

# Abstracción. Tipos de Datos Abstractos

## Tecnología de la Programación

L. Daniel Hernández <ldaniel@um.es>

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

$$p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$$

### 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

- 1 Abstracción
- 2 Mecanismos de Abstracción
- 3 Tipos de Abstracción
  - Abstracción Procedimental
  - Abstracción de Iteración
  - Abstracción de Datos
- 4 Tipos de Datos Abstractos
  - Especificación de TDAs
  - Representación de TDAs
  - Implementación de TDAs



## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

Abstracción Procedimental

Abstracción de Iteración

Abstracción de Datos

## 4 Tipos de Datos Abstractos

Especificación de TDAs

Representación de TDAs

Implementación de TDAs



# Abstracción

- La **abstracción** es el proceso mental por el que captamos las **características principales** de un concepto o proceso descartando los detalles aislando conceptualmente las distintas partes, propiedades o cualidades de un objeto para estudiarlas por separado.
- Este proceso categoriza elementos en grupos.
  - Cada grupo es una abstracción que ignora determinadas características de los elementos que forman parte del grupo
  - Pero a la vez, en cada grupo, se resalta los aspectos más relevantes de los objetos que contienen.
- La abstracción permite estudiar un sistema complejo a diferentes niveles de detalle; es decir, refleja un **modelo jerárquico**.

La abstracción permite hacer una descomposición en la que varía el nivel de detalle. *Ej: Desde el concepto de bisagra al concepto de vivienda.*
- Es muy importante que cada uno de los niveles de abstracción estén al mismo nivel.

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

Abstracción Procedimental

Abstracción de Iteración

Abstracción de Datos

## 4 Tipos de Datos Abstractos

Especificación de TDAs

Representación de TDAs

Implementación de TDAs



# Mecanismos de Abstracción

Métodos prácticos que se emplean

- **Abstracción por parametrización.**

- Un parámetro representa a un conjunto de elementos específicos.

**Ejemplo:** `int num`, que es una abstracción de los valores concretos naturales 1, 2, 3, ... con los que podemos identificar `num`.

- Los parámetros se usan en los procedimientos (abstracción procedimental): los parámetros en un función/procedimiento, abstrae un número infinito de acciones.

**Ejemplo:** : En vez de usar el procedimiento `trasladarAlPunto(juan, (1, 3))` se puede usar `trasladarAlPunto(Objeto obj, Punto p)` para representar que se quiere trasladar cualquier objeto a un punto del plano (y no solo a `juan` a un punto concreto).

- **Abstracción por especificación.** Se basa en cumplimentar un documento en el que se indica lo siguiente:

- Un nombre. El identificador de un procedimiento, función, ...
- Una descripción. Explica en qué consiste la abstracción pero solo mencionará lo imprescindible.
- Las condiciones. Los estados que deben cumplirse. Varias partes según los detalles de la abstracción a realizar.

**Ejemplo:** En una abstracción procedimental, se debe determinar, al menos, las **precondiciones** (los requisitos exigidos para el procedimiento) y las **postcondiciones** (el efecto de la ejecución).

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

- Abstracción Procedimental
- Abstracción de Iteración
- Abstracción de Datos

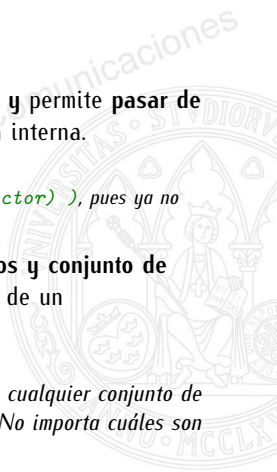
## 4 Tipos de Datos Abstractos

- Especificación de TDAs
- Representación de TDAs
- Implementación de TDAs



# Tipos de Abstracción

- **Abstracción procedimental.** Es definir un conjunto de **procedimientos como abstracción de operaciones**.
  - Funciones y procedimientos responden a esta abstracción.
  - *Ej: la suma de números se abstrae para sumar matrices, listas, ...*
- **Abstracción de iteración.** Es la que usa **colecciones** de objetos y permite **pasar de un elemento al siguiente** sin saber cómo se organizan de forma interna.
  - Intervienen dos elementos: el **iterable** y el **iterador**.
  - *Ej: `while(vector.hasNext())` generaliza a `while( i < len(vector) )`, pues ya no depende del índice y solo depende de la colección.*
- **Abstracción de datos.** Es la que trabaja con **conjunto de objetos y conjunto de operaciones** (procedimientos) sobre los objetos, lo que los dota de un comportamiento.
  - La mayor abstracción define modelos: datos + operaciones
  - *Ej: la estructura de grupo en matemáticas es una abstracción de cualquier conjunto de objetos que tenga un operador cerrado con ciertas propiedades. No importa cuáles son los objetos de la estructura.*





