In [1]:
```
# project 2 to find the strength of concrete
#name: SR VIGNESH
#mail: vignesh.sr2020@vitstudent.ac.in
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

In [4]:
```
df=pd.read_csv("https://raw.githubusercontent.com/Premalatha-success/Yhills_July12_Analytics/main/concrete.csv")
df
```

Out[4]:

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
0	141.3	212.0	0.0	203.5	0.0	971.8	748.5	28	29.89
1	168.9	42.2	124.3	158.3	10.8	1080.8	796.2	14	23.51
2	250.0	0.0	95.7	187.4	5.5	956.9	861.2	28	29.22
3	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85
4	154.8	183.4	0.0	193.3	9.1	1047.4	696.7	28	18.29
...	...	...	...	...	...	...	...	...	...
1025	135.0	0.0	166.0	180.0	10.0	961.0	805.0	28	13.29
1026	531.3	0.0	0.0	141.8	28.2	852.1	893.7	3	41.30
1027	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1028	342.0	38.0	0.0	228.0	0.0	932.0	670.0	270	55.06
1029	540.0	0.0	0.0	173.0	0.0	1125.0	613.0	7	52.61

1030 rows × 9 columns

In [6]:
```
df.dtypes
```

Out[6]:
```
cement          float64
slag            float64
ash             float64
water           float64
superplastic    float64
coarseagg       float64
fineagg         float64
age               int64
strength        float64
dtype: object
```

In [7]:
```
df.describe()
```

Out[7]:

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
cement          0
slag            0
ash             0
water           0
superplastic    0
coarseagg       0
fineagg         0
age             0
strength        0
dtype: int64
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   cement        1030 non-null   float64
 1   slag          1030 non-null   float64
 2   ash           1030 non-null   float64
 3   water         1030 non-null   float64
 4   superplastic  1030 non-null   float64
 5   coarseagg     1030 non-null   float64
 6   fineagg       1030 non-null   float64
 7   age           1030 non-null   int64  
 8   strength      1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

In [17]:

```
num=df.select_dtypes(include=['int64','float64']).keys()
from sklearn.impute import SimpleImputer
impute=SimpleImputer(strategy='mean')
impute_fit=impute.fit(df[num])
df[num]=impute_fit.transform(df[num])
df
```

Out[17]:

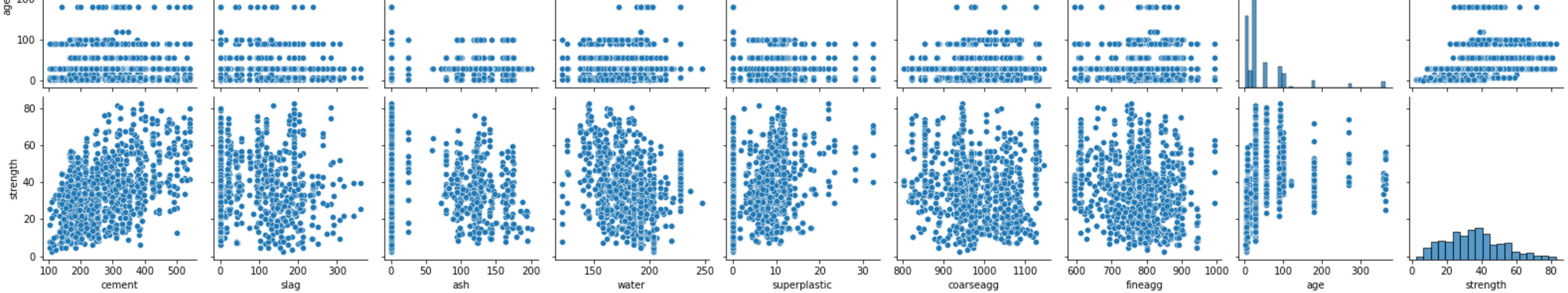
	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
0	141.3	212.0	0.0	203.5	0.0	971.8	748.5	28.0	29.89
1	168.9	42.2	124.3	158.3	10.8	1080.8	796.2	14.0	23.51
2	250.0	0.0	95.7	187.4	5.5	956.9	861.2	28.0	29.22
3	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28.0	45.85
4	154.8	183.4	0.0	193.3	9.1	1047.4	696.7	28.0	18.29
...	...	...	...	...	...	...	...	...	...
