

## ▼ PRUEBA PRÁCTICA #1

```
import pandas as pd
import numpy as np
import pylab as pl
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

```
año_parametro_inicio = None
año_parametro_fin = None
```

```
año_parametro_inicio = '2012'
año_parametro_inicio = año_parametro_inicio[2:4]
año_parametro_fin = '2021'
año_parametro_fin = año_parametro_fin[2:4]
```

```
año_parametro_fin
```

```
'21'
```

```
Poblaciones = pd.read_csv('1.Poblaciones.csv', sep = ';', skiprows=[0,2], usecols=['Periodo',
Poblaciones
```

```

sep_para = Poblaciones['Periodo'].str.split('-', expand=True)
sep_para.columns = ['Mes', 'Año']
Poblaciones = pd.concat([Poblaciones, sep_para], axis=1)
Poblaciones

```

	Periodo	Indicadores	Nacional	Mes	Año
0	dic-07	Población Total	13.682.302	dic	07
1	dic-07	Población menor de 15 años	4.372.812	dic	07
2	dic-07	Población en Edad de Trabajar (PET)	9.309.490	dic	07
3	dic-07	Población Económicamente Activa	6.336.029	dic	07
4	dic-07	Empleo	6.019.332	dic	07
...	...	...	...	...	...
895	oct-21	Desempleo Abierto	336.101	oct	21
896	oct-21	Desempleo Oculto	48.103	oct	21
897	oct-21	Desempleo Cesante	298.846	oct	21
898	oct-21	Desempleo Nuevo	85.358	oct	21
899	oct-21	Población Económicamente Inactiva	4.330.241	oct	21

900 rows × 5 columns

```

empleo = Poblaciones.loc[Poblaciones.Indicadores == 'Empleo']
desempleo = Poblaciones.loc[Poblaciones.Indicadores == 'Desempleo']

```

Codigo para obtener los valores de todos los años

```

año_empl = empleo['Año']
año_empl = año_empl.drop_duplicates()
año_empl = list(año_empl)

empleo_final = []
sumatoria = 0
contador = 0
empleo['Nacional']
for i in año_empl:
    for j in empleo.index:
        valor = empleo['Nacional'][j].split(sep='.')
        operacion = int(valor[0])*1000000+int(valor[1])*1000+int(valor[2])
        if str(empleo['Año'][j]) == i:
            sumatoria = sumatoria + operacion
            contador = contador + 1

```

```

empleo_final.append(sumatoria/contador)
sumatoria = 0
contador = 0

print(empleo_final)

[6019332.0, 6125310.0, 6125135.0, 6143685.5, 6264709.0, 6506555.5, 6695018.0, 6784413.7]

año_empl = desempleo['Año']
año_empl = año_empl.drop_duplicates()
año_empl = list(año_empl)

desempleo_final = []
sumatoria = 0
contador = 0
desempleo['Nacional']
for i in año_empl:
    for j in desempleo.index:
        valor = desempleo['Nacional'][j].split(sep='.')
        operacion = int(valor[0])*1000+int(valor[1])
        if str(desempleo['Año'][j]) == i:
            sumatoria = sumatoria + operacion
            contador = contador + 1
        desempleo_final.append(sumatoria/contador)
        sumatoria = 0
        contador = 0

print(desempleo_final)

[316697.0, 362084.5, 423802.0, 365672.5, 302996.0, 279372.5, 281348.0, 304555.0, 324618

```

## Codigo para un rango de años

```

rango_años = np.arange(int(año_parametro_inicio), int(año_parametro_fin)+1)
rango_años

array([12, 13, 14, 15, 16, 17, 18, 19, 20, 21])

resultados_empleos_anios = []
for anio in rango_años:
    if(anio <10 ):
        anio = '0'+str(anio)
    dataset_temporal = empleo.loc[empleo.Año == str(anio)]

    for j in dataset_temporal.index:
        valor = dataset_temporal['Nacional'][j].split(sep='.')

```

```
operacion = int(valor[0])*1000000+int(valor[1])*1000+int(valor[2])
dataset_temporal['Nacional'][j] = operacion
```

