



Introducción al Pensamiento Computacional -

Semana 8



© Todos los derechos reservados Universidad Rafael Landívar URL.

DESEMPEÑOS ESPERADOS

- ☰ Desempeños esperados

DESARROLLO DE CONOCIMIENTOS

- ☰ Lógica
- ☰ Principales constructos de algoritmos
- ☰ Controlando el estado de la programación

APLICANDO LO APRENDIDO

- ☰ Aplicando lo aprendido
- ☰ Actividad 1

☰ Actividad 2

☰ Actividad 3

☰ Actividad 4

DE LA TEORÍA A LA PRÁCTICA Y REFLEXIÓN

☰ Recursos

CRITERIOS DE EVALUACIÓN

☰ Rúbrica de evaluación

☰ Diario de experiencias de laboratorio

FUENTES DE REFERENCIAS

☰ Referencias

CRÉDITOS

☰ Créditos

Desempeños esperados



El estudiante:

- 1 Aplica la lógica a los lenguajes de programación.
- 2 Utiliza buenas prácticas en el manejo de ciclos y condicionales.
- 3 Explica cómo controlar el estado de los programas.
- 4 Aplica formas que hagan a los programas más fáciles de manejar.

Lógica



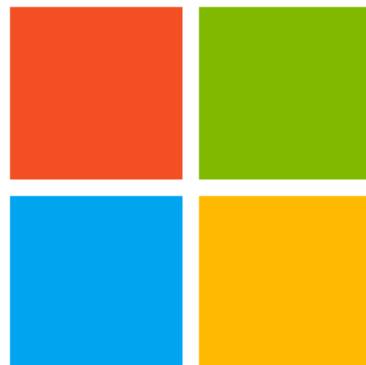
Lógica

En el módulo 2 se aprendió sobre pensamiento lógico. En esta sección se mostrará cómo integrar estos conceptos en la programación.

Recordando: las proposiciones lógicas son enunciados que valúan como falso o verdadero. Para ello se utilizan los operadores: AND, OR, NOT.

C# es un lenguaje de programación, no un lenguaje de lógica. Por lo tanto, no provee el asignar significado a las proposiciones, pero sí provee una forma similar de expresiones.

Analice y realice el siguiente ejercicio de expresiones booleanas y operadores lógicos en C#.

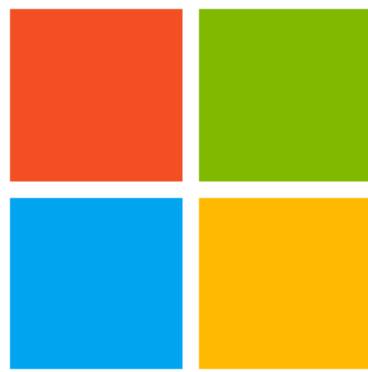


Introducción - Learn

Al agregar lógica de decisión a la aplicación, tiene que evaluar muchos tipos diferentes de condiciones. Para expresar dichas condiciones adecuadamente, necesitará una gama completa de operadores. Imagine que necesita determinar si un valor es mayor, igual o menor que otro valor. O que necesita asegurarse de que la entrada de un usuario coincide con un valor específico.

[MÁS INFORMACIÓN MICROSOFT >](#)





Ejercicio: expresiones booleanas - Learn

"Lógica de decisión" y "lógica de bifurcación" son términos que usan los programadores para describir el cambio de la ruta de ejecución en función de la evaluación de alguna expresión. Por ejemplo, se puede escribir código que evalúe la entrada del usuario.

