Certified Tech Developer
The Ultimate Degree

Fundamentación

Programación Imperativa es una de las primeras materias de la Carrera, la cual busca ser el primer acercamiento de los estudiantes al mundo de la programación.

En la sociedad actual, donde las competencias digitales son primordiales en el ambiente laboral, principalmente en las áreas técnicas, “Programación imperativa” busca transmitir las nociones básicas de la programación para poder generar los cimientos necesarios para el crecimiento de los estudiantes y lograr la incorporación de los conceptos fundamentales para formar un profesional competente.

Los contenidos fueron pensados para que los estudiantes incorporen el pensamiento computacional que les va a permitir estructurar sus mecanismos de resolución de problemas.

Objetivos de aprendizaje

Al finalizar la materia, el estudiante será capaz de generar programas que resuelvan conflictos de su cotidiano y tener las nociones para expandir sus capacidades cuando las necesite. De esta forma, no solo podrá replicar lo aprendido si no también tendrá las herramientas para incorporar nuevos conocimientos.

Metodología de enseñanza-aprendizaje

Desde Digital House, proponemos un modelo educativo que incluye entornos de aprendizaje sincrónicos y asincrónicos con un enfoque que vincula la teoría y la práctica, mediante un aprendizaje activo y colaborativo.

Nuestra propuesta incluye clases en vivo con tu grupo de estudiantes y docentes, a los que podrás sumarte desde donde estés. Además, contamos con un campus virtual a medida, en el cual encontrarás las clases virtuales, con actividades, videos, presentaciones y recursos interactivos, para realizar a tu ritmo antes de cada clase en vivo.

A lo largo de tu experiencia de aprendizaje en Digital House lograrás desarrollar habilidades técnicas y blandas, como ser el trabajo en equipo, creatividad, responsabilidad, compromiso, comunicación efectiva y autonomía.

En Digital House utilizamos la metodología de “aula invertida”. ¿Qué quiere decir? Cada semana te vamos a pedir que te prepares para la que sigue, leyendo textos, viendo videos, realizando actividades, entre otros recursos. De esta forma, cuando llegues al encuentro en vivo, estarás preparado para abordar el tema de manera más rica.

Utilizamos actividades y estrategias basadas en los métodos participativos y activos para ponerte en movimiento ya que uno solo sabe lo que hace por sí mismo. Por ese motivo, organizamos las clases para que trabajes en ella de verdad y puedas poner en práctica las distintas herramientas, lenguajes y competencias que hacen a la formación de un programador. Concebimos la clase como espacio de trabajo.

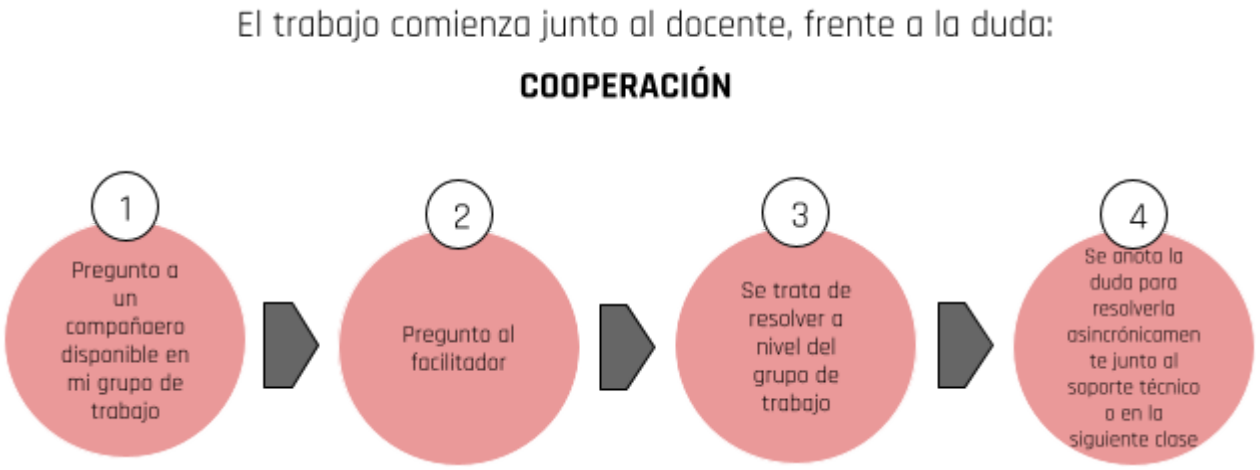
Una de las cuestiones centrales de nuestra metodología de enseñanza es el aprendizaje en la práctica. Por ese motivo, a lo largo de la cursada estarán muy presentes las ejercitaciones, es decir, la práctica de actividades de diversos tipos y niveles de complejidad que te permitirán afianzar el aprendizaje y comprobar que lo hayas asimilado correctamente. De esta forma, se logra un aprendizaje más significativo y profundo, la asimilación de los conocimientos de manera más eficaz y duradera, relacionar lo aprendido con la realidad de los programadores,

