03-practica_datos_politicos

September 10, 2024

1 Aplicando Python a datos políticos

1.1 INDICE

1.1.1 1. Carga de data frame con pandas (no estudiar)

1.1.2 2. Listas

1.1.3 3. Diccionarios

1.1.4 4. Estructuras de control

1.1.5 5. Funciones

2 ==========

2.0.1 1. Carga de data frame con pandas (no estudiar)

[2]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# pip install openpyxl
```

[3]:
```
aprobacion= pd.read_excel("data/aprobacion.xlsx")
```

[4]:
```
aprobacion.head()
```

[4]:

	pais	anio	trimestre	presidente	presidente_genero	aprobacion_neta	pib	corrupcion	poblacion	desempleo
0	Argentina	2000	1	Fernando de la Rúa	Masculino	40.125999	14.015253	5.521512e+11	37057452	15.000000
1	Argentina	2000	2	Fernando de la Rúa	Masculino	16.389999	14.015253	5.521512e+11	37057452	15.000000
2	Argentina	2000	3	Fernando de la Rúa	Masculino	23.968000	14.015253	5.521512e+11	37057452	15.000000
3	Argentina	2000	4	Fernando de la Rúa	Masculino	-18.254000	14.015253	5.521512e+11	37057452	15.000000
4	Argentina	2001	1	Fernando de la Rúa	Masculino	-6.973000	14.015253	5.278078e+11	37471509	18.299999

```
   crecimiento_pib
0             -0.8
1             -0.8
2             -0.8
3             -0.8
4             -4.4
```

Hay numeros que se visualizan en notacion cientifica por ser muy largos. vamos a corregirlo

[5]:
```
pd.set_option('display.float_format', lambda x: f'{x:.0f}')
```

[6]:
```
aprobacion.head()
```

[6]:
```
        pais  anio  trimestre          presidente presidente_genero  \
0  Argentina  2000          1  Fernando de la Rúa         Masculino
1  Argentina  2000          2  Fernando de la Rúa         Masculino
2  Argentina  2000          3  Fernando de la Rúa         Masculino
3  Argentina  2000          4  Fernando de la Rúa         Masculino
4  Argentina  2001          1  Fernando de la Rúa         Masculino

