**Funciones:**

Para evitar duplicidad en el código (acortar y simplificar).

Lograr modularidad, de manera que nuestros programas “tengan sentido”.

Las funciones no siempre tienen argumentos, se utiliza la función main, es como un run(corre todos los módulos).

Las funciones soportan múltiples return statements

A los parámetros se les puede asignar valores por defecto.

Si no se especifica el tipo de datos de datos de salida que se quieres, sale una tupla.

Las tuplas se utilizan cuando no se quieren editar los datos.

Scoping = alcance.

**Diccionarios**

Es la última de las tres estructuras de datos que utiliza Python, las primeras son listas y tuplas. Son colecciones no secuenciales o no ordenadas. En las listas y tuplas el orden importan, mantienen el orden cuando se le pide a Python que la muestre. En los diccionarios no, la razón por la cual no existe este orden de datos tiene que ver con eficiencia computacional, y es que el computador encuentra una manera más eficiente cuando impone su propio orden y no cuando lo impone el usuario. Por ejemplo, en una base de datos con mil cédulas, no importa el orden en que me la muestra el computador. Y, la segunda característica, es que son asociaciones o mapeos entre llaves y valores. Es un mapeo donde el dominio se llama llave y donde los valores del rango se llaman así, valores. La sintaxis es con llaves { }, los corchetes [ ] son para las listas y los paréntesis para las tuplas.

Va así {llave1: valor1, llave2: valor2, llave3: valor3}

Los pares de llaves y valores van separados por comas. Los dos puntos : hacen referencia al mapeo, identifican qué se le asigna a la llave. Los mapeos son conjuntos de pares de tipo llave – valor, donde llave y valor se separan por dos puntos y los pares llave-valor se separan por comas.

Ejemplo:

Sexo = {“Juana”: “F”, “David”: “M”, “Ernesto”: “M”, “Andrea”: “F”}

Muy seguramente en este caso el computador va a organizar el diccionario en orden alfabético de acuerdo con la llave.

El diccionario soporta todo tipo de objetos: strings, enteros…

Ya que el orden no importa, cómo se hace para indexar, por ejemplo en las listas uno puede decir que le muestre el elemento 3 o 4 y lo muestra, pero acá no hay orden entonces, se debe hacer con el ID de la llave y no con la posición, ejemplo, Andrea.

Se pueden agregar elementos al diccionario, se hace con sexo[“Santiago”] = “M”

El tercer elemento importante de los diccionarios es que son mutables.

Si hay dos llaves con el mismo ID, pero diferente valor se deja siempre el último valor.

Los diccionarios tienen sus propios métodos. Para utilizar el nombre del método se pone el nombre del diccionario punto y el método: sexo.keys() muestra la lista de llaves. Otros métodos es values y muestra los valores. Otro, el que más se utiliza, es ítems, muestra las llaves y valores en forma de tuplas, una colección de tuplas en par llave-valor.

Para borrar o eliminar elementos del diccionario se usa: del sexo [“Santiago”]

Para validar si una llave está dentro del diccionario se usa: “andres in sexo y el sistema bota un booleano que dice si está True y si no está False.

Clear remueve todos los elementos del diccionario.

Si no se especifica el tipo de dato que se quiere el computador guarda el diccionario como una tupla.

El count cuenta los elementos en una lista o tupla, el resultado genera un diccionario.

El método get muestra el valor de la llave. Este método soporta más de un argumento.

En caso de que la llave no exista dentro del diccionario le asigna un None, muestra un None (nada, vació). Yo puedo modificar eso y decirle qué quiero que me muestre, pero eso no agrega la llave al diccionario. Se puede agregar la llave al diccionario con este método en caso de que la llave no exista.

**Funcitional programming**

**- Funciones anónimas o “Lambda functions”:** Usa la palabra reservada lamba y con esta se define una función, sirve para ahorrarse líneas de código.

**- map (), filter(), reduce():** La función map está definida para tomar todos los elementos de una lista iterable y aplicarles una función, y además está programa de manera tal que es más eficiente que un Loop. Hace lo mismo que el loop, pero much más rápido, sirve en casos de bases de datos muy grandes. La función Map toma dos argumentos, una función y a qué se lo quiero aplicar, y ese a qué se lo quiero aplicar es una lista de datos.

**List comprehensions**

Sirve para simplificar las funciones vistas anteriormente.

**QUIZ**