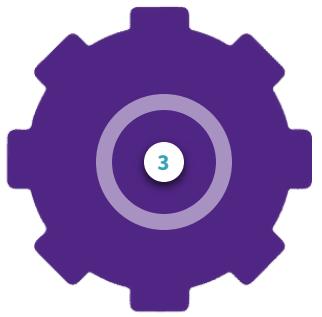
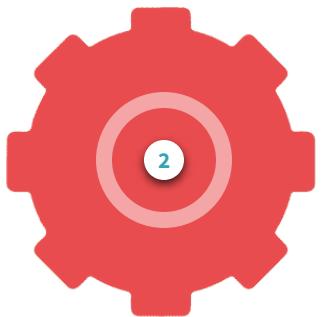
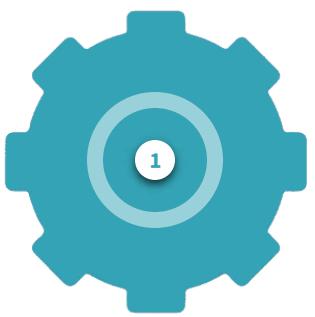
MÁS INFORMACIÓN MICROSOFT >

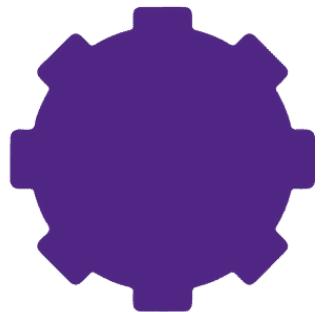
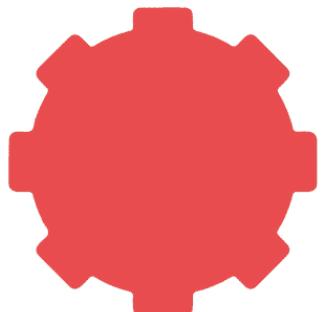
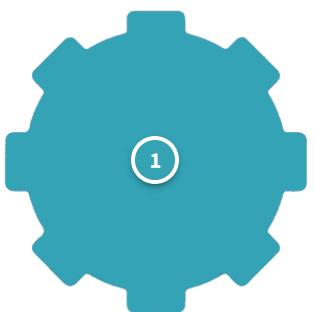


Principales constructos de algoritmos



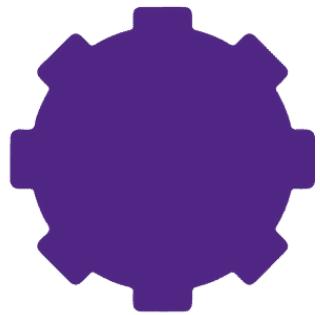
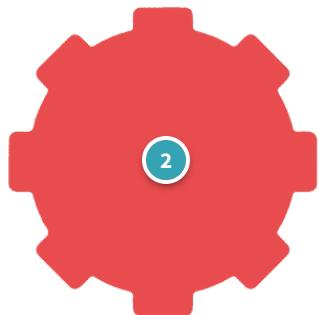
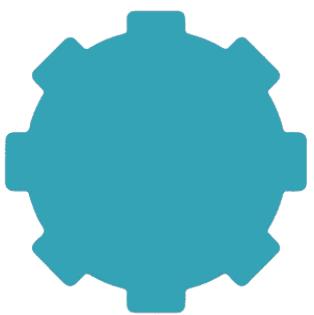
En el módulo 2 se introdujo a los algoritmos como una secuencia de pasos para resolver un problema. Existen muchas formas complejas de algoritmos, pero en esencia, se identifican los siguientes constructos básicos:





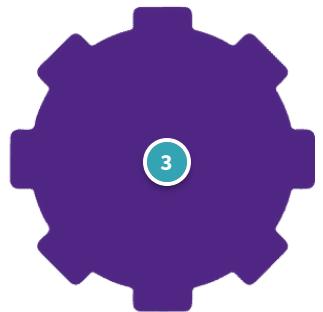
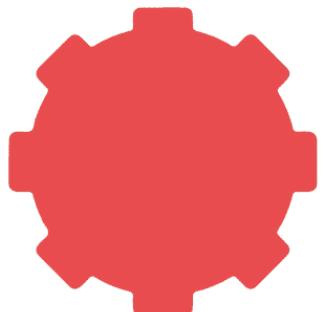
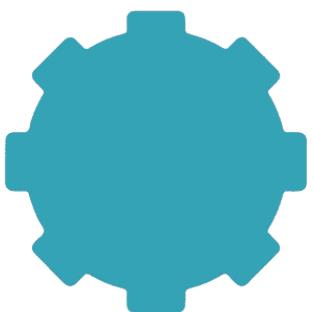
Secuencias

Son series de instrucciones ejecutadas una después de la otra.



