Laboratorio

Ingeniería de Software

@SOYPABLOG – 31 DE ENERO DE 2023

Pablo Gerardo González López

Ciencias de la Computación

Coordinación de los Servicios de Cómputo
Facultad de Ciencias
U.N.A.M.

[Correo] pablog@ciencias.unam.mx
[Telegram] @soyPabloG

Normas y medidas sanitarias del laboratorio

- No se permite ingerir alimentos o bebidas.
- Se debe hacer uso responsable del equipo de cómputo.
  En estas medidas, se incluye cuidar el no dañar el equipo de cómputo con las mochilas al entrar y salir.
  - Usar el equipo únicamente para propósitos autorizados.
  - Proteger sus credenciales de acceso a las aulas y talleres.
  - Acceder solo a archivos y recursos de red que les pertenezcan o que estén disponibles de manera explícita a todo público.
  - Usar solo versiones legales de software protegido por derechos de autor.
- Uso obligatorio de cubrebocas.
- Registrar su asistencia en el formulario correspondiente.
- Evitar asistir si se presentan síntomas de resfriado o COVID-19.

Recursos de apoyo sobre la COVID-19

Pruebas PCR

Laboratorio Nacional de Soluciones Biomiméticas para Diagnóstico y Terapia

FACULTAD DE CIENCIAS

$500 para comunidad U.N.A.M.

http://biosensor.fciencias.unam.mx

Pruebas PCR, de antígenos, serológicas, y asesoría a pacientes

Clínica de diagnóstico COVID, U.N.A.M.

CIUDAD UNIVERSITARIA

COLONIA DEL VALLE

Gratuito para comunidad U.N.A.M.

55 6896 2238 - 744 505 2271

[Instagram] @ClinicaCOVID_UNAM

Atención psicológica ante estrés, ansiedad, o depresión al adaptarse a la “nueva normalidad”

Medicina Conductual y Psicología de la Salud

POSGRADO DE LA FACULTAD DE PSICOLOGÍA

Gratuito para estudiantes U.N.A.M.

