SPRINT 1 - MEETING 10 4

Programación avanzada: clases y objetos

Unirme La meeting comienza en

Lista de recursos

Notebook:

Profundiza

Glosario

Challenge

00:08:18

DS_Bitácora_10_Clases

"Hay sólo dos clases de lenguajes de programación: aquellos de los que la gente está siempre quejándose y aquellos que nadie usa". -Bjarne Stroustrup - Científico de la computación.

Python es lo que se denomina un lenguaje de programación orientado a objetos. No es el único: Java o C++ por ejemplo, también lo son. En esta bitácora conocerás qué significa esta categoría y cuáles son las características de sus principales componentes: Clases y Objetos.

Hay muchas formas de afrontar este tema y el grado de abstracción que podemos darle supone un abanico grande de posibilidades. Aquí optamos por una **perspectiva utilitarista:** vamos a priorizar los conceptos que agreguen más valor a los temas vistos en la carrera.

Sin embargo, es importante que puedas profundizar en este tema e incorporar otros enfoques. ¡Podrás encontrarlos entre los recursos complementarios de esta Bitácora! Tómate el tiempo necesario para digerir y reflexionar sobre esas ideas, ya que te permitirán entender con mayor profundidad algunos temas de Data Science y también sobre la programación en general.

Programación Estructurada y Programación Orientada a Objetos La Programación Estructurada o Procedimental y la Programación Orientada a

Objetos son los dos paradigmas de programación más comunes y utilizados. Ambos conviven entre sí y no existe un estándar aceptado a la hora de definir estos términos. Para empezar, estas son algunas diferencias:

Programación estructurada:

- Es el paradigma con el que vienes trabajando.
- Hace foco en un flujo de código lineal.
- Generalmente defines línea a línea el código, las variables y las acciones que usas, solamente interrumpidas por toma de decisiones con estructuras if/else

Programación orientada a objetos:

- Se complementa con la programación estructurada.
- Incluye la noción de **Objetos**, una suerte de "paquetes" en los que se agrupan las funciones y variables.
- Imagina que quieres definir en un programa que estás escribiendo ciertas

• Incluye la noción de **Clases**, el molde que define a los Objetos.

características de una persona. Podrías crear una variable con su nombre, otra con su edad, otra con su altura, etc. Hasta ahora, tendrías que definir una variable por cada una de esas características, y seguramente las agruparías en una lista o diccionario. El código hasta el momento se vería de la siguiente forma:

```
print(caracteristicas carlos)
{'Carlos': 'Carlos', 34: 34, 1.64: 1.64}
```

Luego, tal vez te interese definir funciones que operen sobre las características

In [4]: caracteristicas_carlos = {nombre : "Carlos", edad: 34, altura : 1.64}

de esas personas. Por ejemplo, si tuvieras la función "cumplir años", se vería algo así:

```
def cumplir años():
    caracteristicas carlos[edad] = caracteristicas carlos[edad] + 1
    return caracteristicas_carlos[edad]
```

Si tuvieras que hacer esto para muchas personas y funciones, el código empezaría a convertirse en un Frankenstein difícil de manejar y poco prolijo. ¿No lo crees?

En general, esta forma de programar y sobretodo de entender el código se lo conoce como Programación Estructurada o Procedimental.

Con un paradigma de programación orientada objetos, crearíamos la Clase

Persona; esta Clase contendría atributos (edad, nombre, etc) y funciones (cumplir_años(), dormir(), etc). A su vez, esta Clase crea Objetos persona. Se pueden crear miles de 'Objeto persona' y todos ellos tendrán los atributos y funciones de la Clase Persona. No te frustres si te has perdido, jahora veremos en detalle cómo funcionan las Clases y Objetos!

