CA#1_[Soroush Mesforush Mashhad]_[810198472]

0.1 EPS CA1

0.1.1 Soroush Mesforush Mashhad

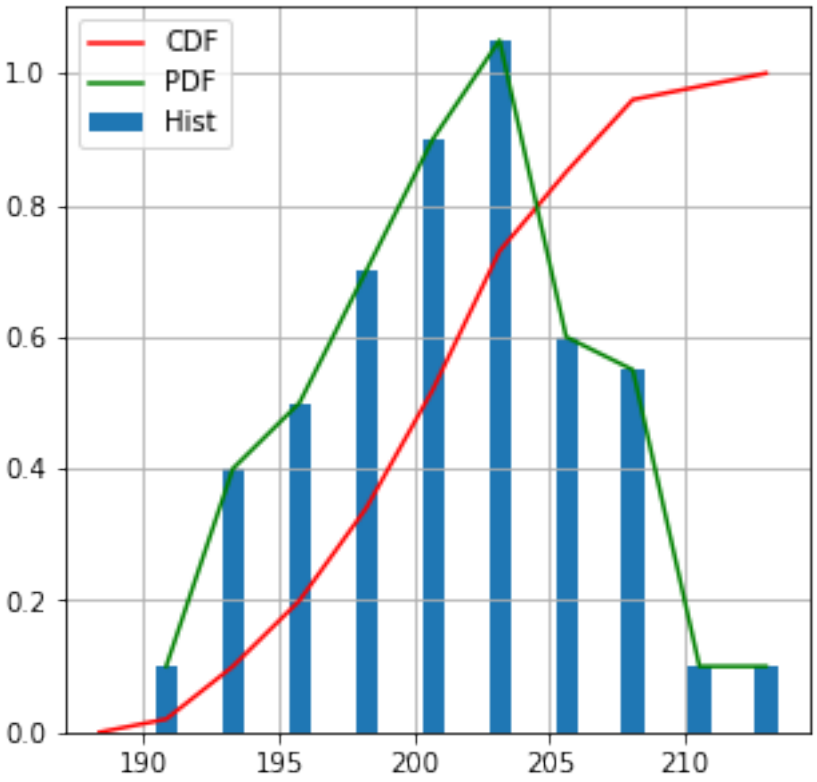
STD NO:810198472

0.2 Section 1

0.3 Q1

Here We Use np.histogram to compute it and by using the definition of the cumilative distribution function(CDF) we shall plot it with the help of the histogram.

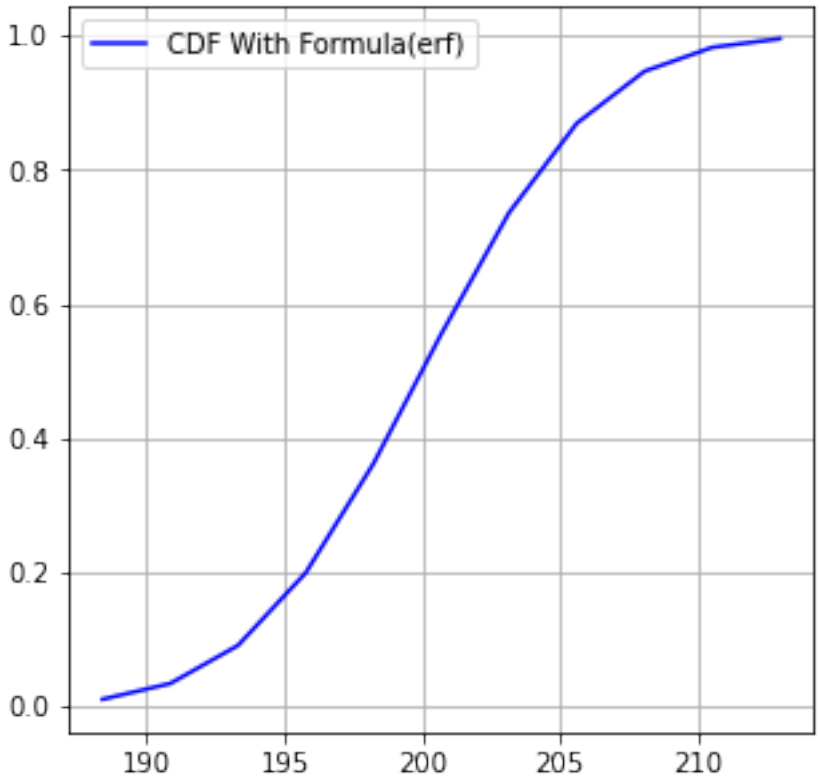
[1]:
```
import numpy as np
import math as m
import matplotlib.pyplot as plt
from IPython.display import Image
#Here I have defined the needed variables
Miu=200
Sigma=5
#We create a random dataset.
np.random.seed(12345679)
RandDset = np.random.normal(Miu,Sigma,100)
hist,bins=np.histogram(RandDset)
plt.figure(figsize=(5,5))
ax = plt.subplot(111)
CDF_Array=[None]*len(bins)
i=0
#In this while loop we create the cdf by adding the his elements.
while i<len(bins):
    CDF_Array[i]=(sum(hist[0:i]))/sum(hist)
    i=i+1
# Here we plot the PDF,CDF and Histogram accordingly
ax.plot(bins,CDF_Array,'r',label='CDF')
ax.bar(bins[1:],hist/20,label='Hist')
ax.plot(bins[1:],hist/20,'g',label='PDF')
ax.grid(True)
ax.legend()
plt.show()
```