medicinaconductual.cu20@gmail.com

[Facebook] @medicinaconductualCU

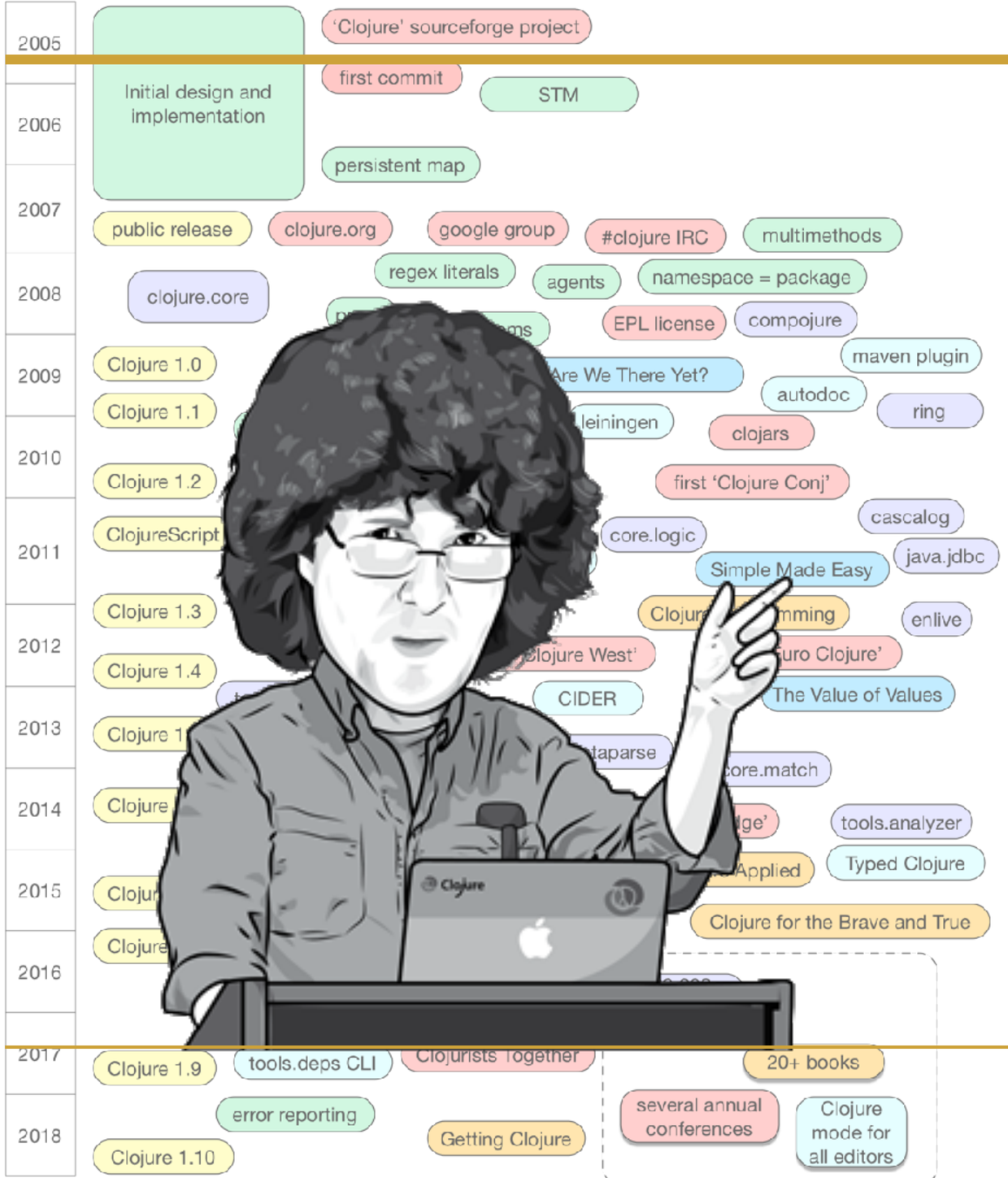
Dinámica de trabajo

- Sesiones por Zoom los martes de 14:15 a 16 hrs.
  - Se registrará la asistencia a estas sesiones.
- Se dejarán diversas actividades a lo largo del semestre. Estas se evaluarán mediante la plataforma Google Classroom.
  - El promedio de estas actividades corresponderá al 20% de la calificación final.

*«The most fundamental problem in computer science is **problem decomposition**: how to take a complex problem and divide it up into pieces that can be solved independently».*

John Ousterhout

Clojure



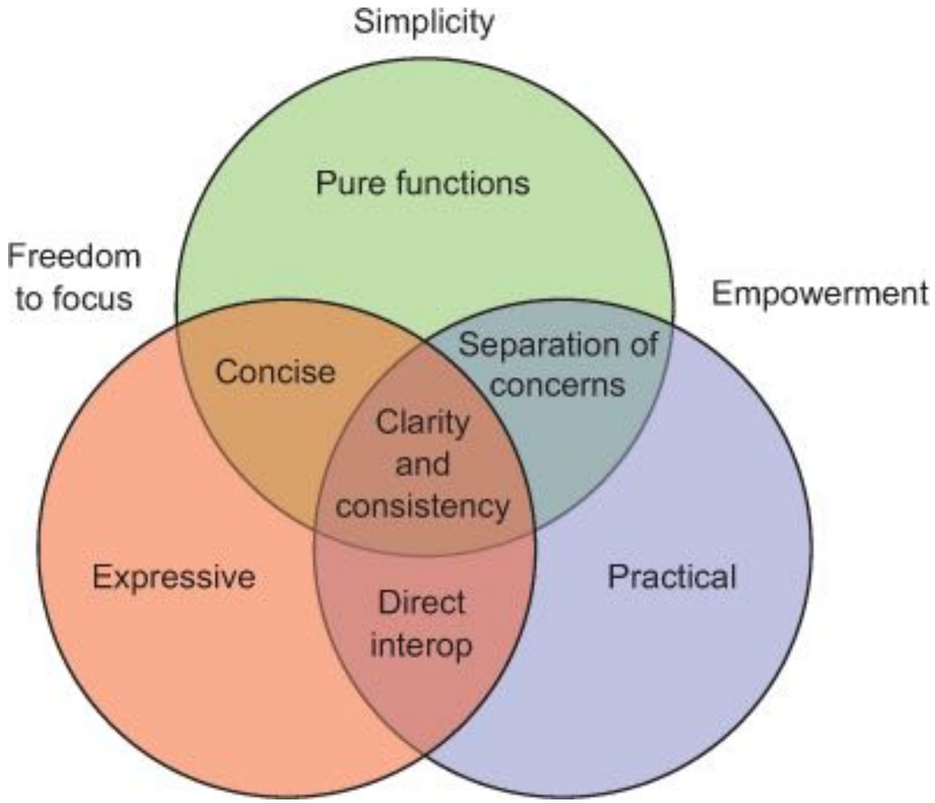
¿Qué es Clojure?

