ORIENTACIÓN A OBJETOS

1º Entrega: Conceptos básicos

Entre los paradigmas de programación soportados por python destacan el procedural, el funcional y la orientación a objetos.

¿Que es la Orientación a objetos?

La orientación a objetos se basa en la definición e interacción de una unidades mínimas de código, llamadas objetos, para el diseño y desarrollo de aplicaciones.

Ventajas:

- re usabilidad del código
- la modularidad
- la facilidad para modelar componentes del mundo real.

Clases Y Objetos

El la POO (**programación orientada a objetos**), un objeto es un componente que tiene un rol especifico y que puede interactuar con otros.

¿Para que se usan los objetos?

Para establecer una equivalencia entre un objeto del mundo real con un componente de software.

¿Que se obtiene al implementar POO?

El modelado para la representación y resolución de problemas del mundo real a través de la programación, es mas sencillo e intuitivo.

Representación:
Todos los objetos presentan dos componentes principales:
1- un conjunto de características y propiedades
2- un comportamiento determinado

Ejemplo:
Pensemos en una moto
Esta tiene características como color, marca y modelo.

Esta tiene como comportamiento: frenar, acelerar, girar.

Si queremos hacer una representación con POO:
1- pasaremos las características del objeto en cuestión como sus **ATRIBUTOS**
2- pararemos el comportamiento del objeto como sus operaciones (**MÉTODOS**)

```
class Moto ():
  def __init__(self, marca, modelo, color):
        self.marca=marca
        self.modelo=modelo
         self.color=color

  def get_marca(self):
        marca="Nueva marca"
        print(self.marca)
```

nota: este código no hace nada todavía, es parte de la definición!

Diferencia Entre Clase Y Objeto:

Decimos que "un objeto es una instancia de una clase"

¿Que es una instancia?

El proceso es así:

1- definimos los atributos y operaciones de un objeto a través de una clase.
2- la instancia es un mecanismo que nos permite crear un objeto que pertenece a una determinada clase.

¿Que ganamos con esto?
De esta manera podemos tener diferentes objetos que pertenezcan a la misma clase.

Ej:
con la clase moto que creamos anteriormente, podríamos instanciar los objetos siguientes:

moto1(Honda,cbr190, azul)
moto2(Suzuki, katana2022, roja)
moto3(Zanella,patagonia150, negro)
...
moto n()

Ahora aplicaremos esto….

1- Para crear una clase se usa la sentencia **class** seguida de un nombre.
Class Moto

2- Por convención, el nombre de la clase empieza en mayúscula.
3- Cambien suele estar contenida en un fichero del mismo nombre de la clase, pero todo en minúscula.

Ejemplo 1: La clase mas sencilla en python

La clase mas sencilla que podemos crear en python es la siguiente:

```
class First:
  pass
```

Luego de definida la clase, creamos un objeto utilizando el siguiente código:

```
a=First()
```

Constructor De La Clase

Cuando creamos una clase podemos hacer uso de un método especial que se llama constructor.

¿Que hace el constructor?
El es método que se encargue de realizar tareas de inicialización.

- El métodos siempre es ejecutado cuando se crea un objeto.

¿Como Se Declara Un Constructor En Python?

Python dispone de dos métodos involucrados en la creación de un objeto.
- El primero se llama __**init**__
- El segundo __**new**__

nota:
Al ser crear un objeto primero se invoca a __new__ y después se invoca a __init__ que el que habitual mente se emplea para ejecutar todas las operaciones iniciales que necesitemos

Importante!
En la practica solo utilizaremos __init__ como nuestro método constructor.

Ejemplo 2: clase con constructor

```
class First:
    def __init__(self):
          print("constructor ejecutado...")
```

Ahora creamos un objeto

```
a=First()
constructor ejecutado…
```

nota: por ahora no te preocupes por las palabras que no se han explicado..

Actividad n.º 1: en un archivo, copia el código anterior y ejecutalo
Actividad n.º 2: en un archivo**,** copia el código de la clase moto y luego genera algunos objetos.

Self

Han notado que tanto el constructor como el método de la clase Moto tienen en su código la palabra self?

```
class Moto ():
    def __init__(self, marca, modelo, color):
          self.marca=marca
          self.modelo=modelo
           self.color=color

    def get_marca(self):
          marca="Nueva marca"
          print(self.marca)
```

Self contiene una referencia al objeto que ejecuta el método y, por lo tanto, puede ser utilizada para acceder al espacio de nombres del objeto.

Debemos tener en cuenta que cada objeto contiene su propio espacio de nombres..
Entonces:
Self es un parámetro que contiene nuestro método constructor, y ademas todo método de instancia deberá contenerlo como su primer parámetro.

Ej: def __init__(self, marca, modelo, color):

nota:
En Python, self no es una palabra clave y podemos emplear cualquier nombre para esta referencia.
-Esto no es aconsejable, ya que, por convención, esta ampliamente aceptado y arraigado el nombre de self para este propósito.
-Por otro lado, cuando se invoca a un método, no es necesario incluir la mencionada referencia.

Ejemplo 3:
Modificamos la clase First anterior añadiendo el siguiente método:

```
class First:
    def __init__ (self):
            print("constructor ejecutado")

    def nuevo (self):
            print ("soy nuevo")

Ahora podemos crear un objeto de la clase First
 f= First()
constructor ejecutado

Para ejecutar su método llamado “nuevo”
f.nuevo()

salida:
>> soy nuevo….
```