Clases y Objetos En Data Science utilizamos —y, de hecho, ya venimos utilizando— muchas

Clases y Objetos. Los DataFrames de librería Pandas son uno de ellos. ¡Fíjate la ventaja que tiene! Podemos trabajar con muchos Datasets distintos y usar las mismas funciones en todos ellos. Esto permite calcular, por ejemplo, promedios sobre los datos usando la misma función. Las Clases son como el molde o la plantilla que va a crear los Objetos y a

trabajamos creando Objetos y las Clases que importamos de las Librerías. En otros lenguajes —como Java o códigos más complejos en Python— es más recurrente crear nuestras propias Clases. Un ejemplo que ya conoces es la **Clase Data Frame**, de la librería Pandas. Esta

determinar sus características y funciones. Los/as Data Scientists normalmente

".head()" o ".length()." Cada Data Frame que creamos es un Objeto de la Clase Data Frame. A ese acto se le llama instanciar un Objeto. Por ejemplo: In [4]: # Creamos el DataFrame

crea Objetos Data Frame, que tienen atributos como "shape" y métodos como

data_pandas = pd.DataFrame(data_dic) data pandas En este caso se está creado el Objeto data pandas de la Clase Data Frame, es decir que jestamos instanciando el objeto datapandas! Este Objeto se está creando en

import numpy as np

#creamos un array a partir de una lista

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

Aquí se instanciaría el Objeto array_list de la Clase Array de la librería Numpy:

base a la información del diccionario data_dic.

```
array_list = np.array(lista)
                  type (array list)
                 numpy.ndarray
Métodos: son las funciones que vienen definidas en las Clases y suelen operar
sobre los atributos.
```

Atributos: estos son las "características" que van a tener los objetos. Al igual que las funciones, pueden tener atributos por default. Es decir, cuando

Imagina las Clases como si fueran una fábrica. Una fábrica que crea ciertos objetos, como por ejemplo, pelotas. La Clase Pelota podría tener atributos como color, tamaño y presión de aire y métodos cómo inflar() o desinflar().

instanciamos un objeto, puede no ser necesario definir algunos atributos.

¿Por qué nos sirve saber de Clases y **Objetos?**

En primer lugar, ¡para comprender el código! A partir del siguiente encuentro empezarás a trabajar con Scikit-Learn, la

librería estándar para Aprendizaje Automático en Python. Verás que saber POO nos ayuda a utilizarla mejor, comprender la documentación y el código.

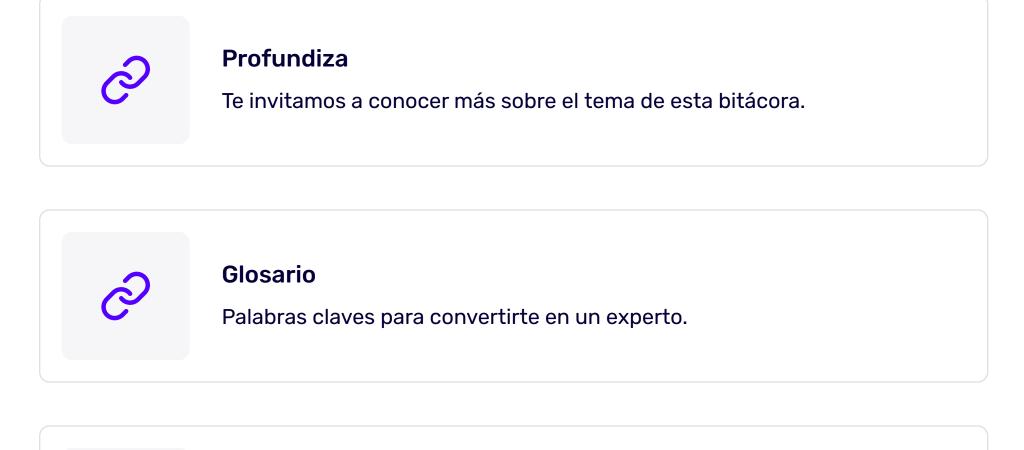
Empieza a trabajar en el notebook

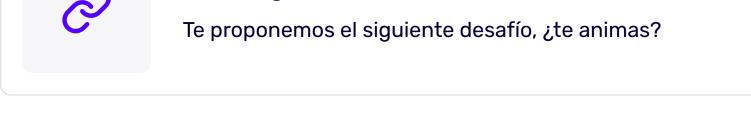
Trabaja en la sección 1 del notebook. La sección 2 la trabajaremos en clase.



¡Prepárate para el próximo encuentro!

Notebook: DS_Bitácora_10_Clases





Challenge