UNIVERSIDAD TECNOLOGICA NACIONAL

Métodos mágicos y clases abstractas

Programación 2

Ing. Tomás Ponce

Metodos magicos

- Los métodos mágicos son aquellos métodos que comienzan con dos guión bajos y terminan con dos guión bajos
- Un ejemplo claro de esto es el constructor de cualquier clase : __init__

Metodos magicos

- El método mágico más utilizado en las clases de python es el constructor. Un método que no tiene una cantidad fija de parámetros, ya que estos dependen de las propiedades que tenga el objeto
- El segundo método más utilizado sea __str__, con el que se crear una representación del objeto que tenga significado para las personas.

Ejemplo modificación de str

```
1.  class Vector():
2.      def __init__(self, data):
3.          self._data = data
4.
5.
6.      def __str__(self):
7.          return f"The values are: {self._data}"
8.
9.
10. v = Vector([1,2])
11. print(v)
```

```
The values are: [1, 2]
```

Clases abstractas

-Una clase abstracta tiene como objetivo definir comportamiento común que múltiples subclases pueden heredar sin tener que implementar toda la clase abstracta.

-Las clases abstractas NO pueden ser instanciadas.

-En resumidas cuentas, me obliga a que las clases que heredan de la clase padre implementen todos los metodos de la misma. Las subclases tienen que implementar todos los métodos abstractos, en el caso de que falta alguno de ellos Python no permitirá instancias tampoco la clase hija.

Ejemplo de clase abstracta : Error

```
class Animal(ABC):
    @abstractmethod
    def mover(self):
        pass

    @abstractmethod
    def comer(self):
        print('Animal come')


class Gato(Animal):
    def mover(self):
        print('Mover gato')

g = Gato() # Error
```

Ejemplo de clase abstracta: forma correcta

```
1.  class Animal(ABC):
2.      @abstractmethod
3.      def mover(self):
4.          pass
5.
6.      @abstractmethod
7.      def comer(self):
8.          print('Animal come')
9.
10.
11. class Gato(Animal):
12.     def mover(self):
13.         print('Mover gato')
14.
15.
16.     def comer(self):
17.         super().comer()
18.         print('Gato come')
19.
20.
21. g = Gato()
22. g.mover()
23. g.comer()
```