

# Lenguajes y Paradigmas de Programación

Toda la información está disponible en la [ficha del campus virtual](#) y en el [sitio Moodle de LPP](#)

## Datos académicos de la asignatura

**Departamento de Ciencia de la Computación e Inteligencia Artificial**

**6 créditos ECTS:** 1 clase de teoría de 2 h. y 1 clase de prácticas de 2 h. a la semana

**Profesores:**

- Antonio Botía ([e-mail](#)): grupos 6 y 9 de prácticas
- Domingo Gallardo ([e-mail](#)): grupos 2 y 4 de teoría y 3, 5 y 8 de prácticas
- Cristina Pomares ([e-mail](#)): grupos 1 y 3 de teoría y 1, 2, 4 y 7 de prácticas, coordinadora de la asignatura

## Recursos de la asignatura

- [Sitio Moodle](#) abierto y accesible a toda la comunidad educativa, contiene los apuntes, transparencias, prácticas y otros materiales docentes
- [Foro de consultas](#) en el sitio Moodle (sólo accesible a estudiantes)
- [Foro de anuncios](#) en el sitio Moodle (sólo accesible a estudiantes)

## Objetivos y competencias

**Objetivos:**

- ¿Qué elementos son comunes a los lenguajes de programación?
- ¿Qué características tienen?
- ¿Cuáles son los elementos esenciales y los accesorios?
- ¿Qué familias o paradigmas de lenguajes podemos identificar?
- ¿Cómo diseñar un buen lenguaje de programación?

Dominando estos contenidos será mucho más fácil aprender nuevos lenguajes de programación, identificar sus aspectos esenciales e incluso ser capaz de diseñar lenguajes específicos orientados a dominios concretos.

**Competencias:**

- Conocer y diferenciar las características de los distintos paradigmas de programación (programación funcional, procedural y orientada a objetos) e identificarlas en lenguajes de programación concretos.
- Conocer los elementos que componen los lenguajes de programación (estructuras de control, procedimientos, tipos de datos) y distintas implementaciones de estos elementos en distintos lenguajes.
- Diferenciar entre tiempo de ejecución y tiempo de compilación en distintos ámbitos: detección de errores o definición, creación o ámbito de vida de variables.
- Conocer modelos de computación específicos que expliquen la semántica de los lenguajes de programación. En concreto: modelo de sustitución para la programación funcional y modelo de entornos para explicar el ámbito y los valores de las variables en la programación procedural.
- Utilizar la abstracción y la recursión para diseñar correctamente procedimientos y estructuras de datos (listas y árboles).
- Ser capaz de diseñar, implementar y corregir programas funcionales, en concreto utilizando el lenguaje de

programación Scheme.

- Ser capaz de implementar programas sencillos en Scala, en los que se utilicen las características multiparadigma del lenguaje.
- Comparar el paradigma orientado a objetos con el paradigma procedural clásico, reconociendo las ventajas que aporta en cuanto a abstracción, reutilización y modificación de código.
- Conocer los elementos fundamentales de la abstracción aplicada al diseño de lenguajes de programación. Ser capaz de diseñar un sencillo lenguaje orientado a un dominio restringido (DSL).
- Conocer los principios básicos del paradigma de programación lógica.