fomentar la autonomía y el autoconocimiento, mejorar el análisis, la relación y la comprensión de conceptos, además, ayuda a ejercitar multitud de competencias.

El aprendizaje entre pares es uno de los elementos centrales de nuestra metodología, por eso, en cada clase te propondremos que trabajes en mesas de trabajo junto a tus compañeros y, a lo largo de la cursada, iremos variando la composición de los grupos para potenciar la cooperación. Lo que se propone es un cambio de mirada sobre el curso en cuestión, ya no se contempla al estudiante transitando su camino académico de manera individual, sino como parte de un equipo que resulta de la suma de las potencialidades de cada uno. La distribución en grupos de trabajo fomenta la diversidad y el aprovechamiento del potencial de cada integrante para mejorar el rendimiento del equipo.

La explicación recíproca como eje del trabajo cotidiano no solo facilita el aprendizaje de los compañeros, sino que sobre todo potencia la consolidación de conocimientos por parte de quien explica. Se promueve la responsabilidad, la autonomía, la proactividad, todo en el marco de la cooperación. Lo que lleva a resignificar la experiencia de aprendizaje y a que la misma esté vinculada con emociones positivas.

El trabajo cooperativo permite entablar relaciones responsables y duraderas, aumenta la motivación y el compromiso, además promueve un buen desarrollo cognitivo y social. La cooperación surge frente a la duda: si un estudiante tiene una pregunta, le consulta a algún miembro de su grupo asignado que esté disponible. Si la duda continúa, se convoca al facilitador. Si no lo resuelven, el facilitador pedirá a todos que se detengan para cooperar como equipo en la resolución del conflicto que ha despertado la duda. Así debatirán todos los integrantes de la mesa buscando la solución. Si aún así no pueden resolverlo, anotarán la duda que será abordada asincrónicamente por el soporte técnico o de forma sincrónica en la siguiente clase por parte del profesor.



Todos los días, finalizada la jornada, los estudiantes reconocerán a uno de los integrantes del grupo con quienes compartió ese día. El criterio para ese reconocimiento es la cooperación.

Cada grupo tendrá un facilitador que será elegido a partir de los reconocimientos y generando un sistema de rotación donde cualquiera pueda pasar por ese rol. El facilitador no es una figura estática, sino que cumple un rol dinámico y versátil. El facilitador es un estudiante que moviliza el alcance de los objetivos comunes del equipo poniendo en juego la cooperación. Es aquel que comparte con la mesa su potencial en favor del resto del equipo y que, por lo tanto, promueve la cooperación.

Información de la materia

- Modalidad 100% a distancia
- Cantidad de semanas totales: 9
- Cantidad de encuentros sincrónicos semanales: 3
- Clases virtuales en nuestro campus Playground: 27
- Cantidad de clases en vivo: 27

Requisitos y correlatividades

¿Qué materias tiene que tener cursada el alumno previamente?

- Ninguna, esta es una materia inicial.

¿Qué materias habilita a cursar esta materia?

- Programación Orientada a Objetos
- Front End II
- Testing I

Modalidad de trabajo

Nuestra propuesta educativa está diseñada especialmente para esta modalidad 100% a distancia, mediante un aprendizaje activo y colaborativo siguiendo nuestro pilar de "aprender haciendo".