```
sumatoria = dataset_temporal['Nacional'].sum()
resultado1 = sumatoria/len(dataset_temporal)
resultados_empleos_anios.append(resultado1)
resultados_empleos_anios
```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html)  
 exec(code\_obj, self.user\_global\_ns, self.user\_ns)

```
[6506555.5,
 6695018.0,
 6784413.75,
 7151139.25,
 7482333.75,
 7766294.0,
 7778951.0,
 7853174.75,
 7673343.25,
 7917790.1]
```

```
resultados_desempleos_anios = []
for anio in rango_años:
    if(anio <10 ):
        anio = '0'+str(anio)
    dataset_temporal = desempleo.loc[desempleo.Año == str(anio)]
```

```
for j in dataset_temporal.index:
    valor = dataset_temporal['Nacional'][j].split(sep='.')
    operacion = int(valor[0])*1000+int(valor[1])
    dataset_temporal['Nacional'][j] = operacion
```

```
sumatoria = dataset_temporal['Nacional'].sum()
resultado1 = sumatoria/len(dataset_temporal)
resultados_desempleos_anios.append(resultado1)
resultados_desempleos_anios
```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html)  
 exec(code\_obj, self.user\_global\_ns, self.user\_ns)

```
[279372.5,
 281348.0,
 304555.0,
 324618.0,
 423871.75,
 358466.5,
```

```
330265.75,
365105.75,
456457.5,
431402.7]
```

## Graficas de resultados

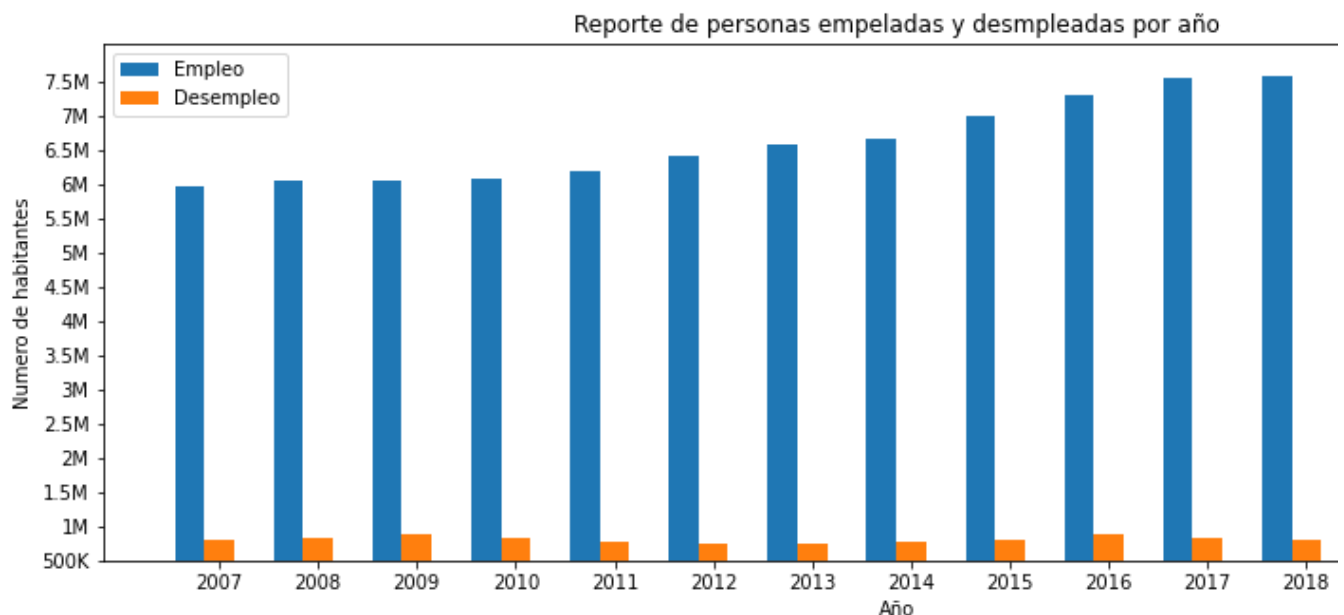
### Empleos de todos los años