0.4 Q2 With erf

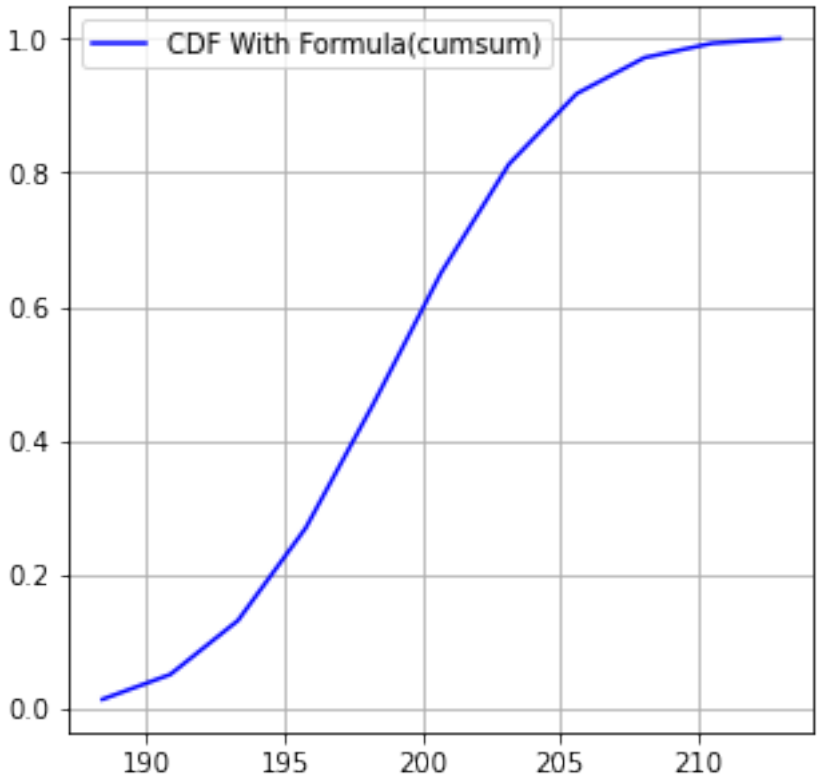
Here We Shall draw the CDF Using the Formula and not by using the histogram.

[2]:
```
CDF_F = [None]*len(bins)
plt.figure(figsize=(5,5))
#In this loop we create the CDF_F using erf.
for k in range(0,len(bins)):
    CDF_F[k] = (1+m.erf((bins[k]-Miu)/(np.sqrt(2)*Sigma)))/2
ax = plt.subplot(111)
#Here we plot it accordingly.
ax.plot(bins,CDF_F,'b',label='CDF With Formula(erf)')
ax.grid(True)
ax.legend()
plt.show()
```



0.5 Q2 with cumsum

[3]:
```
CDF_Fc = [None]*len(bins)
PDF_F = [None]*len(bins)
plt.figure(figsize=(5,5))
#Here we create CDF_F using the PDF integral by using cumsum.
PDF_F=(1/(Sigma*np.sqrt(2*np.pi)))*np.exp(-1*((bins-Miu)**2)/(2*Sigma**2))
#for i in range(0,len(bins)):
 #   CDF_Fc[i]=sum(hist[0:i])/sum(hist)
CDF_Fc=np.cumsum(PDF_F)/PDF_F.sum()
ax = plt.subplot(111)
#Here we plot it accordingly.
ax.plot(bins,CDF_Fc,'b',label='CDF With Formula(cumsum)')
ax.grid(True)
ax.legend()
plt.show()
```