## Temario

- Tema 1. **Lenguajes de programación:** Historia de los lenguajes de programación. Elementos de los lenguajes de programación. Abstracción. Paradigmas de programación. Compiladores e intérpretes.
- Tema 2. **El lenguaje de programación Scheme:** Primitivas. Tipos de datos básicos. Símbolos. Cadenas. Listas. Definición de funciones.
- Tema 3. **Programación Funcional:** Características e historia del paradigma de Programación Funcional. Características declarativas del paradigma funcional. Definición de funciones. Funciones como datos de primer orden. La forma especial lambda. Ámbito de variables y closures. Dualidad entre datos y programas: formas especiales apply y eval. Datos compuestos en Scheme: parejas. Construcción, recorrido y operaciones sobre listas. Listas con elementos compuestos. Listas de listas.
- Tema 4. **Procedimientos y estructuras recursivas:** Diseño de funciones recursivas. Recursión mutua. Procesos recursivos e iterativos. Memoization. Estructuras de datos recursivas: expresiones-s, árboles binarios y árboles genéricos.
- Tema 5. **El lenguaje de programación Scala.** Intérprete y scripts. Tipos de datos básicos. Operadores. Estructuras de control. Ámbito de variables. Tipos de datos compuestos: arrays y colecciones. Recorriendo colecciones. Definición de clases y herencia en Scala. Paquetes y sentencias imports en Scala.
- Tema 6. **Programación funcional en Scala:** Lenguajes multiparadigma. Programación funcional en Scala. Listas. Recursión pura y recursión por la cola. Funciones como datos de primer orden. Clousures y funciones anónimas. Funciones de orden superior: mappings y filtros de colecciones.
- Tema 7. **Programación Imperativa:** Historia del paradigma de programación imperativo. Características principales. Datos mutables. Datos mutables en Scheme. Estructuras de datos mutables en Scheme. Ámbito de variables dinámico y modelo de entornos en Scheme y Scala. Estado local.
- Tema 8. **Programación Orientada a Objetos:** Características e historia del paradigma de Programación Orientada a Objetos. Conceptos avanzados de POO en Scala: Traits en Scala. Traits como modificaciones apilables. Ventajas de los traits frente a la herencia múltiple. Miembros abstractos. Modelo de agentes para la programación concurrente.
- Tema 9. **Construcción de nuevos lenguajes de programación:** Diseño de lenguajes de programación. Macros. Lenguajes de dominio (DSL).
- Tema 10. **Programación Lógica:** El paradigma de programación lógica. Programación lógica en Scheme. Clausulas y objetivos. Backtracking. Unificación. Operadores lógicos. Variables.

El calendario de temas, prácticas y exámenes está disponible en la ficha de la asignatura.

## Prácticas

Durante las horas prácticas se realizarán de forma individual ejercicios de programación relacionados con los conceptos que se están estudiando en ese momento que servirán para reforzar y profundizar en las competencias de la asignatura. Para el desarrollo de las prácticas utilizaremos los siguientes lenguajes de programación y entornos de desarrollo:

- **Racket** (versión de Scheme, lenguaje de programación funcional)
- **Scala** (lenguaje multiparadigma basado en Java con características de programación funcional)

Se realizarán entre 11 sesiones de ejercicios de prácticas de una semana de duración y 2 seminarios de lenguajes. En cada sesión se propondrá una hoja de ejercicios con 5 o 6 problemas de programación relacionados con los temas vistos en las sesiones de teoría que se resolverán de forma individual. El profesor estará disponible en la sesión de prácticas para resolver dudas y dar pistas sobre cómo atacar los problemas.

También cada semana, durante la primera media hora de la sesión práctica, se pondrán en común las soluciones de la hoja de ejercicios anterior. Durante la hora y media restante se resolverán los ejercicios de la semana actual.

Al final de la semana, cuando en todos los grupos de prácticas se hayan puesto en común las soluciones, se publicarán en Moodle las soluciones propuestas por los profesores.

Resumen del funcionamiento:

- Cada semana se publicará la hoja de ejercicios *n* a realizar en las prácticas
- Durante esa semana se trabajará en la hoja de ejercicios *n* y se pondrán en común las soluciones de la hoja *n-1*
- Se publicará la solución de la hoja de ejercicios *n-1*