# Profundizamos en ...

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

### Abstracción Procedimental

Abstracción de Iteración

Abstracción de Datos

## 4 Tipos de Datos Abstractos

### Especificación de TDAs

### Representación de TDAs

### Implementación de TDAs



# Abstracción Procedimental

- Consiste en crear **procedimientos y funciones** como abstracción de operaciones.
- Un procedimiento/función **debe responder sin ambigüedad a qué hace obviando el cómo lo implementa.**
- Abstraemos un conjunto de operaciones (detalles de cómo se realiza) como una única operación (el qué hace), expresada como función o procedimiento.
- Las técnicas de **programación procedimental y modular** usan este tipo de abstracción para romper el problema principal en problemas más pequeños.
- Toda operación **se debe especificar** de la siguiente forma:

```
operacion nombre (id1:tipo1, id2: tipo2, ....) return tipo
Precondiciones: Indican los datos de entrada, los id
Retorna: Indica el tipo de dato que retorna
Descripción: Descripción textual del comportamiento de
              la operación (el efecto)
Excepciones: Indica las excepciones (opcional)
```

- Este tipo de especificaciones los incorporan los IDE y editores de programación.
- A partir de estas especificaciones se genera la documentación (API) para los programadores.  
Un buen ejemplo es  
[https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#substring\(int\)](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#substring(int))

# Profundizamos en ...

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

Abstracción Procedimental

**Abstracción de Iteración**

Abstracción de Datos

## 4 Tipos de Datos Abstractos

Especificación de TDAs

Representación de TDAs

Implementación de TDAs



# Abstracción de Iteración

- Consiste en tener algún mecanismo que permita **acceder** a los elementos de un contenedor **sin tener en cuenta su representación interna**.
- **Representación interna**: Estructura de datos que se utiliza para representar al contenedor.
- Los **bucles con contadores** tienen en cuenta la representación de los datos.

```
for i: int = 0...top:  
    acción sobre elemento[i] # Considera estructura indexada
```

- Es más adecuado tener una **abstracción** de la forma:

```
for cada elemento P de Contenedor  
    acción sobre P
```

- La responsabilidad del recorrido se traslada a un objeto que se llamará **iterador**.
- Sobre un contenedor se podría definir **diferentes iteradores**:
  - Cada iterador sabe cómo recorrer el contenedor para el que se define
  - Cada iterador tiene su propio recorrido sobre la estructura
- Sobre distintos tipos de contenedores los iteradores puede recorrerlas usando una **interfaz común**: ¡no hay que cambiar el código principal! P.e. **iterador.next()**

# Profundizamos en ...

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

Abstracción Procedimental

Abstracción de Iteración

**Abstracción de Datos**

## 4 Tipos de Datos Abstractos

Especificación de TDAs

Representación de TDAs

Implementación de TDAs



# Abstracción de Datos

- Existen 3 tipos (o niveles) de abstracción.
- **Tipos de datos integrados** (o fundamentales). Son los que ofrecen los lenguajes de programación.

**Ejemplo:** En **Python** se distinguen, entre otros<sup>1</sup>:

- Los tipos de datos simples: numéricos (enteros, reales y complejos) y booleanos.
- Los tipos de datos compuestos: cadena de caracteres (str), secuencias (rangos, listas y tuplas), mapas (diccionarios), conjuntos.
- **Tipos de datos definidos por el usuario** o programador. Son los que pueden diseñar los programadores agrupando todos de datos fundamentales.
  - array, record, struct, **class**
- **Tipos de datos abstractos (TDA)**. Construye modelos (matemáticos) usando agrupación de datos.
  - No es lo mismo una estructura de datos formado por dos valores y operar con ellos que trabajar con vectores numéricos 2D con operaciones matemáticas (donde no importa la estructura).

