In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

In [2]:

```
train_df = pd.read_excel(r"C:\Users\user\Desktop\train.xls")
test_df = pd.read_excel(r"C:\Users\user\Desktop\test.xls")
```

In [3]:

```
print(test_df)
```

```
     Roll no test preparation   gender parental level of education  \
0   EXA32000              none    male          associate's degree
1   EXA32001         completed    male            some high school
2   EXA32002              none    male            some high school
3   EXA32003         completed    male            some high school
4   EXA32004              none  female           bachelor's degree
..       ...              ...     ...                         ...
95  EXA32095              none    male           bachelor's degree
96  EXA32096         completed    male          associate's degree
97  EXA32097              none    male                some college
98  EXA32098         completed    male          associate's degree
99  EXA32099              none    male                 high school

           lunch    Section  practical score  viva score
0       standard  Section C               74          89
1       standard  Section E               66          75
2       standard  Section C               52          55
3       standard  Section D               69          85
4       standard  Section E               46          62
..           ...        ...              ...         ...
95      standard  Section B               82          84
96  free/reduced  Section B               70          58
97      standard  Section C               76          67
98      standard  Section A               62          71
99      standard  Section B               58          67

[100 rows x 8 columns]
```

In [4]:

```
print(train_df)
```

```
         Roll no test preparation   gender parental level of education  \
0      EXA000001              none    male                some college   
1      EXA000002              none    male             master's degree   
2      EXA000003              none    male             master's degree   
3      EXA000004              none  female                some college   
4      EXA000005              none  female                 high school   
...          ...               ...     ...                         ...   
31994  EXA031995              none    male            some high school   
31995  EXA031996              none  female                 high school   
31996  EXA031997              none    male           bachelor's degree   
31997  EXA031998              none    male          associate's degree   
31998  EXA031999              none    male            some high school   

              lunch    Section  practical score  viva score  exam score  
0          standard  Section A               70          73          70  
1      free/reduced  Section C               55          54          52  
2      free/reduced  Section E               56          46          43  
3      free/reduced  Section C               35          47          41  
4          standard  Section C               87          92          81  
...             ...        ...              ...         ...         ...  
31994  free/reduced  Section E               63          53          80  
31995      standard  Section B              100          80          68  
31996  free/reduced  Section B               62          61          74  
31997      standard  Section D               75          32          82  
31998      standard  Section C               51          92          82  

[31999 rows x 9 columns]
```

In [5]:

```
# Separate features and target variable
X = train_df.drop(columns=['exam score'])
y = train_df['exam score']
```

In [6]:

```
# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [7]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import warnings

from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score

warnings.filterwarnings('ignore')
```

In [8]:

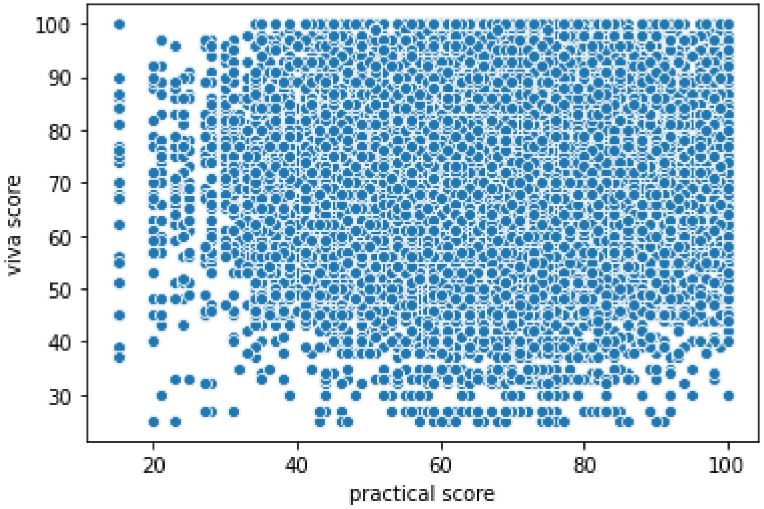
```
# Initialize and train model
model = RandomForestRegressor(random_state=42)
```

In [16]:

```
# plotting a scatterplot
sns.scatterplot(x='practical score',
                y='viva score', data= train_df)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xc57448>
```

