Sistema de tipos

1. ¿Qué es el sistema de tipos?

Es un conjunto de reglas que estructuran y organizan una colección de tipos. Provee **mecanismos de expresión**, esto es, formas de expresar tipos intrínsecos o definir tipos nuevos y asociar los tipos definidos con construcciones del lenguaje. Además define **reglas de resolución**, es decir, equivalencia(¿Dos valores tienen el mismo tipo?), compatibilidad(¿Puedo usar el tipo en este contexto?) e inferencia de tipos(¿Cuál es el tipo se deduce del contexto?). Mientras más flexible el lenguaje, más complejo el sistema.

2. ¿Cómo clasificamos un sistema de tipos?

El objetivo del sistema de tipos es lograr que los programas sean tan seguros como sea posible. Pero la seguridad tiene un costo llamado flexibilidad. Es en este punto en el que podemos dividir al sistema de tipos en dos partes:

- **Tipado Fuerte**: si el sistema de tipos especifica restricciones sobre cómo deben trabajarse las operaciones que involucran valores de diferentes tipos.
- **Tipado débil**: si el sistema de tipos es más flexible con respecto a cómo deben trabajarse las operaciones que involucran valores de diferentes tipos.

3. ¿Qué debo tener en cuenta para crear un sistema de tipos?

Basicamente, cuatro puntos:

- Tipo y tiempo de chequeo.
- Reglas de equivalencia y conversión.
- Reglas de inferencia de tipo.
- Nivel de polimorfismo del lenguaje.

4. ¿A qué apunta el tipo y tiempo de chequeo?

Con respecto al tipo de chequeo, se relaciona con el tipo de ligadura. Hay dos opciones:

- **Tipado estático**: si las ligaduras se dan durante la compilación.
- **Tipado dinámico**: si las ligaduras se dan durante la ejecución. Provoca mas comprobaciones en tiempo de ejecución.

IMPORTANTE: que el tipado sea fuerte, no implica que sea estático. El mejor ejemplo es Python. Tener en cuenta.

En lo concerniente al tiempo de chequeo, se relaciona con el tiempo de ligadura. Otras dos opciones:

- **Tipado estático**: si cada entidad/variable queda ligada a su tipo durante la compilación, sin necesidad de ejecutar el programa.
- **Tipado dinámico**: si la ligadura de la variable/entidad se produce en tiempo de ejecución.

5. ¿Qué es un tipo?

Es una mezcla de tres puntos de vista íntimamente relacionados:

- Desde el punto de vista **denotacional**(la relación entre el concepto y la realidad), es el conjunto de valores sobre un dominio.

- Desde el punto de vista **constructivo**, puede ser primitivo(built-in o predefinido) si lo provee el lenguaje o compuesto(composite o derivado) si emplea constructores de tipos.
- Desde el punto de vista **abstracto**, puede ser una interfaz a una representación o un conjunto de operaciones con semántica bien definida y consistente.

El dominio de un tipo de dato implica el conjunto de valores éste contempla mientras que la abstracción es aquello que permite distinguir qué operaciones pueden realizarse con tal dato sin preocuparse demasiado en cómo se implementará luego todo aquello.

6. ¿Qué tipos de datos existen?

7.

	Elementales	**Compuestos**
Predefinidos	Enteros, reales, caracteres, booleanos, etc	String(mayoría de las veces predefinido)
Definidos por el usuario	Enumerativos	Arreglos, registros, listas, etc

- Tipos predefinidos: son aquellos que ya vienen preestablecidos en el lenguaje. Ventajas:
  - Invisibilidad de la representación.
  - Verificación estática.
  - Desambiguar operadores.
  - Control de precisión.
- Tipos definidos por el usuario: separan la especificación de la implementación. Se definen los tipos que el problema necesita. Permiten definir nuevos tipos e instanciarlos. La idea es que cumplan con algunas características:
  - **Legibilidad**: elección apropiada de nuevos nombres.
  - **Estructura jerárquica de las definiciones de tipos**: proceso de refinamiento.
  - **Modificabilidad**: los cambios solo se realizan sobre la definición.
  - **Factorización**: se usa la cantidad de veces necesarias.