In [36]:
```
# importing the necessary libraries

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

In [37]:
```
#taken data from Cleaveland UCI
#This is a CSV file so we need to import it into our project using pan
# please change the following argument to the filepath where the datas
heart_disease = pd.read_csv('/Users/fizzausman/Desktop/heart_clevelanc
```

In [38]:
```
#this shows the result of the data frame
heart_disease
```

Out[38]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	cor
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
292	40	1	3	152	223	0	0	181	0	0.0	0	0	2	
293	39	1	3	118	219	0	0	140	0	1.2	1	0	2	
294	35	1	3	120	198	0	0	130	1	1.6	1	0	2	
295	35	0	3	138	183	0	0	182	0	1.4	0	0	0	
296	35	1	3	126	282	0	2	156	1	0.0	0	0	2	

297 rows × 14 columns

In [39]:
```
#this shows the column names or attributes
heart_disease.keys()
```

Out[39]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'tha
lach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'condition'],
      dtype='object')
```

In [40]:
```
#shows each row values
heart_disease.values
```

Out[40]:
```
array([[69.,  1.,  0., ...,  1.,  0.,  0.],
       [69.,  0.,  0., ...,  2.,  0.,  0.],
       [66.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [35.,  1.,  3., ...,  0.,  2.,  1.],
       [35.,  0.,  3., ...,  0.,  0.,  0.],
       [35.,  1.,  3., ...,  0.,  2.,  1.]])
```

In [41]:
```
#shows whether heart condition is present for each row in dataset. 0 =
print(heart_disease['condition'])
```

```
0      0
1      0
2      0
3      1
4      0
      ..
292    1
293    1
294    1
295    0
296    1
Name: condition, Length: 297, dtype: int64
```

In [42]:
```
#another way to get column names
print(heart_disease.columns.tolist())
```

```
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
'exang', 'oldpeak', 'slope', 'ca', 'thal', 'condition']
```

In [43]:
```
#to get the shape of the data. here it is 297 by 14
heart_disease.shape
```

Out[43]:
```
(297, 14)
```

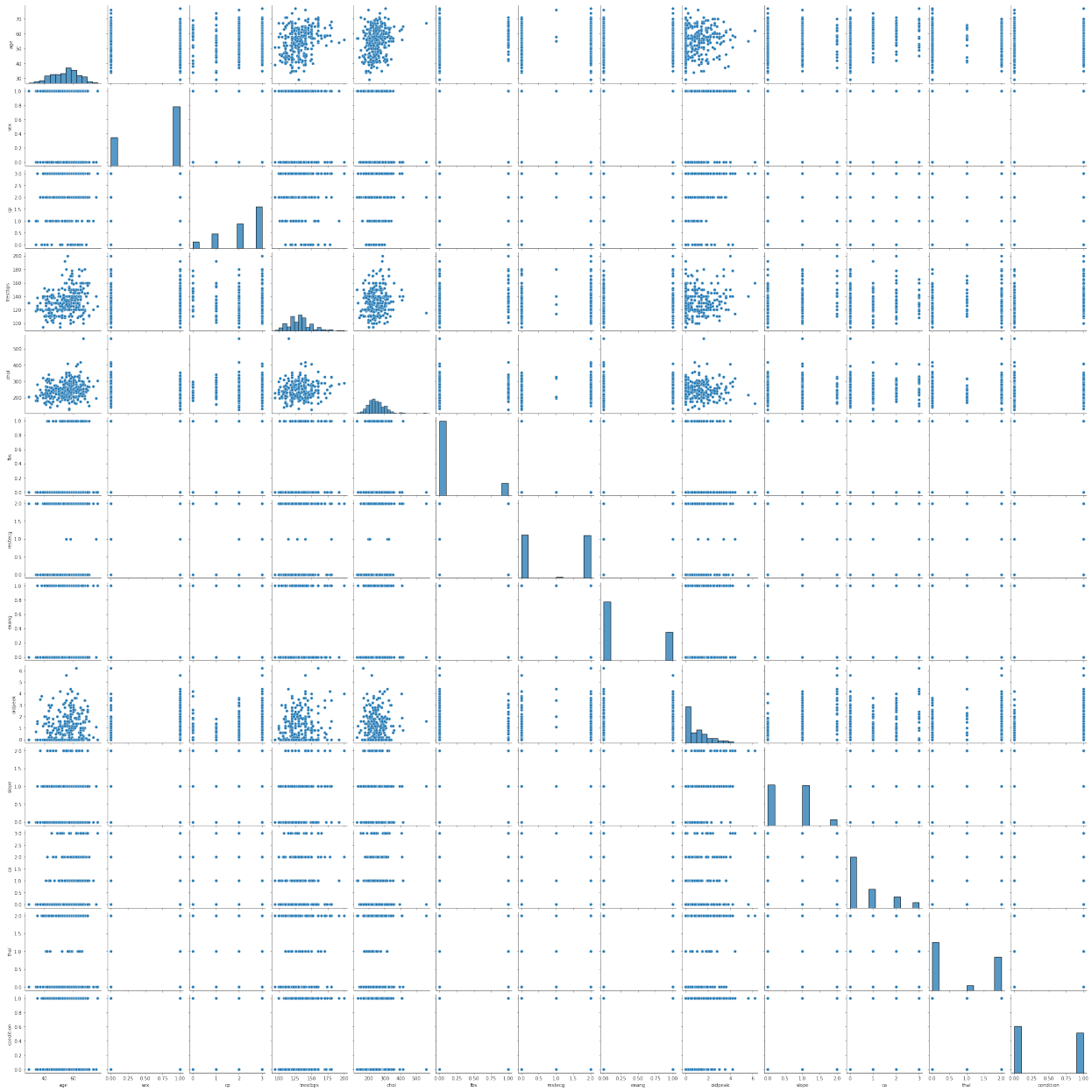
In [44]:
```
#shows first few rows for the dataset
heart_disease.head()
```