1025	135.0	0.0	166.0	180.0	10.0	961.0	805.0	28.0	13.29
1026	531.3	0.0	0.0	141.8	28.2	852.1	893.7	3.0	41.30
1027	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28.0	44.28
1028	342.0	38.0	0.0	228.0	0.0	932.0	670.0	270.0	55.06
1029	540.0	0.0	0.0	173.0	0.0	1125.0	613.0	7.0	52.61

1030 rows × 9 columns

In [18]:
```
#EDA
sb.pairplot(df)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x27899ab9b50>
```

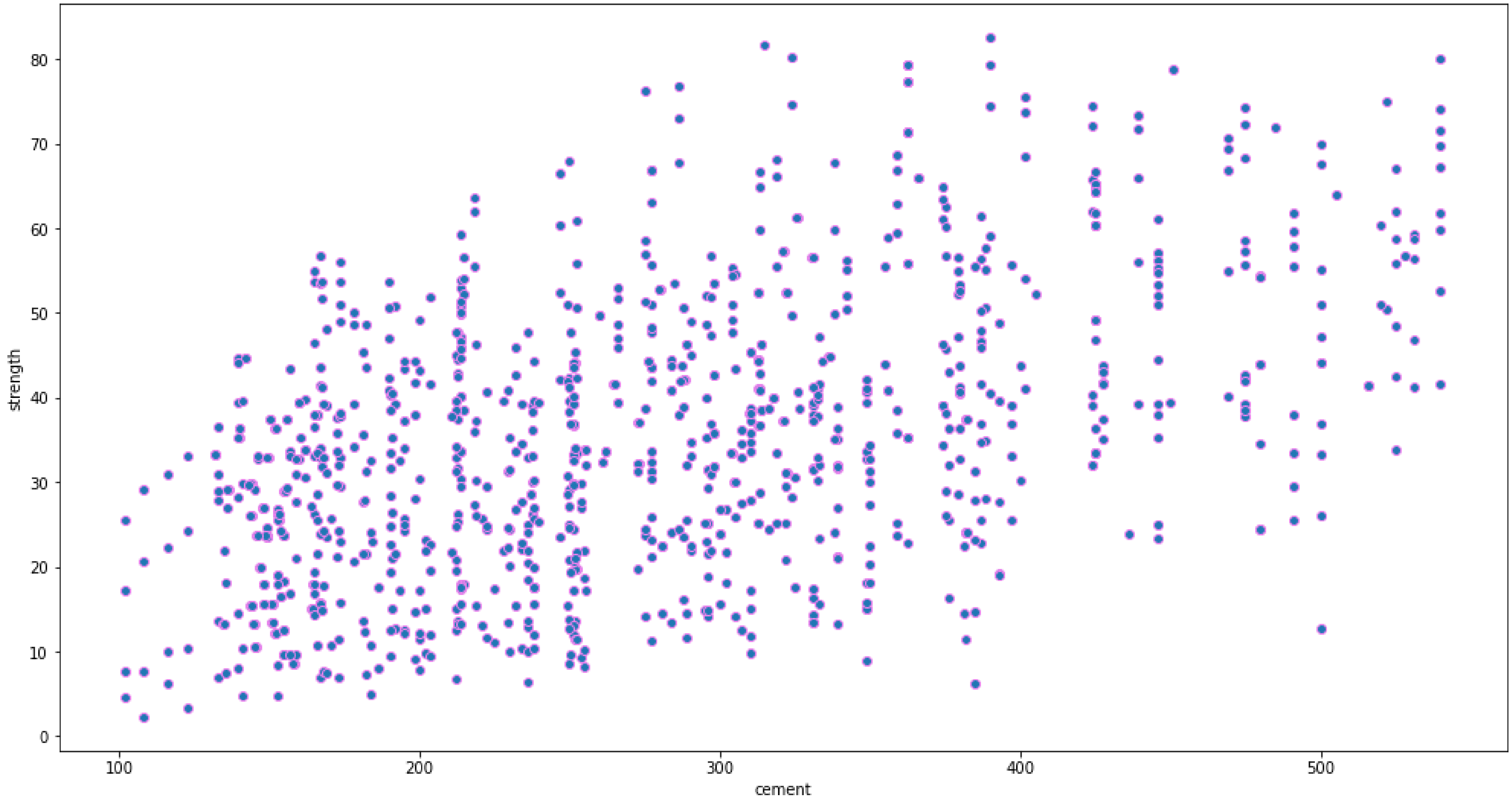

cement
slag
ash
water
superplastic
coarseagg
fineagg



age
strength
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
strength

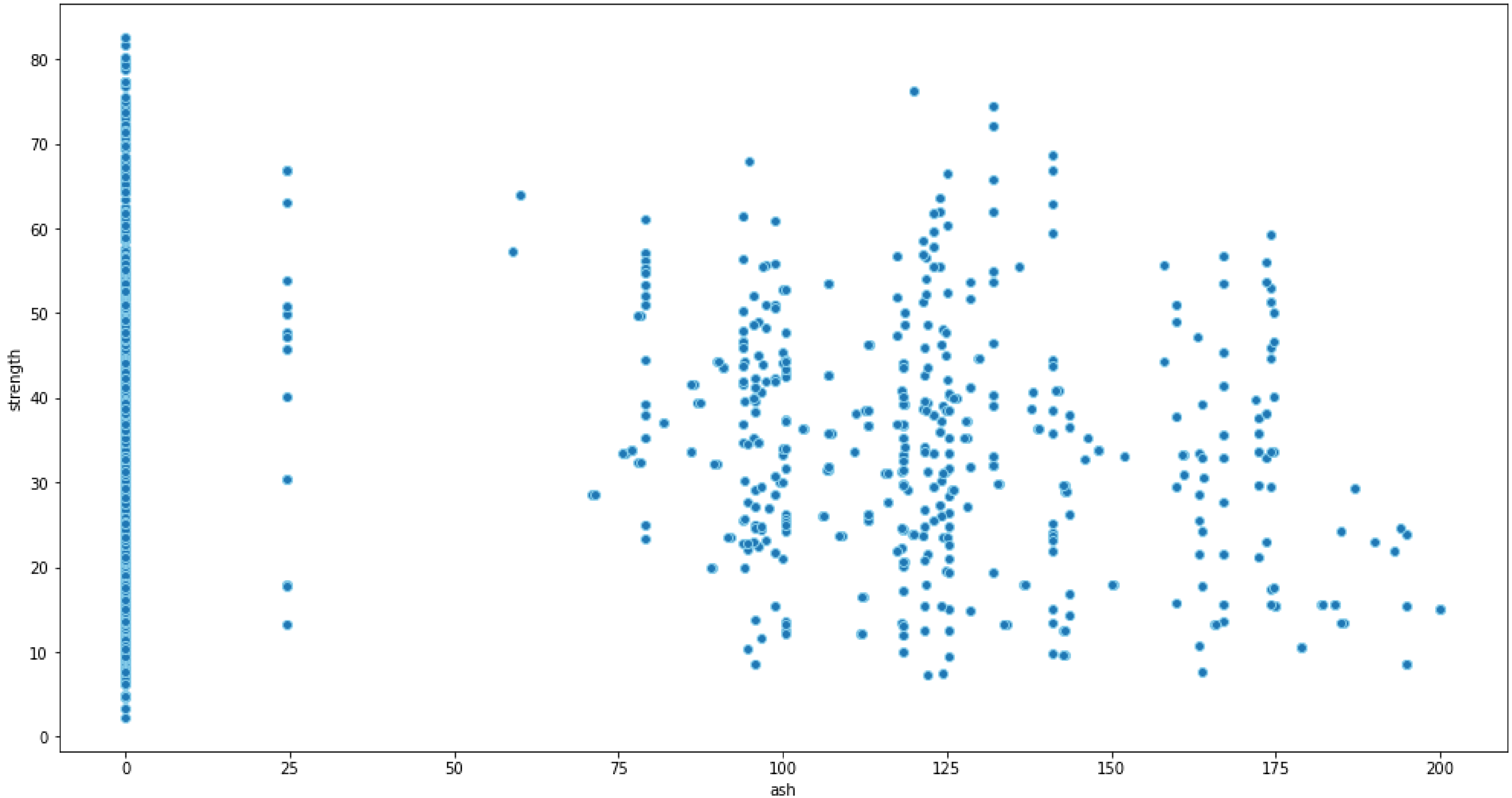
In [19]:
```python
#scatter plot
plt.figure(figsize=[17,9])
plt.scatter(y='strength',x='cement',edgecolors='violet',data=df)