Condicionales

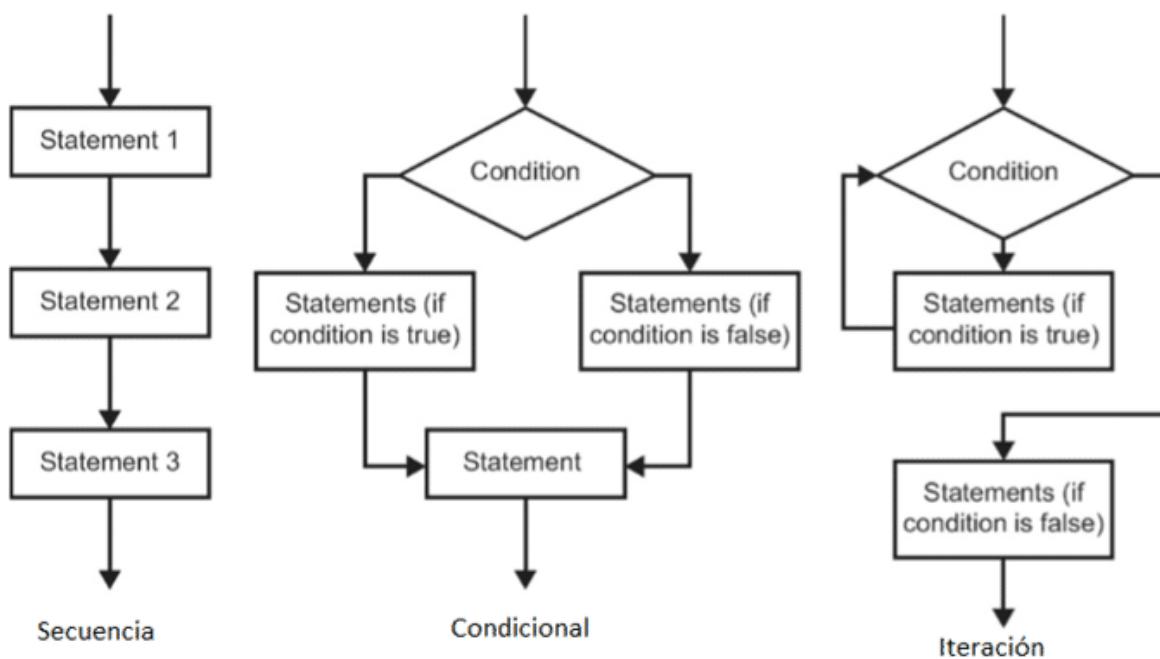
Son puntos de bifurcación donde las instrucciones se ejecutan solo si cierta condición se cumple.



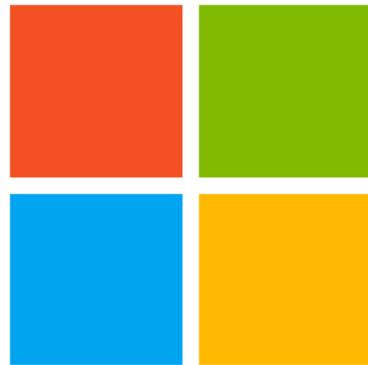
Iteraciones

Son secuencias que se ejecutan repetidamente tanto como se cumplan ciertas condiciones.

En la siguiente figura se muestra cada constructo como un diagrama de flujo.



Realice el siguiente ejercicio de secuencias y condicionales en C#



Ejercicio: operador condicional - Learn

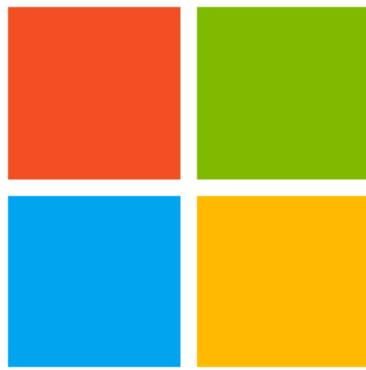
Imagine que necesita determinar rápidamente si la compra de un cliente es válida para un descuento promocional. Si el importe de la venta es superior a 1000, el descuento de la compra es de 100 dólares. Si el importe es de 1000 o menos, solo se descuentan 50 dólares.