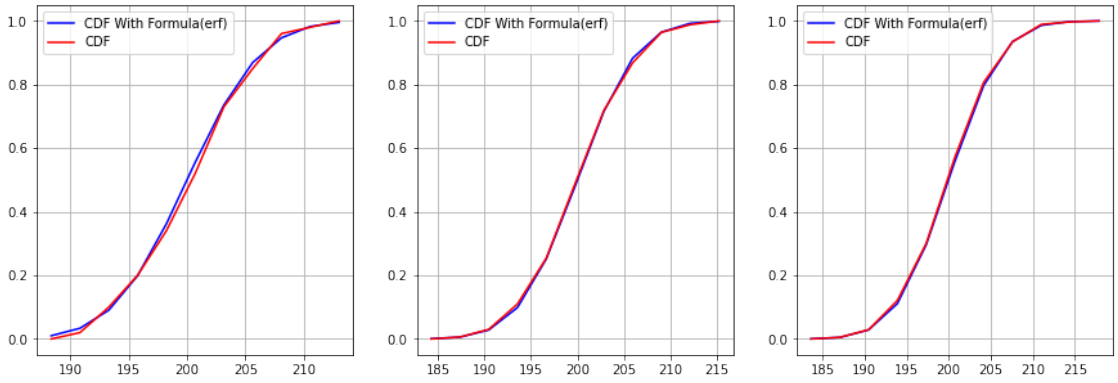
0.6 Q3

Here we shall draw the CDF's from Q1 and Q2 for different N values in three subplots respectively.

0.7 With erf

[4]:
```
#This part is exactly like Q1 and Q2 combined,thus I have refrained from adding␣
↪any extra comments.
Miu=200
Sigma=5
np.random.seed(12345679)
RandDset1 = np.random.normal(Miu,Sigma,100)
hist1,bins1=np.histogram(RandDset1)
RandDset2 = np.random.normal(Miu,Sigma,500)
hist2,bins2=np.histogram(RandDset2)
RandDset3 = np.random.normal(Miu,Sigma,1000)
hist3,bins3=np.histogram(RandDset3)
plt.figure(figsize=(15,5))
ax = plt.subplot(131)
CDF_Array1=[None]*len(bins1)
CDF_F1 = [None]*len(bins1)
for k in range(0,len(bins1)):
```

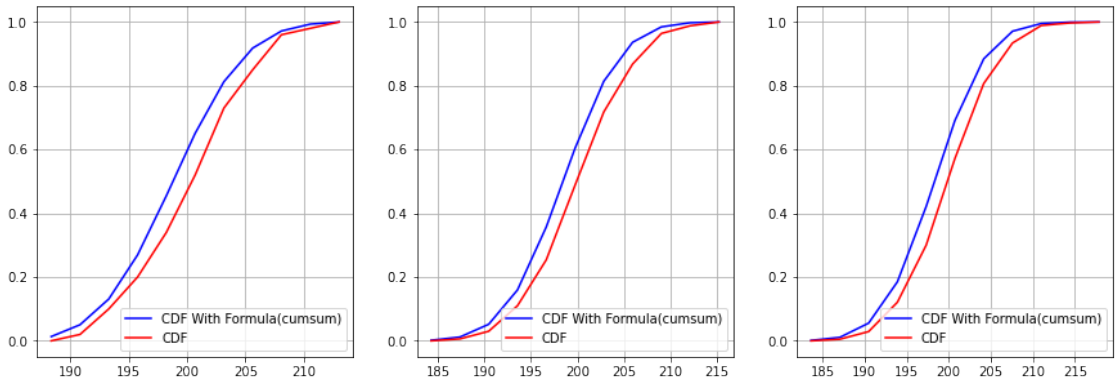
```
    CDF_F1[k] = (1+m.erf((bins1[k]-Miu)/(np.sqrt(2)*Sigma)))/2
ax.plot(bins1,CDF_F1,'b',label='CDF With Formula(erf)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins1):
    CDF_Array1[i]=(sum(hist1[0:i]))/sum(hist1)
    i=i+1
ax.plot(bins1,CDF_Array1,'r',label='CDF')
ax.grid(True)
ax.legend()
ax = plt.subplot(132)
CDF_Array2=[None]*len(bins2)
CDF_F2 = [None]*len(bins2)
for k in range(0,len(bins2)):
    CDF_F2[k] = (1+m.erf((bins2[k]-Miu)/(np.sqrt(2)*Sigma)))/2
ax.plot(bins2,CDF_F2,'b',label='CDF With Formula(erf)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins2):
    CDF_Array2[i]=(sum(hist2[0:i]))/sum(hist2)
    i=i+1
ax.plot(bins2,CDF_Array2,'r',label='CDF')
ax.grid(True)
ax.legend()
ax = plt.subplot(133)
CDF_Array3=[None]*len(bins3)
CDF_F3 = [None]*len(bins3)
for k in range(0,len(bins3)):
    CDF_F3[k] = (1+m.erf((bins3[k]-Miu)/(np.sqrt(2)*Sigma)))/2
ax.plot(bins3,CDF_F3,'b',label='CDF With Formula(erf)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins3):
    CDF_Array3[i]=(sum(hist3[0:i]))/sum(hist3)
    i=i+1
ax.plot(bins3,CDF_Array3,'r',label='CDF')
ax.grid(True)
ax.legend()
plt.show()
```



