UNIVERSIDAD EAFIT

Científico de Datos

Nivel Básico

Aliados:

Microsoft

idata
Advanced analytics for business

Vigilada Mineducación

UNIVERSIDAD EAFIT

Tema:

Python como lenguaje de DS

Lenguajes más usados en DS

Profundizar en lenguajes

Operaciones lógicas

Un tipo importante de operación en programación son las **operaciones lógicas**. Estas pueden realizarse sobre **variables booleanas**.

```
In [27]: variable_1 = True
         variable_2 = False
         print(variable_1 or variable_2)
```

```
True
```

```
In [28]: print(not(variable_1))
```

```
False
```

A	B	A & B
False	False	False
False	True	False
True	False	False
True	True	True

A	B	A or B
False	False	False
False	True	True
True	False	Ture
True	True	True

Condicionales -if

Los **condicionales** son bloques de código que se ejecutan únicamente si se cumple una condición. El resultado de esta condición debe ser un **Booleano** (True o False). Esto se logra mediante el condicional **if.**

```
[10]: valor = 5
      if valor > 10:
          print('El valor es mayor que 10')
```

```
5 > 10
```

```
False
```

No se cumple la condición.

```
[11]: valor = 15
      if valor > 10:
          print('El valor es mayor que 10')
```

```
El valor es mayor que 10
```

```
15 > 10
```

```
True
```

Se cumple la condición.

Condicionales –if/else

Además uno puede agregar un código que se ejecute si la condición no se cumple. Para esto se utiliza el condicional **else**.

```
In [77]: nombre = 'Pedro'

         if nombre == 'Juan':
             print('Esta persona se llama Juan')
         else:
             print('Esta persona NO se llama Juan')
```

```
Esta persona NO se llama Juan
```

```
'Pedro'=='Pedro'
```

```
True
```

```
'Juan'=='Pedro'
```

```
False
```

La comparación entre strings también genera un booleano.

Nota: Para condicionales usamos doble igual ==, ya que nos reservamos el igual simple = para la asignación de variables.

Condicionales –if/elif/else

Además del **if** y el **else**, uno puede agregar más condiciones a través de condicional **elif** (else if). De esta forma se puede agregar un número arbitrario de condiciones.

In [80]:
```
edad = 20

if edad < 18:
    print('Esta persona tiene menos de 18 años')
elif edad > 18:
    print('Esta persona tiene mas de 18 años')
else:
    print('Esta persona tiene justo 18 años')
```

```
Esta persona tiene mas de 18 años
```

Numpy

- Es una librería para cálculo numérico
- Trae una estructura de datos: los **arrays.**

Ver Documentación Numpy

Uso de los arrays

Es una lista

```
[2]: import numpy as np

     arreglo = np.array([0,1,2,3,4,5])
     arreglo

[2]: array([0, 1, 2, 3, 4, 5])

[3]: print(arreglo)

     [0 1 2 3 4 5]
```

```
[2]: import numpy as np

     arreglo = np.array([0,1,2,3,4,5])
     arreglo

[2]: array([0, 1, 2, 3, 4, 5])

[3]: print(arreglo)

     [0 1 2 3 4 5]

[ ]: arreglo + 2

[ ]: array([2, 3, 4, 5, 6, 7])
```

Aliados:

Microsoft

idata Advanced analytics for business

Vigilada Mineducación

Formas de crear arrays (1/2)

1. Listas

```
[2]: import numpy as np

     arreglo = np.array([0,1,2,3,4,5])
     arreglo

[2]: array([0, 1, 2, 3, 4, 5])

[3]: print(arreglo)

     [0 1 2 3 4 5]
```

2. np.arange

```
[5]: import numpy as np

[6]: arreglo_1 = np.arange(2,9)
     arreglo_1

[6]: array([2, 3, 4, 5, 6, 7, 8])

[7]: arreglo_2 = np.arange(2,9,2)
     arreglo_2

[7]: array([2, 4, 6, 8])
```

Formas de crear arrays (2/2)

3. np.linspace

```
[5]: import numpy as np

[8]: arreglo_3 = np.linspace(2,9,3)
     arreglo_3

[8]: array([2. , 5.5, 9. ])

[9]: arreglo_4 = np.linspace(2,9,20)
     arreglo_4

[9]: array([2.        , 2.36842105, 2.73684211, 3.10526316, 3.47368421,
            3.84210526, 4.21052632, 4.57894737, 4.94736842, 5.31578947,
            5.68421053, 6.05263158, 6.42105263, 6.78947368, 7.15789474,
            7.52631579, 7.89473684, 8.26315789, 8.63157895, 9.        ])
```

