In [10]:
```
#Import neccessry libraries
```

In [ ]:
```
#import pandas as pd
#import numpy as np
#import matplotlib.pyplot as plt
#import seaborn as sns
#from sklearn.model_selection import train_test_split
#from sklearn.linear_model import LinearRegression
#from sklearn.preprocessing import StandardScaler
#from sklearn.feature_selection import SelectFromModel
#from sklearn.metrics import mean_squared_error, accuracy_score
```

Load the dataset

In [11]:
```
import pandas as pd
import numpy as np
```

In [43]:
```
import pandas as pd
```

In [42]:
```
data = pd.read_csv("C:\\Users\\HP\\Desktop\\Student_performance 02.csv")
data
```

Out[42]:

	Reading_score	Writing_score
0	72	74
1	90	88
2	95	93
3	57	44
4	78	75
...	...	...
995	99	95
996	55	55
997	71	65
998	78	77
999	86	86

1000 rows × 2 columns

In [13]:

```
x= np.array(data["Reading_score"]).reshape((-1,1))
x
```

Out[13]:

```
array([[ 72],
       [ 90],
       [ 95],
       [ 57],
       [ 78],
       [ 83],
       [ 95],
       [ 43],
       [ 64],
       [ 60],
       [ 54],
       [ 52],
       [ 81],
       [ 72],
       [ 53],
       [ 75],
       [ 89],
       [ 32],
       [ 42],
```

In [14]:

```
y=np.array(data["Writing_score"]).reshape((-1,1))
y
```

Out[14]:

```
array([[ 74],
       [ 88],
       [ 93],
       [ 44],
       [ 75],
       [ 78],
       [ 92],
       [ 39],
       [ 67],
       [ 50],
       [ 52],
       [ 43],
       [ 73],
       [ 70],
       [ 58],
       [ 78],
       [ 86],
       [ 28],
       [ 46],
```

In [15]:

```
#checking for miising data
data.isnull().sum()
```

Out[15]:

```
Reading_score    0
Writing_score    0
dtype: int64
```

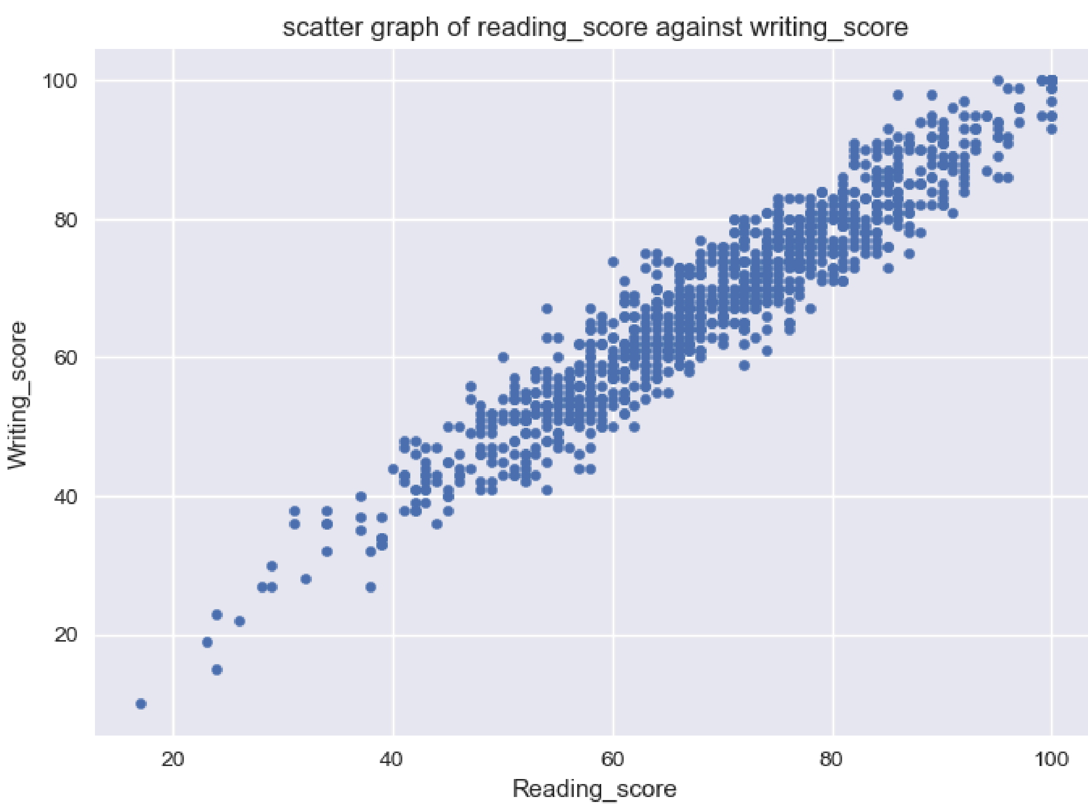
In [16]:
```
#Visualization
```

In [17]:
```
#importing necessary libraries
import matplotlib.pyplot as plt
from matplotlib import style
from statistics import mean
```