```
numero_de_grupos = len(empleo_final)
plt.figure(figsize=(15,5))
indice_barras = np.arange(numero_de_grupos)
ancho_barras = 0.30

plt.bar(indice_barras, empleo_final, width=ancho_barras, label='Empleo', )
plt.bar(indice_barras + ancho_barras, desempleo_final, width=ancho_barras, label='Desempleo')
plt.legend(loc='best')

## Se colocan los indicadores en el eje x
plt.xticks(indice_barras + ancho_barras, ('2007', '2008', '2009', '2010','2011','2012', '2013', '2014', '2015', '2016', '2017', '2018'))
plt.yticks(np.arange(800000, step=550000 ),['500K','1M','1.5M','2M','2.5M','3M','3.5M','4M',
plt.ylabel('Numero de habitantes')
plt.xlabel('Año')
plt.title('Reporte de personas empedadas y desempleadas por año')

plt.show()
```



### Empleos y desempleos por un rango de años

```

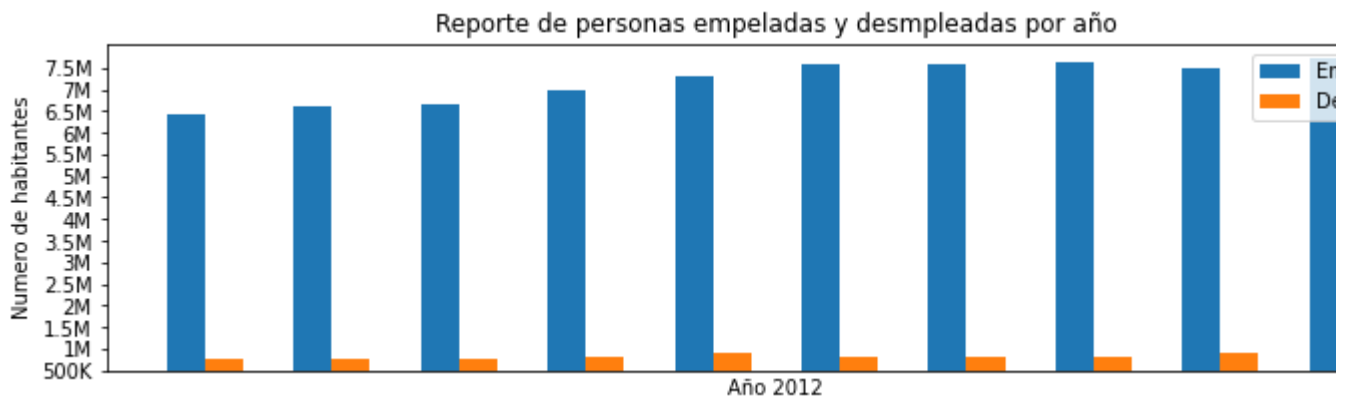
plt.figure(figsize=(12,3))
indice_barras = np.arange(len(resultados_empleos_anios))
ancho_barras =0.30
print(resultados_empleos_anios)
plt.bar(indice_barras, resultados_empleos_anios, width=ancho_barras, label='Empleo', )
plt.bar(indice_barras + ancho_barras, resultados_desempleos_anios, width=ancho_barras, label=
plt.legend(loc='best')

## Se colocan los indicadores en el eje
plt.xticks([])
plt.yticks(np.arange(8000000, step=550000 ),['500K','1M','1.5M','2M','2.5M','3M','3.5M','4M',
plt.ylabel('Numero de habitantes')
plt.xlabel('Año+' 20'+año_parametro_inicio)
plt.title('Reporte de personas empeladas y desmpleadas por año')

plt.show()

```

[6506555.5, 6695018.0, 6784413.75, 7151139.25, 7482333.75, 7766294.0, 7778951.0, 785317



Segundo grafico, sectorizacion

