Unidad II: Programación Funcional

2.1. El tipo de datos

Python como varios otros lenguajes modernos como ser ruby, scala, etc., tiene influencia del paradigma funcional. No vamos acá a enseñar el paradigma funcional, pero sí sucede que varias de estas ideas están relacionadas con la manipulación de estructuras de datos contenedoras, como las listas, diccionarios, sets, etc.

Entonces en esta sección vamos a ver algunas ideas que, luego de entenderlas aplicadas sobre las colecciones, podremos utilizarlas luego para modelar cualquier problema. Y tendremos un conjunto de "herramientas" para pensar soluciones a problemas.

En general estas ideas llevan a combinar el paradigma estructurado y objetos con la simplicidad y legibilidad del paradigma funcional y declarativo.

2.2. Funciones

En general las funciones que veníamos viendo hasta el momento se denominan funciones de primer nivel. Porque existe la idea de funciones de orden superior. Que se refiere a funciones que:

- Reciben otra función como uno o varios parámetro/s.
- O bien retornan otra función como resultado.

Y claro, se llaman de orden superior porque operan sobre funciones. El dominio o la imagen de estas funciones son funciones.

Veamos un ejemplo. Empezamos por uno simple, para no decir bastante "pavo",

porque no es justamente para lo que uno usaría estas funciones realmente. Supongamos que tenemos una función que sirve para "saludar". Símplemente hace un print.

2.3. Intervalos

Funciones devuelven siempre el mismo valor

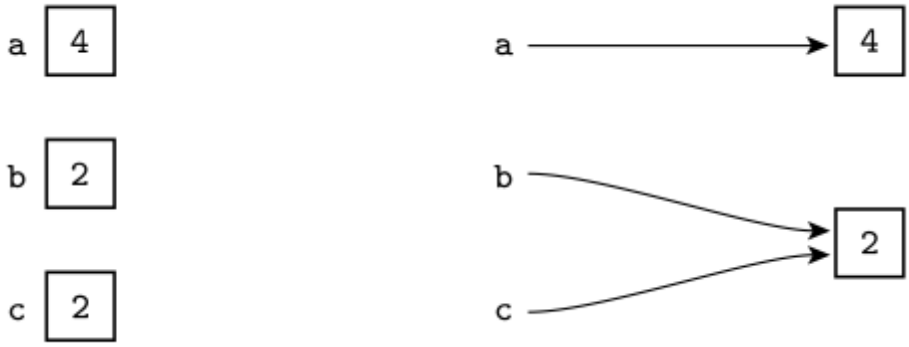
- Los lenguajes funcionales puros tienen la propiedad de transparencia referencial
- Como consecuencia, en programación funcional, una función siempre devuelve el mismo valor cuando se le llama con los mismos parámetros
- Las funciones no modifican ningún estado, no acceden a ninguna variable ni objeto global y modifican su valor

Diferencia entre declaración y modificación de variables

- En programación funcional pura una vez declarada una variable no se puede modificar su valor
- En algunos lenguajes de programación (como Scala) este concepto se refuerza definiendo la variable como inmutable (con la directiva `val`).
- En programación imperativa es habitual modificar el valor de una variable en distintos pasos de ejecución

2.4. Operadores

- En programación declarativa sólo existen valores, no hay referencias.
- La distinción entre valores y referencias es fundamental, sin embargo, en la programación imperativa.



Diferencia entre valor y referencia

- Cuando se realiza una asignación **de un valor** a una variable debemos considerar que estamos dando un nombre a un objeto matemático que no puede ser modificado o que estamos copiando el valor en la variable.

  Por ejemplo, en Java, los tipos de datos primitivos son valores. Las asignaciones valores de estos tipos a variables realizan copias de valores:

- En la variable `a` se copia el valor 4 y en las variables `b` y `c` se copia el valor `2`. No hay forma de modificar (*mutar*) esos valores. Podríamos cambiar las variables guardando en ella otros valores, pero los valores propiamente dichos son inmutables. En la última instrucción modificamos el valor de la variable `b`, pero el valor de la variable `c` sigue siendo 2.

  Los tipos de datos cuyos valores son inmutables y sus asignaciones tienen una semántica de copia reciben el nombre de **tipos de valor** (*value types* en inglés).

- Los **tipos de referencia** son tipos de datos mutables en los que la asignación funciona con semántica de referencia.

  Por ejemplo, cualquier objeto en Java tiene una semántica de referencia. Cuando asignamos un objeto a una variable, estamos guardando en la variable una referencia al objeto.

2.5. Aplicaciones de las listas

En Scheme todos los datos compuestos se construyen a partir de las parejas. En concreto las listas se definen de una forma recursiva muy elegante como

secuencias de parejas. Esta característica se remonta al origen del LISP en el que John McCarthy definió el concepto de *S-expression* e introdujo la notación del "." para definir una pareja.

2.6. Árboles

Clase de la asignatura Lenguajes y Paradigmas de Programación de Ingeniería Informática.

En el segundo vídeo se explican algunos algoritmos recursivos en el lenguaje de programación funcional Scheme que trabajan sobre árboles binarios. En concreto la obtención de una lista con sus elementos y la búsqueda en un árbol binario ordenado.

En este tercer vídeo se explican las operaciones de inserción en árboles binarios ordenados desde el punto de vista de la programación funcional. Se incluyen ejemplos y demostraciones con el lenguaje de programación Scheme.

En el útimo vídeo se explican dos versiones del algoritmo recursivo que obtiene la lista de elementos de un árbol genérico. Los algoritmos se realizan con el paradigma de programación funcional, utilizando en concreto el lenguaje de programación Scheme.