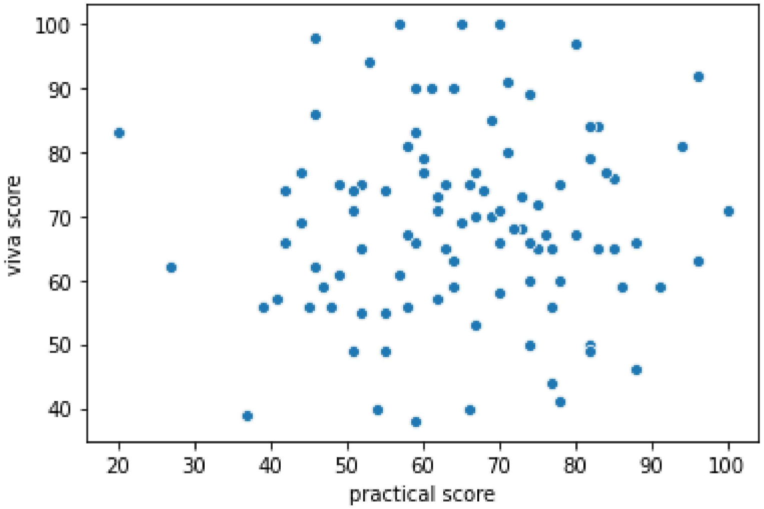


In [17]:

```
sns.scatterplot(x='practical score',
                y='viva score', data= test_df)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10e8dc0>
```

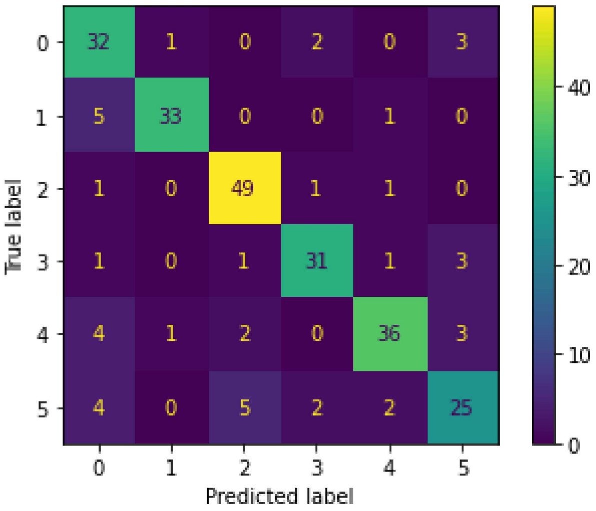


In [23]:

```
import numpy
import matplotlib.pyplot as plt
#Extracting Independent and dependent Variable
x= train_df.iloc[:, 1:2].values
y= train_df.iloc[:, 2].values
```

In [25]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC


# generate some sample data
X, y = make_classification(n_samples=1000,
n_features=10,
n_informative=6,
n_redundant = 2,
n_repeated = 2,
n_classes = 6,
n_clusters_per_class=1,
random_state = 42
)


# split the data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=0)


# initialize and train a classifier
clf = SVC(random_state=0)
clf.fit(X_train, y_train)


# get the model’s prediction for the test set
predictions = clf.predict(X_test)


# using the model’s prediction and the true value,
# create a confusion matrix
cm = confusion_matrix(y_test, predictions, labels=clf.classes_)


# use the built-in visualization function to generate a plot
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
disp.plot()
plt.show()
```



```
In [29]: train_df.shape
```

```
Out[29]: (31999, 9)
```

```
In [30]: test_df.shape
```

```
Out[30]: (100, 8)
```

```
In [31]: train_df.dropna().shape
```

```
Out[31]: (31999, 9)
```

In [33]:

```
test_df['viva score'].value_counts()
```

```
Out[33]: 65     6
         75     5
         66     5
         56     5
         71     4
         59     4
         74     4
         77     4
         67     3
         100    3
         90     3
         49     3
         60     2
         70     2
         62     2
         63     2
         57     2
         55     2
         50     2
         40     2
         68     2
         61     2
         69     2
         73     2
         79     2
         81     2
         83     2
         84     2
         89     2
         98     1
         92     1
         39     1
         97     1
         41     1
         44     1
         46     1
         94     1
         86     1
         53     1
         91     1
         72     1
         58     1
         85     1
```