plt.ylabel('strength')
plt.xlabel('cement')
```

Out[19]:
```
Text(0.5, 0, 'cement')
```



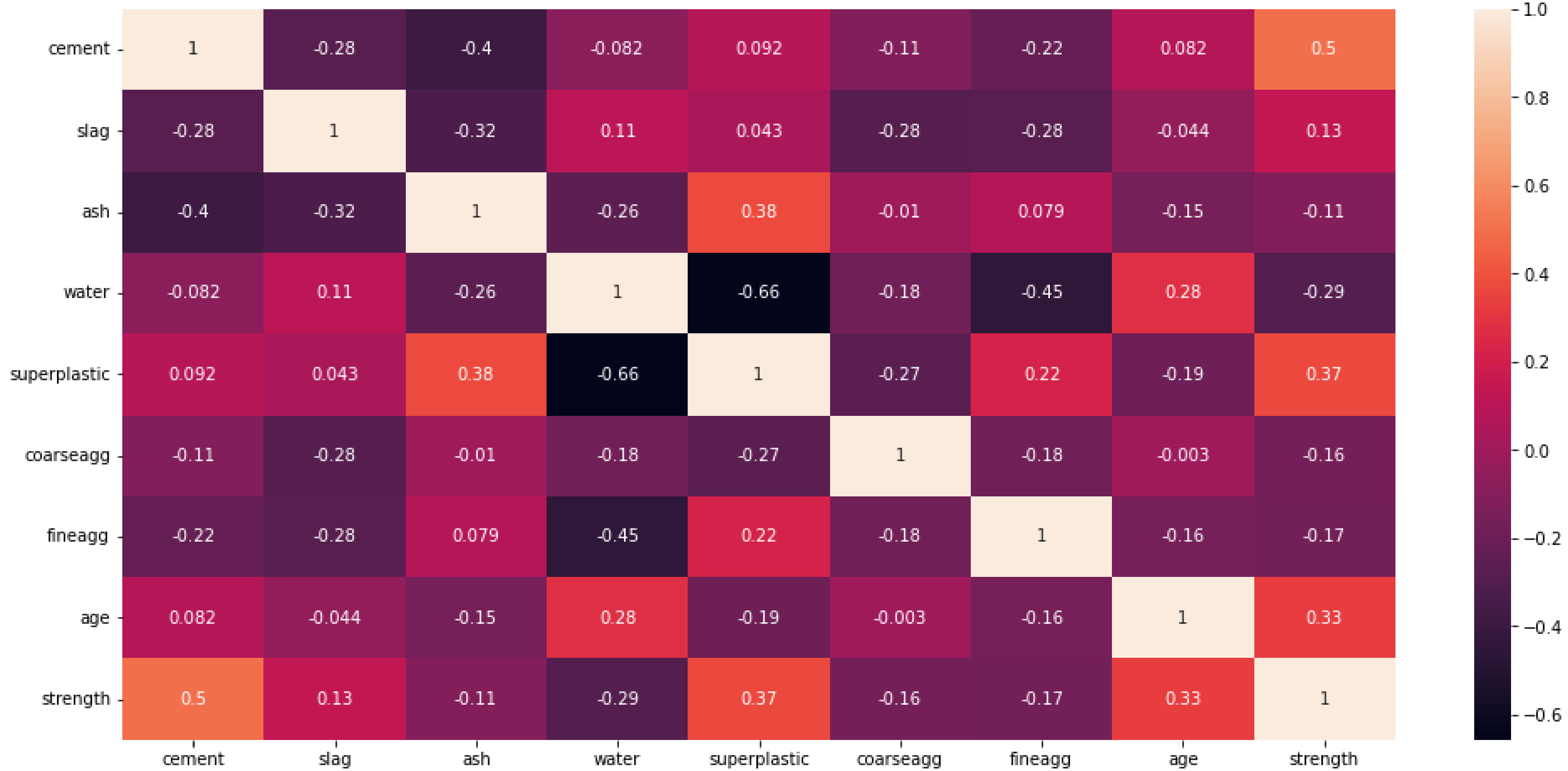
In [21]:
```
plt.figure(figsize=[17,9])
plt.scatter(y='strength',x='ash',edgecolors='skyblue',data=df)
plt.ylabel('strength')
plt.xlabel('ash')
```

Out[21]: Text(0.5, 0, 'ash')

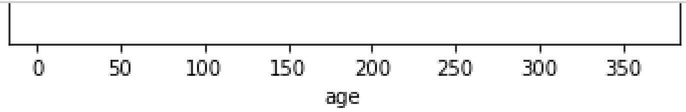


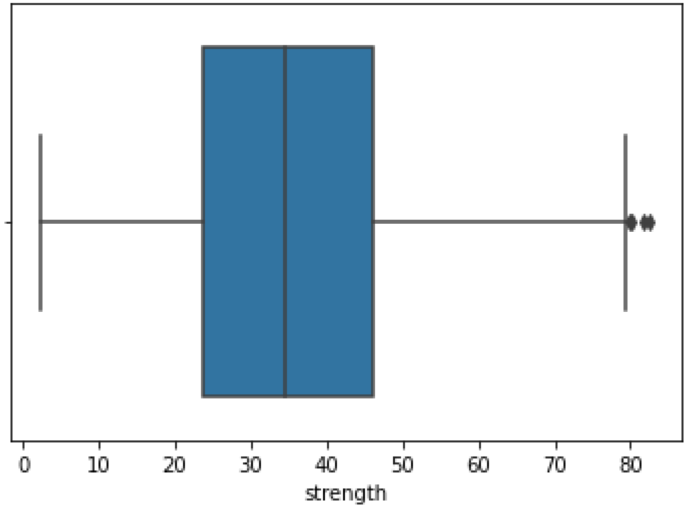
In [22]:
```
#correlation plot
plt.figure(figsize=[17,8])
sb.heatmap(df.corr(),annot=True)
```

Out[22]: <AxesSubplot:>



In [25]:
```python
l=['cement','slag','ash','water','superplastic','coarseagg','fineagg','age','strength']
for c in l:
    sb.boxplot(x=df[c])
    plt.show()
```





In [26]:
```python
#dividing independent and dependent variables
x=df.drop(['strength'],axis=1)
y=df['strength']
```

In [27]:
```python
#splitting data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

In [28]:
```python
#Feature scaling
from sklearn.preprocessing import StandardScaler
stand=StandardScaler()
Fit=stand.fit(x_train)
x_train_scl=Fit.transform(x_train)
xtest_scl=Fit.transform(x_test)
```

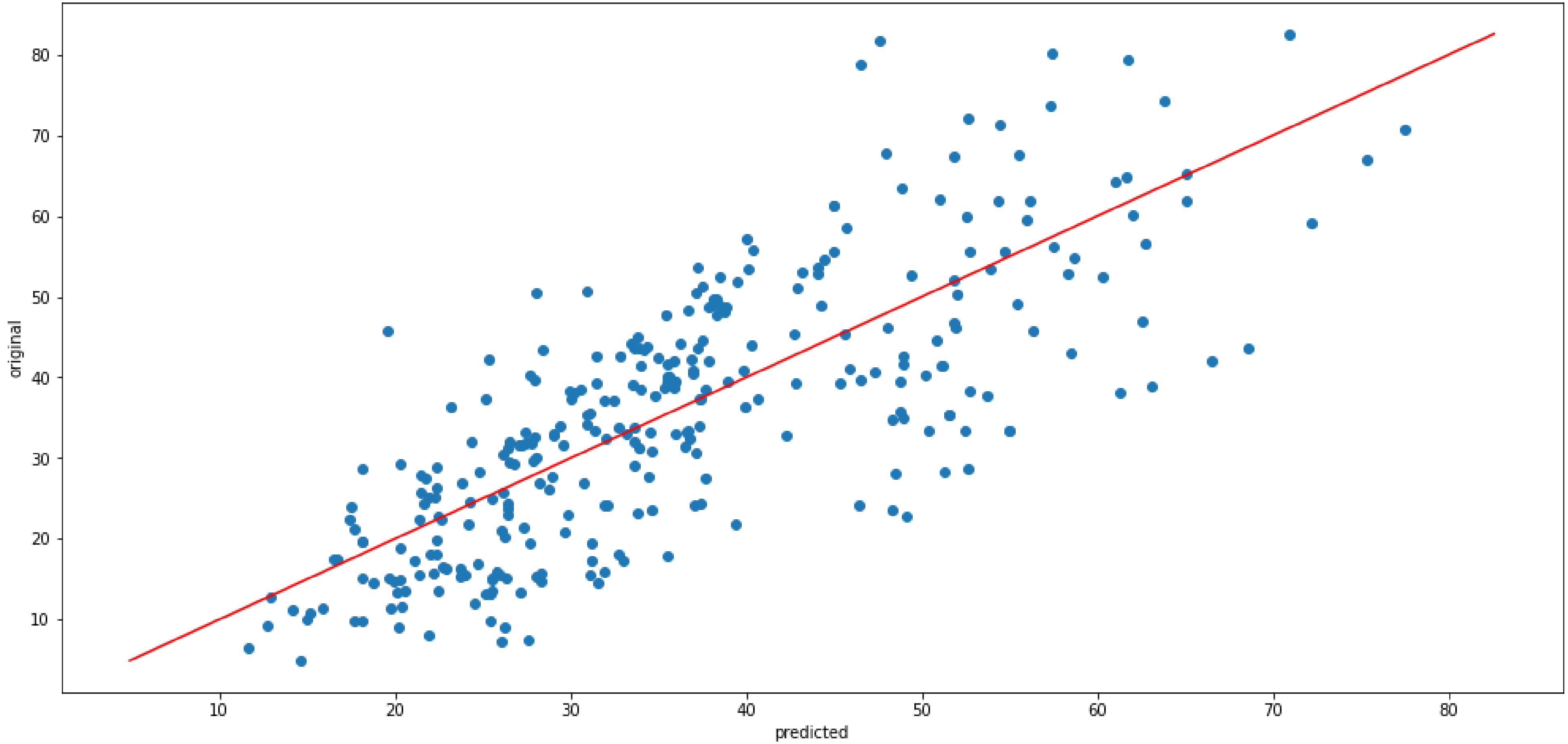
In [33]:
```
#linear regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
lr=LinearRegression()
fit=lr.fit(x_train_scl,y_train)
score=lr.score(xtest_scl,y_test)
print('predicted score is:{}'.format(score))
print()
y_predict=lr.predict(xtest_scl)
print('mean_sqrd_error is :',mean_squared_error(y_test,y_predict))
rms=np.sqrt(mean_squared_error(y_test,y_predict))