Arreglos equiespaciados

Slicing o Indexación en arrays

- Ver posición específica

```
[21]: arreglo = np.arange(2,20,4)
      arreglo

[21]: array([ 2,  6, 10, 14, 18])

[22]: print(arreglo[0], arreglo[2], arreglo[-1], arreglo[-4])
      2 10 18 6
```

- Y si queremos rangos:

```
[32]: arreglo = np.arange(0,15)
      arreglo

[32]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

[33]: arreglo[2:12:2]

[33]: array([ 2,  4,  6,  8, 10])
```

salto

comienzo

final

Aliados:

Microsoft

idata Advanced analytics for business

Selección en arrays

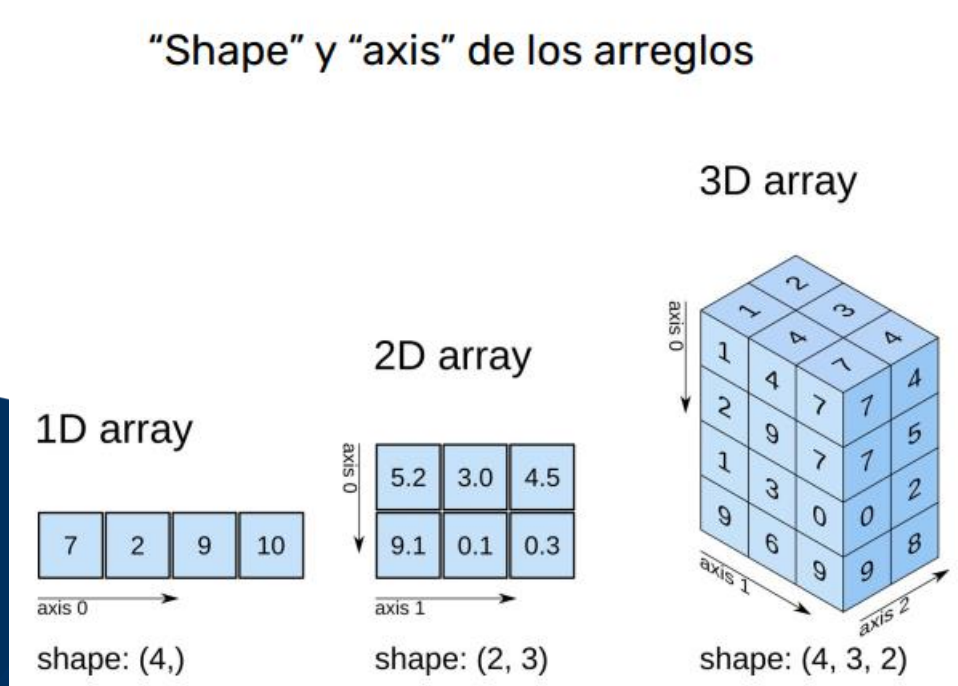
```
[34]: arreglo = np.arange(0,15)
      arreglo

[34]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

[35]: arreglo[2:7] = 25
      arreglo

[35]: array([ 0,  1, 25, 25, 25, 25, 25,  7,  8,  9, 10, 11, 12, 13, 14])
```

Arrays multidimensionales (1/2)



No es la forma más cómoda de crearlo

```
[39]: arreglo2d = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
      arreglo2d

[39]: array([[ 1,  2,  3,  4],
             [ 5,  6,  7,  8],
             [ 0, 10, 11, 12]])

[40]: arreglo2d.shape

[40]: (3, 4)
```

filas columnas

Aliados:

Microsoft

idata Advanced analytics for business

Vigilada Mineducación

Arrays multidimensionales (2/2)

```
[42]: arreglo2d = np.arange(100).reshape(10,10)
      arreglo2d

[42]: array([[ 0,  1,  2,  3,  4,  5,  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, 33, 34, 35, 36, 37, 38, 39],
             [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
             [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
             [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
             [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
             [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
             [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[43]: arreglo2d[2:5,::2]

[43]: array([[20, 22, 24, 26, 28],
             [30, 32, 34, 36, 38],
             [40, 42, 44, 46, 48]])
```

¿Cómo opera este slicing)

Tips

Leer casos y practicar con códigos de:

1. Medium
2. TowardsDataScience
3. GitHub
4. Stack Overflow
5. Kaggle

Otros…

Material complemetario (DS)

- Lenguajes de programación en DS

- Python Data Science Handbook – Capítulo 2: Intro to Numpy

Contenido asincrónico

- Azure – Core solutions and Management tools
- Azure – General security and network security features

UNIVERSIDAD EAFIT

¡Gracias!

Aliados:

Microsoft

idata
Advanced analytics for business

Vigilada Mineducación