01

January 18, 2022

[1]: #Python program to perform linear search

def linearSearch(array, n, x):

# Going through array sequencially

for i in range(0, n):

if (array[i] == x):

return i

return -1

array = [2, 4, 0, 1, 9]

x = eval(input("enter the element to be searched: "))

n = len(array)

result = linearSearch(array, n, x)

if(result == -1):

print("Element not found")

else:

print("Element found at index: ", result)

02

import bisect

def insert(list, n):

bisect.insort(list, n)

return list

list = [1, 2, 4]

n = eval(input("enter the value to be inserted "))

print(insert(list, n))

03

class Base:

def \_\_init\_\_(self):

self.a = 10

self.\_b = 20

def display(self):

print(" the values are :")

print(f"a={self.a} b={self.\_b}")

class Derived(Base): # Creating a derived class

def \_\_init\_\_(self):

Base.\_\_init\_\_(self) # Calling constructor of Base class

self.d = 30

def display(self):

Base.display(self)

print(f"d={self.d}")

def \_\_add\_\_(self, ob):

return self.a + ob.a+self.d + ob.d

#return self.a + ob.a+self.d + ob.d+self.b + ob.b

obj1 = Base()

obj2 = Derived()

obj3 = Derived()

obj2.display()

obj3.display()

print("\n Sum of two objects :",obj2 + obj3)

04

import pandas as

pd df=pd.read\_csv("C:\\Users\\Apple\\Desktop\\toyota.csv")

import pandas as pd

df=pd.read\_csv("C:\\Users\\Apple\\Desktop\\toyota.csv",skiprows=1)

import pandas as pd

df=pd.read\_csv("C:\\Users\\Apple\\Desktop\\toyota. csv",skiprows=1,names=['a','b','c','d','e','f','g','h'])

import pandas as pd

df=pd.read\_csv("C:\\Users\\Apple\\Desktop\\toyota.csv",nrows=5)

import pandas as pd

df=pd.read\_excel("C:\\Users\\Apple\\Desktop\\Toyota1.xls")

: import pandas as pd

import numpy as np

df=pd.read\_csv("C:\\Users\\Apple\\Desktop\\Toyota.csv")

new\_df=df.replace(-77777,np.NaN)

import pandas as pd df=pd.DataFrame(weather\_data,columns=["day","temperature","windspped","event"])

df

new\_df=df.replace(-99,np.NaN)

new\_df

new\_df=df.replace([-99999,-88888],np.NaN)

new\_df

new\_df=df.replace( { 'temperature':-99999,

'windspeed':-99999,

'event':'0'

},np.NaN)

new\_df

import pandas as pd

df=pd.DataFrame(weather\_data)

new\_df=df.replace(

{ 'temperature':-99999, '

windspeed':-99999,

'event':'0'

},np.NaN)

new\_df=df.replace( { -99999:np.NaN, 'No Event':'Sunny' })

new\_df=df.replace({ 'temperature':'[A-Za-z]', 'windspeed':'[A-Za-z]'},'',regex=True)

df=pd.DataFrame({ 'Score':['exceptional','average','good','poor','average','exceptional'], 'student':['rob','maya','parthiv','tom','julian','erica'] }) df.replace(['poor','average','good','exceptional'],[1,2,3,4])

df.replace(['poor','average','good','exceptional'],[1,2,3,4])

df['Price'].max()

df['Price'].max()

df.describe()

05

import numpy as np

arr1=([[1,2,3],[4,5,6]])

arr2=([[7,8,9],[9,10,11]])

print("concatenating of two arrays \n ",np.concatenate([arr1,arr2],axis=1))

print("Vertical stacking \n",np.vstack((arr1,arr2)))

print("Horizontal stacking \n",np.hstack((arr1,arr2)))

arr =np.array([1,2,3,4,5,4,4])

x=np.where(arr==4)

print(x)

arr =np.array([6,7,8,9])

x=np.searchsorted(arr,5)

print(x)

arr =np.array([1,3,5,7])

x=np.searchsorted(arr,[2,4,6])

print(x)

a =np.array([[1,4],[3,1]])

print("sorted array : ",np.sort(a))

print("\n Flattened sorted array is ",np.sort(a,axis=0))

x =np.array([3,1,2])

print("\n indices that would sort an array ",np.argsort(x))

print("\n sorted complex number", np.sort\_complex([5,3,6,2,1]))

import numpy as np

x=np.arange(9,0)

print(np.split(x,3))

print(np.split(x,[3,5,6,10]))

x=np.arange(9)

np.array\_split(x,4)

a=np.array([[1,3,5,7,9,11],[2,4,6,8,10,12]])

print("splitting along horizontal axis into 2 parts\n", np.hsplit(a,2))

print("\n splitting along Vertical axis into 2 parts\n", np.vsplit(a,2))

import numpy as np

v=np.array([1,2,3])

w=np.array([4,5])

print("v = ", v)

print("w = ", w)

print("\n outer product of v & w is \n")

print(np.reshape(v,(3,1)) \* w)

x=np.array([[1,2,3],[4,5,6]])

print("x = ", x)

print("v = ", v)

print("\n x + v = ", x+v)

print("\n transposing this final result")

print((x.T+w).T)

print("\n x+np.reshape(w, (2,1))")

print(x+np.reshape(w,(2,1)))

print(x\*2)\

import numpy as np

import matplotlib.pyplot as plt

x=np.arange(0,3 \* np.pi, 0.1)

y\_sin= np.sin(x)

y\_cos= np.cos(x)

plt.plot(x,y\_sin)

plt.plot(x,y\_cos)

plt.xlabel('x\_ axis label')

plt.ylabel('y\_axis label')

plt.title('sine and cosine')

plt.legend(['sine' , 'cosine'])

plt.show()