```
80     1
76     1
38     1
Name: viva score, dtype: int64
```

In [34]:

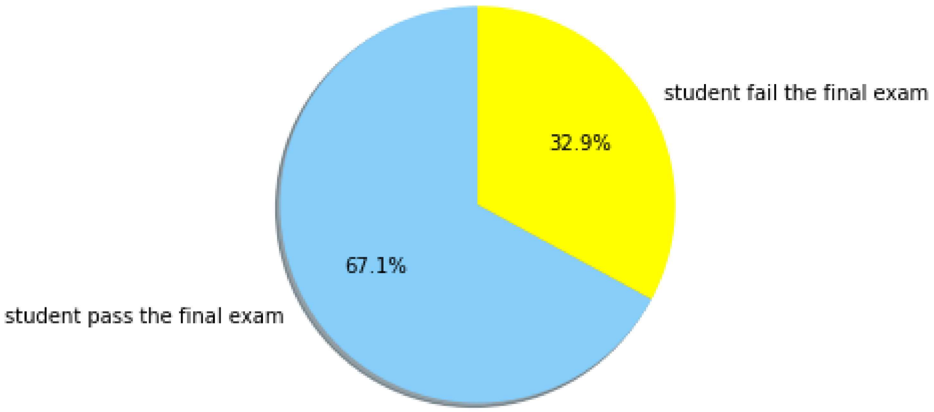
```
train_df['viva score'].value_counts()
```

Out[34]:

```
72    1164
77    1139
66    1061
68    1045
69     974
      ...
32      33
34      33
25      33
41      32
37      32
Name: viva score, Length: 71, dtype: int64
```

In [35]:

```
labels = 'student pass the final exam ', 'student fail the final exam'
sizes = [265, 130]
colors=['lightskyblue','yellow']
fig1, ax1 = plt.subplots()
ax1.pie(sizes,  labels=labels, autopct='%1.1f%%',colors=colors,
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



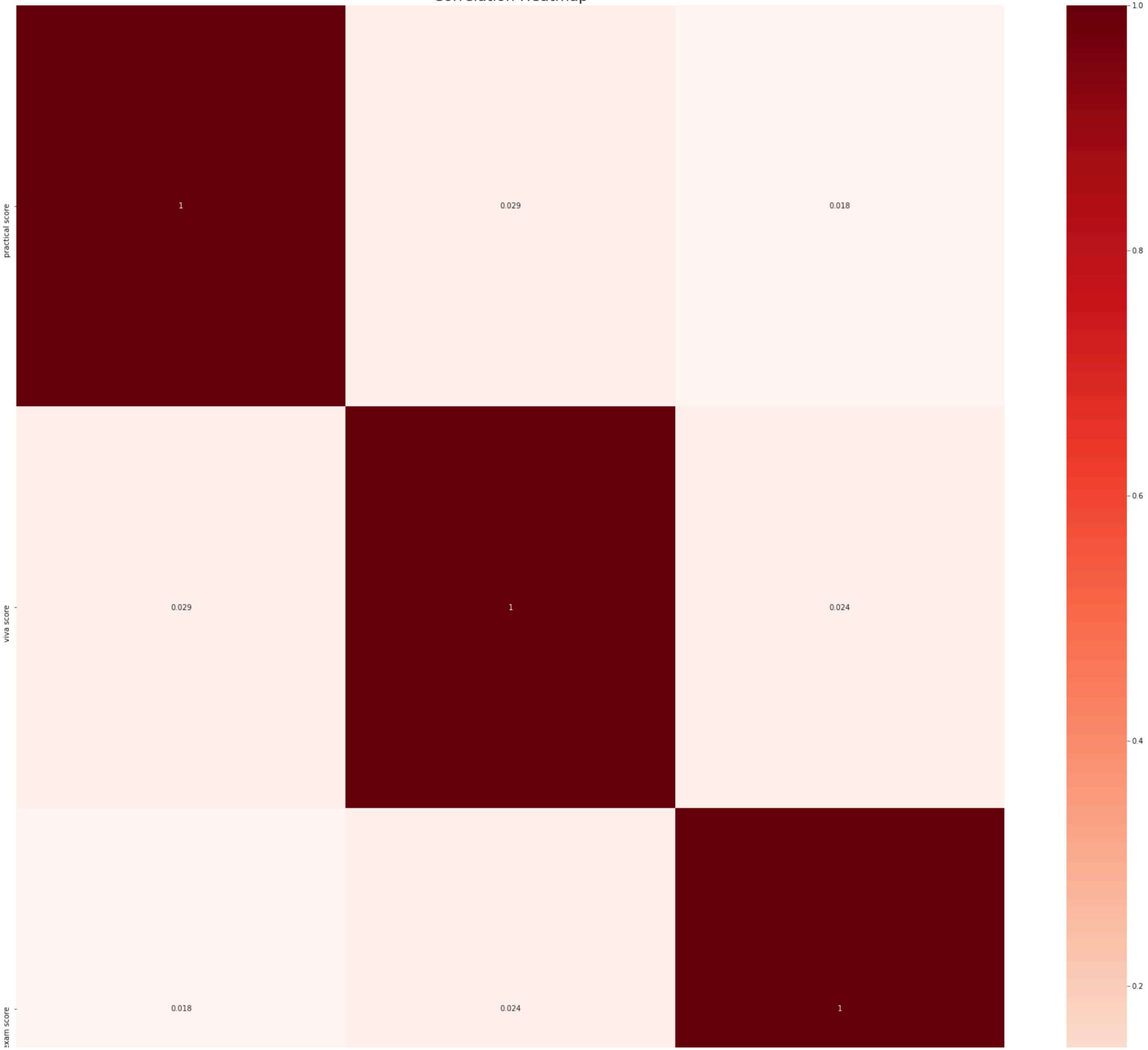
In [36]:

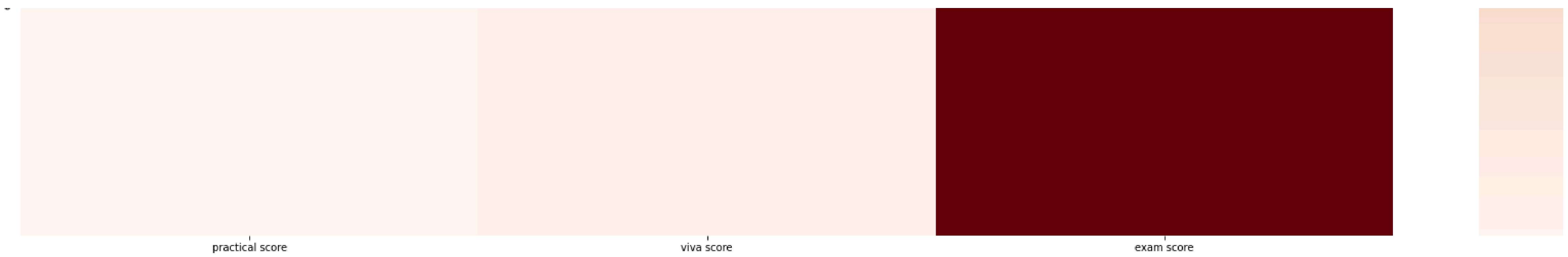
```
corr = train_df.corr()
plt.figure(figsize=(30,30))
sns.heatmap(corr, annot=True, cmap="Reds")
plt.title('Correlation Heatmap', fontsize=20)
```