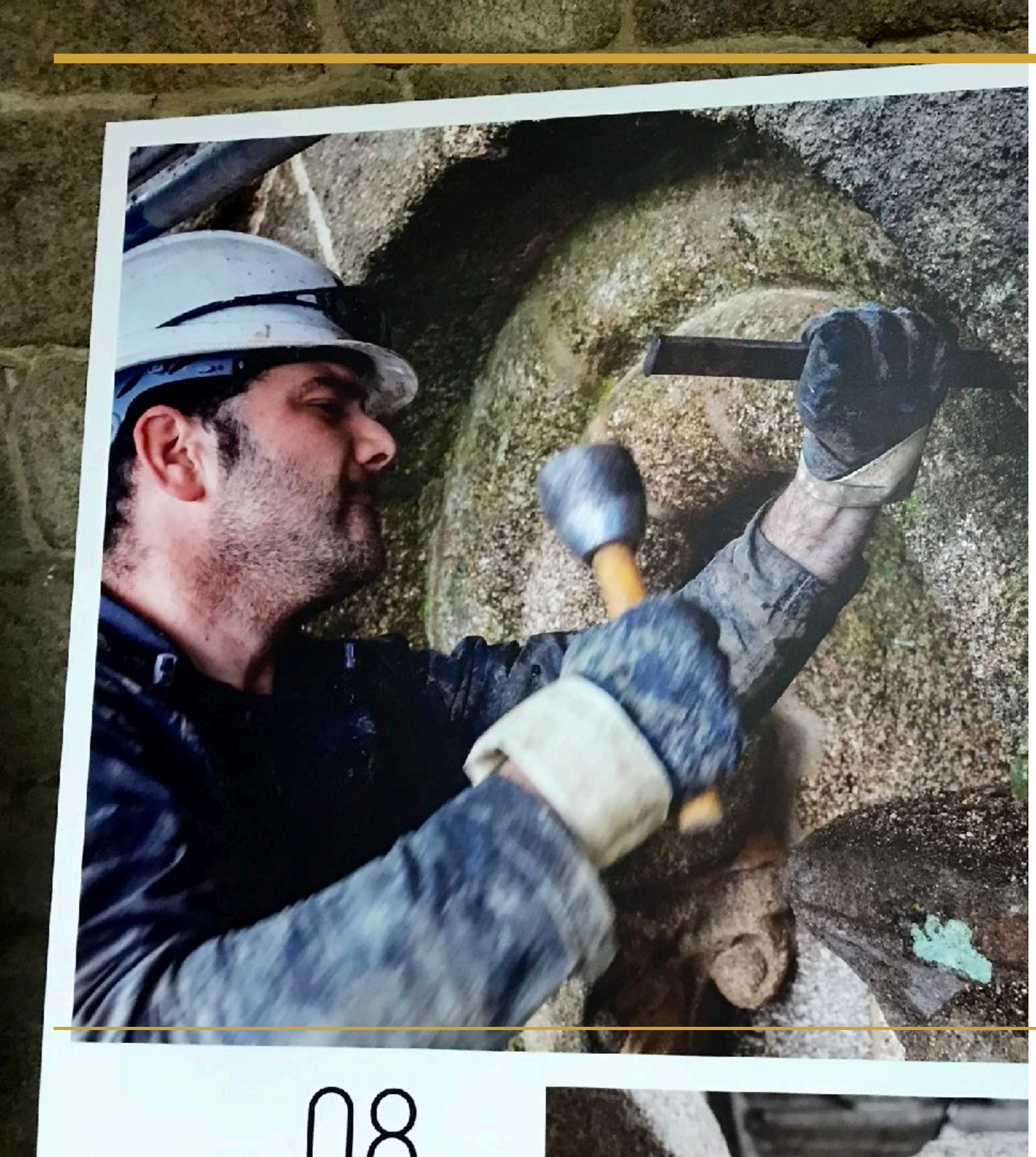
- Ideado por Rich Hickey durante el segundo lustro del siglo XXI.
- Miembro de la familia de los lenguajes de programación LISP.
- Idóneo para desarrollar *software* concurrente y soportar técnicas de programación funcional.
- Simbiótico (*hosted language*). Se ejecuta sobre la JVM, aunque existe una versión (ClojureScript) que se ejecuta en entornos de JavaScript.