---

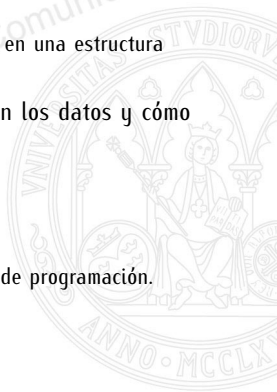
<sup>1</sup><https://docs.python.org/es/3/library/stdtypes.html>

- 1 Abstracción
- 2 Mecanismos de Abstracción
- 3 Tipos de Abstracción
  - Abstracción Procedimental
  - Abstracción de Iteración
  - Abstracción de Datos
- 4 Tipos de Datos Abstractos
  - Especificación de TDAs
  - Representación de TDAs
  - Implementación de TDAs



# Tipos de Datos Abstractos

- Los **Tipos de Datos Abstractos** (TDA) son **modelos matemáticos** que constan de
  - un nombre publico para
  - identificar a un conjunto de datos (valores), junto con
  - un conjunto de operaciones bien definidas sobre los datos (como en una estructura algebraica).
- Como modelo, le es **irrelevante** cómo se almacenan o estructuren los datos y cómo se implementan las operaciones.
- Para todo TDA hemos de abordar **tres tareas**:
  - **Especificación**: definición del TDA
  - **Representación**: estructura con la que representar el TDA.
  - **Implementación**: cómo implementar la estructura en un lenguaje de programación.





# Profundizamos en ...

- 1 Abstracción
- 2 Mecanismos de Abstracción
- 3 Tipos de Abstracción
  - Abstracción Procedimental
  - Abstracción de Iteración
  - Abstracción de Datos
- 4 Tipos de Datos Abstractos
  - Especificación de TDAs
  - Representación de TDAs
  - Implementación de TDAs



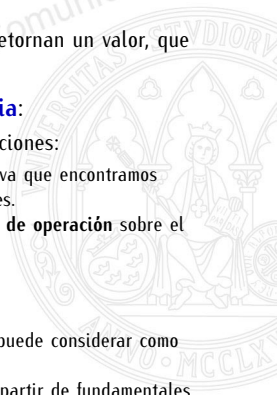
# Especificación de TDAs

- La **especificación** de un TDA (Datos+Operaciones) se rige por las normas generales de la **Abstracción por Especificación**
- La especificación consta de **tres partes**: nombre y descripción, la definición de los datos y la definición de los operadores.
- **Nombre y descripción.**
  - Se le dará un nombre que identifica al conjunto de datos y las operaciones.
  - Se indicará qué representan.
- **Especificación de los datos.**
  - Pueden venir dados por reglas.  
**Ejemplo:** Definiciones recursivas (p.e. una lista).
  - Puede usar notaciones matemáticas conocidas.  
**Ejemplo:** Conjuntos conocidos ( $\mathbb{N}$ ,  $\mathbb{R}$ , ...), conjuntos  $\{s_1, s_2, \dots\}$ , intervalos  $[a, b]$ , expresiones regulares, ...
  - Nunca se definirá pensando en estructuras concretas de un lenguaje de programación.
- **Especificación de las Operaciones** (abstractas).
  - Se indicará tanto la sintaxis como la semántica de cada una.
  - Se emplearán las especificaciones de la abstracción procedimental.
- **IMPORTANTE. Un TDA define un valor de tipo  $T$ :**

