# CIENCIATURA EN CIENCIA DE DATOS

## PROGRAMACIÓN AVANZADA (189)

Lic. Felipe Morales Clase N. 4. Unidad I. Metaclases y decoradores.







## **TEMARIO**

- T0: **Revisión** de Clase anterior
  - Revisión de dudas
  - Análisis de resoluciones de estudiantes
  - Conclusiones
- T1: Metaclases
- T2: Decoradores
- T3: Codificación en Python 3
- Temas relacionados
- Links



## T0: Revisión de **Clase Anterior**:

Revisión de dudas

Análisis de resoluciones de estudiantes

Conclusiones



## T1: Metaclases

## **Explicación:**

Vamos a ver cómo podemos usar una **metaclass** para crear una clase, en vez de usar la sentencia **class** como lo vimos anteriormente. La metaclass predeterminada es **type**. Por lo tanto, podemos usar la siguiente sentencia para crear una nueva clase:

```
new_class = type('myClass',(object, ),{'a' : True})
```

Es similar a:

class myClass(object):

a = True

Nota: En ambos casos la palabra object se puede omitir.



## T1: Metaclases

## Sintaxis para type():

Esta función devuelve el tipo de clase del argumento(objeto) pasado como parámetro. La función **type**() se utiliza principalmente con fines de depuración. Se pueden pasar dos tipos diferentes de argumentos a la función **type**(), uno y tres argumentos. Si **type**(obj) se pasa un solo argumento, devuelve el tipo de objeto dado. Si **type**(name, bases, dict)se pasan tres argumentos, devuelve un nuevo tipo de objeto. Se puede utilizar de dos formas, como se muestra a continuación:

type(object)

type(namr, bases, dict)

#### Parámetros: tipo (objeto)

• *Objeto*: Este es un parámetro obligatorio. Si esto se hace solo un parámetro pasado a type (), no le devuelva el tipo de parámetros.

#### Parámetros: tipo (nombre, bases, dictado)

- name: nombre de la clase.
- **bases**: (opcional). Este es un parámetro opcional y es la clase de la cual hereda comportamiento.
- dict: (opcional). Este es un parámetro opcional y es un nombre de espacio con la definición de clase.



#### T1: Metaclases

## Ejemplos con método (utilizando dict):

```
def dice(self):
    print('Guauuuuu')

Animal = type('Perro', (), {'nombre':'a definir'})

Perro = type('Perro', (Animal, ), dict(cantidad patas=4, dice=dice))

p = Perro()

print(p.nombre)  # Imprime a definir

print(p.cantidad patas)  # Imprime 4

p.dice()  # Imprime Guauuuuuu
```

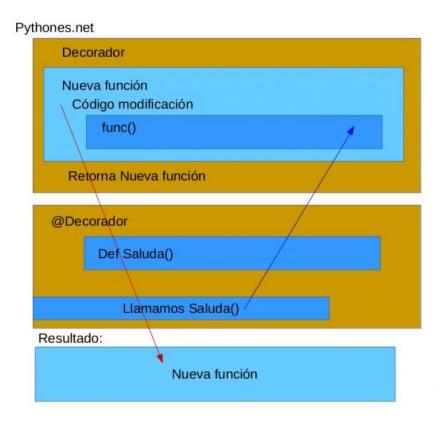
## Clase 4. Unidad I.



## **T2: Decoradores**

## Definición:

Un **decorador** es una función que recibe como parámetro una función y que retorna otra función. Esto lo utilizamos para alterar el comportamiento de una función sin tener que modificar su código. Suena un poco complicado pero con el uso de ejemplos se puede comprender perfectamente.



- 1- La flecha azul simboliza lo que sucede cuando llamamos a una función definida dentro del decorador.
- 2 -La flecha roja simboliza lo que se obtiene como resultado



Ahora tambien hago diagramas!



## **T2: Decoradores (decorators)**

## **Ejemplo 1:**

```
def my primer decorador (function):
  def funcion de retorno():
      print('Cargando...')
       function()
      print('Proceso finalizado')
   return funcion de retorno
@my primer decorador
def funcion de entrada():
  print('Hola mundo')
            == " main
    name
   funcion de entrada()
```





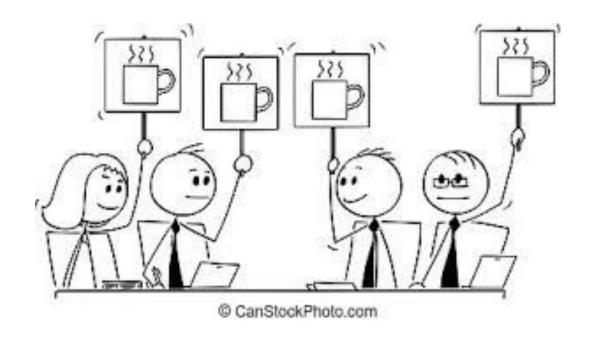
## **T2: Decoradores (decorators)**

## Ejemplo 2 (decorador con argumentos):

```
def calcular area triangulo (function):
   def funcion de retorno (*args, **kwargs):
      res = function(*args, **kwargs)
      return res / 2
   return funcion de retorno
@calcular area triangulo
def calcular area(base, altura):
  return base * altura
           == " main ":
    name
   print(calcular area(4, 10))
```



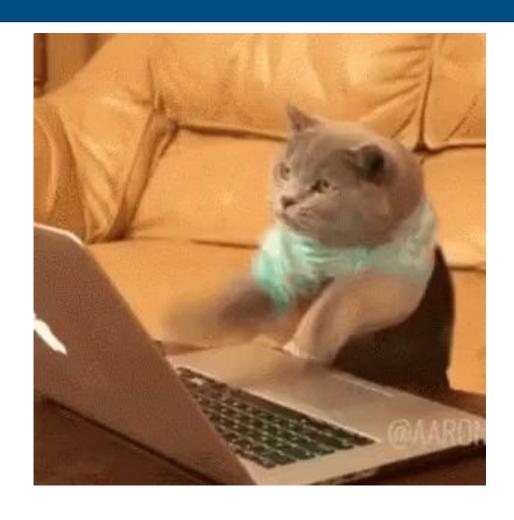
# **BREAK**





¿CHALLENGE?





¿Seguimos trabajando?

## Clase 4. Unidad I.



## T3: Codificación en python 3

## Ejercicio 1:

- a) Rehaga al menos 3 clases con el constructor type
- b) ¿Cómo se deben definir los métodos con el constructor type?

## **Ejercicio 2:**

a) Defina una función de nombre **mensaje**, la cual debe imprimir "Esto es Programación Avanzada", la cual a través de un decorator poder imprimir el texto "Hola" y "Chau", La salida debe ser de la forma:

Hola

Esto es Programación Avanzada.

Chau

- a) Programar un decorador que en caso de querer dividir por 0 emita un mensaje por pantalla. Integrar en un ejemplo
- b) Programar un decorador para que imprima la fecha y hora. Integrar en un ejemplo.
- c) Investigar: i) Cómo invocar a 2 decoradores a la vez?
  - ii) Cómo invocar a un decorador que está programado en otro archivo?

## Unidad I. Links



## Links interesantes:

## Aprenda a pensar como un programado con Python

https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf

#### Documentación oficial:

https://www.python.org/doc/

#### **Curso de Python desde 0 (pildorasinformaticas)**

https://www.youtube.com/playlist?list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS

#### **Otros links:**

https://ellibrodepython.com/polimorfismo-en-programacion

https://parzibyte.me/blog/2019/06/30/clases-constructores-python-poo/





¡Muchas gracias por tu atención!