Out[36]:

```
Text(0.5, 1.0, 'Correlation Heatmap')
```

Correlation Heatmap



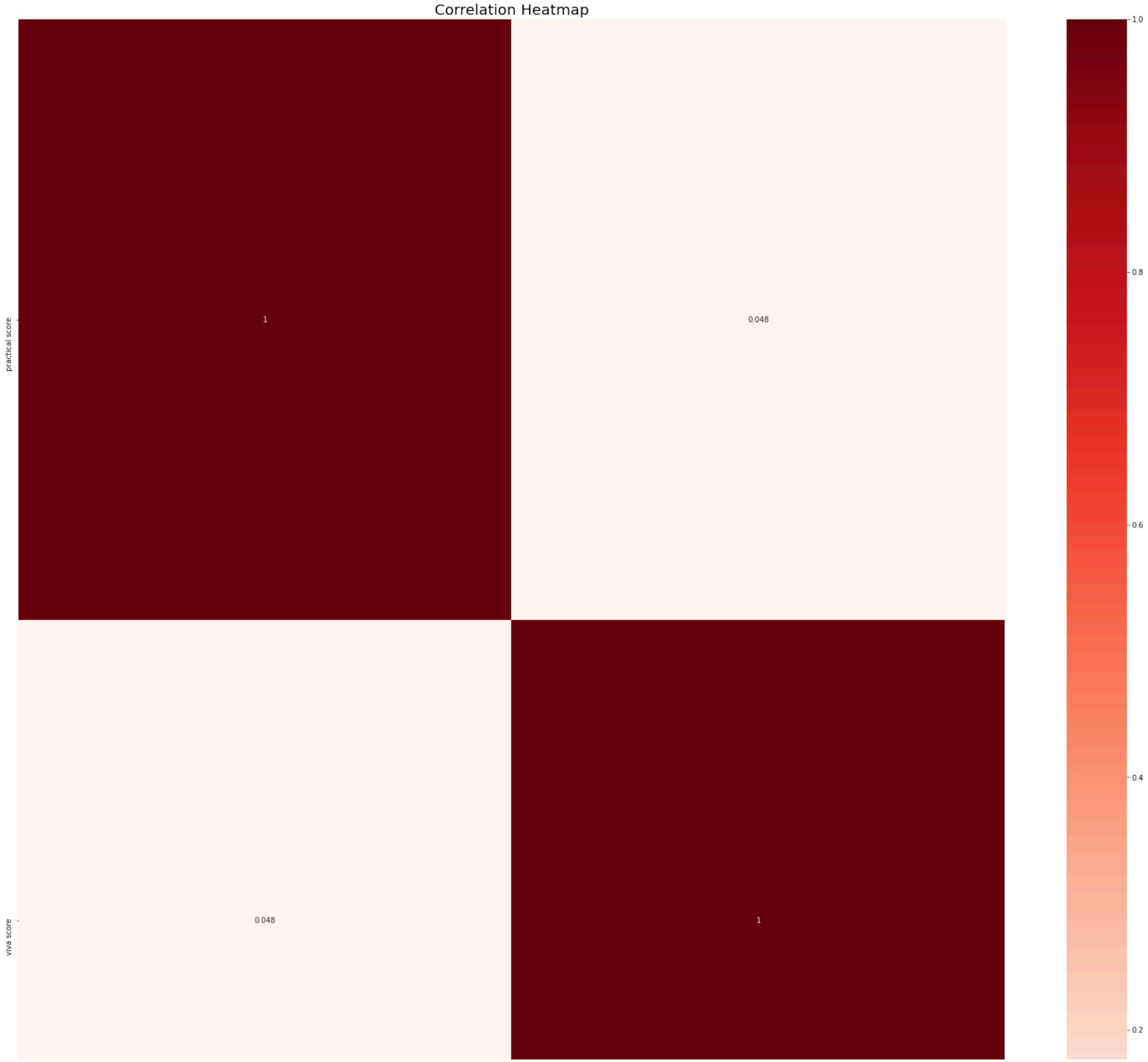

practical score
viva score
exam score

In [37]:

```
corr = test_df.corr()
plt.figure(figsize=(30,30))
sns.heatmap(corr, annot=True, cmap="Reds")
plt.title('Correlation Heatmap', fontsize=20)
```

Out[37]:

```
Text(0.5, 1.0, 'Correlation Heatmap')
```


Correlation Heatmap
practical score
viva score
1
0.048
0.048
1
1.0
0.8
0.6
0.4
0.2

In [39]:

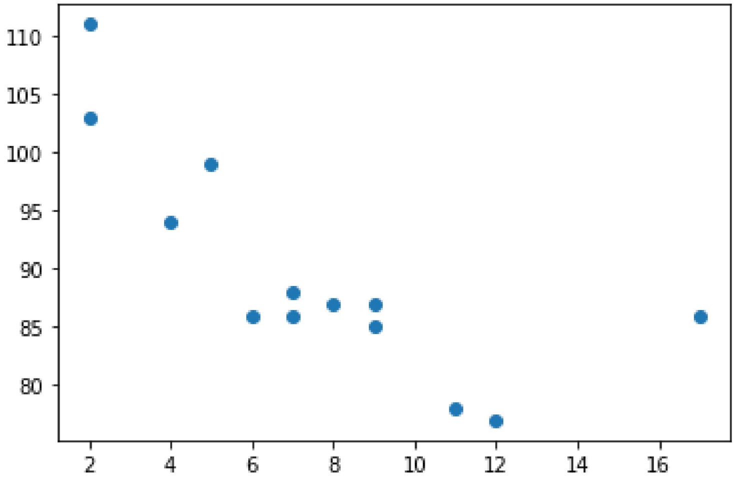
```
train_df["exam score"].unique()
```

Out[39]:

```
array([ 70,  52,  43,  41,  81,  85,  74,  62,  76,  71,  86,  88,  72,
        51,  59,  79,  75,  37,  82,  54,  87,  78,  48,  77,  67,  65,
        90,  68,  56,  80,  84,  63,  61,  93,  66,  73,  36,  57,  33,
        46,  89,  95,  42,  91,  60,  58,  38,  83,  97,  64,  53, 100,
        55,  47,  50,  69,  94,  44,  99,  92,  49,  15,  40,  98,  19,
        96,  35,  32,  26,  28,  45,  27,  30,  23], dtype=int64)
```

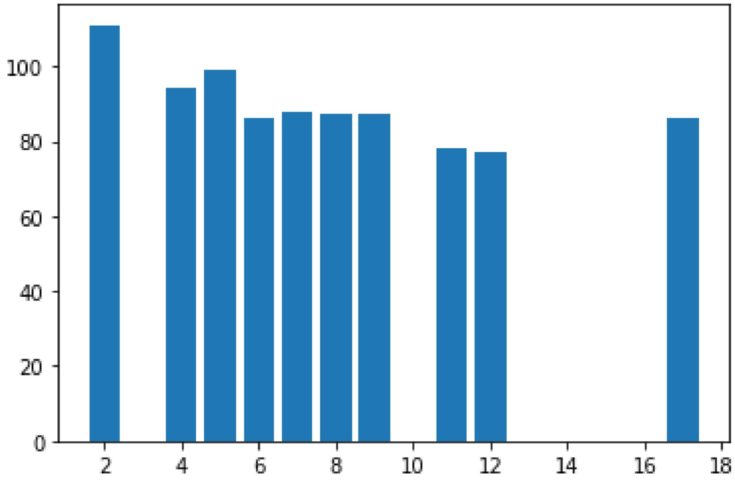
In [49]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



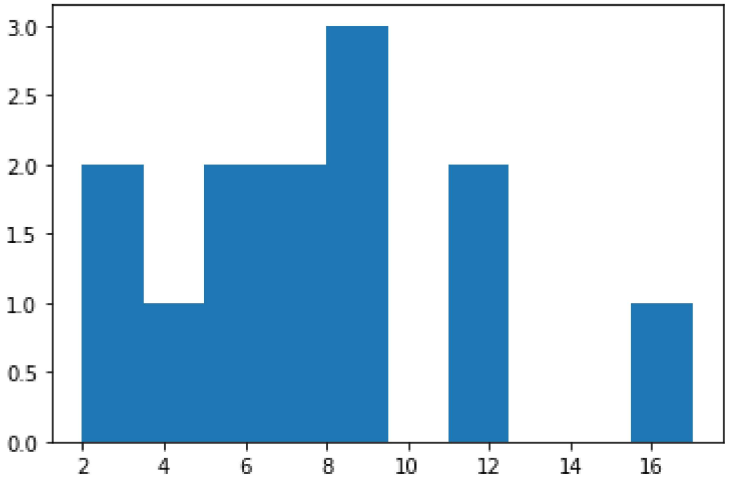
In [50]:

```
plt.bar(x,y)
plt.show()
```



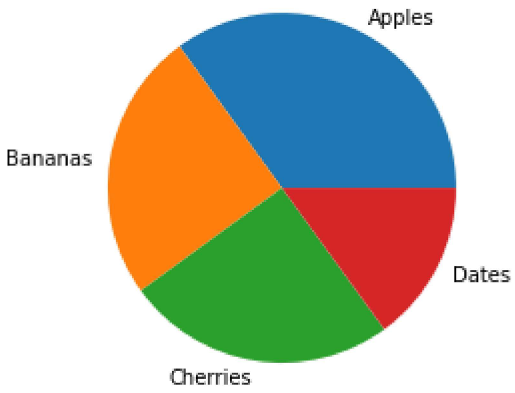
In [51]:

```
plt.hist(x)
plt.show()
```



In [53]:

```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```


Apples
Bananas
Dates
Cherries