*Todos los entes/valores que respondan al TDA  $T$  son de tipo  $T$ .*

# Tipos de Operadores de los TDAs

- Los operadores de un TDA se pueden dividir por su **objetivo**:
  - **Constructores.**  
Los que indican cuáles son los datos necesarios para construir un valor de tipo  $T$ .
  - **Modificadores.**  
Los que construyen un nuevo valor de tipo  $T$  a partir de un valor de tipo  $T$  dado.
  - **Consulta.**  
El conjunto de operaciones que a partir de un valor de tipo  $T$  retornan un valor, que no es de tipo  $T$ .
- Las operaciones de un TDA se pueden dividir por su **importancia**:
  - **Fundamentales**, también llamadas primitivas. Cumplen dos condiciones:
    - **No se puede quitar ninguna:** La supresión de una de ellas conlleva que encontramos problemas que no se pueden resolver porque nos faltan operaciones.
    - Ese conjunto de operaciones **permite construir cualquier otro tipo de operación** sobre el TDA.
    - **Todas deben poder usarse:** todas deben estar visibles.
  - **No fundamentales.**
    - **Apoyan la definición de una operación fundamental**, pero no se puede considerar como tal. No deben de usarse (deben estar ocultas)
    - Las que **aumentan el conjunto de operaciones** y se construyen a partir de fundamentales (deberían añadirse las menos posibles)



# Un ejemplo de definición de TDA.

## Ejemplo: TDA Polinomio

- Es una expresión algebraica  $p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$  compuesta por la suma de dos o más monomios (es un **coeficiente** y una **variable** con **exponente**). No existen dos monomios con el mismo exponente. Todos los monomios usan la misma variable. El  $k$ -ésimo monomio es el que tiene una variable con coeficiente  $\neq 0$ .
- 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*
  - *etc...*
- **NOTA.** Aquí se muestra una versión simplificada. Hay que seguir la especificación procedimental.



# Profundizamos en ...

- 1 Abstracción
- 2 Mecanismos de Abstracción
- 3 Tipos de Abstracción
  - Abstracción Procedimental
  - Abstracción de Iteración
  - Abstracción de Datos
- 4 Tipos de Datos Abstractos
  - Especificación de TDAs
  - Representación de TDAs
  - Implementación de TDAs



# Representación de TDAs

- Se debe de elegir una estructura **rep** indicando qué datos de un valor de tipo  $T$  se almacenan en la estructura **rep**.
- **La función de abstracción.** Una función sobreyectiva  $Abst : \text{rep} \longrightarrow \mathcal{A}$
- **El invariante de la representación.** Es un predicado  $I : \text{rep} \longrightarrow \mathbb{B}$  que es cierto para los objetos de **rep** que sean legítimos.

- **Ejemplo:** Representación de Polinomios

- Para los polinomios  $p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$  se opta por la estructura:

```
struct rep {  
    entero grado  
    real[] coef  
}
```

- grado es **un entero** para representar el grado del polinomio, y
- coef es **una lista** indexada de reales para representar a los coeficientes  
coef NO ES UN ARRAY.
- $Abst(r) = r.coef[0] + r.coef[1]x + r.coef[2]x^2 + \dots$   
 $\dots + r.coef[r.grado]x^{r.grado}$
- $I(r) = (r.grado \geq 0) \text{ AND } r.grado = \text{len}(r.coef)$

# Profundizamos en ...

## 1 Abstracción

## 2 Mecanismos de Abstracción

## 3 Tipos de Abstracción

Abstracción Procedimental

Abstracción de Iteración

Abstracción de Datos

## 4 Tipos de Datos Abstractos

Especificación de TDAs

Representación de TDAs

Implementación de TDAs



# Implementación de TDAs

- lenguaje: lenguaje de programación en el que se implementa el TDA.
- Lo que **necesita conocer** el usuario del TDA en ese lenguaje es:
  - su **nombre**,
  - su **dominio** (conjuntos de datos con los que trabaja y su tipo),
  - su **interface** (los nombres de las operaciones asociadas).

Estos 3 elementos caracterizan a un TDA y se llama **parte pública**.

- Lo que **no necesita conocer** el usuario del TDA en ese lenguaje es:
  - la **estructura de datos** usada (cómo está codificados los conjuntos de datos)
  - cómo se han **implementado** los algoritmos (operadores)

Estos elementos reciben el nombre de **parte privada**.

- Cuando se programe el TDA se deberá de tener en cuenta estas dos partes.
- Se deberá de implementar considerando:
  - **Encapsulamiento**, agrupando en un objeto atributos (variables) y métodos (operaciones).
  - **Ocultación** de información, pues establecer qué atributos y métodos pueden permanecer ocultas (o visibles).
- La POO aparece de forma natural para resolver esta situación.

- **Nuestro Objetivo:**

Usar la POO para implementar TDAs (que son modelos matemáticos) en Python.