Programación Funcional

- Aplicación y composición de funciones.
- Las funciones son objetos de primera clase. Es decir se pueden:
  - Almacenar como valores.
  - Ser pasadas como argumento.
  - Ser devueltas por otra función.
- Funciones puras.

Filosofía de Clojure (*The Clojure way*)





¡Vamos allá!

👩‍💻👨‍💻

Leiningen

¿Qué es y cómo se instala?

- Una herramienta gestora de proyectos de Clojure, capaz de manejar paquetes y automatizar el proceso de compilación.
- Instalarlo requiere de 4 sencillos pasos:
    1. Descargar el `script` (https://raw.githubusercontent.com/technomancy/leiningen/stable/bin/lein).
    2. Agregarlo al $PATH donde el `shell` pueda encontrarlo.
    3. Volverlo un fichero ejecutable.
    4. Ejecutarlo. Automáticamente descargará e instalará todo lo necesario.

https://leiningen.org/

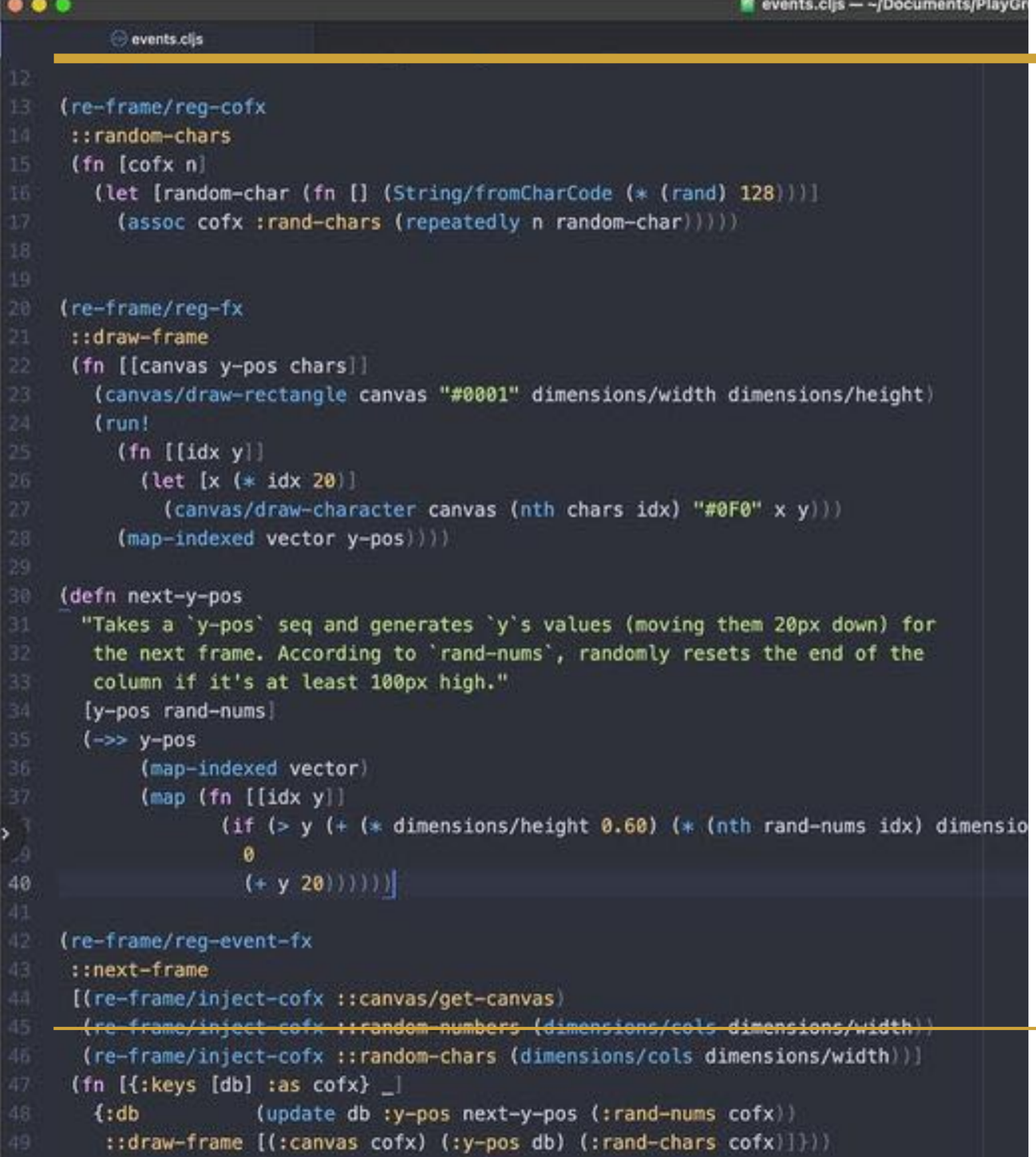
REPL := ***r****eads your input,* ***e****valuates it,* ***p****rints the result, and* ***l****oops, presenting you with a prompt again.*

