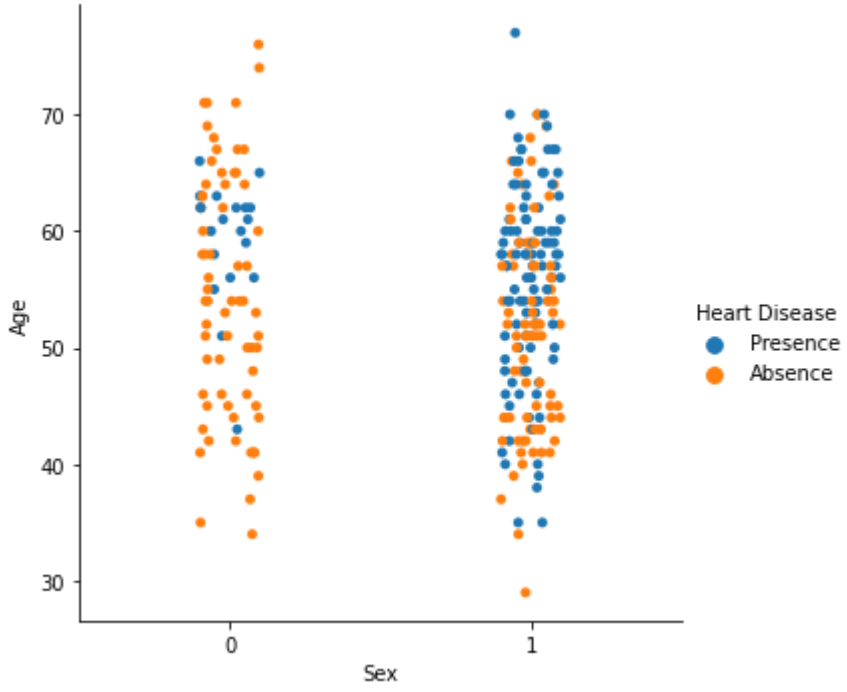


In [19]:

```
sns.catplot(data=df, x='Sex', y='Age', hue='Heart Disease', palette='tab10')
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x205a367dcc8>
```



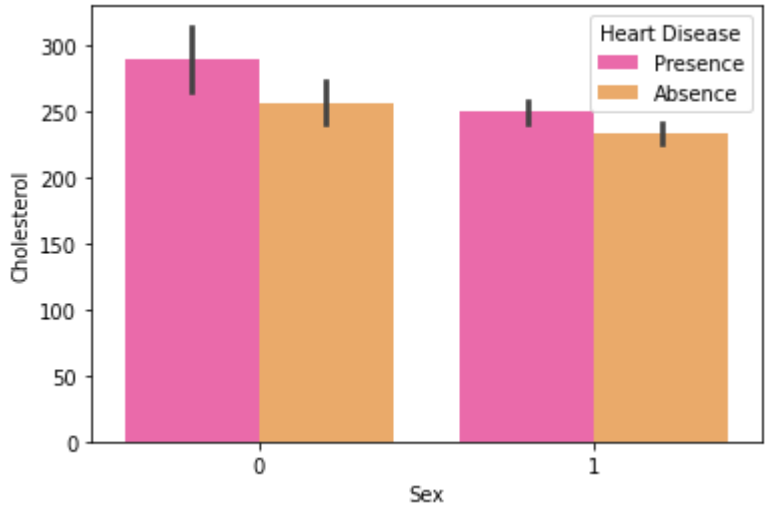
In [20]:

In

```
sns.barplot(data=df, x='Sex', y='Cholesterol', hue='Heart Disease', palette='spring')
```

Out[20]: <AxesSubplot:xlabel='Sex',

ylabel='Cholesterol'>



In [21]:

```
df['Sex'].value_counts()
```

Out[21]:

```
1    183
0     87
Name: Sex, dtype: int64
```

In [22]:

```
df['Chest pain type'].value_counts()
```

Out[22]:

```
4    129
3     79
2     42
1     20
Name: Chest pain type, dtype: int64
```

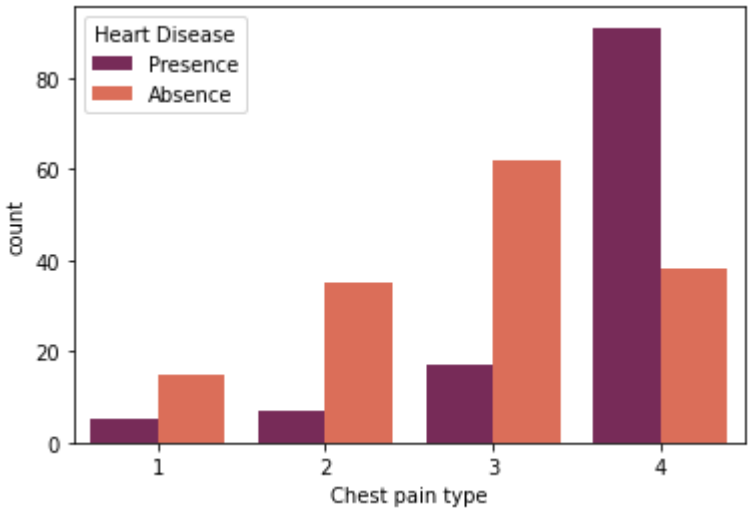
[23]:

```
sns.countplot(x='Chest pain type', hue='Heart Disease' , data=df, palette='rocket')
```

In

Out[23]: <AxesSubplot:xlabel='Chest pain type',

ylabel='count'>



In [24]:

```
gen = pd.crosstab(df['Sex'], df['Heart Disease'])
print(gen)
```

```
Heart Disease  Absence  Presence
Sex
0                   67        20
1                   83       100
```