print('root mean squared error is: {}'.format(rms))
```

```
predicted score is:0.5845101408706099

mean_sqrd_error is : 110.77145201748968
root mean squared error is: 10.524801756683576
```

In [34]:

```
plt.figure(figsize=[17,8])
plt.scatter(y_predict,y_test)
plt.plot([y_test.min(),y_test.max()], [y_test.min(),y_test.max()],color='red')
plt.xlabel('predicted')
plt.ylabel('original')
plt.show()
```



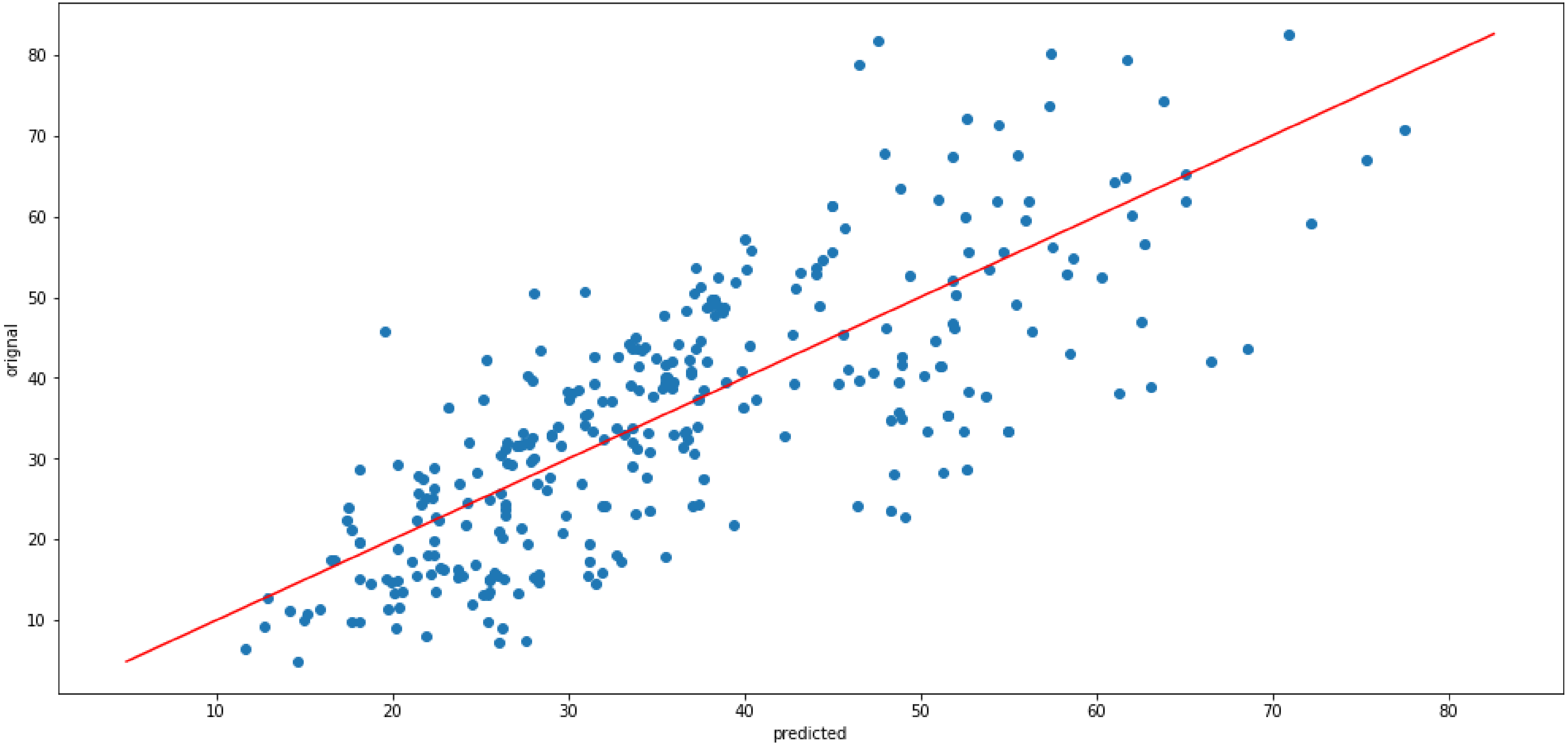
In [38]:

```
# lasso and rigid regression
from sklearn.linear_model import Ridge,Lasso
from sklearn.metrics import mean_squared_error
rd= Ridge(alpha=0.4)
ls= Lasso(alpha=0.3)
fit_rd=rd.fit(x_train_scl,y_train)
fit_ls = ls.fit(x_train_scl,y_train)
print('score od ridge regression is:-',rd.score(xtest_scl,y_test))