0.8 With Cumsum

[5]:
```
#This part is exactly like Q1 and Q2 combined,thus I have refrained from adding␣
 ↪any extra comments.
Miu=200
Sigma=5
np.random.seed(12345679)
RandDset1 = np.random.normal(Miu,Sigma,100)
hist1,bins1=np.histogram(RandDset1)
RandDset2 = np.random.normal(Miu,Sigma,500)
hist2,bins2=np.histogram(RandDset2)
RandDset3 = np.random.normal(Miu,Sigma,1000)
hist3,bins3=np.histogram(RandDset3)
plt.figure(figsize=(15,5))
ax = plt.subplot(131)
CDF_Array1=[None]*len(bins1)
CDF_F1c = [None]*len(bins1)
PDF_F1 = [None]*len(bins1)
PDF_F1=(1/(Sigma*np.sqrt(2*np.pi)))*np.exp(-1*((bins1-Miu)**2)/(2*Sigma**2))
CDF_F1c=np.cumsum(PDF_F1)/PDF_F1.sum()
ax.plot(bins1,CDF_F1c,'b',label='CDF With Formula(cumsum)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins1):
    CDF_Array1[i]=(sum(hist1[0:i]))/sum(hist1)
    i=i+1
ax.plot(bins1,CDF_Array1,'r',label='CDF')
ax.grid(True)
ax.legend()
ax = plt.subplot(132)
CDF_Array2=[None]*len(bins2)
CDF_F2c = [None]*len(bins2)
```

```
PDF_F2 = [None]*len(bins1)
PDF_F2=(1/(Sigma*np.sqrt(2*np.pi)))*np.exp(-1*((bins2-Miu)**2)/(2*Sigma**2))
CDF_F2c=np.cumsum(PDF_F2)/PDF_F2.sum()
ax.plot(bins2,CDF_F2c,'b',label='CDF With Formula(cumsum)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins2):
    CDF_Array2[i]=(sum(hist2[0:i]))/sum(hist2)
    i=i+1
ax.plot(bins2,CDF_Array2,'r',label='CDF')
ax.grid(True)
ax.legend()
ax = plt.subplot(133)
CDF_Array3=[None]*len(bins3)
CDF_F3c = [None]*len(bins3)
PDF_F3 = [None]*len(bins1)
PDF_F3=(1/(Sigma*np.sqrt(2*np.pi)))*np.exp(-1*((bins3-Miu)**2)/(2*Sigma**2))
CDF_F3c=np.cumsum(PDF_F3)/PDF_F3.sum()
ax.plot(bins3,CDF_F3c,'b',label='CDF With Formula(cumsum)')
ax.grid(True)
ax.legend()
i=0
while i<len(bins3):
    CDF_Array3[i]=(sum(hist3[0:i]))/sum(hist3)
    i=i+1
ax.plot(bins3,CDF_Array3,'r',label='CDF')
ax.grid(True)
ax.legend()
plt.show()
```



0.9 Q3 Explanation

As you can see with both the erf function and by using cumsum when we increase the amount of N the two CDF's shall get closer to each other,of course this is seen better with the erf method because the formula for the CDF of a normal(gaussian) distribution is described very accurately with the error funcion,hence in this question when we increase the amount of N the CDF's shall match better accordingly.