   aprobacion_neta  pib   corrupcion  poblacion  desempleo  crecimiento_pib
0               40   14 552151219031   37057452         15               -1
1               16   14 552151219031   37057452         15               -1
2               24   14 552151219031   37057452         15               -1
3              -18   14 552151219031   37057452         15               -1
4               -7   14 527807756979   37471509         18               -4
```

[101]:
```
aprobacion.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pais               1020 non-null   object
 1   anio               1020 non-null   int64
 2   trimestre          1020 non-null   int64
 3   presidente         1020 non-null   object
 4   presidente_genero  1020 non-null   object
 5   aprobacion_neta    1020 non-null   float64
 6   pib                1020 non-null   int64
 7   corrupcion         1020 non-null   float64
 8   poblacion          1020 non-null   int64
 9   desempleo          1020 non-null   float64
 10  crecimiento_pib    1020 non-null   float64
dtypes: float64(4), int64(4), object(3)
memory usage: 87.8+ KB
```

[102]:
```
aprobacion.describe()
```

[102]:

	anio	trimestre	aprobacion_neta	pib	corrupcion
count	1020	1020	1020	1020	1020
mean	2007	2	15	415993735215580	411402368992
std	4	1	28	258665895292915	687789664895
min	2000	1	-66	642001953125	17373017702
25%	2003	2	-7	212949919700623	40130146629
50%	2007	2	16	375663681030273	91972026250
75%	2011	3	38	662462463378906	400986433732
max	2014	4	87	944239273071289	3139626109943

	poblacion	desempleo	crecimiento_pib
count	1020	1020	1020
mean	31379665	7	4
std	47560715	4	4
min	3030347	1	-11
25%	5867626	4	2
50%	13072060	6	4
75%	30317848	9	6
max	204213133	21	18

2.0.2 2.Listas

2.1 Pueden repasar los metodos de creacíon de listas en el material de computación. Nosotros vamos a crarlas rapidamente desde el dataframe para trabajar sobre ellas

[103]:
```
# Convertir cada columna a una lista
pais_list = aprobacion['pais'].tolist()
anio_list = aprobacion['anio'].tolist()
trimestre_list = aprobacion['trimestre'].tolist()
presidente_list = aprobacion['presidente'].tolist()
presidente_genero_list = aprobacion['presidente_genero'].tolist()
aprobacion_neta_list = aprobacion['aprobacion_neta'].tolist()
pib_list = aprobacion['pib'].tolist()
corrupcion_list = aprobacion['corrupcion'].tolist()
poblacion_list = aprobacion['poblacion'].tolist()
desempleo_list = aprobacion['desempleo'].tolist()
crecimiento_pib_list = aprobacion['crecimiento_pib'].tolist()
```

[104]:
```
presidente_list[0:5]
```

[104]:
```
['Fernando de la Rúa',
 'Fernando de la Rúa',
 'Fernando de la Rúa',
 'Fernando de la Rúa',
 'Fernando de la Rúa']
```

[105]:
```
pais_unique = list(set(pais_list))
anio_unique = list(set(anio_list))
trimestre_unique = list(set(trimestre_list))
presidente_unique = list(set(presidente_list))
presidente_genero_unique = list(set(presidente_genero_list))
aprobacion_neta_unique = list(set(aprobacion_neta_list))
pib_unique = list(set(pib_list))
corrupcion_unique = list(set(corrupcion_list))
poblacion_unique = list(set(poblacion_list))
desempleo_unique = list(set(desempleo_list))
crecimiento_pib_unique = list(set(crecimiento_pib_list))
```

[106]:
```
presidente_unique[:11]
```

[106]:
```
['Jorge Quiroga Ramírez',
 'Porfirio Lobo Sosa',
 'Eduardo Alberto Duhalde',
 'Juan Orlando Hernández',
 'Vicente Fox',
 'Alan García',
 'Gonzalo Sánchez de Lozada',
 'Michelle Bachelet',
 'Valentín Paniagua',
 'Alberto Fujimori',
 'Evo Morales']
```

Metodos para listas

[107]:
```
# 1. Longitud de la Lista
print(len(pais_unique))  # No modifica la lista original
```

```
17
```

[108]:
```
# 2. Ordenar la Lista
print(pais_unique)

pais_unique.sort()  # Modifica la lista original [ascendete]
print(pais_unique)

pais_unique.sort(reverse=True)  # Modifica la lista original  [descendente]
print(pais_unique)
```

```
['Brasil', 'Costa Rica', 'Nicaragua', 'Uruguay', 'Perú', 'Venezuela',
'Argentina', 'Chile', 'Ecuador', 'México', 'Panamá', 'Colombia', 'Paraguay', 'El
Salvador', 'Bolivia', 'Honduras', 'Guatemala']
['Argentina', 'Bolivia', 'Brasil', 'Chile', 'Colombia', 'Costa Rica', 'Ecuador',
'El Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá',
'Paraguay', 'Perú', 'Uruguay', 'Venezuela']
['Venezuela', 'Uruguay', 'Perú', 'Paraguay', 'Panamá', 'Nicaragua', 'México',
```

```
'Honduras', 'Guatemala', 'El Salvador', 'Ecuador', 'Costa Rica', 'Colombia',
'Chile', 'Brasil', 'Bolivia', 'Argentina']
```

[109]:
```
# 3. Invertir el Orden de la Lista
pais_unique.reverse()  # Modifica la lista original
print(pais_unique)
```