Out[44]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condi
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	

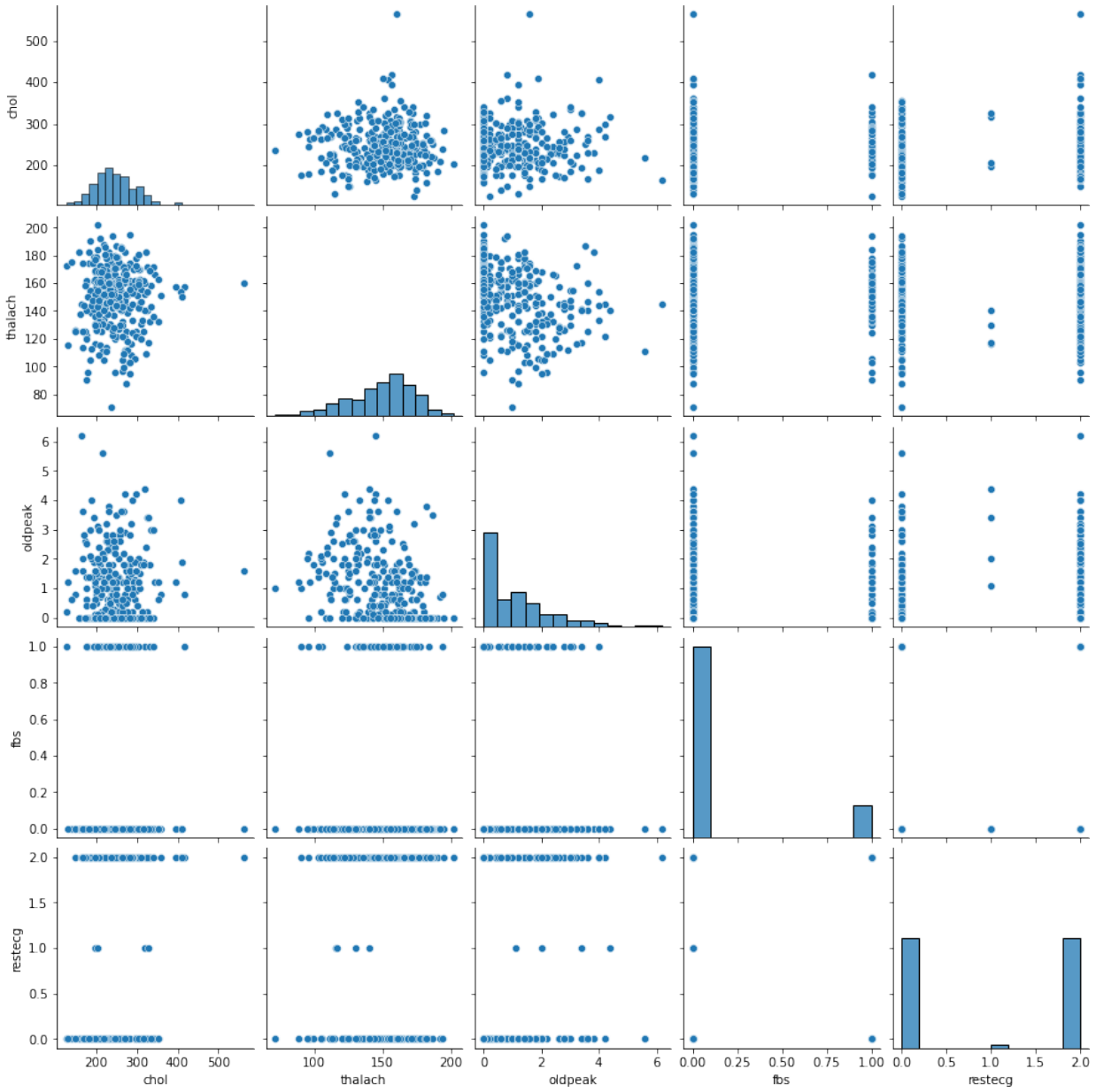
In [45]:
```
#makes a pairplot with each of the variables on both x and y axis to s
sns.pairplot(heart_disease, vars = ['age','sex','cp','trestbps','chol'
```

Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x7fc163b37eb0>
```



In [46]:
```
#taking only a few variables to show how powerpul pairplots are
sns.pairplot(heart_disease, vars=['chol','thalach','oldpeak','fbs','re
```

Out[46]: <seaborn.axisgrid.PairGrid at 0x7fc16166adf0>

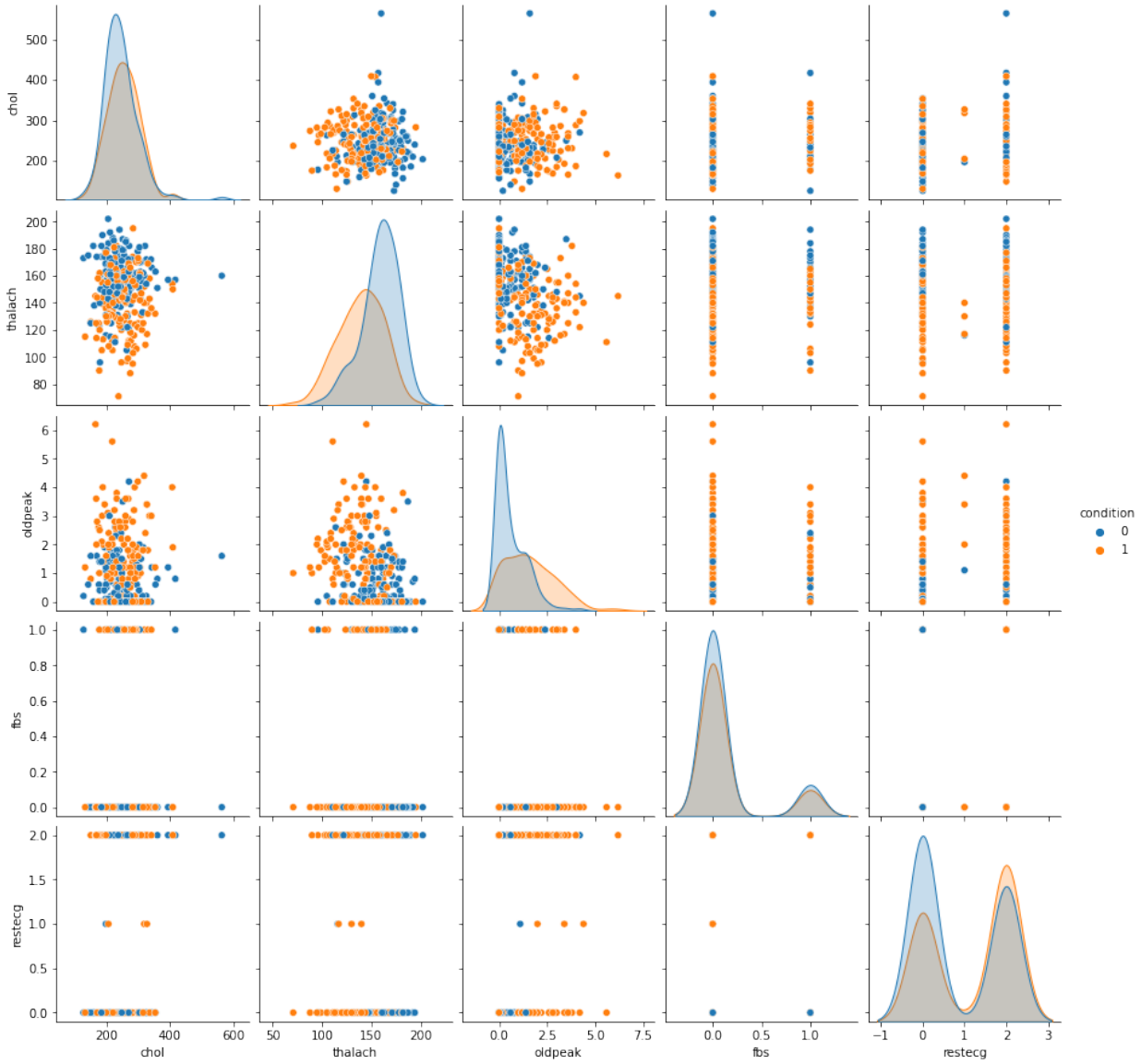


In [47]:
```
plt.figure(figsize=(10,8), dpi= 80)
sns.pairplot(heart_disease, kind="scatter", hue="condition", plot_kws=
plt.show()
```