0.10 Q4 with erf

[6]:
```
#Here we define the needed variables.
Miu1=100
Sigma1=np.sqrt(10)
Miu1=100
Sigma1=np.sqrt(10)
Miu2=10
Sigma2=np.sqrt(2.5)
a0=Miu+Sigma
b0=Miu-Sigma
a1=Miu1+Sigma1
b1=Miu1-Sigma1
a2=Miu2+Sigma2
b2=Miu2-Sigma2
#Here we calculate the CDF values then display the wanted probabilities.
CDF_a0 = (1+m.erf((a0-Miu)/(np.sqrt(2)*Sigma)))/2
CDF_b0 = (1+m.erf((b0-Miu)/(np.sqrt(2)*Sigma)))/2
CDF_a1 = (1+m.erf((a1-Miu1)/(np.sqrt(2)*Sigma1)))/2
CDF_b1 = (1+m.erf((b1-Miu1)/(np.sqrt(2)*Sigma1)))/2
CDF_a2 = (1+m.erf((a2-Miu2)/(np.sqrt(2)*Sigma2)))/2
CDF_b2 = (1+m.erf((b2-Miu2)/(np.sqrt(2)*Sigma2)))/2
print(f'For Miu=200 and Sigma=5 we have : {CDF_a0-CDF_b0:.6f}')
print(f'For Miu=100 and Sigma=sqrt(10) we have : {CDF_a1-CDF_b1:.6f}')
print(f'For Miu=10 and Sigma=sqrt(2.5) we have : {CDF_a2-CDF_b2:.6f}')
```

```
For Miu=200 and Sigma=5 we have : 0.682689
For Miu=100 and Sigma=sqrt(10) we have : 0.682689
For Miu=10 and Sigma=sqrt(2.5) we have : 0.682689
```

0.11 Q4 with CDF like Q1

[7]:
```
#Here we define the needed variables.
Miu1=100
Miu2=10
Sigma1=np.sqrt(10)
Sigma2=np.sqrt(2.5)
#The probability for Miu=200 and Sigma=5 is projected as follows:
ProbA0=np.interp(Miu+Sigma,bins,CDF_Array)
ProbB0=np.interp(Miu-Sigma,bins,CDF_Array)
```

```
np.random.seed(12345679)
RandDset1 = np.random.normal(Miu1,Sigma1,100)
hist1,bins1=np.histogram(RandDset1)
CDF_Array1=[None]*len(bins1)
i=0
while i<len(bins1):
    CDF_Array1[i]=(sum(hist1[0:i]))/sum(hist1)
    i=i+1
ProbA1=np.interp(Miu1+Sigma1,bins1,CDF_Array1)
ProbB1=np.interp(Miu1-Sigma1,bins1,CDF_Array1)
np.random.seed(12345679)
RandDset2 = np.random.normal(Miu2,Sigma2,100)
hist2,bins2=np.histogram(RandDset2)
CDF_Array2=[None]*len(bins2)
j=0
while j<len(bins2):
    CDF_Array2[j]=(sum(hist2[0:j]))/sum(hist2)
    j=j+1
ProbA2=np.interp(Miu2+Sigma2,bins2,CDF_Array2)
ProbB2=np.interp(Miu2-Sigma2,bins2,CDF_Array2)
print(f'For Miu=200 and Sigma=5 we have : {ProbA0-ProbB0:.6f}')
print(f'For Miu=100 and Sigma=sqrt(10) we have : {ProbA1-ProbB1:.6f}')
print(f'For Miu=10 and Sigma=sqrt(2.5) we have : {ProbA2-ProbB2:.6f}')
```

```
For Miu=200 and Sigma=5 we have : 0.651497
For Miu=100 and Sigma=sqrt(10) we have : 0.651497
For Miu=10 and Sigma=sqrt(2.5) we have : 0.651497
```

0.12 Q4 Explanation

As we can see in both the Cumsum and erf version of the CDF function when we calculate the probability we reach the same values for any Miu and Sigma(there is a slight difference in the amount in the Cumsum and erf version of the CDF because the cumsum version isn't 100% accurate,the erf version is very precise.) The reason that the probability never changes is because the CDF of the Gaussian(Normal) distribution has nothing to do with Miu and Sigma,I have written this in a PDF file accordingly.)