## Horarios

Horario LPP 2014-2015

	Lunes	Martes	Miércoles	Jueves	Viernes
8:30-9:00					
9:00-9:30			Grupo T2 <b>Domingo</b>	Grupo T1 <b>Cristina</b>	
9:30-10:00			ARA D28	D26	
10:00-10:30			Asociado a T2		
10:30-11:00					
11:00-11:30		Grupo P3 <b>Domingo</b>	Grupo P7 <b>Cristina</b>	Grupo P1 <b>Cristina</b>	Grupo P9 <b>Antonio</b>
11:30-12:00		L24	L25	L27	L16
12:00-12:30		Asociado a T2	Asociado a T1	Asociado a T1	Asociado a T1
12:30-13:00					
13:00-13:30		Grupo P2 <b>Cristina</b>			
13:30-14:00		L13			
14:00-14:30		Asociado a T1			
14:30-15:00					
15:00-15:30				Grupo T4 <b>Domingo</b>	
15:30-16:00	Grupo T3 <b>Cristina</b>	E12		D26	
16:00-16:30					
16:30-17:00					
17:00-17:30		Grupo P6		Grupo P8 <b>Domingo</b>	
17:30-18:00	Grupo P4 <b>Cristina</b>	Antonio		L23	
18:00-18:30	L15	L13		Asociado a T4	
18:30-19:00	Asociado a T3	Asociado a T4			
19:00-19:30				Grupo P5 <b>Domingo</b>	
19:30-20:00				L13	
20:00-20:30				Asociado a T4	
20:30-21:00					
21:00-21:30					
21:30-22:00					

## Evaluación

### Convocatoria normal (evaluación continua)

En la convocatoria de junio se realizará una evaluación continua con tres exámenes parciales y con la evaluación semanal de los ejercicios prácticos. El primer examen ponderará un 30% y los otros dos un 32% en la nota final. Los exámenes consistirán en pruebas y ejercicios similares a las realizadas en las clases prácticas, por lo que es fundamental su seguimiento puntual.

Un 6% de la calificación se conseguirá con la asistencia a las clases prácticas (siendo necesario asistir a al menos 2/3 de dichas clases) y la entrega en Moodle de al menos 7 hojas de ejercicios de las 11 planteadas. Con esas entregas se obtendrán 0,4 puntos en la nota final, las entregas 8, 9, 10 y 11 añadirán 0,05 puntos cada una a la nota final. Esta parte de la calificación no es recuperable.

En el primer parcial no se exige nota mínima en ninguno de los parciales. En los parciales 2 y 3 se debe obtener más de un 3 para poder ponderar la nota con los demás. La nota final será la suma ponderada de todas las notas de la evaluación continua. El parcial 3 se realizará en la fecha oficial asignada al examen final de la convocatoria de junio.

- Convocatoria normal (evaluación continua)
  - Parcial 1 (30%, sin nota mínima): Temas 1, 2 y 3 - Semana 2 de marzo
  - Parcial 2 (32%, nota mínima de 3): Temas 4, 5, 6 y 7 - Semana 27 de abril
  - Parcial 3 (32%, nota mínima de 3): Temas 8, 9, 10 - 8 de junio
  - Asistencia y participación en prácticas (6%)
  - La nota final se calculará con la media ponderada de las 4 notas anteriores

## Convocatoria extraordinaria

- En la convocatoria de julio se podrán recuperar los parciales en los que se haya obtenido una nota inferior a 5. La nota final se calculará con la misma ponderación que en la primera convocatoria, un 94%.
- La nota de prácticas no se recupera, se mantiene la de la evaluación continua

## Consejos para aprender con éxito los contenidos de la asignatura

El consejo fundamental para aprobar la asignatura es **trabajar todas las semanas e intentar seguir el ritmo de la asignatura**. Los conceptos de la asignatura se van construyendo de forma progresiva y lo visto en una semana depende muchas veces de lo aprendido en semanas anteriores.

¿Cómo estudiar estos conceptos? Con la excepción de algunos temas y apartados concretos (como la historia de los lenguajes de programación o las características de los distintos paradigmas) la asignatura es fundamentalmente práctica. No tiene sentido aprender de memoria los ejercicios y los ejemplos vistos en clase. Hay que *trabajárselos*. Eso significa que, primero, hay que entenderlos sobre el papel y *muy importante* hay que **probar todos los ejemplos en el intérprete (el REPL)**. Y *probar* significa escribir los ejemplos y jugar con ellos, proponiendo pequeñas variantes, preguntándose *¿qué pasaría si...?* y probándolo.