```
<Figure size 800x640 with 0 Axes>
```

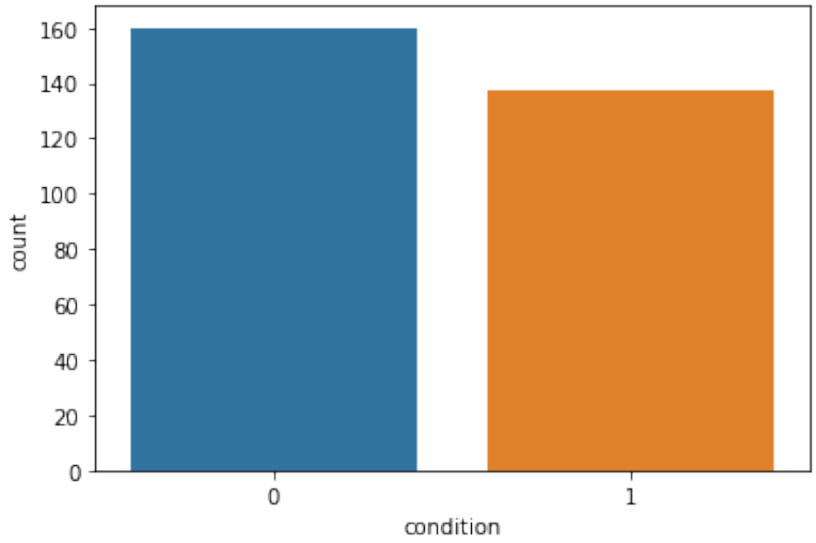
In [48]:
```
#blue points are where heart disease is not present and orange points
sns.pairplot(heart_disease, hue = 'condition', vars = ['chol', 'thalac
 'restecg'])
```

Out[48]: <seaborn.axisgrid.PairGrid at 0x7fc173bc88e0>



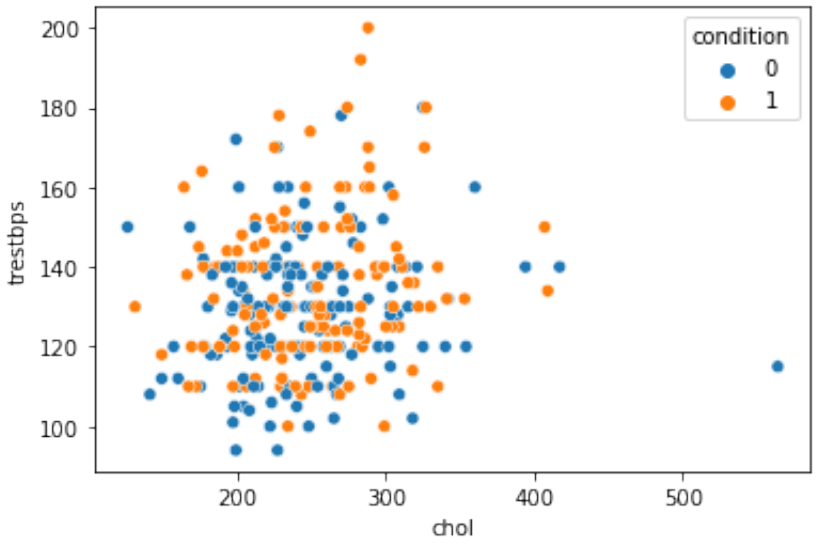
In [49]:
```
#this simply tells how many heart cases have heart disease in the data