[MÁS INFORMACIÓN MICROSOFT >](#)

Realice los siguientes ejercicios de iteraciones en C#, ponga atención a la sintaxis y la ejecución.

While y Do-While:



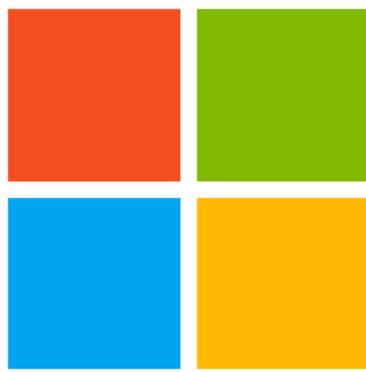


Ejercicio: Instrucciones do-while, while y continue - Learn

A primera vista, las instrucciones do-while y while son otra instrucción de iteración que permite recorrer en iteración un bloque de código y, de este modo, cambiar el flujo de ejecución del código. Pero una vez que se examine su funcionamiento, se pueden identificar mejor los matices de cada instrucción de iteración y cuándo usarlas.

[MÁS INFORMACIÓN MICROSOFT >](#)

For:



Ejercicio: instrucción de iteración for - Learn

A primera vista, la instrucción for es otra instrucción de iteración que permite recorrer en iteración un bloque de código y, de este modo, cambiar el flujo de ejecución del código. Sin embargo, una vez que se examina su funcionamiento, podemos identificar mejor los matices de cada instrucción de iteración y cuándo usarlas.

[MÁS INFORMACIÓN MICROSOFT >](#)

Terminación anticipada de ciclos con break (y otros saltos):

 MICROSOFT



[Instrucciones de salto: Referencia de C#](#)

Las siguientes instrucciones transfieren el control sin condiciones: Para obtener información sobre la instrucción throw que genera una excepción y también transfiere el control sin condiciones, consulte throw. La instrucción break termina la break contenedora más próxima (es decir, los bucles for, foreach, while o do) o la for.

MÁS INFORMACIÓN MICROSOFT >



Controlando el estado de la programación



Lea comprensivamente y realice los ejercicios de tipos de datos, variables y asignación de valores a variables.