0.13 Section 2

0.14 Q1

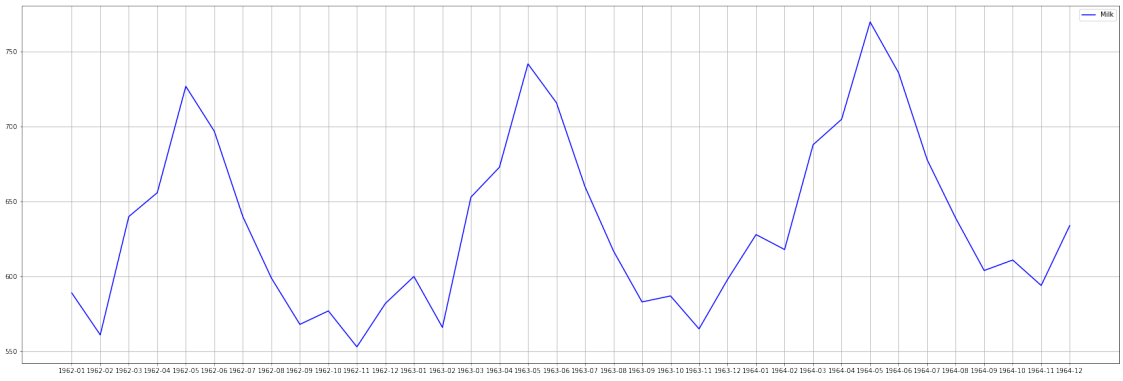
[8]:
```
import pandas as pd
#Here by using pandas I read the csv file and by using T.values.tolist() method␣
 ↪I convert it into a list of lists.
df = pd.read_csv('tesla-stock-price.csv')
data_list = df.T.values.tolist()
Opening_Value=data_list[3]
Value=int(input("Enter Value : "))
```

```
#Closing_Value=data_list[1]
Count=0
for i in range(0,len(Opening_Value)):
    if Opening_Value[i]>Value:
        Count=Count+1
    else:
        continue
print(f'The count is : {Count}')
```

```
Enter Value : 250
The count is : 436
```

0.15 Q2 Part1

[9]:
```
df = pd.read_csv('monthly-milk.csv')
data_list = df.T.values.tolist()
Month=data_list[0]
Pounds_of_milk=data_list[1]
#Month=np.arange(0,len(Pounds_of_milk))
#print(Pounds_of_milk)
plt.figure(figsize=(30,10))
ax = plt.subplot(111)
ax.plot(Month[0:36],Pounds_of_milk[0:36],'b',label='Milk')
ax.grid(True)
ax.legend()
plt.show()
```



0.16 Q2 Part 2

By plotting the milk produced by the cows in each year(in my case i have plotted it for three years) we deduce that in each year the cows produce the most milk in a certain part of the year(in this case in summer).

0.17 Q2 Part 3

[10]:
```
df = pd.read_csv('monthly-milk.csv')
data_list = df.T.values.tolist()
Month=data_list[0]
Pounds_of_milk=data_list[1]
#In this loop I have found the month with maximum milk in every year,because␣
 ↪the function is periodic
#all of them occur in the 5th month of the year which is may.
print(f'Month with most milk in each year')
print(f'--------------------------------')
for n in range(1,15):
    Max_Milk_Index = np.argmax(Pounds_of_milk[12*(n-1):12*n])
    print(Month[Max_Milk_Index*3*n-8])
# Here I shall display the most milk production in the first period which is␣
 ↪the first year
#for k in range()
```

```
Month with most milk in each year
---------------------------------
1962-05
1963-05
1964-05
1965-05
1966-05
1967-05
1968-05
1969-05
1970-05
1971-05
1972-05
1973-05
1974-05
1975-05
```

0.18 Q3

[11]:
```
df = pd.read_csv('migrants.csv')
df.drop(["percentarea", "ccode"], axis = 1, inplace = True)
data_list = df.T.values.tolist()
All_Country_Names=data_list[0]
All_Country_Migrants=data_list[1]
Sorted_All_Country_Migrants=All_Country_Migrants.copy()
Sorted_All_Country_Migrants.sort()
Top_Ten_Migrants=Sorted_All_Country_Migrants[len(All_Country_Migrants)-10:
 ↪len(All_Country_Migrants)]
Top_Ten_Country_Names=[None]*10
```

```
for i in range(0,10):
    for j in range(0,len(All_Country_Migrants)):
        if Top_Ten_Migrants[i]==All_Country_Migrants[j]:
            Top_Ten_Country_Names[i]=All_Country_Names[j]
            break
        else:
            continue
plt.figure(figsize=(20,10))
ax = plt.subplot(111)
ax.bar(Top_Ten_Country_Names,Top_Ten_Migrants,color='black',width=0.8)
plt.xticks(Top_Ten_Country_Names,rotation=90)
ax.grid(True)
plt.show()
```