sns.countplot(x='condition', data= heart_disease)
```

Out[49]: <AxesSubplot:xlabel='condition', ylabel='count'>



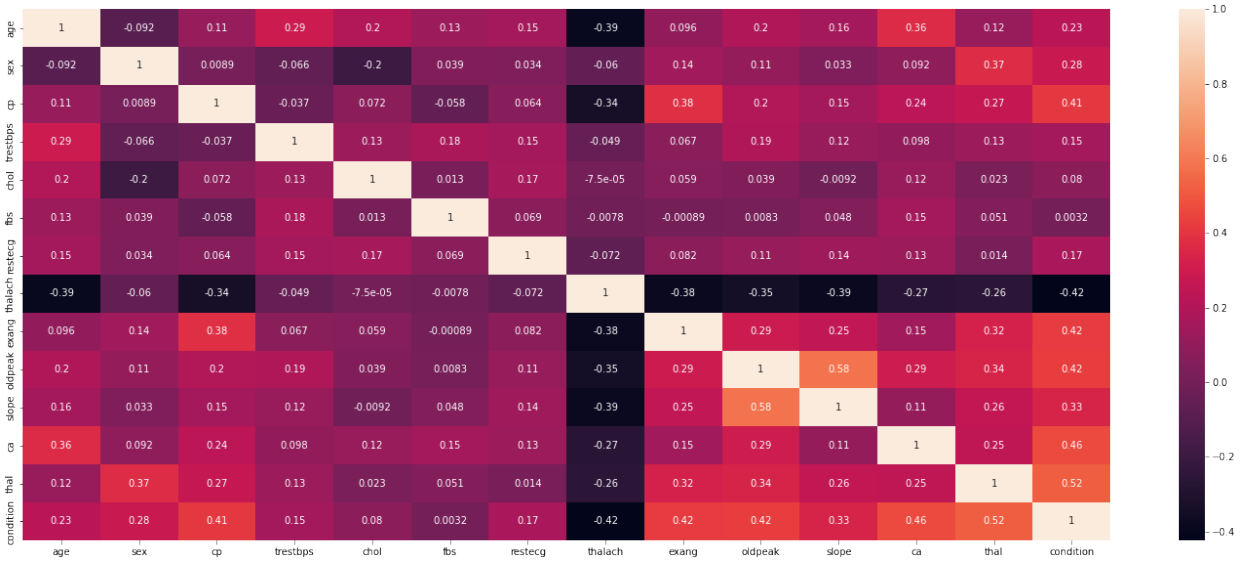
In [50]:
```
# scatterplot shows the relationship between cholestrol levels and res
sns.scatterplot(x='chol',y='trestbps',hue='condition',data=heart_disea
```

Out[50]: <AxesSubplot:xlabel='chol', ylabel='trestbps'>



In [51]:
```
#heatmap is plotted to show correlation of all variables in the datase
plt.figure(figsize=(25,10))
sns.heatmap(heart_disease.corr(), annot=True)
```

Out[51]: <AxesSubplot:>



DATA SPLITTING FOR MACHINE LEARNING

Here we break the data into smaller data frames x and y so that we can split them into training and testing data to create machine learning model

In [52]:
```
x = heart_disease.drop(['condition'],axis=1)
```

In [53]:
```
x
```

Out[53]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
292	40	1	3	152	223	0	0	181	0	0.0	0	0	2
293	39	1	3	118	219	0	0	140	0	1.2	1	0	2
294	35	1	3	120	198	0	0	130	1	1.6	1	0	2
295	35	0	3	138	183	0	0	182	0	1.4	0	0	0
296	35	1	3	126	282	0	2	156	1	0.0	0	0	2

297 rows × 13 columns

In [54]:
```
y = heart_disease['condition']
```

In [55]:
```
y
```

Out[55]:
```
0      0
1      0
2      0
3      1
4      0
      ..
292    1
293    1
294    1
295    0
296    1
Name: condition, Length: 297, dtype: int64
```

In [56]:
```
from sklearn.model_selection import train_test_split
```

In [57]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0
```

