```
#1. a.
def age_avg(n):
    age=[]
    for i in range(n):
        age.append(int(input(f"Enter age of {i+1}\n")))
    _sum=0
    for i in age:
        _sum=_sum+i
    avg=_sum/n
    print(avg,"years")


#1. b.
def dictionary():
    fruits=["Apple","Mango","Peach","Banana"]
    prices=[100,80,150,70]
    d={'a':5,'b':10}
    d=dict(zip(fruits,prices))
    print(d)

if __name__=='__main__':
    print("What you want to do? \n1. age avg(1)\n2. dictionary fruits(2)")
    n=int(input(""))
    if (n==1):
        age_avg(10)
    elif(n==2):
        dictionary()
    else:
        print("Invalid Input")
```

```
import numpy as np
import time
import sys
def list_array():
    arr1=np.ones(10000)
    arr2=2*np.ones(10000)
    l1=list(arr1)
    l2=list(arr2)
    print("numpy array size : ",arr1.nbytes+arr2.nbytes ,"bytes")
    print("list size : ",sys.getsizeof(l1)+sys.getsizeof(l2),"bytes")

    #numpy array time
    start=time.time()
    arr3=arr1+arr2
    numpy_time=time.time()-start

    #list time
    start=time.time()
    l3=[]
    for i in range(len(l1)):
        l3.append(l1[i]+l2[i])
    python_time=time.time()-start

    print("Time for NumPy array",numpy_time,"Seconds")
    print("Time for Python list",python_time,"Seconds")

if __name__=="__main__":
    list_array()
```

```
import numpy as np
#3. a.
def FiveX2():
    ar=np.arange(50,100,5)
    ar=ar.reshape(5,2)
    ar1=ar.reshape(10,1)
    print("Reshaped array\n",ar)
    print("Original array\n",ar1)
#b.
def Randomarray():
    ar=np.random.randint(1,101,size=10)

    min_val=np.min(ar)
    max_val=np.max(ar)
    mean_val=np.mean(ar)
    median_val=np.median(ar)
    std_dev=np.std(ar)
    unique_val=np.unique(ar)
    count_unique_val=len(unique_val)
    print(ar)
    print("\nmin value :-",min_val)
    print("max value :-",max_val)
    print("mean value :-",mean_val)
    print("median value :-",median_val)
    print("std dev value :-",std_dev)
    print("unique value :-",unique_val)
    print("count of unique value :-",count_unique_val)
#c.
def sqroot():
    ar1=np.random.randint(1,101,size=(3,3))
    ar2=np.random.randint(1,101,size=(3,3))
    ar3=np.sqrt(ar1+ar2)
    print(ar1)
    print(ar2)
    print("\nArray addtion and sqroot\n",ar3)
#d.
def oprt():
    ar=np.array([[34,43,73],[82,22,12],[53,94,66]])

    ar_del_col=np.delete(ar,1,axis=1)
    ar_sor_row=ar[:,ar[1].argsort()]
    ar_sor_col=ar[ar[:,1].argsort()] # sort by 2nd column
    rowmax=np.max(ar,axis=1)
    colmax=np.max(ar,axis=0)
    ar_swap=ar.copy()
    ar_swap[:,[0,1]]=ar_swap[:,[1,0]]
    even=ar[ar%2==0]
    #e.
    idmat=np.eye(5)*5
```

```
    print(ar)
    print("Array after 2nd col delete\n",ar_del_col)
    print("Array sorted by 2nd row\n",ar_sor_row)
    print("Array sorted by 2nd col\n",ar_sor_col)
    print("row wise max elem\n",rowmax)
    print("col wise max elem\n",colmax)
    print("Array with first two columns swapped\n",ar_swap)
    print("Even numbers from array\n",even)
    print("Identity matrix with diagonal element at 5:\n",idmat)


if __name__=='__main__':
    print("What you want to do? \n1. FiveX2(1)\n2. Random Array(2)\n3. Square root(3)\n4. Operations and identity matrix(4)")
    n=int(input(""))
    if (n==1):
        FiveX2()
    elif(n==2):
        Randomarray()
    elif(n==3):
        sqroot()
    elif(n==4):
        oprt()
    else:
        print("Invalid Input")
```

```
#4
import pandas as pd
import numpy as np
def panda_op():

    #a.
    d={'XII':25,'XI':30,'X ':50}
    series=pd.Series(d)
    print("Sorted series on Index:\n",series.sort_index())
    print("Sorted series on Values:\n",series.sort_values())

    #b.