```
['Argentina', 'Bolivia', 'Brasil', 'Chile', 'Colombia', 'Costa Rica', 'Ecuador',
'El Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá',
'Paraguay', 'Perú', 'Uruguay', 'Venezuela']
```

[110]:
```
# 4. Agregar Elementos a la Lista
pais_unique.append('Jamaica')  # Modifica la lista original
print(pais_unique)

pais_unique.insert(1, 'EEUU')  # Modifica la lista original
print(pais_unique)
```

```
['Argentina', 'Bolivia', 'Brasil', 'Chile', 'Colombia', 'Costa Rica', 'Ecuador',
'El Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá',
'Paraguay', 'Perú', 'Uruguay', 'Venezuela', 'Jamaica']
['Argentina', 'EEUU', 'Bolivia', 'Brasil', 'Chile', 'Colombia', 'Costa Rica',
'Ecuador', 'El Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua',
'Panamá', 'Paraguay', 'Perú', 'Uruguay', 'Venezuela', 'Jamaica']
```

[111]:
```
# 5. Eliminar Elementos de la Lista
pais_unique.remove('Chile')  # Modifica la lista original
print(pais_unique)

del pais_unique[2]  # Modifica la lista original
print(pais_unique)

last_element = pais_unique.pop()  # Modifica la lista original y guarda el␣
 ↪elemento eliminado
print(pais_unique)
print(f'Elemento eliminado: {last_element}')
```

```
['Argentina', 'EEUU', 'Bolivia', 'Brasil', 'Colombia', 'Costa Rica', 'Ecuador',
'El Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá',
'Paraguay', 'Perú', 'Uruguay', 'Venezuela', 'Jamaica']
['Argentina', 'EEUU', 'Brasil', 'Colombia', 'Costa Rica', 'Ecuador', 'El
Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá', 'Paraguay',
'Perú', 'Uruguay', 'Venezuela', 'Jamaica']
['Argentina', 'EEUU', 'Brasil', 'Colombia', 'Costa Rica', 'Ecuador', 'El
Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá', 'Paraguay',
'Perú', 'Uruguay', 'Venezuela']
Elemento eliminado: Jamaica
```

[112]:
```
# 6. Buscar un Elemento en la Lista
print('Argentina' in pais_unique)  # No modifica la lista original

index = pais_unique.index('Argentina')  # No modifica la lista original
print(f'Índice de Argentina: {index}')
```

```
True
Índice de Argentina: 0
```

[113]:
```
# 7. Contar Elementos en la Lista
count = pais_unique.count('Argentina')  # No modifica la lista original
print(f'Cantidad de veces que aparece Argentina: {count}')
```

```
Cantidad de veces que aparece Argentina: 1
```

[114]:
```
# 8. Unir Listas
anio_unique = [2020, 2021, 2022, 2023, 2024]
combined_list = pais_unique + anio_unique  # Crea una nueva lista combinada
print(combined_list)
```

```
['Argentina', 'EEUU', 'Brasil', 'Colombia', 'Costa Rica', 'Ecuador', 'El
Salvador', 'Guatemala', 'Honduras', 'México', 'Nicaragua', 'Panamá', 'Paraguay',
'Perú', 'Uruguay', 'Venezuela', 2020, 2021, 2022, 2023, 2024]
```

[115]:
```
# 10. Convertir la Lista a una Cadena
pais_string = ', '.join(pais_unique)  # Crea una nueva cadena
print(pais_string)
type(pais_string)  #comprobamos el tipo de dato
```

```
Argentina, EEUU, Brasil, Colombia, Costa Rica, Ecuador, El Salvador, Guatemala,
Honduras, México, Nicaragua, Panamá, Paraguay, Perú, Uruguay, Venezuela
```

[115]:
```
str
```

[116]:
```
# 10. Obtener un subconjunto de la lista
primeros_tres = pais_unique[:3]
primeros_tres
```

[116]:
```
['Argentina', 'EEUU', 'Brasil']
```

2.1.1 algunos metodos mas pero para datos numericos

[117]:
```
desempleo_unique = [5.2, 7.1, 6.5, 4.8, 5.9, 8.3, 7.0, 5.4]