In [58]:
```
x_train
```

Out[58]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
19	42	1	0	148	244	0	2	178	0	0.8	0	2	0
258	52	1	3	125	212	0	0	168	0	1.0	0	2	2
82	66	0	2	146	278	0	2	152	0	0.0	1	1	0
66	41	1	1	110	235	0	0	153	0	0.0	0	0	0
203	60	1	3	117	230	1	0	160	1	1.4	0	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
58	45	1	1	128	308	0	2	170	0	0.0	0	0	0
87	64	1	2	140	335	0	0	158	0	0.0	0	0	0
197	61	1	3	148	203	0	0	161	0	0.0	0	1	2
174	64	1	3	120	246	0	2	96	1	2.2	2	1	0
122	51	0	2	130	256	0	2	149	0	0.5	0	0	0

207 rows × 13 columns

In [59]:
```
y_train
```

Out[59]:
```
19     0
258    1
82     0
66     0
203    1
      ..
58     0
87     1
197    1
174    1
122    0
Name: condition, Length: 207, dtype: int64
```

In [60]:
```
x_test
```

Out[60]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
245	54	1	3	120	188	0	0	113	0	1.4	1	1	2
61	44	1	1	120	220	0	0	170	0	0.0	0	0	0
69	35	1	1	122	192	0	0	174	0	0.0	0	0	0
24	71	0	1	160	302	0	0	162	0	0.4	0	2	0
116	52	0	2	136	196	0	2	169	0	0.1	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
103	57	1	2	128	229	0	2	150	0	0.4	1	1	2
113	54	0	2	160	201	0	0	163	0	0.0	0	1	0
48	52	1	1	128	205	1	0	184	0	0.0	0	0	0
266	49	0	3	130	269	0	0	163	0	0.0	0	0	0
55	46	0	1	105	204	0	0	172	0	0.0	0	0	0

90 rows × 13 columns

In [61]:
```
y_test
```

Out[61]:
```
245    1
61     0
69     0
24     0
116    0
      ..
103    1
113    0
48     0
266    0
55     0
Name: condition, Length: 90, dtype: int64
```

LOGISTIC REGRESSION AND MACHINE LEARNING MODEL

In [62]:
```
from sklearn.linear_model import LogisticRegression

#confusion matrix is a 2d array that is used to compare predicted data
#false positive, and false negative

#classification report measures the quality of predictions against act
#for us. Accuracy is the measure of how many data points were predicte

from sklearn.metrics import classification_report , confusion_matrix
```