Tipos de datos “primitivos” en C#

 MICROSOFT



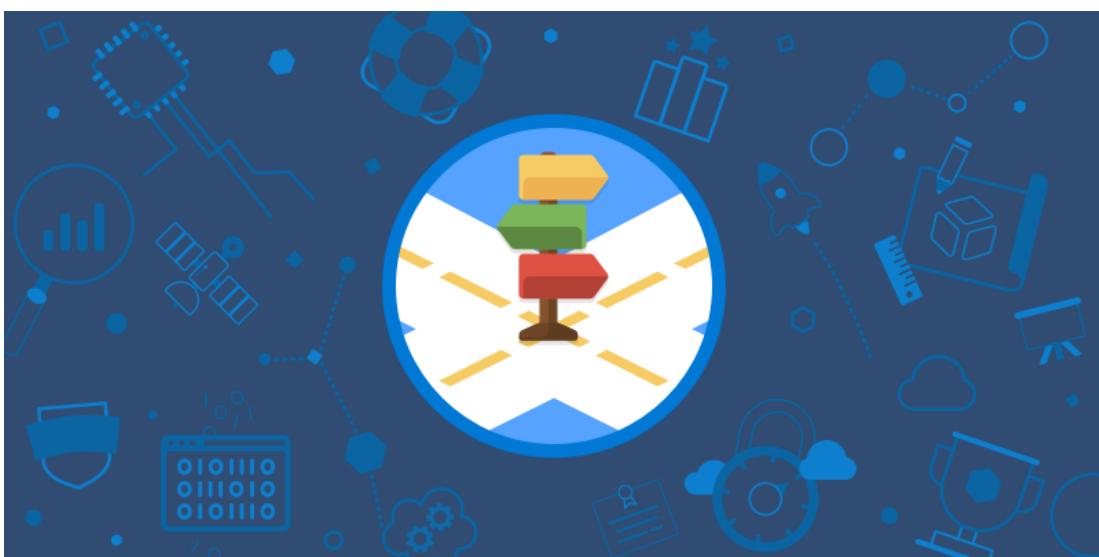
El sistema de tipos de C#

C# es un lenguaje fuertemente tipado. Todas las variables y constantes tienen un tipo, al igual que todas las expresiones que se evalúan como un valor. Cada declaración del método especifica un nombre, el tipo y naturaleza (valor, referencia o salida) para cada parámetro de entrada y para el valor devuelto.

[MÁS INFORMACIÓN MICROSOFT >](#)

Tipos de datos

 MICROSOFT

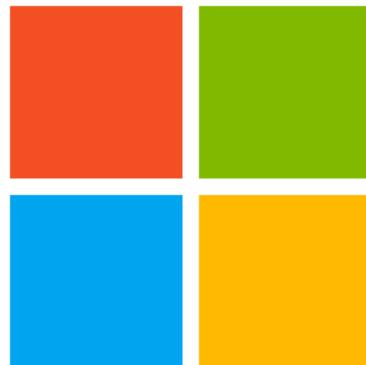


Elección del tipo de datos correcto en el código de C# - Learn

Obtenga información sobre la diferencia entre muchos tipos de datos, cómo funcionan, qué hacen y cómo elegir uno u otro.

[MÁS INFORMACIÓN MICROSOFT >](#)

Variables numéricas (operaciones sencillas)



Ejercicio: Suma sencilla y conversión de datos implícita - Learn

A menudo queremos realizar operaciones matemáticas con datos numéricos. Comenzaremos por la suma en esta unidad y veremos otras operaciones en la unidad siguiente, ya que hay una lección importante que aprender sobre la forma en que el compilador de C# analiza e interpreta el código.

[MÁS INFORMACIÓN MICROSOFT >](#)

Variables literales

 MICROSOFT

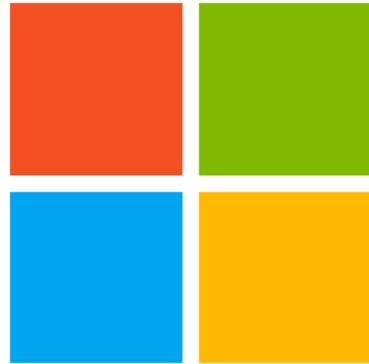


Ejercicio: Valores literales - Learn

En este ejercicio se imprimen mensajes que contienen otros tipos de datos y se aprende por qué los tipos de datos son tan importantes en C#. Un valor literal es un valor codificado de forma rígida que nunca cambia. Anteriormente se ha mostrado una cadena de literal en el panel de salida.

[MÁS INFORMACIÓN MICROSOFT >](#)

Asignación de valores



Ejercicio: Los bloques de código y el ámbito de las variables - Learn

Un bloque de código es una o varias instrucciones de C# que definen una ruta de acceso de ejecución. Normalmente, las instrucciones fuera del bloque de código influyen en cómo, si y con qué frecuencia se ejecuta ese bloque de código en tiempo de ejecución.

[MÁS INFORMACIÓN MICROSOFT >](#)

