PROGRAMACIÓN LÓGICA Y FUNCIONAL

Unidad 1: Conceptos básicos

Sesión 1: Introducción a los Paradigmas Declarativos

Docente: Carlos Reynaldo Portocarrero Tovar

UTP Universidad Tecnológica del Perú

Sobre el docente

- Profesional apasionado por la docencia y la investigación científica.
- Docente de la Universidad Tecnológica del Perú (UTP)
- Doctor y Magister de Ciencias en Computación por la Universidad Federal do ABC (UFABC) - Brasil.
- Ingeniero en Sistemas e Informática.
- Técnico en Redes y Comunicaciones de datos.
- Líneas de Investigación: Bioinformática y Ciencia de Datos.
- Email: c31644@utp.edu.pe
- Teléfono: +51 987 412 264

Universidad Tecnológica del Perú

INICIO
Objetivo de la sesión

Al finalizar la sesión el estudiante reconoce los fundamentos de la programación declarativa y diferencia los enfoques imperativo y declarativo.

UTILIDAD
¿Porque es importante conocer los paradigmas de programación?

1. Amplía tu forma de pensar como programador
2. Te da herramientas para elegir la mejor solución
3. Mejora tu adaptabilidad a nuevos lenguajes
4. Aumenta la calidad y mantenibilidad del código
5. Es clave en entornos profesionales
6. Te hace más creativo

TRANSFORMACIÓN
¿Qué es un Paradigma de Programación?

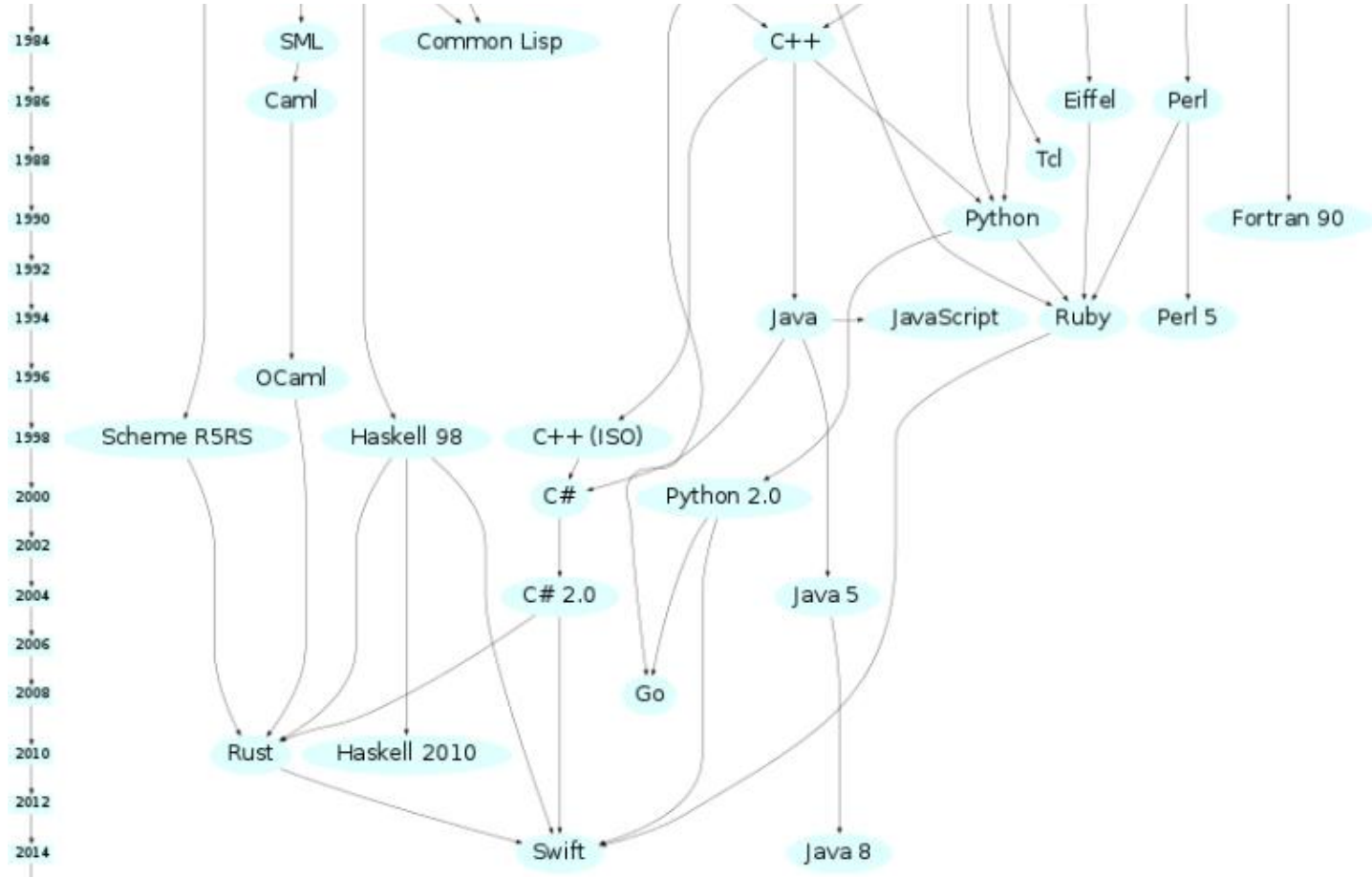
Un paradigma de programación es un estilo o enfoque para resolver problemas computacionales mediante la escritura de programas.

