UNIVERSIDAD DE MURCIA

# Tipos de Datos Abstractos Sin Orden

## Tecnología de la Programación

L. Daniel Hernández < *ldaniel@um.es* >

Dpto. Ingeniería de la Información y las Comunicaciones
Universidad de Murcia
19 de septiembre de 2023

---

$$p(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

**TDA Polinomio**

- Usa: naturales (exponentes), reales (coeficientes)
- Operaciones:
  - `Crear (real *coef) : Polinomio`
    *El k-ésimo coeficiente representa al k-ésimo monomio.*
  - `suma (Polinomio f, Polinomio g) : Polinomio`
    *Retorna la suma de polinomios*
  - `termino (Polinomio f, int k) : real`
    *Retorna el coeficiente del k-ésimo monomio*
  - ...

```
class  Polinomio:
    grado: int
    coef: list

    def suma(p: Polinomio):
        ...
```

# Índice de Contenidos

# Desarrollo

# Qué son los TDAs sin relación de orden

- Los TDAs sin relación de orden hacen referencia a contenedores donde no establecemos ningún criterio de ordenación entre los elementos que almacena.
- Ejemplos:
  - La cesta de la compra en un supermercado.
  - Las personas que están en un sala de espera, identificadas con un boleto.
- Podemos distinguir las bolsas (bag), conjuntos (set) y mapas (map).
- Las operaciones generales que podemos definir en este tipo de contenedor, donde los objetos simplemente se almacenan son:
  - Consultar el número de objetos que tiene el contenedor,
  - Determinar si el contenedor está vacío,
  - Añadir un nuevo objeto en el contenedor,
  - Informar si un objeto está en el contenedor (pertenencia),
  - Retirar un objeto del contenedor, y
  - Quitar todos los objetos de (limpiar) el contenedor.

# Desarrollo

# Bag

- Contenedor que almacena una colección de elementos (objetos o items) donde todos son del mismo tipo y no importa la posición dónde se almacenó cada uno ni el número de veces que aparecen.
- Operaciones
  - `Bag(): Bag`. Crea un nuevo *bag*, inicialmente vacío.
  - `len(): int`. Retorna el número de elementos en el *bag*.
    Para una bag cualquiera $< a_0, a_1, \ldots, a_{n-1} >$ retornará el valor $n$.
  - `contains(item): bool`. Indica si el elemento `item` se encuentra en el *bag*.
    Retorna `True` si está contenido y `False` si no está contenido.
  - `add(item): None`. Modifica el *bag* añadiendo el elemento `item` al *bag*.
  - `remove(item): None`. Elimina y retorna la ocurrencia `item` del *bag*. Lanza un error si el elemento no existe.
  - `iterator(): IteratorBag`. Crea y retorna un iterador para el *bag*.

# Desarrollo

# Set - I

- Un Bag donde **no se permite** la repetición de elementos.
- Operaciones:
  - `Set(): Set`. Crea un nuevo conjunto, *set*, inicialmente vacío.
  - `len(): int`. Retorna el número de elementos en el conjunto.
    Para una lista cualquiera $< a_0, a_1, \ldots, a_{n-1} >$ retornará el valor $n$.
  - `contains(element): bool`. Indica si el elemento `element` se encuentra en el conjunto. Retorna `True` si está contenido y `False` si no está contenido.
  - `add(element): None`. Modifica el conjunto añadiendo el elemento `element` al conjunto.
  - `remove(element): None`. Elimina el elemento `element` del conjunto. Lanza un error si el elemento no existe.
  - `iterator(): IteratorSet`: Crea y retorna un iterador para el conjunto.
  - `isSubsetOf(setB): bool`. Determina si un conjunto es subconjunto del conjunto dado.
    Un conjunto A es subconjunto de B si todos los elementos de A están en B.
  - `equal(setB): bool`. Determina si el conjunto es igual al conjunto dado.
    Dos conjuntos son iguales si ambos contienen el mismo número de elementos y todos los elementos del conjunto está en el conjunto B. Si ambos están vacíos entonces son iguales.

# Set - II

- `union(setB): Set`. Retorna un nuevo conjunto que es la unión del conjunto con el conjunto dado.
  La unión del conjunto A con el conjunto de B es un nuevo conjunto que está formado por todos los elementos de A y todos los elementos de B que no están en A.
- `difference(setB): Set`. Retorna un nuevo conjunto que es la diferencia del conjunto con el conjunto dado.
  La diferencia del conjunto A con el conjunto de B es un nuevo conjunto que está formado por todos los elementos de A que no están en B.
- `intersect(): Set`. Retorna un nuevo conjunto que es la intersección del conjunto con el conjunto dado.
  La intersección del conjunto A con el conjunto de B es un nuevo conjunto que está formado por todos los elementos que están en A y también en B.

- Notar que los 6 primeros operadores del TDA Set son análogos a los operadores del TDA Bag. Los restantes son operaciones bien conocidas de los conjuntos matemáticos.

# Desarrollo

# Map

- Colección de registros no repetidos donde cada uno consta de una clave y un valor. La clave debe ser comparable y es la que se utiliza para poder acceder al valor.
- Ejemplos: tener las notas de los estudiantes por su identificador de estudiante, los datos fiscales por algún número de identificación, los datos de un conductor por el número de matricula de su vehículo, etc ...
- Operaciones:
  - `Map()`: `Map`. Crea un nuevo map vacío.
  - `len()`: `int`. Retorna el número de registros clave/valor que existen en el map..
  - `contains(key)`: `bool`. Indica si la clave `key` se encuentra en el contenedor. Retorna `True` si la clave está contenida y `False` si no está contenida.
  - `remove(key)`: `None`. Elimina el registro que tiene como clave el valor `key`. Lanza un error si el elemento no existe.
  - `add(key, value)`: `None`. Modifica el map añadiendo el par `key/value` al contenedor. Si existiera un registro con la clave `key` se sustituye el par `key/value` existente por el nuevo par `key/value`. Retorna `True` si la clave es nueva y `False` si se realiza una sustitución.
  - `valueOf(key)`: `TipoValor`. Retorna el valor asociado a la clave dada. La clave debe de existir en el Map.
  - `iterator()`: `IteratorMap`: Crea y retorna un iterador para el conjunto.

# Desarrollo

# Cómo implementar los TDAs sin orden

- **Estructuras integradas del lenguaje.**
  - Algunas estructuras conocidas son: `array`, `list`, `tuple` y `dictionary`, `namedtuple`, `chainmap`, etc ...
  - Dependerá del lenguaje particular.
  - Python no tiene arrays.
- **Estructuras enlazadas.**
  - Una estructura enlazada es aquella que se construye utilizando como estructura básica un **nodo**:

    ```
    struct Nodo
        valor: TDA
        referencia1 Nodo
        referencia2 Nodo
        ...
    ```

  - Es una estructura construida por el programador.
  - Un nodo sirve para representar muchos conceptos.
    Ejemplo: elementos de un conjunto, los de una lista o los vértices de un árbol o un grafo.
  - Distintos tipos de nodos nos sirve para implementar el mismo concepto de TDA.
    Ejemplo: un conjunto se puede implementar con un una referencia o con dos referencias.

# Desarrollo

# Estructuras útiles de Python

La estructuras de datos que presenta Python y que te pueden ayudar a implementar estos TDAs (y otros) son:

- Rangos
- Sets
- Listas
- Tuplas
- Diccionarios

Consulte la documentación oficial sobre cada uno de estos tipos de datos.

Recuerda

- Un TDA no es una estructura de datos.
- Un TDA no es una clase.