Los entornos de aprendizaje son tanto sincrónicos como asincrónicos, con un enfoque que vincula teoría y práctica, por lo que ambas están presentes en todo momento.

Contamos con un campus virtual propio en el cual vamos a encontrar actividades, videos, presentaciones y recursos interactivos con instancias de trabajo individual y en equipo para profundizar en cada uno de los conceptos.

Además, realizaremos encuentros online y en vivo con el grupo de estudiantes y docentes, a los que podremos sumarnos desde donde estemos a través de una plataforma de videoconferencias con nuestra cámara y micrófono para generar una experiencia cercana.

Metodología de evaluación

La evaluación formativa es un proceso continuo que genera información sobre la formación de nuestros estudiantes y de nosotros como educadores.

A su vez, se genera conocimiento de carácter retroalimentador, es decir, tiene una función de conocimiento ya que nos permite conocer acerca de los procesos de enseñanza y aprendizaje También tiene una función de mejora continua porque nos permite saber en qué parte del proceso nos encontramos, validar si continuamos por el camino planificado o necesitamos tomar nuevas decisiones para cumplir los objetivos propuestos.

Por último, la evaluación desempeña un papel importante en términos de promover el desarrollo de competencias muy valiosas.

Nuestro objetivo es correrno de la evaluación tradicional, donde muchas veces resulta un momento difícil, aburrido y tenso. Para ello, vamos a utilizar la gamificación, la cual es una técnica donde se aplican elementos de juego para que el contenido sea más atractivo, los

participantes se sientan motivados e inmersos en el proceso, utilicen los contenidos de aprendizaje como retos que realmente quieren superar y aprendan del error.
A su vez, para registrar dicha formación, se utiliza un conjunto de instrumentos, para los cuales es fundamental utilizar la mayor variedad posible y técnicas de análisis.

Criterios de aprobación

- Realizar las actividades de Playground (80% de completitud)
- Asistencia a los encuentros sincrónicos (90% de asistencia)*
- Obtener un puntaje de 7 o más en la evaluación final.
- Obtener un puntaje de 7 o más en la nota final de la materia.

Contenidos

Módulo 1: Pensamiento Computacional

Comprender las nociones básicas de programación, identificando las estructuras cognitivas del pensamiento computacional.

Clase 1: Bienvenida

- Presentación estudiantes
- Pensamiento Computacional

Clase 2: Pensando como la Computadora

- Pasos para hacer un origami
- Del símbolo al texto
    - Comunicación y lenguaje
    - Órdenes mediante símbolos (lightbot)
    - Lenguaje y ambigüedad
    - Órdenes mediante lenguaje textual

Clase 3: Cierre de Semana

- Integrando pensamiento computacional con herramientas lúdicas para programar

Módulo 2: Intro a JavaScript

Comienza la etapa en la que vamos a conocer el entorno de trabajo y las estructuras más elementales y básicas de la programación. Estos elementos son usados por la mayoría de los lenguajes más populares (C, C++, C#, JAVA, Python, JS, etc) por lo que aprenderlas y dominarlas es absolutamente necesario.

Clase 4: Conociendo el entorno de desarrollo: Node y javascript

- Guías de Instalación (Visual Studio Code y Node.js)
    - Instalación de herramientas Windows
    - Instalación de herramientas Linux
    - Instalación de herramientas MacOS
- Visual Studio Code
    - Entorno
    - Crear un archivo
    - Extensiones y atajos
- Más sobre Node.js
    - Arquitectura
    - Verificación de Instalación
- Variables y tipos de Datos
    - Tipos de variables
    - Declaración de una variable
    - Asignación de un valor
    - Declaración con var, let o const
    - Tipos de datos simples (string, number, boolean)
    - Tipos de datos especiales (null, undefined, etc...)

