In [25]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

df = pd.read_csv('BostonHousing.csv')
```

In [28]:
```
df.corr()['medv'].sort_values()
```

Out[28]:
```
lstat     -0.737663
ptratio   -0.507787
indus     -0.483725
tax       -0.468536
nox       -0.427321
crim      -0.388305
rad       -0.381626
age       -0.376955
chas       0.175260
dis        0.249929
b          0.333461
zn         0.360445
rm         0.695360
medv       1.000000
Name: medv, dtype: float64
```

In [30]:
```
x = df.loc[:, df.columns !='medv'].values
y = df.loc[:,df.columns == 'medv'].values
x_train,x_test,y_train,y_test= train_test_split(x,y,random_state = 45, test
```

In [32]:
```
scaler=StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [35]:
```
3,),activation='relu', name='dense_1'),Dense(64,activation='relu',name='den
['mae'])
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 128)               1792

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 10,113
Trainable params: 10,113
Non-trainable params: 0
_________________________________________________________________
```

In [37]:
```
model.fit(x_train,y_train,epochs=100,validation_split=0.05,verbose='auto')
y_pred=model.predict(x_test)
```

```
12/12 [==============================] - 0s 14ms/step - loss: 76.2766
- mae: 6.8955 - val_loss: 100.1668 - val_mae: 8.8121
Epoch 7/100
12/12 [==============================] - 0s 14ms/step - loss: 55.1359
- mae: 5.5588 - val_loss: 64.4091 - val_mae: 7.0157
Epoch 8/100
12/12 [==============================] - 0s 13ms/step - loss: 40.5468
- mae: 4.6741 - val_loss: 40.6364 - val_mae: 5.5201
Epoch 9/100
12/12 [==============================] - 0s 14ms/step - loss: 31.1596
- mae: 4.0035 - val_loss: 27.2923 - val_mae: 4.3666
Epoch 10/100
12/12 [==============================] - 0s 15ms/step - loss: 26.5183
- mae: 3.6514 - val_loss: 16.6301 - val_mae: 3.2562
Epoch 11/100
12/12 [==============================] - 0s 14ms/step - loss: 23.8151
- mae: 3.4333 - val_loss: 13.6345 - val_mae: 2.9521
Epoch 12/100
12/12 [==============================] - 0s 14ms/step - loss: 21.6712
- mae: 3.3061 - val_loss: 11.2334 - val_mae: 2.6816
```

In [11]:
```
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

Type *Markdown* and LaTeX: α^2

In [20]:
```
model = Sequential(layers=[Dense(128,input_shape=(13,),activation='relu',na
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
y_pred = model.predict(x_test)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 128)               1792