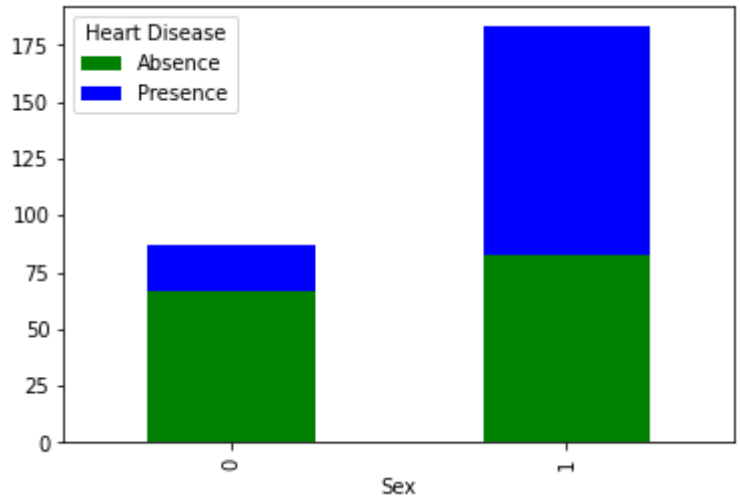
[25]:

```
gen.plot(kind='bar', stacked='True', color=['green','blue'],grid=False)
```

Out[25]:

<AxesSubplot:xlabel='Sex'>

In



In [42]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale=['Age', 'EKG results', 'Cholesterol', 'Thallium', 'Number of vessels fluro

df[columns_to_scale] = StandardScaler.fit_transform(df[columns_to_scale]) In [43]:
```

```
df.head()
```

Out[43]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	
0	1.712094	1	4	130	1.402212	0	0.981664	109	0	2.4	2	
1	1.382140	0	3	115	6.093004	0	0.981664	160	0	1.6	2	
2	0.282294	1	2	124	0.219823	0	-1.026285	141	0	0.3	1	
3	1.052186	1	4	128	0.258589	0	-1.026285	105	1	0.2	2	
4	2.152032	0	2	120	0.374890	0	0.981664	121	1	0.2	1	

[44]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale=['Age', 'EKG results', 'Cholesterol', 'Thallium', 'Number of vessels fluro

df[columns_to_scale] = StandardScaler.fit_transform(df[columns_to_scale]) In [45]:
```

In

```
df.head()
```

Out[45]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	
0	1.712094	1	4	130	1.402212	0	0.981664	109	0	2.4	2	
1	1.382140	0	3	115	6.093004	0	0.981664	160	0	1.6	2	
2	0.282294	1	2	124	0.219823	0	-1.026285	141	0	0.3	1	
3	1.052186	1	4	128	0.258589	0	-1.026285	105	1	0.2	2	
4	2.152032	0	2	120	0.374890	0	0.981664	121	1	0.2	1	

In [47]:

```
x=df.drop(['Heart Disease'], axis=1)
y=df['Heart Disease']
```

In [48]:

```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3, random_state=40)
```

In [49]:

```
print('x_train-', x_train.size)
print('x_test-', x_test.size)
print('y_train-', y_train.size)
print('x_test-', x_test.size)
```

```
x_train- 2457
x_test- 1053
y_train- 189 x_test-
1053
```

[73]:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
model1=lr.fit(x_train,y_train)
prediction1=model1.predict(x_test)
```

In [54]:

In

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,prediction1)
cm
```

Out[54]:

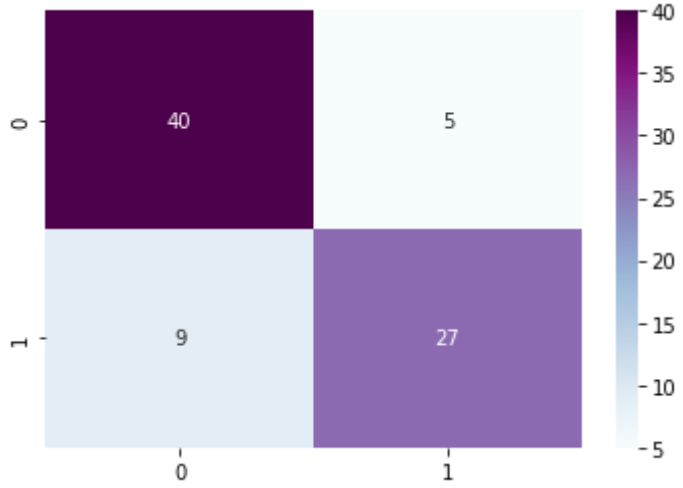
```
array([[40,  5],
       [ 9, 27]], dtype=int64)
```

In [56]:

```
sns.heatmap(cm, annot=True,cmap='BuPu')
```

Out[56]:

<AxesSubplot:>



In [60]:

```
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]
print('Testing Accuracy:', (TP+TN+FN)/(TP+TN+FN+FP))
```

Testing Accuracy: 0.9382716049382716

In [70]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,prediction1)
```

Out[70]:

```
0.8271604938271605
```

In [62]:

In

```
from sklearn.metrics import classification_report
print(classification_report(y_test, prediction1))
```

```
              precision    recall  f1-score   support

     Absence       0.82      0.89      0.85        45
    Presence       0.84      0.75      0.79        36

    accuracy                           0.83        81
   macro avg       0.83      0.82      0.82        81
weighted avg       0.83      0.83      0.83        81
```

In [77]:

```
print('NB  :', accuracy_score(y_test, prediction1))
```

```
NB  : 0.8271604938271605
```