Clase 5: Trabajando con Funciones

- Operadores Matemáticos
    - Asignación
    - Aritméticos
    - Concatenación
- Funciones
    - Declaración y estructura de funciones expresadas y declaradas
    - Invocación
    - Scope

Clase 6: Cierre de Semana

- Construimos una calculadora

Clase 7: Operando Lógicamente

- Retomando Operadores.
    - De comparación.
    - Logicos (AND, OR y NOT)
- True & False Como Conceptos
    - Truthy & Falsy, ¿Que son realmente?

Clase 8: Controlando el flujo de la aplicación

- If / Else
    - Componentes de un if (if, else, else if)
    - Funcionamiento de un if
- If Ternario / Switch
    - If ternario, estructura básica
    - switch, estructura básica, agrupamiento de casos, bloque default

Clase 9: Primer Parcial

Módulo 3: JavaScript intermedio

Continuamos profundizando en el uso del lenguaje, esta vez trabajando con tipos de datos más complejos como son los strings, los arrays y los objetos literales. Vamos a aprender cómo trabajar con estas estructuras de datos, iterarlos y trabajar con ellos dinámicamente.

También vamos a adentrarnos en el sistema de módulos de Node, y comprender cómo podemos usarlo para persistir los datos de nuestras aplicaciones.

Clase 10: Strings y arrays: Trabajando con colecciones

- Métodos de strings
    - length
    - indexOf()
    - slice()
    - split()
    - replace()
- Intro arrays
    - Estructura
    - Posiciones dentro de un array
    - longitud de un array
- Métodos de arrays
    - .push()
    - .pop()
    - .shift()
    - .unshift()
    - .join()
    - .indexOf()
    - .lastIndexOf()
    - .includes()

Clase 11: Ciclos: Repetir...repetir...repetir

- For loop
    - Estructura básica
    - Funcionamiento

- While / do while
    - Estructura básica (while y do while)
    - Funcionamiento

Clase 12: Cierre de Semana

Clase 13: Literalmente: Objetos y Viernes 13! (JSON)

- Objetos Literales
    - Estructura básica
    - Propiedades
    - Métodos
    - Ejecución de un método
    - Trabajando dentro del objeto - this
- JSON
    - JSON vs Objeto Literal
    - JSON.parse()
    - JSON.stringify()

Clase 14: Práctica

Desarrollo de una aplicación completa utilizando lo aprendido hasta el momento como preparación para el segundo parcial.

Clase 15: Segundo Parcial

Módulo 4: JavaScript avanzado

Cerramos la materia poniendo todo lo aprendido en práctica a lo largo del desarrollo de dos pequeñas aplicaciones.

En el camino aprenderemos a usar herramientas que nos van permitir crear un código más eficiente y que sea mantenible a lo largo del tiempo.

Clase 16: Modul-ando

- Modules (Require, Exports)
    - Tipos de módulos (nativos, de terceros, creados)
    - Requerir un Módulo
- Lectura y escritura de archivos
    - writeFileSync()
    - appendFileSync()
    - readFileSync()

Clase 17: Avanzando con funciones

- Arrow Functions
    - Declaración y estructura
- Callbacks
    - Callbacks anónimos y definidos

Clase 18: Cierre de Semana

Clase 19: Arrays y más arrays

- Métodos de arrays Avanzados
    - map()
    - filter()
    - reduce()
    - foreach()

Clase 20: Más métodos de arrays

- Más métodos de arrays
    - slice()
    - splice()
    - sort()
    - find()

Clase 21: Cierre de Semana

Clase 22: Práctica

Desarrollo de una aplicación completa utilizando lo aprendido hasta el momento.

Clase 23: Práctica

Desarrollo de una aplicación completa utilizando lo aprendido hasta el momento.

Clase 24: Examen Final

Módulo 5: Cierre

Como parte final veremos buenas prácticas del lenguaje y temas adicionales con los que pueden seguir su camino de aprendizaje del lenguaje por su cuenta.

Clase 25: JS y Node ++ (parte 1)

- Tecnicismos JS
- Event loop y Asincronismo (como funciona JavaScript por detrás)

Clase 26: JS y Node ++ (parte 2)

- JS en él navegador
- Buenas Prácticas

Clase 27: Cierre de Materia

- Retro Final