Python - Funciones y Clases

Autor:
Juan David Argüello Plata - Ingeniero Mecánico

Profesor tutor:
Jairo René Martínez Morales - Químico PhD

CENIVAM
Universidad Industrial de Santander

Introducción

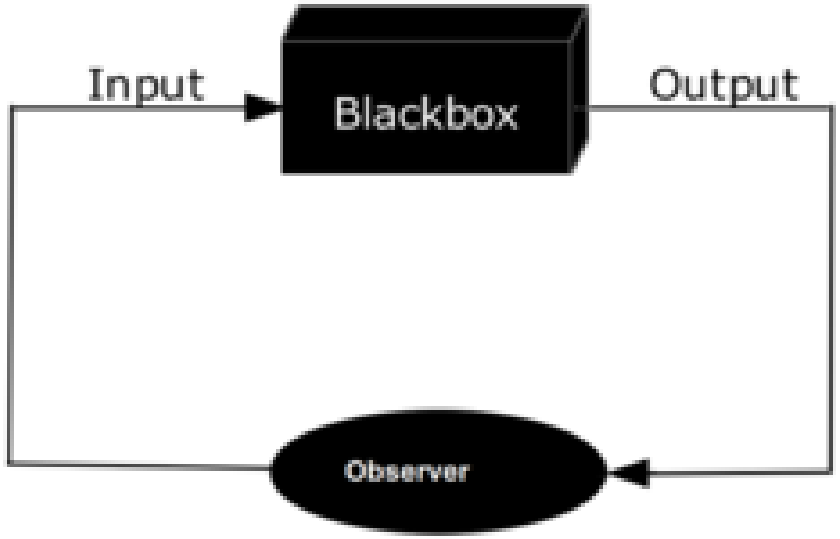
Objetivos

Los objetivos del día de hoy...

- Conocer la utilidad de las funciones y su sintaxis.
- Clases y herencia.
- Librerías.

Funciones

Las funciones son *procedimientos* que toman entradas y producen salidas (como una "caja negra"). Los podemos utilizar una y otra vez a lo largo del código de programación.



Las principales ventajas de las funciones son:

- Mejora la lectura del código.
- Disminuye la escritura de líneas.
- Organiza los algoritmos.

Funciones - sintaxis

La **sintaxis** de una función es la siguiente:

```
def <nombre func> (<entradas>):
        <algoritmo>
        return salidas  (opcional)
```

Por ejemplo: una función de suma de dos números sería algo como:

```
def suma (a,b):
        return a+b

if __name__ == '__main__':
        print(suma(2,3))
```

Funciones - Ejercicio

Desarrolla los siguientes ejercicios:

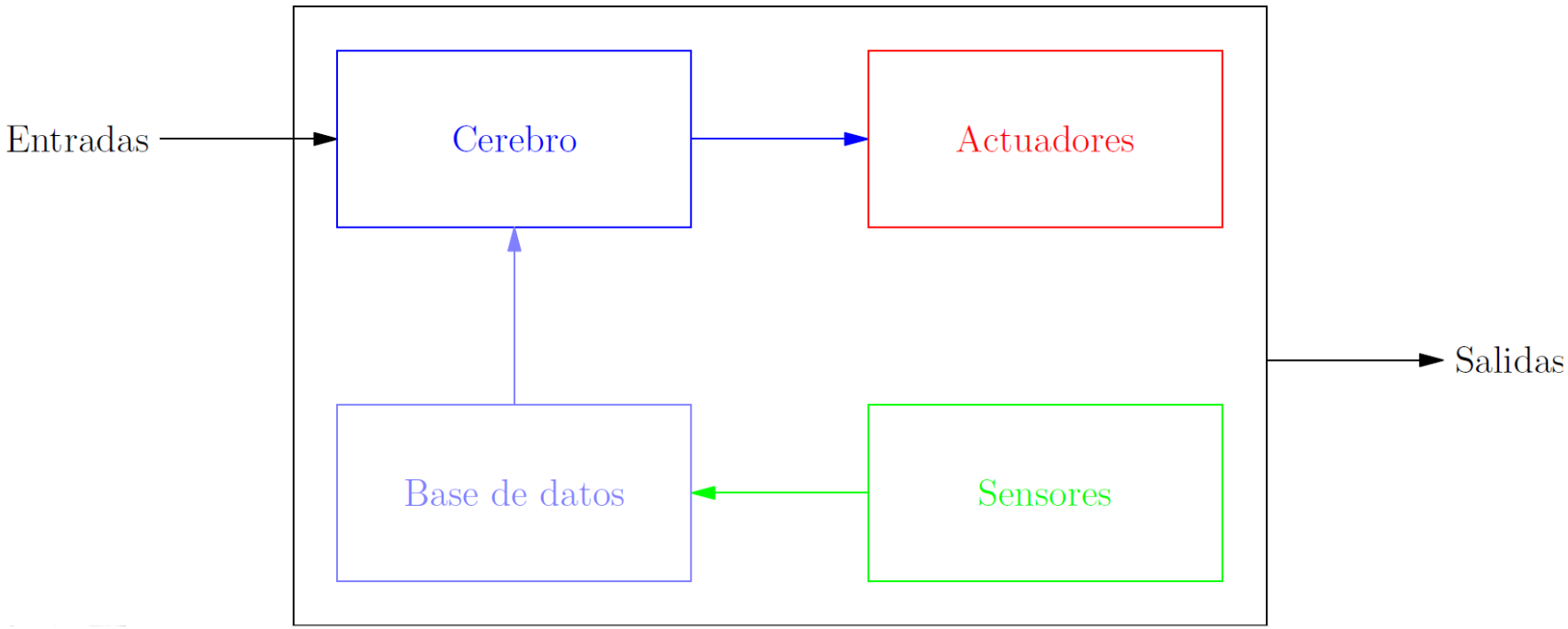
- Define una función que identifique si un número es par o impar.
- Las siguientes personas desean ingresar a un bar:

Persona	**Edad**
Juan	26
Sebastián	14
Ramiro	22
María	16
Fabián	20
Diego	17

  Imprime una lista de quiénes pueden entrar.
- *¿La revancha de Fibonacci?* Desarrolla un algoritmo que permita obtener la lista de números de Fibonacci en un rango aleatorio...

Clases I

Las clases son *estructuras* o algoritmos que permiten dividir un concepto complejo en conceptos simples que pueden ser tanto generales como detallados. Para el caso del semillero, podríamos pensar en algo como esto:



Clases II

Las **entradas** del sistema son:

- Tipo de cultivo.
- Ubicación de plantas.
- Fecha de desarrollo.

Las **salidas** son:

- Cantidad de producto.
- Datos recolectados durante el desarrollo.

Clases - sintaxis

La sintaxis más básica es la siguiente:

```
class <Nombre de la clase>:
    ...
```

La sintaxis más completa:

```
class <Nombre de la clase> (<Clase padre>):
  <Propiedades generales>
  def __init__(self, <entradas>):
    (...)
    super().__init__(<entr. padre>)      (OPCIONAL)

  def <nombre func. interna> (self, <entradas>):
    (...)
    return salidas          (OPCIONAL)

  def __call__(self):
      return <resultados>
```

Clases - sintaxis

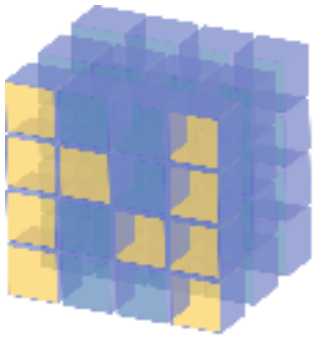
1. "def __init__ (self, <entradas>)": corresponde al algoritmo que se inicializará el momento en que se llame a una clase (una clase es llamada de la misma manera que una función: "ClassName()").
2. "super().__init__(<entr. padre>)": llama la función *def __init__* de la clase **padre**.
3. "def <nombre func. interna> (self, ...)": se conocen como *métodos* de la clase. Se pueden llamar por fuera de la misma. Por ejemplo: tenemos una clase llamada "Alumno", y uno de sus métodos es "horario". Se llamaría: Alumno.horario(<entradas del método>).
4. "def __call__ (self)": permite obtener resultados de las clases. Cuando las clases son llamadas desde *afuera*, se convierten en **objetos**. NO podemos obtener información útil de objetos a menos que los "llamemos". Se invoca la función *call* poniendo un paréntesis al final; por ejemplo: Juan = Alumno() $\rightarrow$ nos da un objeto. pero Juan(), nos da los resultados de la clase.

Librerías I

Python es reconocido como el lenguaje por defecto de la *ciencia de datos*. En parte se debe a las librerías disponibles.

Instalar una librería...

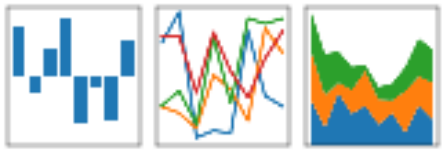
pip install <nombre de la libreria>



NumPy

SciPy

pandas



matplotlib

"Python's Scientific Ecosystem"