**INSTITUTO SUPERIOR TECNOLÓGICO DEL SUR**

**CARRERA**

**DISEÑO Y PROGRAMACIÓN WEB**

**LENGUAJE WEB 2**

**“Decoradores en Python”**

**PROFESOR(A) : Amado Cerpa Juan Andrés**

**ALUMNO : Vilca Apaza Christian**

**SEMESTRE : V**

4/11/2020

Índice

[Decoradores en python 3](#_Toc55424353)

[Creando un decorador en python 3](#_Toc55424354)

[Uso de \*args 4](#_Toc55424355)

[Uso de \*\*kwargs 4](#_Toc55424356)

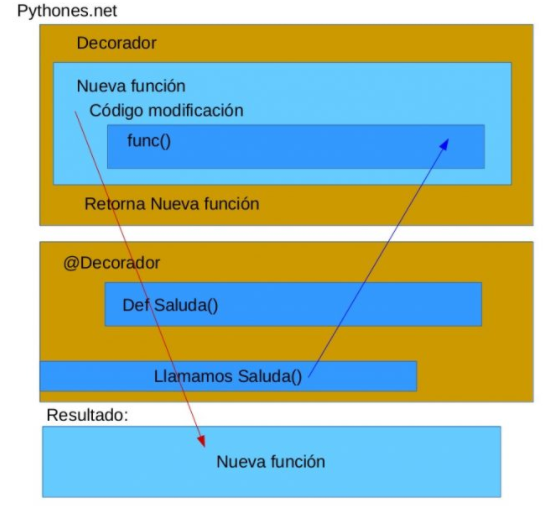
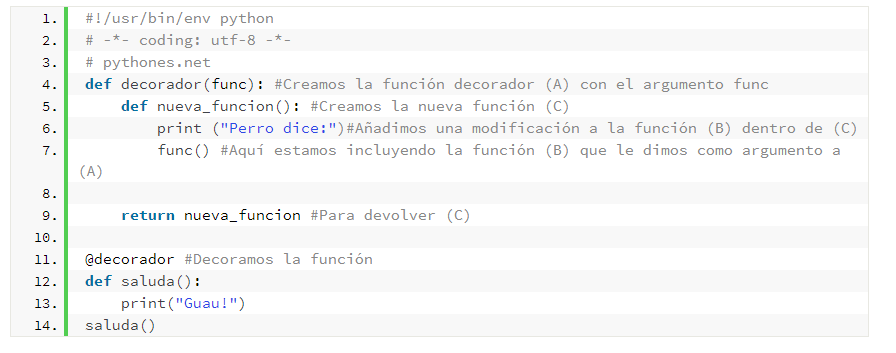
[Usando \*args y \*\*kwargs para llamar a una función 5](#_Toc55424357)

[Bibliografía 6](#_Toc55424358)

# Decoradores en python

Un decorador es una función que recibe como argumento a otra función para devolver otra función.

# Creando un decorador en python



# Uso de \*args

El principal uso de \*args y \*\*kwargs es en la definición de funciones. Ambos permiten pasar un número variable de argumentos a una función, por lo que si quieres definir una función cuyo número de parámetros de entrada puede ser variable, considera el uso de \*args o \*\*kwargs como una opción. De hecho, el nombre de args viene de argumentos, que es como se denominan en programación a los parámetros de entrada de una función.

A continuación, te mostramos un ejemplo para que veas el uso de \*args:

**def** test\_var\_args(f\_arg, \*argv):

print("primer argumento normal:", f\_arg)

**for** arg **in** argv:

print("argumentos de \*argv:", arg)

test\_var\_args('python', 'foo', 'bar')

Y la salida que produce el código anterior al llamarlo con 3 parámetros es la siguiente:

primer argumento normal: python

argumentos de \*argv: foo

argumentos de \*argv: bar

# Uso de \*\*kwargs

\*\*kwargs permite pasar argumentos de longitud variable asociados con un nombre o **key** a una función. Deberías usar \*\*kwargs si quieres manejar argumentos con nombre como entrada a una función. Aquí tienes un ejemplo de su uso.

**def** saludame(\*\*kwargs):

**for** key, value **in** kwargs.items():

print("*{0}* = *{1}*".format(key, value))

>>> saludame(nombre="Covadonga")

nombre = Covadonga

Es decir, dentro de la función no solo tenemos acceso a la variable como con \*args, sino que también tenemos acceso a un nombre o key asociado. A continuación, veremos cómo se puede usar \*args y \*\*kwargs para llamar a una función con una lista o diccionario como argumentos.

# Usando \*args y \*\*kwargs para llamar a una función

Ahora veremos cómo se puede llamar a una función usando \*args y \*\*kwargs. Consideremos la siguiente.

**def** test\_args\_kwargs(arg1, arg2, arg3):

print("arg1:", arg1)

print("arg2:", arg2)

print("arg3:", arg3)

Ahora puedes usar \*args o \*\*kwargs para pasarle argumentos a la función. Se puede hacer de la siguiente manera:

*# Primero con \*args*

>>> args = ("dos", 3, 5)

>>> test\_args\_kwargs(\*args)

arg1: dos

arg2: 3

arg3: 5

*# Ahora con \*\*kwargs:*

>>> kwargs = {"arg3": 3, "arg2": "dos", "arg1": 5}

>>> test\_args\_kwargs(\*\*kwargs)

arg1: 5

arg2: dos

arg3: 3

Por último, si quieres usar los tres tipos de argumentos de entrada a una función: normales, \*args y \*\*kwargs, deberás hacerlo en el siguiente orden.

funcion(fargs, \*args, \*\*kwargs)

# Bibliografía

<https://python-intermedio.readthedocs.io/es/latest/args_and_kwargs.html>

<https://recursospython.com/guias-y-manuales/decoradores/>

<https://pythones.net/decoradores-en-python-oop/>