Keywords

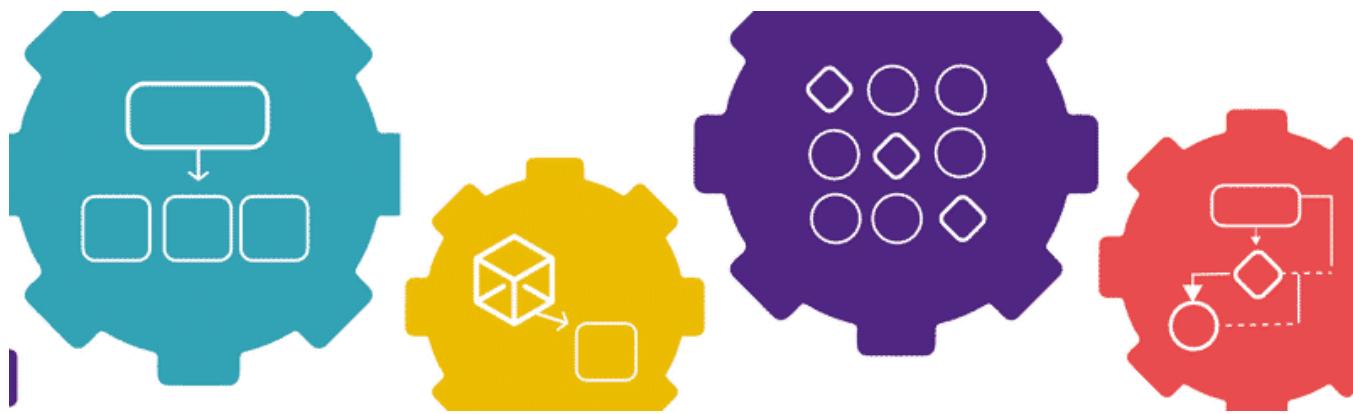




C# Keywords

C# Keywords

MÁS INFORMACIÓN MICROSOFT >



Aplicando lo aprendido



Para los siguientes ejercicios, escriba un programa en C# para responder a las preguntas y retos.

Recomendación: realice el algoritmo en Blockly y luego transfíralo a C#.

 GOOGLE DEVELOPERS

Google Developers

Blockly | Google Developers

A JavaScript library for building visual programming editors.

MÁS INFORMACIÓN GOOGLE DEVELOPERS >



Actividad 1



Actividad 1



Instrucciones

Marque si los siguientes enunciados son verdaderos o falsos.

A las variables se les pueden asignar valores.

Verdadero



Falso

SUBMIT

El uso de “break” rompe la ejecución de un ciclo (iteración).



Verdadero



Falso

SUBMIT

Una condicional es un trozo de código que evalúa un resultado.



Verdadero



Falso

SUBMIT

Actividad 2



Actividad 2



Instrucciones

Regrese al ejercicio del módulo 2 sobre la canción de “Los 12 días de Navidad” ([The 12 Days of Christmas](#)). Convierta el algoritmo en un programa de C#.

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Actividad 3



Actividad 3



Escriba un programa que tome una lista de medidas en grados Fahrenheit y las convierta a Celsius.

Recordatorio: $C = F - 32 \times \frac{5}{9}$

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Actividad 4



Actividad 4

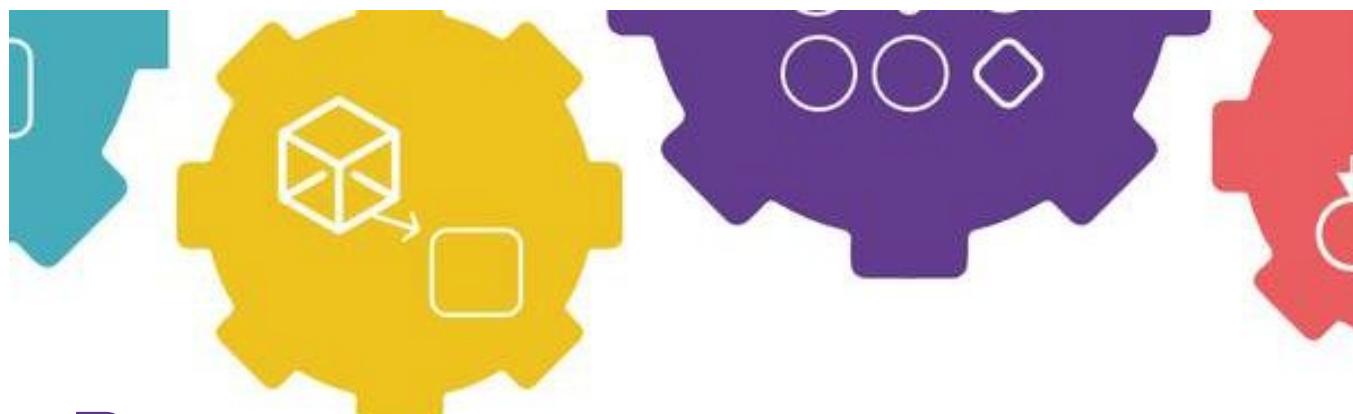


Instrucciones

Escriba un programa que dada una lista de 10 palabras (pueden estar predefinidas o las ingrese el usuario) liste las que son “molestas”. Una palabra molesta es la que cumple con estos criterios: está escrita toda en mayúsculas o terminan con muchos signos de admiración o exclamación.