    rv=np.random.rand(10)   #random values
    rs=pd.Series(rv)        #random series
    print("Maximum Value:",rs.max(),"at position",rs.idxmax())
    print("Minimum value:",rs.min(),"at position",rs.idxmin())

    #c.
    t={'Monday':25,'Tuesday':40,'Wednesday':18,'Thusrday':27,'Friday':32,'Saturday':39,'Sunday':28}   #temperature
    ts=pd.Series(t)
    t4d=ts.iloc[3]
    tt=ts.loc['Tuesday'] #temperature tuesday
    print("Temperature of the 4th day of the week:",t4d)
    print("Temperature of Tuesday:",tt)

    #d.
    a={'Name':['Aroma','Kiran','Riyan','Rohan','Amit','Yash','Mona','Kartik','Kavita','Pooja'],'Percentage':[79.5,29.0,90.5,np.nan,32.0,65.0,56.0,np.nan,29.0
    b={'Name':['Parveen','Ahil','Shaila','Shruti','Mark'],'Percentage':[89.5,92.5,90.5,91.5,90.0],'Qualify':['yes','yes','yes','yes','yes']}
    dfa=pd.DataFrame(a)
    dfb=pd.DataFrame(b)
    print("\tFirst Data Frame\n",dfa)
    print("\tSecond Data Frame\n",dfb)
    #e.
    merged=pd.concat([dfa,dfb],ignore_index=True)
    print("Merged Data Frame First & Second \n",merged)
    #f.
    filtered=merged[merged['Percentage']>=80]
    missing=merged.isnull().sum().sum()
    print("\nFiltered dataframe with 80 percentage and above count missing data:")
    print(filtered)
    print("Count of missing data:",missing)


if __name__ == '__main__':
    panda_op()
```

```
#5
import seaborn as sns


penguins_df = sns.load_dataset('penguins')

num_observations = len(penguins_df)
num_attributes = len(penguins_df.columns)

print("Number of Observations/Records:", num_observations)
print("Number of Attributes:", num_attributes)

print("\nAttributes and Data Types:")
print(penguins_df.dtypes)

print("\nFirst 5 records:")
print(penguins_df.head())

print("\nLast 5 records:")
print(penguins_df.tail())

values = penguins_df.iloc[2:4, 1]
print("\nValue of the second column for the third and fourth records:")
print(values)
```

```
#6
import pandas as pd
import numpy as np

np.random.seed(0)
data = np.random.rand(30, 6)
columns = ['A', 'B', 'C', 'D', 'E', 'F']
df = pd.DataFrame(data, columns=columns)

null_indices = np.random.choice(df.index, size=20, replace=False)
for index in null_indices:
    column = np.random.choice(df.columns)
    df.loc[index, column] = np.nan

missing_values_per_column = df.isnull().sum()
print("Number of missing values in each attribute:")
print(missing_values_per_column)

threshold = 0.7 * len(df.columns)
df = df.dropna(thresh=threshold)

df_normalized = (df - df.min()) / (df.max() - df.min())
print("\nNormalized dataframe:")
```

```
print(df_normalized)
```

```
#7
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
import pandas as pd

iris_data = load_iris()
iris_df = pd.DataFrame(data=iris_data.data, columns=iris_data.feature_names)
iris_df['target'] = iris_data.target
iris_df['species'] = iris_df['target'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

plt.figure(figsize=(8, 6))
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='species', data=iris_df)
plt.title('Relationship between Petal Length and Petal Width')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.show()

plt.figure(figsize=(10, 8))
for i, feature in enumerate(iris_data.feature_names):
    plt.subplot(2, 2, i + 1)
    sns.histplot(data=iris_df, x=feature, kde=True)
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()

plt.figure(figsize=(6, 6))
iris_df['species'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Frequency Count of Flower Types')
plt.ylabel('')
plt.show()

sns.pairplot(iris_df, hue='species')
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()
```

```
#8
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

mpg_df = sns.load_dataset('mpg')
```

```
X = mpg_df[['weight', 'cylinders', 'displacement']]
y = mpg_df['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

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

print("Mean Squared Error:", mse)
print("R-squared Score:", r2)
```

```
#9
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

penguins_df = sns.load_dataset('penguins')

penguins_df.dropna(inplace=True)

species_mapping = {'Adelie': 0, 'Gentoo': 1, 'Chinstrap': 2}
penguins_df['species'] = penguins_df['species'].map(species_mapping)
```