{desafío}
latam_

Estructuras de datos _

Sesión Experimental 2

Itinerario

Activación de conceptos

Desarrollo Desafío

Panel de discusión

/* Activación de conceptos */

¿Cómo accedemos al número “3” en la siguiente lista?

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

- lista[3]
- lista[“3”]
- lista[4]
- lista[2]

¿Cuál de las siguientes funciones tiene retorno?

1. list.append(x)
2. list.insert(num, x)
3. list.pop()
4. list.remove(str)
5. list.sort()

¿Cuál de las siguientes relaciones es correcta?

1. map() y sumatoria
2. filter() y transformación
3. reduce() y sumatoria
4. reduce() y filtrar
5. filter() y transformación

Pandas

- Librería orientada a la manipulación y limpieza de datos
- Se importa cómo “import pandas as pd”
- Sus principales estructuras son el DataFrame y las Series
- Se puede crear un DataFrame a partir de un csv con la función pd.read_csv()

DataFrame

```
: 1 import pandas as pd
  2
  3 df = pd.read_csv("natons.csv")
  4
  5 # vamos a extraer las primeras tres observaciones
  6 df.head(3)
```

:

	Unnamed: 0	country	region	gdp	school	adfert	chldmort	life	pop	urban	femlab	literacy	co2	gini
0	1	Algeria	Africa	7300.399902	6.716667	7.300000	34.75	72.316666	34172236	64.933334	0.4522	72.599998	15.0	NaN
1	2	Benin	Africa	1338.800049	3.100000	111.699997	122.75	54.733334	8237634	41.000000	0.8482	41.700001	1.2	NaN
2	3	Botswana	Africa	12307.400391	8.600000	52.099998	60.25	52.250000	1941233	59.250000	0.8870	84.099998	9.2	NaN

Serie y value_counts()

```
1 df['region'].value_counts()
```

```
Africa      52
Asia        49
Europe      43
Americas    35
Oceania     15
Name: region, dtype: int64
```

Subsets

Utilizando loc

```
# Queremos extraer sólo las columnas gdp school adfert chldmort de este subset
df_col_subset = df.loc[:, ['gdp', 'school', 'adfert', 'chldmort']]
df_col_subset.head(1)
```

	gdp	school	adfert	chldmort
0	7300.399902	6.716667	7.3	34.75

Utilizando filtro

```
df_americas = df[df['region'] == 'Americas']
```

```
# tamaño
df_americas.shape
```

```
(35, 13)
```

Iteración con iteritems()

```
1 for colname, serie in df.iteritems():
2     print(colname)
```

```
country
region
gdp
school
adfert
chldmort
life
pop
urban
femlab
literacy
co2
gini
```

Iteración con iterrows()

```python
for index, row_serie in df.iterrows():
    if index == 1:
        print(row_serie)
        print(type(row_serie))
```

```
country       Benin
region       Africa
gdp          1338.8
school          3.1
adfert         111.7
chldmort      122.75
life         54.7333
pop          8237634
urban             41
femlab        0.8482
literacy        41.7
co2              1.2
gini             NaN
Name: 1, dtype: object
<class 'pandas.core.series.Series'>
```

Numpy

- Librería para la computación numérica
- Usado por muchos módulos de Python
- Permiten crear arreglos n-dimensionales
- Poseen funciones matemáticas orientadas a la velocidad de implementación
- Permite trabajar con álgebra lineal
- Se importa cómo “import numpy as pd”

Arreglos de Numpy (ndarray)

```
1  import numpy as np
2
3  # La función np.array() recibe una lista nativa de python
4  array_de_numpy = np.array(["perro", "gato", "erizo"])
5  array_de_numpy
```

```
array(['perro', 'gato', 'erizo'], dtype='<U5')
```

Otras funciones que retornan ndarrays

- **np.asarray:** Convierte inputs sólo si éstos no son ndarray.
- **np.arange:** Genera un rango de manera similar a la implementación nativa de range(), devolviendo una lista en formato ndarray.
- **np.ones, np.ones_like:** Generan arrays poblados de 1 con las dimensiones especificadas. np.ones_like toma como referencia las dimensiones y tipo de dato de otro objeto.

Otras funciones que retornan ndarrays

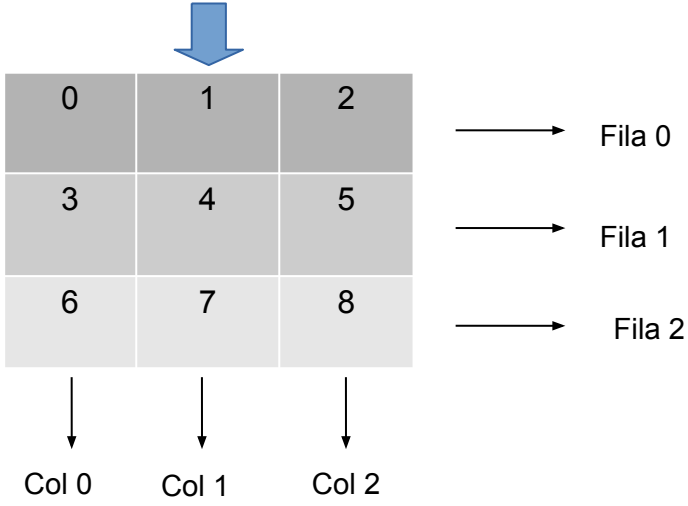
- **np.zeros, np.zeros_like:** Generan arrays poblados de 0 con las dimensiones especificadas. np.zeros_like toma como referencia las dimensiones y tipo de dato de otro objeto.
- **np.empty, np.empty_like:** Generan arrais poblados mediante la asignación de memoria, pero no completa los arrays con elementos.
- **np.full, np.full_like:** Produce un array con determinadas dimensiones y tipo de dato, rellenadas con un valor determinado.

Array n-dimensional (2 dimensiones)

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

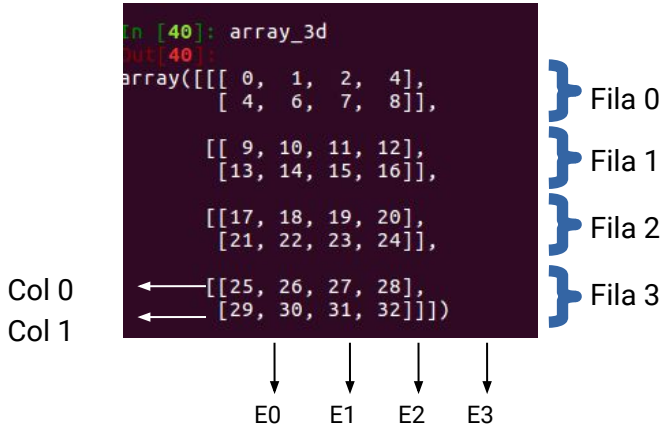
```
In [28]: matriz = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])

In [29]: matriz
Out[29]:
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```



Array n-dimensional (3 dimensiones)

```
array_3d = np.array([ [ [0, 1, 2, 4], [4, 6, 7, 8] ], [ [9, 10, 11, 12], [13, 14, 15, 16] ], [
[17, 18, 19, 20], [21, 22, 23, 24] ], [ [25, 26, 27, 28], [29, 30, 31, 32] ] ])
```



[ ]: 4 Filas, primera dimensión
[ ]: 2 Columnas por fila, segunda dimensión
26: 4 elementos dentro de cada columna dentro de cada fila, tercera dimensión

/* Desafío */

/* Panel de discusión */

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com