- Imperativo
- Declarativo
- Orientado a Objetos
- Funcional
- Lógico

Principales Paradigmas de Programación

- Imperativo – Basado en secuencias de instrucciones (C, Java, Python).
- Declarativo – Describe el resultado deseado, no los pasos (SQL, Prolog, Haskell).
- Orientado a Objetos (OOP) – Modela entidades como objetos con atributos y métodos (Java, C#, Python).
- Funcional – Basado en funciones puras y sin estado (Haskell, Lisp, Scala).
- Lógico – Basado en hechos y reglas lógicas (Prolog).
- Reactivo/Event-Driven – Responde a eventos o flujos de datos (JavaScript, RxJS).

Evolución Lenguajes de Programación



https://ingenieriadesoftware.es/historia-visual-lenguajes-programacion/

Imperativo vs Declarativo

Imperativo: describe paso a paso cómo se realiza una tarea.
Declarativo: describe qué se desea lograr, sin detallar el proceso.

Ejemplos:

- Imperativo: C, Java
- Declarativo:
  - Haskell (Funcional)
  - Prolog (Lógico)

Características del Paradigma Declarativo

- Se enfoca en el resultado, no en los pasos.
- Uso de expresiones matemáticas o lógicas.
- Facilidad para pruebas y verificación formal.
- Menor cantidad de errores por efectos colaterales.

Lenguajes Declarativos

- **Haskell** (paradigma funcional):
  - Funciones puras
  - Evaluación perezosa
  - Sin efectos secundarios
- **Prolog** (paradigma lógico):
  - Programación basada en hechos y reglas
  - Resolución automática por inferencia
  - Uso de unificación y backtracking

Casos de Uso: Áreas de Aplicación

- Haskell:
  - Data
  - Programación Funcional (PF)
- Prolog
  - Inteligencia Artificial (IA)
  - Reglas

Comparativa Paradigmas

Característica	Imperativo (Python)	Declarativo (Haskell/Prolog)
Enfoque	"Cómo" resolver	"Qué" resolver
Ejemplo	for i in range(10): print(i)	map print [1..10]
Ventaja	Control detallado	Código conciso

PRACTICA
Actividades en Clase

- Presentación de conceptos clave
- Preguntas abiertas y lluvia de ideas
- Demostración práctica en GHCi y SWI-Prolog

Recursos:

- SWI-Prolog
- Replit https://replit.com/

Ejemplos

Haskell

-- Calcular el cuadrado de un número

```
cuadrado x = x * x
```

Prolog

% Definir relaciones padre-hijo

```
padre(juan, maria).
padre(juan, jose).
```

Demo Haskell

```
-- Evaluación perezosa
module Main where
fibs :: [Integer]
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)
main :: IO ()
main = print (take 15 fibs)
```

Demo Prolog

```
% Sistema experto simple
sintoma(fiebre).
sintoma(tos).
diagnostico(gripe) :- sintoma(fiebre), sintoma(tos).
```

Tarea

Investigar un ejemplo de problemas donde sea mejor usar un enfoque declarativo que uno imperativo. Explicarlo en media página escrita a mano.

CIERRE
Conclusiones

- Comprender los paradigmas amplía la forma de pensar como programador.
- Elegir el paradigma correcto mejora la calidad y mantenibilidad del código.
- Conocer varios enfoques aumenta la adaptabilidad a nuevos lenguajes y entornos.
- Los paradigmas declarativos ofrecen soluciones más concisas y menos propensas a errores en ciertos contextos.

Universidad
Tecnológica
del Perú