```

Sectorizacion = pd.read_csv('4. Sectorización del empleo.csv',sep = ';', skiprows=[0], encodi
Sectorizacion

```

	Característica	Unnamed: 1	jun-07	sep-07	dic-07	mar-08	jun-08	sep-08	dic-08	mar-09
0	Nacional	Sector Formal	-	-	41,0%	-	42,5%	-	43,9%	-
1	NaN	Sector Informal	-	-	45,1%	-	45,4%	-	43,5%	-
2	NaN	Empleo Doméstico	-	-	3,3%	-	3,3%	-	3,5%	-
3	NaN	No Clasificados por Sector	-	-	10,6%	-	8,8%	-	9,2%	-
4	Urbano	Sector Formal	54,7%	55,7%	54,1%	54,4%	55,8%	56,0%	56,2%	57,4%
5	NaN	Sector Informal	36,9%	36,1%	34,0%	36,6%	34,0%	35,0%	33,2%	34,0%
6	NaN	Empleo Doméstico	4,0%	4,1%	4,3%	3,9%	4,0%	3,9%	4,2%	3,7%
7	NaN	No Clasificados por Sector	4,4%	4,1%	7,7%	5,1%	6,2%	5,1%	6,4%	4,8%
8	Urbano	Sector Formal	45,5%	45,7%	45,5%	45,5%	45,8%	46,0%	46,2%	46,4%
9	NaN	Sector Informal	36,9%	36,1%	34,0%	36,6%	34,0%	35,0%	33,2%	34,0%
10	NaN	Empleo Doméstico	-	-	1,5%	-	1,8%	-	1,9%	-

```
print(año_parametro_inicio)

12
nombres_columnas= Sectorizacion.keys()
contador = 2
posiciones_años=[]
for i in nombres_columnas[2:]:

    if i.split(sep='-')[1] == año_parametro_inicio:
        posiciones_años.append(contador)
        contador+=1

print(posiciones_años)
llaves=Sectorizacion.keys()[posiciones_años]
dataset_preparado = Sectorizacion[llaves]
dataset_preparado
```

[21, 22, 23, 24]

	mar-12	jun-12	sep-12	dic-12
<b>0</b>	-	47,4%	-	48,7%
<b>1</b>	-	41,4%	-	40,8%
<b>2</b>	-	2,7%	-	2,5%
<b>3</b>	-	8,5%	-	8,0%
<b>4</b>	63,1%	61,6%	61,6%	62,5%
<b>5</b>	31,4%	29,2%	30,6%	29,8%
<b>6</b>	2,9%	3,3%	3,1%	2,9%
<b>7</b>	2,5%	5,9%	4,6%	4,9%
<b>8</b>	-	19,9%	-	21,2%
<b>9</b>	-	64,9%	-	62,8%

```
Dataset_Nacional = dataset_preparado[:4]
Dataset_Urbano = dataset_preparado[4:8]
Dataset_Rural = dataset_preparado[8:12]
```

```
elementosFormal=[]
elementosInformal=[]
elementosDomestico=[]
elementosNoClasificado=[]
```

```
valores_pieN=[]
valores_pieU=[]
valores_pieR=[]
```

```
etiquetas = ['Sector Formal', 'Sector Informal', 'Empleo Doméstico', 'Clasificados por Sec
```

```
for i in range(len(Dataset_Nacional)):
    llave = Dataset_Nacional.keys()[i]

    datosNacional = Dataset_Nacional[llave]

    if(datosNacional[0] != '-'):
        elementosFormal.append(datosNacional[0])
    if(datosNacional[1] != '-'):
        elementosInformal.append(datosNacional[1])
    if(datosNacional[2] != '-'):
        elementosDomestico.append(datosNacional[2])
    if(datosNacional[3] != '-'):
        elementosNoClasificado.append(datosNacional[3])
```



```

total=0
for i in elementosFormal:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieN.append(round(total/(len(elementosFormal)),2))

total=0
for i in elementosInformal:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieN.append(round(total/(len(elementosInformal)),2))

total=0
for i in elementosDomestico:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieN.append(round(total/(len(elementosDomestico)),2))

total=0
for i in elementosNoClasificado:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieN.append(round(total/(len(elementosNoClasificado)),2))
valores_pieN

```

```
[48.05, 41.1, 2.6, 8.25]
```

```

for i in range(len(Dataset_Urbano)):
    llave = Dataset_Urbano.keys()[i]

    datosNacional = Dataset_Urbano[llave]
    datosNacional

    if(datosNacional[4] != '-'):
        elementosFormal.append(datosNacional[4])
    if(datosNacional[5] != '-'):
        elementosInformal.append(datosNacional[5])
    if(datosNacional[6] != '-'):
        elementosDomestico.append(datosNacional[6])
    if(datosNacional[7] != '-'):
        elementosNoClasificado.append(datosNacional[7])

total=0
for i in elementosFormal:
    i=i.replace('%','')
    i=i.replace(',','.')

```

```

    total=total+float(i)
valores_pieU.append(round(total/(len(elementosFormal)),2))

total=0
for i in elementosInformal:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieU.append(round(total/(len(elementosInformal)),2))

total=0
for i in elementosDomestico:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieU.append(round(total/(len(elementosDomestico)),2))

total=0
for i in elementosNoClasificado:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieU.append(round(total/(len(elementosNoClasificado)),2))

valores_pieU

[57.48, 33.87, 2.9, 5.73]

for i in range(len(Dataset_Urbano)):
    llave = Dataset_Rural.keys()[i]

    datosNacional = Dataset_Rural[llave]
    datosNacional

    if(datosNacional[8] !='-'):
        elementosFormal.append(datosNacional[8])
    if(datosNacional[9] !='-'):
        elementosInformal.append(datosNacional[9])
    if(datosNacional[10] !='-'):
        elementosDomestico.append(datosNacional[10])
    if(datosNacional[11] !='-'):
        elementosNoClasificado.append(datosNacional[11])

total=0
for i in elementosFormal:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieR.append(round(total/(len(elementosFormal)),2))

total=0

```

```
for i in elementosInformal:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieR.append(round(total/(len(elementosInformal)),2))

total=0
for i in elementosDomestico:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieR.append(round(total/(len(elementosDomestico)),2))

total=0
for i in elementosNoClasificado:
    i=i.replace('%','')
    i=i.replace(',','.')
    total=total+float(i)
valores_pieR.append(round(total/(len(elementosNoClasificado)),2))

valores_pieR
```

```
[48.25, 41.36, 2.59, 7.78]
```




```
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(20,5))
fig.suptitle('Sectorizacion')
ax1.pie(valores_pieN, labels=etiquetas, autopct='%2f %%')
ax2.pie(valores_pieU, labels=etiquetas, autopct='%2f %%')
ax3.pie(valores_pieR, labels=etiquetas, autopct='%2f %%')

plt.figure()
```

&lt;Figure size 432x288 with 0 Axes&gt;

Sectorizacion

## Pregunta 3.




```
Subempleo = Poblaciones.loc[Poblaciones.Indicadores == 'Subempleo']
Pleno = Poblaciones.loc[Poblaciones.Indicadores == 'Empleo Adecuado/Pleno']
NoPleno = Poblaciones.loc[Poblaciones.Indicadores == 'Otro Empleo no pleno']
```





Clasificados por Sector

## Valores para el subempleo

```
# Primero extraemos el valor del
resultados_subempleos = []
for anio in rango_años:
    if(anio <10 ):
        anio = '0'+str(anio)
        dataset_temporal = Subempleo.loc[Subempleo.Año == str(anio)]

        for j in dataset_temporal.index:
            valor = dataset_temporal['Nacional'][j].replace('.', '')
            #operacion = int(valor[0])*1000000+int(valor[1])*1000+int(valor[2])

            dataset_temporal['Nacional'][j] = int(valor)

        sumatoria = dataset_temporal['Nacional'].sum()
        resultado1 = sumatoria/len(dataset_temporal)
        resultados_subempleos.append(resultado1)
resultados_subempleos
```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: SettingW  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user>  
exec(code\_obj, self.user\_global\_ns, self.user\_ns)

```
[634436.0,
 813697.5,
 890444.5,
1034029.25,
1437778.75,
1669343.5,
1493030.0,
1570431.25,
1978116.75,
1933593.1]
```

## Valores para el empleo pleno

```
resultados_empleo_pleno = []
```

```

for anio in rango_años:
    if(anio <10 ):
        anio = '0'+str(anio)
    dataset_temporal = Pleno.loc[Pleno.Año == str(anio)]

    for j in dataset_temporal.index:
        valor = dataset_temporal['Nacional'][j].replace('.', '')
        dataset_temporal['Nacional'][j] = int(valor)

    sumatoria = dataset_temporal['Nacional'].sum()
    resultado1 = sumatoria/len(dataset_temporal)
    resultados_empleo_pleno.append(resultado1)
resultados_empleo_pleno

```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html#setting-with-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#setting-with-copy)

