**FUNDAMENTOS DE PROGRAMACIÓN – PREGUNTAS RETO 4**

**Mariana Duque Rúa**

**C.C. 1214747015**

**mariana.duque3@udea.edu.co**

* **¿Pueden existir funciones dentro de funciones?**

Sí, en Python es posible definir funciones dentro de otras funciones. Estas se conocen como funciones anidadas. Las funciones anidadas tienen acceso a las variables locales de la función en la que están definidas. Por ejemplo:

**def funcion\_padre():**

**def funcion\_hija():**

**print("Esta es la función hija.")**

**print("Esta es la función padre.")**

**funcion\_hija() # Llamada a la función hija desde la función padre**

**funcion\_padre()**

* **¿Se puede enviar como parámetro de una función otra función con retorno?**

Sí, en Python es posible pasar funciones como parámetros a otras funciones. Esto se llama funciones de orden superior o funciones de primera clase. Por ejemplo:

**def funcion\_a():**

**return 5**

**def funcion\_b(func):**

**resultado = func()**

**return resultado**

**resultado\_final = funcion\_b(funcion\_a)**

**print(resultado\_final) # Esto imprimirá 5**

* **¿Qué sucede si dentro de una función se hace un llamado a sí misma?**

Cuando una función se llama a sí misma dentro de su propio cuerpo, se llama recursión. Esto puede llevar a un comportamiento recursivo, donde la función se llama una y otra vez hasta que se cumple una condición de terminación. Es necesario tener una condición de terminación (caso base) para evitar una recursión infinita. Una función recursiva es, por ejemplo:

**def contar\_hasta\_cero(numero):**

**if numero <= 0:**

**print("¡He terminado!")**

**else:**

**print(numero)**

**contar\_hasta\_cero(numero - 1)**

**contar\_hasta\_cero(5)**

* **¿Una variable creada como global dentro de una función, ¿sigue existiendo fuera de la función?**

Sí, en Python, si se declara una variable como global dentro de una función, esa variable será accesible y modificable fuera de la función, en el ámbito global del programa. Por ejemplo:

**def definir\_variable\_global():**

**global variable\_global**

**variable\_global = 42**

**definir\_variable\_global()**

**print(variable\_global) # Esto imprimirá 42**

* **Después de indicar una variable en una función como global, ¿hay alguna manera de volver a usar la variable local?**

Sí, es posible volver a usar la variable local después de haberla declarado como global, pero es necesario tener en cuenta que cualquier modificación en la variable dentro de la función global también afectará a la variable local.

**Punto 1 – Reto 4**

La solución presentada en el Reto 3 incorpora elementos del paradigma funcional al utilizar funciones puras para llevar a cabo tareas específicas. No obstante, es importante destacar que existen oportunidades para una aplicación más completa de este paradigma.

Una de las áreas de mejora reside en la manipulación de variables globales, como *"datos\_cultivo"* y *"costos"*, dentro de algunas funciones. Esta práctica no sigue rigurosamente el principio de funciones puras, que se caracterizan por no tener efectos secundarios y no modificar directamente variables globales. En lugar de alterar las variables existentes, sería beneficioso crear nuevas variables o estructuras de datos con los cambios requeridos, lo que promovería la inmutabilidad de los datos y una mayor coherencia con el enfoque funcional.

Otro aspecto que podría optimizarse es la introducción de funciones de alto orden. Estas funciones, que pueden aceptar otras funciones como argumentos o devolver funciones como resultados, permitirían una mayor modularidad y abstracción en el código. La incorporación de funciones de alto orden facilitaría la reutilización de lógica y promovería un diseño más funcional y declarativo.

Además, la reducción de la dependencia de variables globales sería deseable. Esto se puede lograr mediante la transmisión de datos entre funciones a través de argumentos y el retorno de resultados en lugar de modificar directamente variables globales. Este enfoque contribuiría a la encapsulación de la lógica y a la creación de funciones más autónomas y cohesivas.

En resumen, si bien la solución planteada del Reto 3 incorpora elementos del paradigma funcional, se podría mejorar mediante la adopción de prácticas más estrictas de programación funcional, como la inmutabilidad de datos, el uso de funciones puras, la introducción de funciones de alto orden y la reducción de variables globales. Esto conduciría a un código más coherente con los principios del paradigma funcional y, en última instancia, a un código más modular y mantenible.