# 1. Obtener el valor mínimo y máximo
min_desempleo = min(desempleo_unique)
print(min_desempleo)
max_desempleo = max(desempleo_unique)
print(max_desempleo)
```

```
4.8
8.3
```

[118]:
```
# 2. Suma de todos los elementos
suma_desempleo = sum(desempleo_unique)
suma_desempleo
```

[118]:
```
50.2
```

[119]:
```
# 3. Promedio de los elementos
promedio_desempleo = sum(desempleo_unique) / len(desempleo_unique)
print(f"Promedio: {promedio_desempleo}")
```

```
Promedio: 6.275
```

[120]:
```
# Otra forma de calcular promedio

import statistics
```

[121]:
```
statistics.mean(desempleo_unique)
```

[121]:
```
6.275
```

[122]:
```
# 4. Redondear los valores
desempleo_redondeado = [round(num, 0) for num in desempleo_unique]
desempleo_redondeado
```

[122]:
```
[5.0, 7.0, 6.0, 5.0, 6.0, 8.0, 7.0, 5.0]
```

2.1.2 Diccionario a partir de listas

Nos apuramos y qusimos generar rapidamente un diccionario con nuestras listas_unique pero hubo un problema...

[123]:
```
# Crear un diccionario con 5 elementos
aprobacion_dict = {
    'pais': pais_unique[:5],  # Primeros 5 elementos de la columna 'pais'
    'anio': anio_unique[:5],  # Primeros 5 elementos de la columna 'anio'
    'presidente': presidente_unique[:5],  # Primeros 5 elementos de la columna␣
 ↪'presidente'
    'aprobacion_neta': aprobacion_neta_unique[:5],  # Primeros 5 elementos de␣
 ↪la columna 'aprobacion_neta'
    'pib': pib_unique[:5]  # Primeros 5 elementos de la columna 'pib'
}
aprobacion_dict
```

[123]:
```
{'pais': ['Argentina', 'EEUU', 'Brasil', 'Colombia', 'Costa Rica'],
 'anio': [2020, 2021, 2022, 2023, 2024],
 'presidente': ['Jorge Quiroga Ramírez',
```

```
 'Porfirio Lobo Sosa',
 'Eduardo Alberto Duhalde',
 'Juan Orlando Hernández',
 'Vicente Fox'],
'aprobacion_neta': [1.12699997425079,
 2.927000046,
 3.76200008392334,
 4.059999943,
 5.1300001139999996],
'pib': [666188659667969,
 497184448242188,
 278633213043213,
 291961498260498,
 702841491699219]}
```

Estos valores nunca van a tener sentido porque con unique rompimos la relacion original de los datos entre sí.

2.1.3 Diccionario a partir de listas pero con datos manuales

[124]:
```
# SE UTILIZAN NUMEROS AL AZAR SOLO PARA EJEMPLIFICAR
sample_dict = {
    'pais': ['Argentina', 'Brasil', 'Chile', 'Uruguay', 'Colombia'],
    'presidente': ['Javier Milei', 'Lula Da Silva', 'Gabriel Boric ', 'Luis␣
 ↪Lacalle Pou', 'Gustavo Petro'],
    'aprobacion_neta': [50.5, 55.0, 35.2, 34.5, 41.0],
    'pib': [100000, 300000, 90000, 30000, 60000],
    'poblacion': [46000000, 210000000, 19000000, 3500000, 51000000],
    'desempleo': [13.5, 3.0, 7.2, 6.1, 7.5],
    'crecimiento_pib': [-2.5, 2.5, 1, 0.8, 0.5]
}
```

[125]:
```
sample_dict
```

[125]:
```
{'pais': ['Argentina', 'Brasil', 'Chile', 'Uruguay', 'Colombia'],
 'presidente': ['Javier Milei',
  'Lula Da Silva',
  'Gabriel Boric ',
  'Luis Lacalle Pou',
  'Gustavo Petro'],
 'aprobacion_neta': [50.5, 55.0, 35.2, 34.5, 41.0],
 'pib': [100000, 300000, 90000, 30000, 60000],
 'poblacion': [46000000, 210000000, 19000000, 3500000, 51000000],
 'desempleo': [13.5, 3.0, 7.2, 6.1, 7.5],
 'crecimiento_pib': [-2.5, 2.5, 1, 0.8, 0.5]}
```

[126]:
```
# Métodos básicos de diccionarios
print(sample_dict.keys())  # Muestra las claves
```