 dense_2 (Dense)             (None, 64)                8256

 dense_output (Dense)        (None, 1)                 65

=================================================================
Total params: 10,113
Trainable params: 10,113
Non-trainable params: 0
_________________________________________________________________
4/4 [==============================] - 0s 3ms/step
```

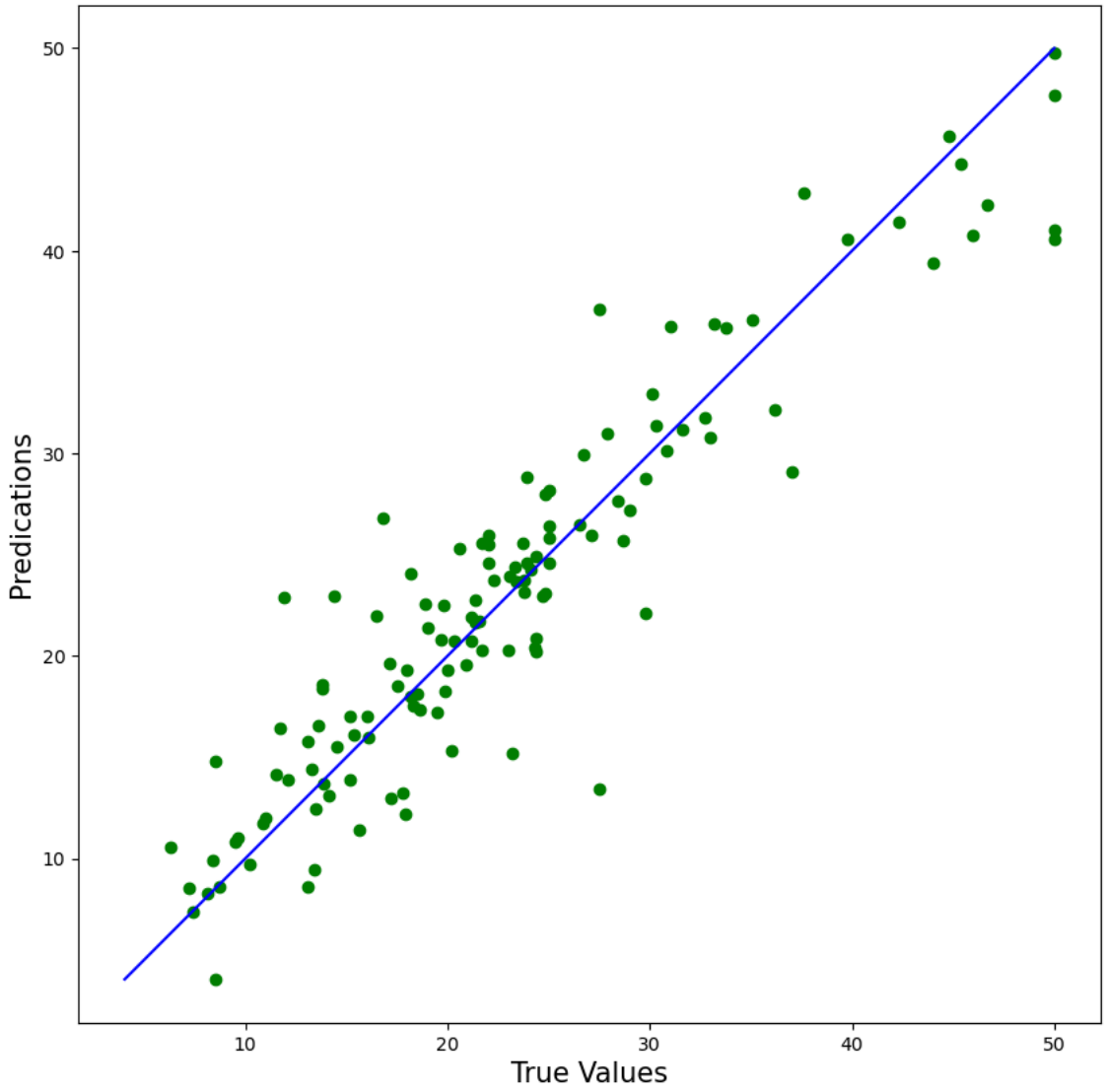
In [21]:
```
model.fit(x_train, y_train,epochs=100, validation_split=0.05,verbose='auto'
```

```
mae: 3.1592 - val_loss: 13.6953 - val_mae: 2.9135
Epoch 16/100
12/12 [==============================] - 0s 6ms/step - loss: 20.0171 -
mae: 3.0759 - val_loss: 13.7497 - val_mae: 2.8731
Epoch 17/100
12/12 [==============================] - 0s 6ms/step - loss: 18.8681 -
mae: 3.0049 - val_loss: 12.0295 - val_mae: 2.7179
Epoch 18/100
12/12 [==============================] - 0s 6ms/step - loss: 18.0340 -
mae: 2.9003 - val_loss: 10.4263 - val_mae: 2.5326
Epoch 19/100
12/12 [==============================] - 0s 6ms/step - loss: 17.2821 -
mae: 2.8350 - val_loss: 9.0797 - val_mae: 2.3955
Epoch 20/100
12/12 [==============================] - 0s 6ms/step - loss: 16.6764 -
mae: 2.7844 - val_loss: 8.1228 - val_mae: 2.2281
Epoch 21/100
12/12 [==============================] - 0s 7ms/step - loss: 16.0062 -
mae: 2.7213 - val_loss: 7.4336 - val_mae: 2.1446
Epoch 22/100
```

In [40]:
```
mae,mse=model.evaluate(x_test,y_test)
print(mae)
print(mse)
plt.figure(figsize=(10,10))
plt.scatter(y_test, y_pred, c='green')

p1= max(max(y_pred), max(y_test))
p2 = min(min(y_pred), min(y_test))
plt.plot([p1,p2],[p1,p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predications', fontsize=15)
plt.axis('equal')
plt.show()
```

```
4/4 [==============================] - 0s 7ms/step - loss: 14.1755 - mae:
2.7194
14.175531387329102
2.7193710803985596
```



In [ ]: