**PYTHON**

**DECORADORES EN PYTHON**

**DEFINICIÓN:**

**Un decorador en Python es cualquier objeto invocable y que es usado para modificar una función o una clase.**

**DECORADORES DE FUNCIONES**

**DECORADORES DE CLASES**

**En el siguiente ejemplo se muestra un decorador que recibe como entrada una función (callable object) y devuelve otra función (callable object) , las diferencia ahora radica en la función que se devuelve contiene una versión modificada de la función de entrada.**

**DECORADOR**

FUNC MODIFICADA

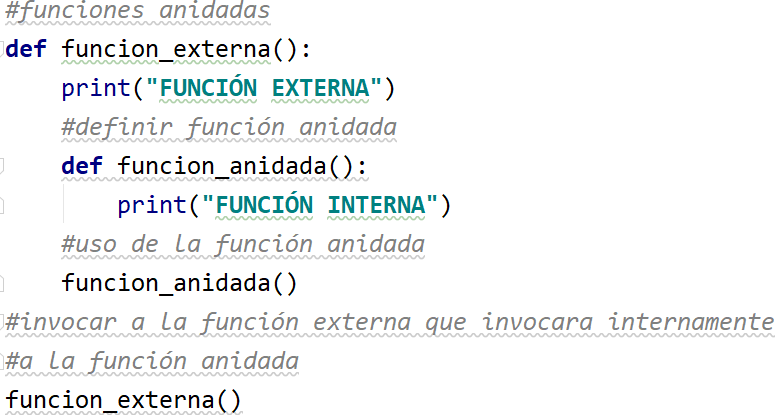
FUNC

**Con el fin de entender el concepto de un decorador, a continuación, se tratará el tópico de funciones anidadas.**

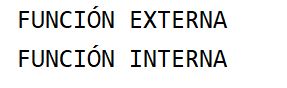
**Las funciones anidadas permiten definir funciones dentro de una función, este concepto es nuevo para los programadores en C , C++ .**

**FUNCIONES ANIDADAS:**

**CODIGO:**



**RESULTADO:**



**NOTA:**

**La función anidada (interna) solo puede ser invocada dentro de la función externa, es decir no se puede invocar externamente a la función anidada.**

**La función externa se puede invocar como cualquier función en Python, puede recibir parámetros, devolver parámetros etc.**

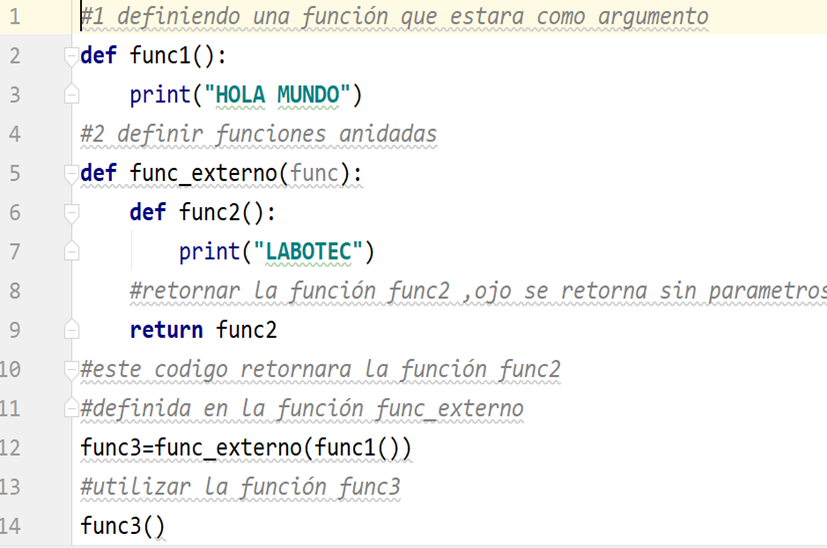
**FUNCIONES COMO PARAMETROS DE UN FUNCIÓN**

**En Python podemos crear funciones que tengan como argumento una función.**

**FUNCIONES QUE RETORNAN FUNCIONES:**

**Además de tener la posibilidad de pasarle como argumento una función a otra función, también se puede retornar funciones.**

**CODIGO:**

****

**RESULTADO:**

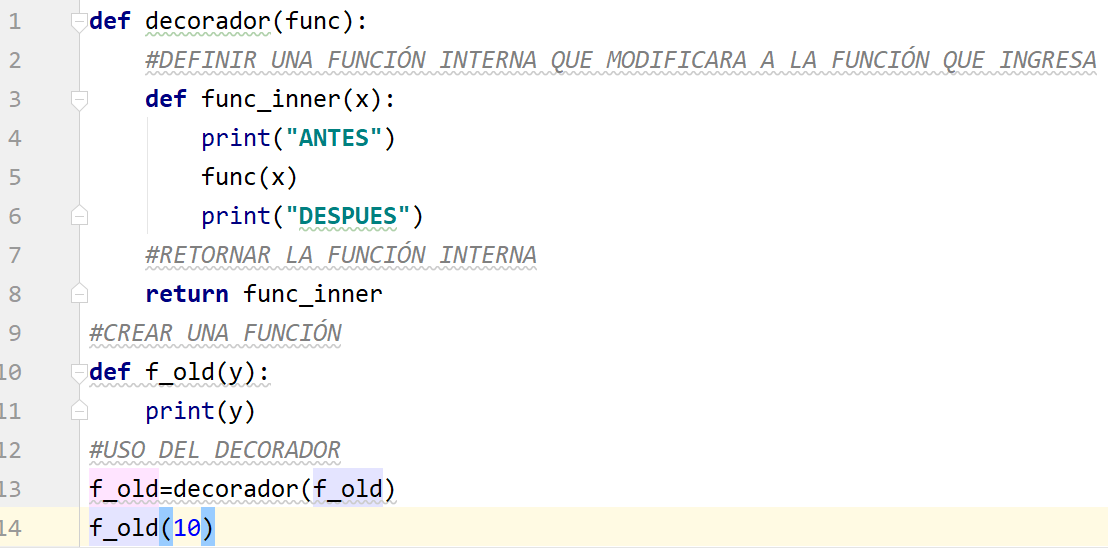


**DECORADORES:**

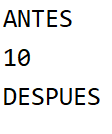
**Como definimos anteriormente, un decorador modifica una función o clase. Los requisitos para entender es saber usar las funciones anidadas.**

**PRIMERA FORMA DE UTILIZAR UN DECORADOR**

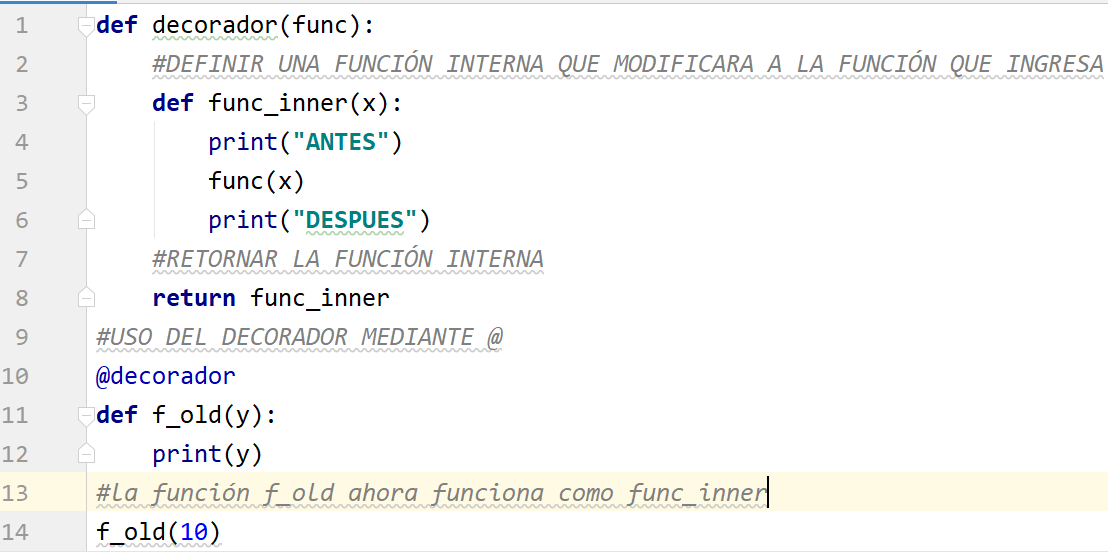
**Código:**



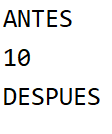
**Resultado:**



**SEGUNDA FORMA DE UTILIZAR UN DECORADOR**



**RESULTADO:**



**Los decorados los podemos encontrar en herramientas para el diseño de páginas web (flask, django), inteligencia artificial (tensorflow, keras , pytorch entre otras).**

**MODULO:**

**Archivo de Python con extensión .py que contiene un conjunto de clases, funciones, constantes que serán reutilizadas por otro archivo de Python con el fin de reducir la cantidad de líneas de código.**

**PAQUETE**

**Es un directorio que contiene sub-paquetes y módulos. Los paquetes en Python tiene que tener un archivo \_\_init\_\_.py y tiene una organización del tipo árbol (tree).**

**REPOSITORIO PYPI**

**Los módulos y paquetes que se requieran usar y no vienen instalados por defecto en Python , son instalados de un repositorio conocido como PYPI , existen otros repositorios para instalar paquetes o módulos , pero PYPI es por excelencia el lugar perfecto para poder descargar los módulos y paquetes que uno requiere para un proyecto en especifico .**

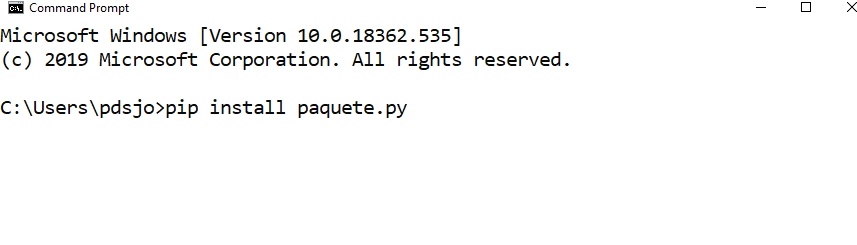
**¿Qué usar para instalar nuevos módulos o paquetes?**

**Para poder instalar nuevos módulos en Python podemos utilizar el Package Installer conocido como pip.**

**¿De dónde se descargan los módulos o paquetes?**

**El instalador de paquetes pip descarga e instala los módulos desde el repositorio PYPI , es necesario tener conexión a internet .**

**Ejemplo de como instalar un paquete en el terminal usando pip**



**NOTA: tener en cuenta que para poder utilizarlo de esa forma, se tiene que agregar PYTHO al PATH que representa el entorno de variables (ENVIRONMENT VARIABLES) de su sistema, con esto agregado se puede ejecutar python desde el terminal usando python e instalar paquetes utilizando el pip.**