Flujo de desarrollo

- Se utiliza el REPL como la principal herramienta para experimentar con el código.
- El REPL nos permite interactuar con el proceso de `Clojure` que se está ejecutando en la JVM (o en el entorno del lenguaje anfitrión [`Clojure` es un *hosted language*]).
- Comúnmente el desarrollo de *software* en `Clojure` consta de:
    1. Experimentar en el REPL.
    2. Definir el conjunto de entradas y salidas de una función.
    3. Implementar una solución que transforme las entradas en las salidas esperadas.
    4. Definir la implementación dentro de una función (`defn`) y situarla en el módulo (*namespace*) apropiado.
    5. *Re-factorizar*.
    6. Repetir el proceso.

Flujo de desarrollo

	0	1	2
0	(xor 0 0)	(xor 1 0)	(xor 2 0)
1	(xor 0 1)	(xor 1 1)	(xor 2 1)
2	(xor 0 2)	(xor 1 2)	(xor 2 2)
3	(xor 0 3)	(xor 1 3)	(xor 2 3)



¿Qué más se necesita?

- Para facilitar el flujo de desarrollo es **indispensable** integrar el REPL a nuestro editor de textos.
  https://youtu.be/s5C7NDdp5P0?t=280
- Existen distintas herramientas para los editores más populares. Por mencionar algunas:
  - `Cider` para `Emacs`.
  - `Cursive` para `IntelliJ`.
  - `Calva` para `Visual Studio Code`.
- Cada una puede ofrecer características distintas, pero destacan:
  - Evaluación *inline*.
  - Consulta de documentación.
  - Resalto de sintaxis.
  - Formato automático del código.
  - Navegación dentro del código.
  - *Debuggers*.
- Además, dado que trabajamos con un LISP es ampliamente recomendado el uso de `Parinfer` o `Paredit`.

Clojure 101 - Sintaxis

- **Expresiones (o formas; en inglés *forms*)**
  Código que es válido dentro de `Clojure`. Existen dos tipos de estructuras que son válidas:

  - Representaciones literales de estructuras de datos (conocidas como **literales**, como los números, cadenas, mapas y vectores).

  - Operaciones.

Clojure 101 - Control de flujo

- Existen (al menos) 3 operadores básicos para este fin:
  - `if`
  - `do`
  - `when`

Clojure 101 - def

Clojure 101 - Estructuras de datos... o simplemente los datos

- En `Clojure` existen representaciones de:
  - Números.
  - Cadenas de texto.
  - Mapas (los primos de los diccionarios).
  - `Keywords`.
  - Vectores.
  - Listas.
  - Conjuntos.
  - Expresiones regulares.

Clojure 101 - Funciones

Clojure 101 - *Destructing*

🏋️ Ejercicio

- Escribir una función que regrese la cantidad de elementos en una secuencia.
  - `(= (__ '(1 2 3 3 1)) 5)`
  - `(= (__ "Hello World") 11)`
  - `(= (__ [[1 2] [3 4] [5 6]]) 3)`
  - `(= (__ '(13)) 1)`
  - `(= (__ '(:a :b :c)) 3)`

En resumen...

Clojure es un lenguaje de programación pragmático, funcional, que se ejecuta sobre la JVM.

La principal herramienta de desarrollo es el REPL. Este se debe integrar al editor de texto para permitir un flujo de trabajo óptimo.

Existen distintas estructuras que nos permitirán representar datos en Clojure.

Para programar en Clojure, debemos comenzar a pensar en funciones en lugar de métodos, que es como lo hacíamos en otros lenguajes de programación.

Homework

- Instalar `Leiningen`.
- Instalar una extensión en su editor favorito para desarrollar utilizando el REPL.
- Ver *4 Programming Paradigms In 40 Minutes* (https://www.youtube.com/watch?v=cgVVZMfLjEI)