print()
print('score of lasso is:',ls.score(xtest_scl,y_test))
print('mean_sqrd_roor of ridig is:',mean_squared_error(y_test,rd.predict(xtest_scl)))
print('mean_sqrd_roor of lasso is:',mean_squared_error(y_test,ls.predict(xtest_scl)))
print('root_mean_squared error of ridge is:',np.sqrt(mean_squared_error(y_test,rd.predict(xtest_scl))))
print('root_mean_squared error of lasso is:',np.sqrt(mean_squared_error(y_test,lr.predict(xtest_scl))))
```

```
score od ridge regression is:- 0.5846262362109313

score of lasso is: 0.5826518196738566
mean_sqrd_roor of ridig is: 110.74050048127938
mean_sqrd_roor of lasso is: 111.2668887477882
root_mean_squared error of ridge is: 10.523331244490947
root_mean_squared error of lasso is: 10.524801756683576
```

In [37]:
```
plt.figure(figsize=[17,8])
plt.scatter(y_predict,y_test)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.xlabel('predicted')
plt.ylabel('orignal')
plt.show()
```



In [39]:
```
#random forest regression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
rnd= RandomForestRegressor(ccp_alpha=0.0)
fit_rnd= rnd.fit(x_train_scl,y_train)
print('score is:',rnd.score(xtest_scl,y_test))
print()
print('mean_sqrd_error is:',mean_squared_error(y_test,rnd.predict(xtest_scl)))
print('root_mean_squared error of is:',np.sqrt(mean_squared_error(y_test,rnd.predict(xtest_scl))))
```

```
score is: 0.9017473783777163

mean_sqrd_error is: 26.194587719735516
root_mean_squared error of is: 5.118064841298469
```

In [40]:

```
x_predict = list(rnd.predict(x_test))
predicted_df = {'predicted_values': x_predict, 'original_values': y_test}
#creating new dataframe
pd.DataFrame(predicted_df).head(20)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but RandomForestRegressor was f
itted without feature names
  warnings.warn(
```

Out[40]:

	predicted_values	original_values
31	46.39440	39.29
109	52.98054	38.63
136	55.26670	43.57
88	64.29460	35.30
918	55.26670	39.44
1025	52.98054	13.29
870	55.26670	10.09
318	52.98054	51.06
261	46.39440	17.24
535	52.98254	19.93
919	46.39440	38.11
596	64.26860	28.94
76	64.35790	24.13
107	46.39440	14.50
424	46.39440	27.74
584	64.29460	60.20
853	64.35790	33.31
664	55.26670	11.39
829	52.98054	22.32
420	64.29460	35.30