In [48]:
```
style.use("seaborn")
plt.title("scatter graph of reading_score against writing_score")
plt.xlabel("Reading_score")
plt.ylabel("Writing_score")
plt.scatter(x,y,label="Data points" ,s=20,alpha=1)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11720\2026039172.py:1: Matplotl
ibDeprecationWarning: The seaborn styles shipped by Matplotlib are depre
cated since 3.6, as they no longer correspond to the styles shipped by s
eaborn. However, they will remain available as 'seaborn-v0_8-<style>'. A
lternatively, directly use the seaborn API instead.
  style.use("seaborn")
```

Out[48]: <matplotlib.collections.PathCollection at 0x17956607c10>



In [19]:
```
#splitting data
```

In [20]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=20,random_sta
#x_train
#x_test
#y_train
#y_test
```

In [21]:
```
#standardizing data
```

In [22]:
```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
x_train_scaled=Scaler.fit_transform(x_train)
#x_train_scaled
x_test_scaled=Scaler.transform(x_test)
#x_test_scaled
```

In [23]:
```
#Building the model
```

In [24]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

In [25]:
```
#Fitting the model
```

In [26]:
```
model.fit(x_train_scaled,y_train)
```

Out[26]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [27]:
```
#Making prediction
```

In [28]:
```
y_pred=model.predict(x_test_scaled)
y_pred
```

Out[28]:
```
array([[84.75119417],
       [64.92874298],
       [71.86660089],
       [75.83109113],
       [81.77782649],
       [73.84884601],
       [68.89323322],
       [59.97313018],
       [71.86660089],
       [53.03527227],
       [46.09741435],
       [25.2838406 ],
       [78.80445881],
       [63.93762042],
       [79.79558137],
       [76.82221369],
       [51.05302715],
       [45.10629179],
       [56.9997625 ],
       [63.93762042]])
```

In [29]:
```
#getting the coefficeint
model.coef_
```

Out[29]:
```
array([[14.45177144]])
```

In [30]:
```
#getting the intercept
model.intercept_
```

Out[30]:
```
array([68.1377551])
```

MODEL EVALUATION

In [31]:
```
#model accuracy on train values
model.score(x_train_scaled,y_train)
```

Out[31]:
```
0.9109678497742956
```

In [32]:
```
#model accuracy on test values
model.score(x_test_scaled,y_test)
```

Out[32]:
```
0.9171117131668868
```

In [33]:
```
#getting the errors
```

In [34]:
```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_errc

mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)

print(f"mse:{mse}")
print(f"r2:{r2}")
print(f"mae:{mae}")
```

```
mse:23.428167052662367
r2:0.9171117131668868
mae:3.780159548387915
```

MODEL OPTIMIZATION

In [35]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

In [36]:
```
#Perform GridSeracrchCV to find optimal alpha for Ridge Regression
param_grid = {'alpha':[0.1, 1, 10,100]}
ridge_model = Ridge()
grid_search = GridSearchCV(ridge_model, param_grid, cv=5)
grid_search.fit(x_train_scaled, y_train)
best_alpha = grid_search.best_params_['alpha']
```

In [37]:
```
# Train the Ridge Regression model with the best alpha
ridge_model = Ridge(alpha=best_alpha)
ridge_model.fit(x_train_scaled, y_train)
```

Out[37]:
```
Ridge(alpha=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
# Mke predictions
y_pred_ridge = ridge_model.predict(x_test_scaled)
```

In [39]:
```
#Evaluate model performance
mae_ridge = mean_absolute_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
```

In [40]:

```
print("Ridge Regression:")
print(f"Best alpha: {best_alpha}")
print(f"MAE: {mae_ridge}")
print(f"R^2: {r2_ridge}")
print(f"MSE: {mse_ridge}")
```

```
Ridge Regression:
Best alpha: 1
MAE: 3.784730002987117
R^2: 0.9169271613991024
MSE: 23.480330148447223
```