```

exec(code_obj, self.user_global_ns, self.user_ns)
[3111499.0,
 3163264.5,
 3391765.25,
 3404390.25,
 3188784.0,
 3275341.0,
 3249694.5,
 3148216.75,
 2395943.75,
 2646380.1]

```

## Valores empleo no pleno

```

resultados_empleo_no_pleno = []
for anio in rango_años:
    if(anio <10 ):
        anio = '0'+str(anio)
    dataset_temporal = NoPleno.loc[NoPleno.Año == str(anio)]

    for j in dataset_temporal.index:
        valor = dataset_temporal['Nacional'][j].replace('.', '')
        #operacion = int(valor[0])*1000000+int(valor[1])*1000+int(valor[2])

        dataset_temporal['Nacional'][j] = int(valor)

    sumatoria = dataset_temporal['Nacional'].sum()
    resultado1 = sumatoria/len(dataset_temporal)
    resultados_empleo_no_pleno.append(resultado1)
resultados_empleo_no_pleno

```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user>

```
exec(code_obj, self.user_global_ns, self.user_ns)
```

```
[2040986.5,  
 2082403.5,  
 1983901.75,  
 2032688.75,  
 2070205.75,  
 1986290.0,  
 2162808.0,  
 2207743.5,  
 2253043.75,  
 2265989.9]
```

## Graficas de resultados

```
import matplotlib.pyplot as plot
```

```
plt.figure(figsize=(12,3))
```

```
indice_barras = np.arange(int(año_parametro_inicio), int(año_parametro_fin)+1)
```

```
indice_barras
```

```
ancho_barras =0.30
```

```
plt.bar(indice_barras, resultados_subempleos, width=ancho_barras, label='Subempleo', )
```

```
plt.bar(indice_barras + ancho_barras, resultados_empleo_pleno, width=ancho_barras, label='Empl
```

```
plt.bar(indice_barras + ancho_barras*2, resultados_empleo_no_pleno, width=ancho_barras, label
```

```
plt.legend(loc='best')
```

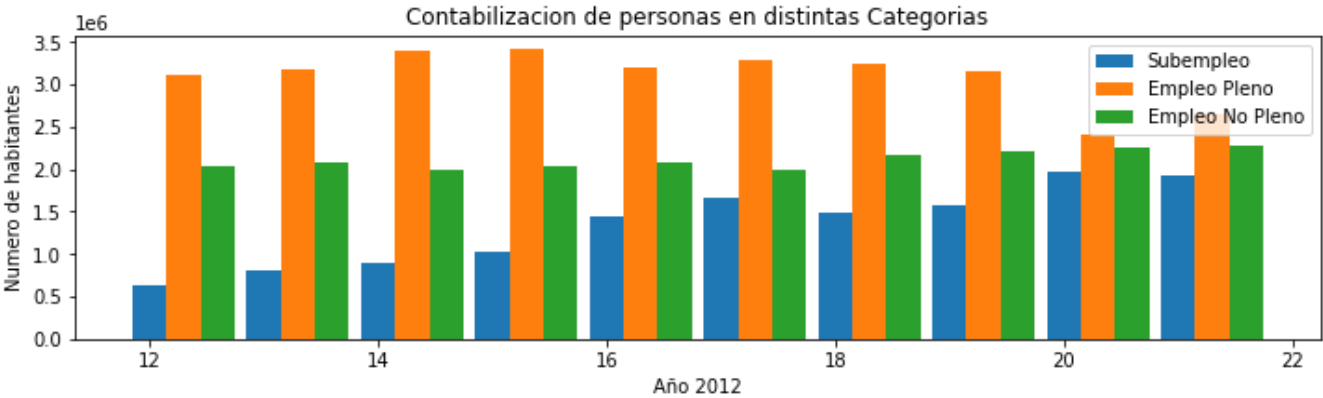
```
## Se colocan los indicadores en el eje
```

```
plt.ylabel('Numero de habitantes')
```

```
plt.xlabel('Año'+ ' 20'+año_parametro_inicio)
```

```
plt.title('Contabilizacion de personas en distintas Categorías')
```

```
plt.show()
```



✓ 0s completed at 8:14 PM