```
dict_keys(['pais', 'presidente', 'aprobacion_neta', 'pib', 'poblacion',
'desempleo', 'crecimiento_pib'])
```

[127]:
```
print(sample_dict.values())  # Muestra los valores
```

```
dict_values([['Argentina', 'Brasil', 'Chile', 'Uruguay', 'Colombia'], ['Javier
Milei', 'Lula Da Silva', 'Gabriel Boric ', 'Luis Lacalle Pou', 'Gustavo Petro'],
[50.5, 55.0, 35.2, 34.5, 41.0], [100000, 300000, 90000, 30000, 60000],
[46000000, 210000000, 19000000, 3500000, 51000000], [13.5, 3.0, 7.2, 6.1, 7.5],
[-2.5, 2.5, 1, 0.8, 0.5]])
```

[128]:
```
print(sample_dict.items())  # Muestra las pares clave-valor
```

```
dict_items([('pais', ['Argentina', 'Brasil', 'Chile', 'Uruguay', 'Colombia']),
('presidente', ['Javier Milei', 'Lula Da Silva', 'Gabriel Boric ', 'Luis Lacalle
Pou', 'Gustavo Petro']), ('aprobacion_neta', [50.5, 55.0, 35.2, 34.5, 41.0]),
('pib', [100000, 300000, 90000, 30000, 60000]), ('poblacion', [46000000,
210000000, 19000000, 3500000, 51000000]), ('desempleo', [13.5, 3.0, 7.2, 6.1,
7.5]), ('crecimiento_pib', [-2.5, 2.5, 1, 0.8, 0.5])])
```

[129]:
```
print(sample_dict.get("poblacion"))  # Obtiene el valor asociado a una clave
```

```
[46000000, 210000000, 19000000, 3500000, 51000000]
```

[130]:
```
sample_dict["ideología"] =␣
↪["globalista",None,"globalista","globalista","nacionalista"]  # Añade una␣
↪nueva clave-valor
```

[131]:
```
sample_dict
```

[131]:
```
{'pais': ['Argentina', 'Brasil', 'Chile', 'Uruguay', 'Colombia'],
 'presidente': ['Javier Milei',
  'Lula Da Silva',
  'Gabriel Boric ',
  'Luis Lacalle Pou',
  'Gustavo Petro'],
 'aprobacion_neta': [50.5, 55.0, 35.2, 34.5, 41.0],
 'pib': [100000, 300000, 90000, 30000, 60000],
 'poblacion': [46000000, 210000000, 19000000, 3500000, 51000000],
 'desempleo': [13.5, 3.0, 7.2, 6.1, 7.5],
 'crecimiento_pib': [-2.5, 2.5, 1, 0.8, 0.5],
 'ideología': ['globalista', None, 'globalista', 'globalista', 'nacionalista']}
```

2.1.4 4.Estructuras de control

for

[132]:
```
# Convertir cada elemento de la lista redondeada en enteros
desempleo_redondeado
desempleo_entero = [int(num) for num in desempleo_redondeado]
print(desempleo_entero)
```

```
[5, 7, 6, 5, 6, 8, 7, 5]
```

Una variante del ejemplo anterior

[133]:
```
# Lista vacía para almacenar los valores enteros
desempleo_entero2 = []