In [63]:
```
logistic_model = LogisticRegression(random_state = 0)
logistic_model.fit(x_train, y_train)
```

```
/Users/fizzausman/opt/anaconda3/lib/python3.9/site-packages/sklearn/l
inear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sho
wn in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options
:
    https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression (https://scikit-learn.org/stable/modules/linear_model.ht
ml#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[63]:
```
LogisticRegression(random_state=0)
```

In [66]:
```
y_predict =logistic_model.predict(x_test)
y_predict
```

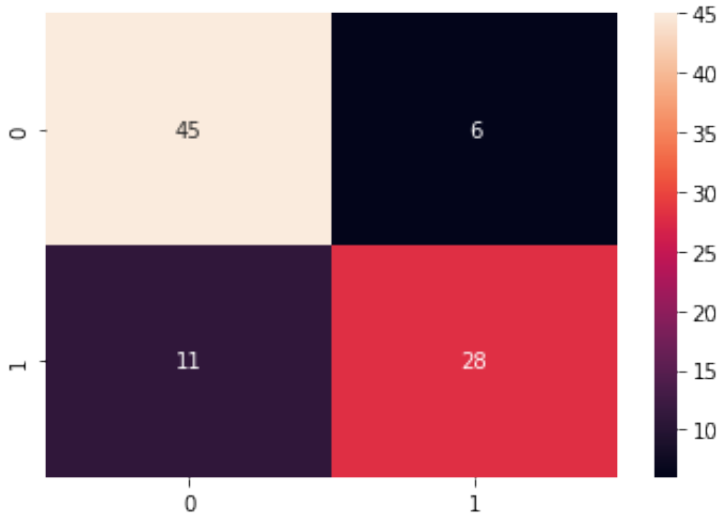
Out[66]:
```
array([1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
0,
       0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0,
0,
       0, 0])
```

In [67]:
```
cm = confusion_matrix(y_test,y_predict)
```

the below heatmap shows that out of 90 data points, 45 were true negatives, 28 were true positives, 11 were false positives and 6 were false negatives

In [68]:
```
sns.heatmap(cm ,annot=True)
```

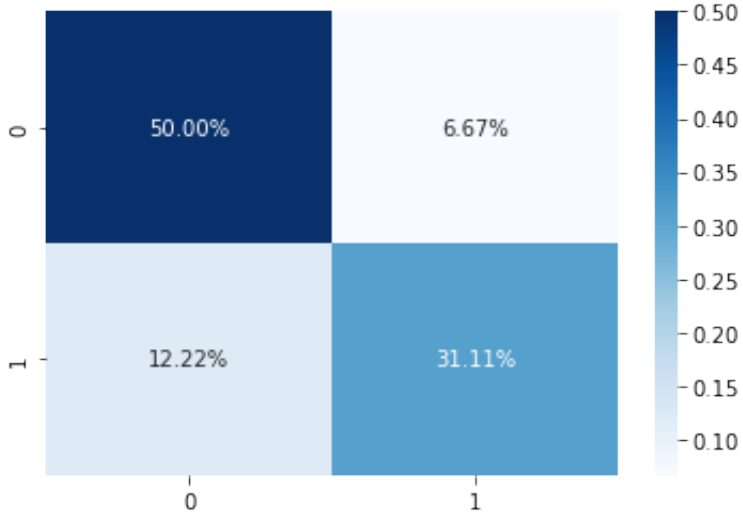
Out[68]: <AxesSubplot:>



the below heatmap shows a confusion matrix in terms of percentages

In [69]:
```
sns.heatmap(cm/np.sum(cm), annot=True,
            fmt='.2%', cmap='Blues')
```

Out[69]: <AxesSubplot:>



IMPROVING THE MODEL

In [70]:
```
logistic_model.fit(x_train,y_train)
```