6

import matplotlib.pyplot as plt

###Line plot

x1=[1,2,3,4,5]

y1=[2,5,2,6,8]

x2=[1,2,3,4,5]

y2=[4,5,8,9,10]

plt.xlabel("X Axis ",fontsize=12,fontstyle='italic')

plt.ylabel("Y Axis ",fontsize=12)

plt.title("LINE PLOT ",fontsize=15,fontname='DejaVu Sans')

plt.plot(x1,y1,color='red',label='First graph')

plt.plot(x2,y2,color='green',label='Second graph')

plt.legend(loc=2)

plt.grid()

#plt.axis('off')

plt.show()

x=['A','B','C','D','E']

y=[20,40,20,60,80]

plt.xlabel("X Axis ",fontsize=12)

plt.ylabel("Y Axis ",fontsize=12)

plt.title("BAR PLOT ", fontsize=15)

plt.bar(x1,y1,color='red',width=0.5)

plt.show()

x1=[1,2,3,4,5]

y1=[2,5,2,6,8]

x2=[1,2,3,4,5]

y2=[4,5,8,9,10]

plt.xlabel("X Axis ",fontsize=12,fontstyle='italic')

plt.ylabel("Y Axis ",fontsize=12)

plt.title("LINE PLOT ",fontsize=15,fontname='DejaVu Sans')

plt.scatter(x1,y1,color='red',label='First graph')

plt.scatter(x2,y2,color='green',s=150,marker="\*",label='Second graph')

plt.plot(x2,y2,color='green')

plt.legend(loc=3)

#plt.axis('off')

plt.show()

import numpy as np

sample=np.random.randint(10,70,100)

plt.title("HISTOGRAM ",fontsize=15,fontname='DejaVu Sans')

plt.hist(sample,rwidth=0.7)

plt.show()

plt.figure(figsize=(7,7))

slices=[10,30,20,50,70]

act=["A","B","C","D","E"]

cols=["pink","red","green","blue","yellow"]

plt.title("PIE Chart ",fontsize=15,fontname='DejaVu Sans')

plt.pie(slices,labels=act,colors=cols,autopct="%1.1f%%",explode=(0,0,0,0,0.1))

plt.show()

07

import numpy as np

a=np.array([[1,2,3],[4,5,6],[7,8,9]])

print("Printing array")

print()

print(a)

print("Printing numpy array attributes")

print("1)Array dimension: ", a.ndim)

print("2) Array shape:", a.shape)

print("3) Array size: ", a.size)

print("4) Array data type :", a.dtype)

print("5) The length of each array item in bytes is : ",a.itemsize)

08

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x, y):

# number of observations/points

n = np.size(x)

# mean of x and y vector

m\_x = np.mean(x)

m\_y = np.mean(y)

# calculating cross-deviation and deviation about x

SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x

SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x

# calculating regression coefficients

b\_1 = SS\_xy / SS\_xx

b\_0 = m\_y - b\_1\*m\_x

return (b\_0, b\_1)

def plot\_regression\_line(x, y, b):

# plotting the actual points as scatter plot

plt.scatter(x, y, color = "m",

marker = "o", s = 30)

# predicted response vector

y\_pred = b[0] + b[1]\*x

# plotting the regression line

plt.plot(x, y\_pred, color = "g")

# putting labels

plt.xlabel('x')

plt.ylabel('y')

# function to show plot

plt.show()

def main():

# observations / data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

# estimating coefficients

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}".format(b[0], b[1]))

# plotting regression line

plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

main()

09

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset = pd.read\_csv('C://Users//Apple//Desktop//User\_Data.csv')

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size = 0.25,␣

↪random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(xtrain)

xtest = sc\_x.transform(xtest)

print (xtrain[0:10, :])

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(xtrain, ytrain)

y\_pred = classifier.predict(xtest)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(ytest, y\_pred)

print ("Confusion Matrix : \n", cm)

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(ytest, y\_pred))

from matplotlib.colors import ListedColormap

X\_set, y\_set = xtest, ytest

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,

stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1,

stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(

np.array([X1.ravel(), X2.ravel()]).T).reshape(

X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Classifier (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

10

1

import pandas as pd

df = pd.read\_csv("aapl.csv")

df.head()

df = pd.read\_csv("aapl.csv",parse\_dates=["Date"], index\_col="Date")

df.tail()

df.index

2

df.loc['2017-06-30']

df.loc["2017-01"]

df.loc['2017-06'].head()

3

df.loc['2017-06'].Close.mean()

df.loc['2017'].head(2)

df.loc['2017-01-08':'2017-01-03']

df.loc['2017-01']

df['Close'].resample('M').mean().head()

df.loc['2016-07']

%matplotlib inline

df['Close'].plot()

11

: import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

cars\_data=pd.read\_csv('Toyota.csv', index\_col=0, na\_values=["??,????"])

cars\_data.dropna(axis=0,inplace=True)

sns.set(style="darkgrid")

sns.regplot(x=cars\_data['Age'], y=cars\_data['Price'])

sns.regplot(x=cars\_data['Age'],y=cars\_data['Price'], marker='\*', fit\_reg=False)

sns.lmplot(x='Age',y='Price',data=cars\_data, fit\_reg=False,hue="FuelType",legend=True, palette='Set1')

sns.histplot(cars\_data['Age'])

sns.histplot(cars\_data['Age'],kde=False,bins=8)

sns.countplot(x="FuelType",data=cars\_data)

sns.countplot(x="FuelType",data=cars\_data, hue='Automatic')

sns.boxplot(x=cars\_data["FuelType"],y=cars\_data["Price"])

sns.boxplot(x=cars\_data["FuelType"],y=cars\_data["Price"],hue="Automatic",data=cars\_data)