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Recursos



Recursos

Instrucciones:

Haber instalado el lenguaje de programación en las computadoras que utilizará para realizar las prácticas.



Rúbrica de evaluación



Rúbrica



Descargue la siguiente rúbrica de evaluación.



Rúbrica de Evaluación.pdf

163.6 KB

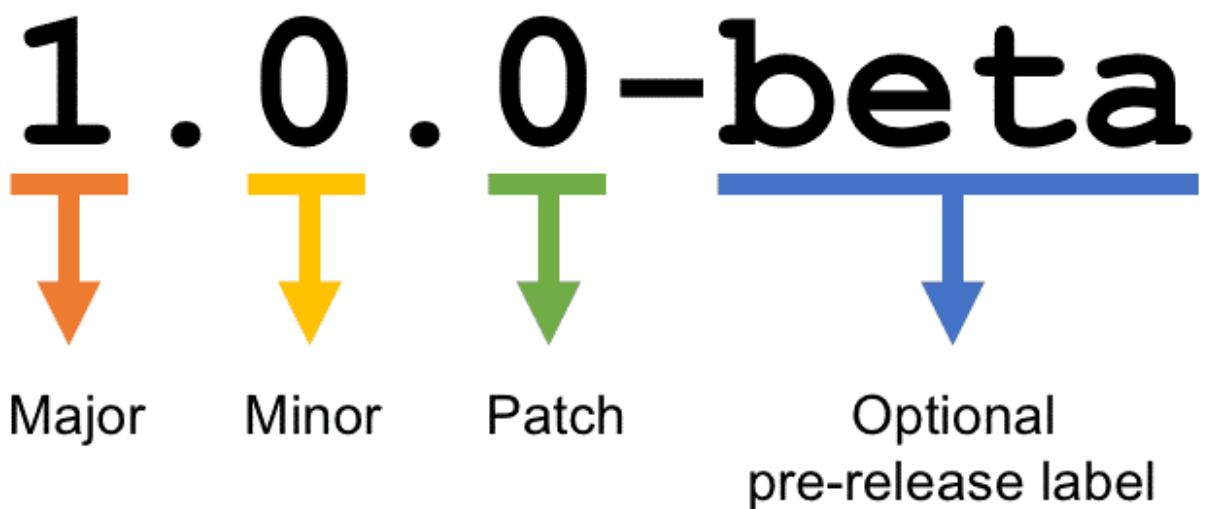


Diario de experiencias de laboratorio



Diario de experiencias del laboratorio

Cada práctica de laboratorio deberá publicarse en una carpeta en el diario de experiencias de laboratorio, bajo la versión 1.0.0.



Referencias

Referencias

Bordigón, F. e Iglesias, A. (2020). Introducción al pensamiento computacional. Universidad

Pedagógica Nacional, Educar Sociedad del Estado y el Ministerio de Educación Argentina.

Beecher, K. (2017). Computational Thinking. A beginner's guide to problem-solving and programming.

ISTE (2018). Computational Thinking. Meets Students Learning. International Society for Technology in Education.

Microsoft (2022). Documentación de C#. <https://docs.microsoft.com/es-es/dotnet/csharp/>

MIT (2022). Computational Thinking, a live online Julia/Pluto textbook. Julia: A Fresh Approach to Computing. computationalthinking.mit.edu. Massachusetts Institute of Technology

Pourbahrami & Tritty (2018). Computational Thinking: How Computer Science Is Revolutionizing Science and Engineering. ENGenious (15) 8-11.

<https://resolver.caltech.edu/CaltechCampusPubs:20181025-110029157>.

Créditos