# Iterar sobre los elementos de la lista redondeada y convertir a enteros
for num in desempleo_redondeado:
    desempleo_entero2.append(int(num))

# Mostrar la lista resultante
print(desempleo_entero2)
```

```
[5, 7, 6, 5, 6, 8, 7, 5]
```

2.1.5 5.Funciones

[134]:
```
aprobacion
```

[134]:

	pais	anio	trimestre	presidente	presidente_genero	\
0	Argentina	2000	1	Fernando de la Rúa	Masculino	
1	Argentina	2000	2	Fernando de la Rúa	Masculino	
2	Argentina	2000	3	Fernando de la Rúa	Masculino	
3	Argentina	2000	4	Fernando de la Rúa	Masculino	
4	Argentina	2001	1	Fernando de la Rúa	Masculino	
…	…	…	…	…	…	
1015	Venezuela	2013	4	Nicolás Maduro	Masculino	
1016	Venezuela	2014	1	Nicolás Maduro	Masculino	
1017	Venezuela	2014	2	Nicolás Maduro	Masculino	
1018	Venezuela	2014	3	Nicolás Maduro	Masculino	
1019	Venezuela	2014	4	Nicolás Maduro	Masculino	

	aprobacion_neta	pib	corrupcion	poblacion	desempleo	\
0	40	140152530670166	552151219031	37057452	15	
1	16	140152530670166	552151219031	37057452	15	
2	24	140152530670166	552151219031	37057452	15	
3	-18	140152530670166	552151219031	37057452	15	
4	-7	140152530670166	527807756979	37471509	18	
…	…	…	…	…	…	
1015	-17	944239273071289	535572061141	30317848	8	
1016	-18	944239273071289	514714815230	30738378	7	
1017	-19	944239273071289	514714815230	30738378	7	
1018	-22	944239273071289	514714815230	30738378	7	

```
1019              -25  944239273071289 514714815230   30738378          7

      crecimiento_pib
0                  -1
1                  -1
2                  -1
3                  -1
4                  -4
…                   …
1015                1
1016               -4
1017               -4
1018               -4
1019               -4

[1020 rows x 11 columns]
```

[142]:
```python
# Creamos una funcion que nos devuelva si el crecimiento del PBI es
# negativo o positivo para el pais y años indicados:

def crecimiento_PBI(pais_buscado, ano_buscado):
    for i in range(len(pais_list)):
        if pais_list[i] == pais_buscado and anio_list[i] == ano_buscado:
            crecimiento = crecimiento_pib_list[i]
            if crecimiento > 0:
                return 'Positivo'
            elif crecimiento < 0:
                return 'Negativo'
            else:
                return 'Neutro'

    return f"No se encontró el crecimiento del PIB para {pais_buscado} en el␣
 ↪año {ano_buscado}."


# Ejemplo de uso:
resultado = crecimiento_PBI('Venezuela', 2013)
print(resultado)
resultado = crecimiento_PBI('Venezuela', 2014)
print(resultado)
```

```
Positivo
Negativo
```

[136]:
```python
crecimiento_PBI('Argentina', 2014)
```

[136]:
```
'Negativo'
```

[137]:
```python
# Ahora queremos crear una funcion que nos devuelva el calculo de
# PBI per capita (PBI / poblacion) para el pais y año indicados

def pbi_pc(pais_buscado, ano_buscado):
    for i in range(len(pais_list)):
        if pais_list[i] == pais_buscado and anio_list[i] == ano_buscado:
            pib = pib_list[i]
            poblacion = poblacion_list[i]
            return pib / poblacion

    return f"No se encontró el PIB o la población para {pais_buscado} en el año␣
↪{ano_buscado}."
```

[140]:
```python
pbi_pc('Argentina', 2000)
```

[140]:
```
3782033.6560151516
```