```
/Users/fizzausman/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

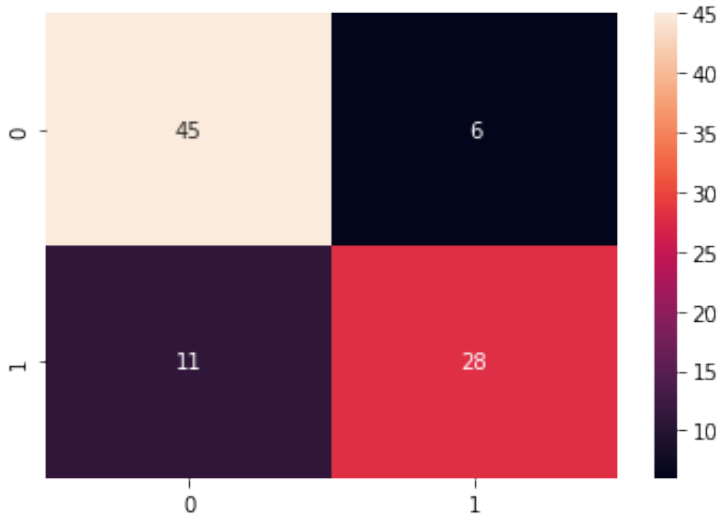
Out[70]:
```
LogisticRegression(random_state=0)
```

In [71]:
```
y_predict =logistic_model.predict(x_test)
```

In [72]:
```
cn = confusion_matrix(y_test,y_predict)
```

In [73]:
```
sns.heatmap(cn,  annot = True)
```

Out[73]:
```
<AxesSubplot:>
```



In [74]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84        51
           1       0.82      0.72      0.77        39

    accuracy                           0.81        90
   macro avg       0.81      0.80      0.80        90
weighted avg       0.81      0.81      0.81        90
```

In [ ]:
```
e accuracy we recieved is 81% or 81.1% for exact accuracy. The accurac
```

Using KNN classifier or K - nearest neighbours classification algorithm

In [85]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report , confusion_matrix
```

Since we are calculating standard euclidean distance we will use minkowski metric with a power parameter of 2

In [86]:
```
knn_model = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski'
knn_model.fit(x_train, y_train)
```

Out[86]:
```
KNeighborsClassifier(n_neighbors=3)
```

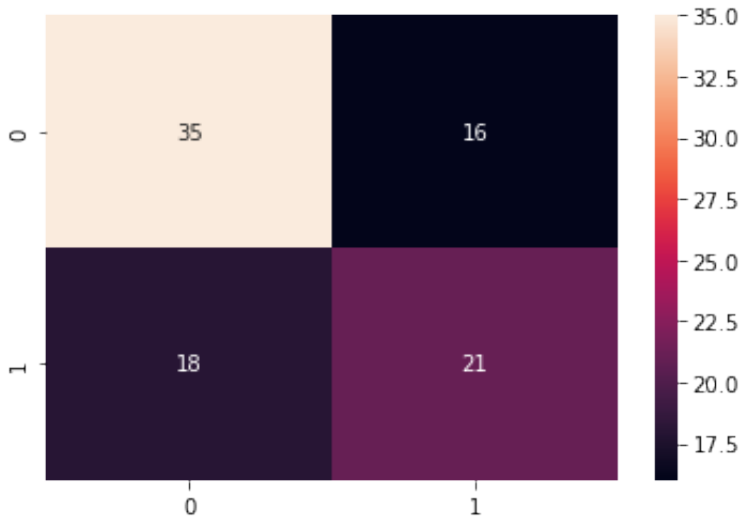
In [87]:
```
y_predict =knn_model.predict(x_test)
y_predict
```

Out[87]:
```
array([1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0])
```

In [88]:
```
cm = confusion_matrix(y_test, y_predict)
```

In [95]:
```
sns.heatmap(cm ,annot=True)
```

Out[95]: <AxesSubplot:>



In [98]:
```
from sklearn import metrics
print("Accuracy for training set: ", metrics.accuracy_score(y_train, k
print("Accuracy for test set : ", metrics.accuracy_score(y_test, knn_m
```

```
Accuracy for training set:  0.8019323671497585
Accuracy for test set :  0.6222222222222222
```

In [99]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.66      0.69      0.67        51
           1       0.57      0.54      0.55        39

    accuracy                           0.62        90
   macro avg       0.61      0.61      0.61        90
weighted avg       0.62      0.62      0.62        90
```

In order to get higher accuracy, you can change the number of neighbours and run the algorithm again.