En cuanto a las prácticas y a los ejercicios propuestos es fundamental pelearse con ellos e intentar hacerlos por uno mismo **sin ver ninguna solución**. Es la única forma de aprender: probando, equivocándose y encontrando la solución por uno mismo.

A la hora de enfrentarse con un problema es fundamental también **usar lápiz y papel** para probar enfoques y encontrar la solución más sencilla sobre el papel antes de probarla en el REPL. Los ejercicios que proponemos no son excesivamente complicados. Todos se resuelven con muy pocas líneas de código y su codificación en el ordenador no tiene dificultad, una vez que se ha encontrado la solución que lo resuelve. Al usar el lápiz y papel también estarás practicando una situación similar a la que te vas a encontrar en los exámenes de la asignatura.

Resumiendo: trabajar todas las semanas, hacer uno mismo todos los ejercicios y no desanimarse ni descolgarse.

Son muy interesantes algunos comentarios de antiguos estudiantes que han aprobado la asignatura.

Cómo dominar los conceptos:

“Para superar la asignatura lo que hice fue estudiar mucho. Hay que practicar y sobre todo

entender los ejercicios y no sabérselos de memoria. Una vez dominados los ejercicios yo mismo me propuse variantes de los mismos. Así es como se domina.”

El problema del cambio de paradigma:

“El problema principal de la asignatura es enfrentarse a un cambio del paradigma de programación.”

Trabajar día a día:

“Lo que hice fue tratar de llevar al día toda la asignatura, además de trabajar con material adicional para poder ampliar y profundizar conocimientos. LPP es una asignatura de fondo en la que tienes que mantener el ritmo de trabajo de principio a fin de cuatrimestre.”

No copiar las prácticas:

“El mayor problema que creo que existe es que muchas personas se relajan y se copian las prácticas en cuanto les resultan un poco difíciles o les lleva algo más del tiempo que les gustaría. Esta asignatura si no haces tu los ejercicios y te peleas con ellos es prácticamente imposible de sacar.”

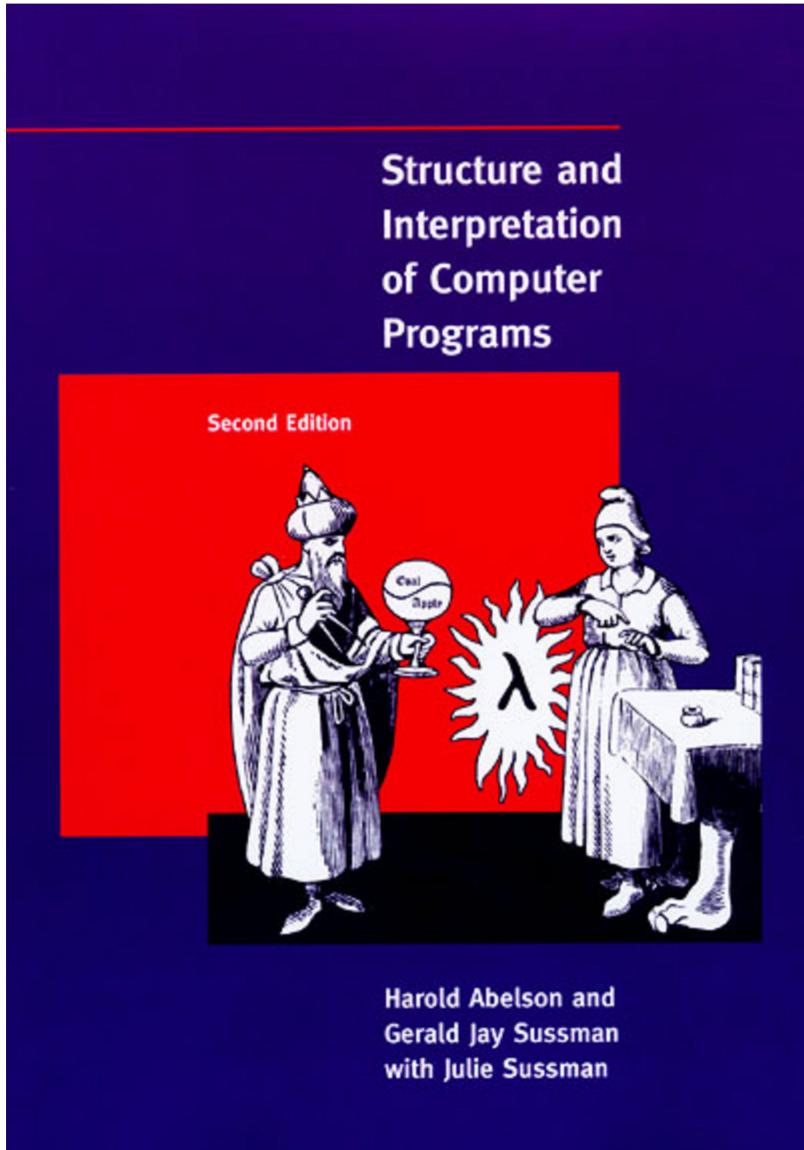
No memorizar:

“Otra de las cosas es que tienes que cambiar la forma de estudiar, no vale memorizar, ni hacer muchos ejercicios sin más. Tienes que entender bien el funcionamiento de la recursión para luego poder practicar con ejercicios, sino no sirve. [...] En mi opinión el problema de LPP para mucha gente es que para los exámenes se memorizan los ejercicios de prácticas de las soluciones que se dan en clase. [...] Por último, otro problema es la carga de trabajo que se tiene en grado, no con esta asignatura que de verdad realiza evaluación continua, pero te encuentras con otras cuatro asignaturas que te piden un mínimo en exámenes parciales, otro en prácticas con más carga lectiva que esta asignatura y además un examen final que incluye toda la asignatura y al final la asignatura que menos bien llevas (o peor) es la que terminas abandonando.

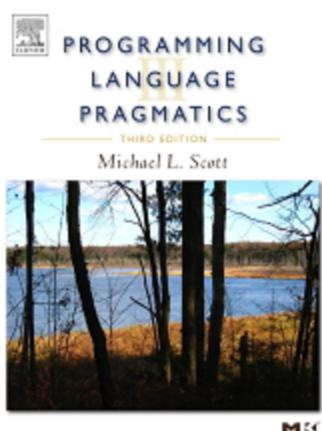
Plantearse uno mismo problemas:

“Los problemas que me encontré a la hora de cursar LPP fue que eran dos lenguajes completamente nuevos, otro tipo de programación que nunca había visto, otra forma de estudiar distinta. Para poder superarla simplemente tienes que hacer ejercicios y también plantearte tú mismo nuevos ejercicios.”

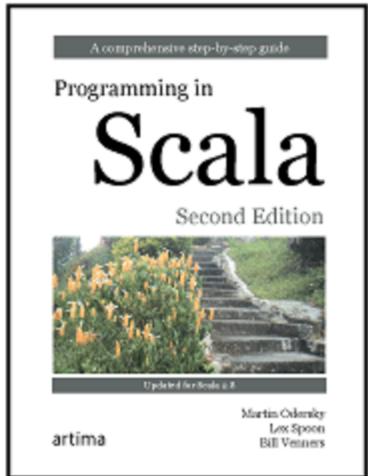
## Bibliografía



- Harold Abelson y Gerald Jay Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, 1996
- [Enlace a la edición on-line](#)
- Signatura en la Biblioteca Politécnica: I.06/ABE/STR



- Michael L. Scott, *Programming Language Pragmatics*, Morgan Kaufmann Publishers, 2009 (Third Edition)
- [Web](#)
- Signatura en la Biblioteca Politécnica: POE 004.43/SCO/PRO



- Martin Odersky, *Programming in Scala*, Artima, 2011
- [Web](#)
- Signatura en la Biblioteca Politécnica: POE 004.43/ODE/PRO

Lenguajes y Paradigmas de Programación, curso 2014-15

© Departamento Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante  
Domingo Gallardo, Cristina Pomares