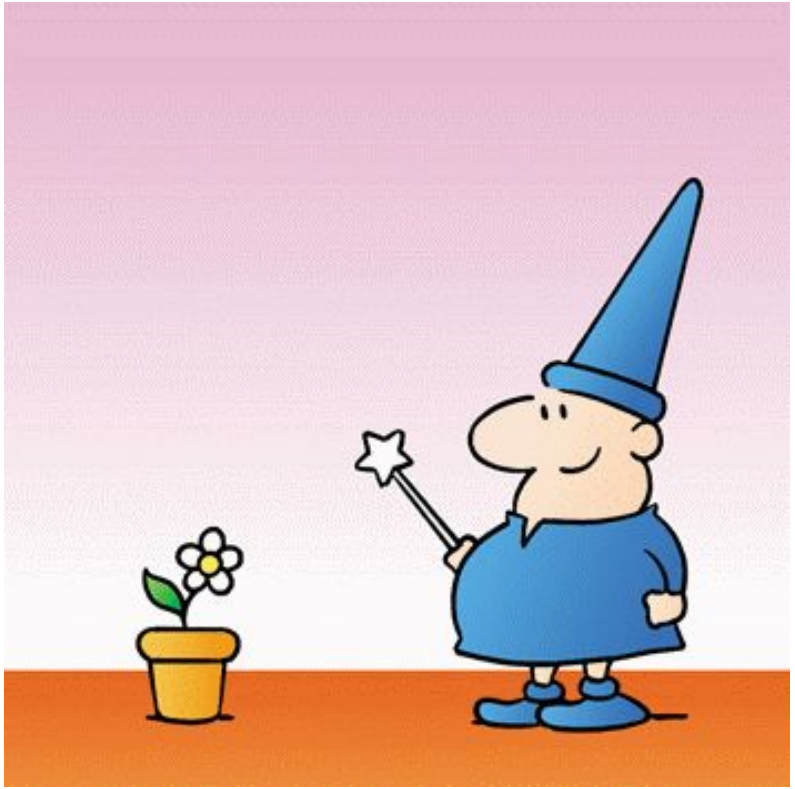
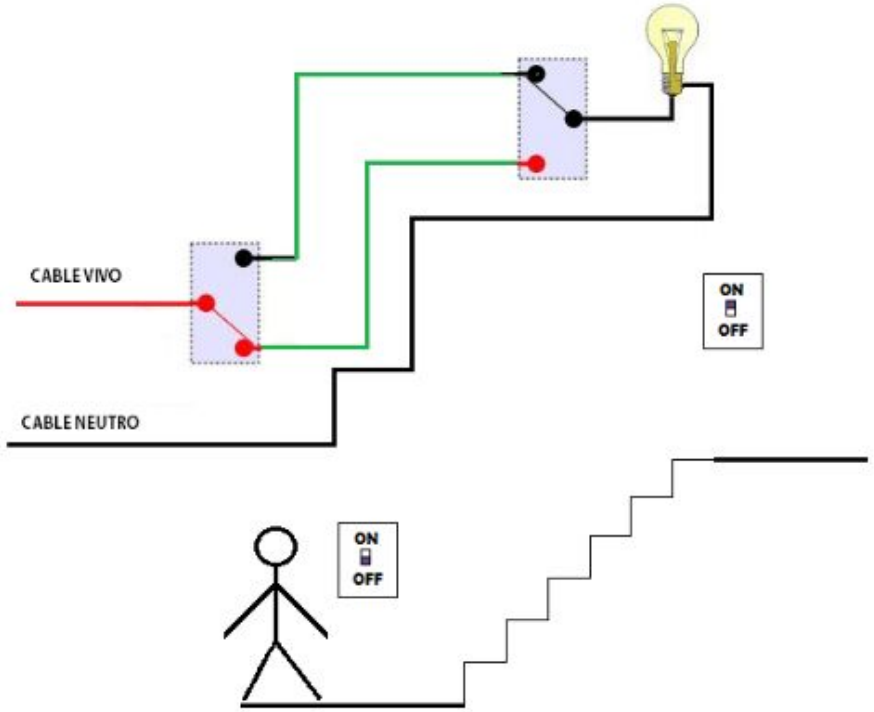
Unidad I: Introducción a la programación en Python

1- Conceptos básicos de programación y su importancia en las ciencias de la Tierra.



¿Cómo funciona?


CABLE VIVO
CABLE NEUTRO
ON
OFF
ON
OFF

¿Qué es?

Hardware y software

Hardware y software, cuerpo y alma

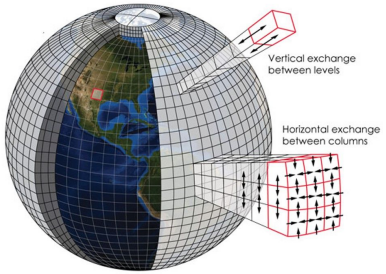
¿Cómo funcionan?

¿Cómo funcionan?

¿Todos pueden crear aplicaciones?

Programación:

- Resolver ecuaciones complejas
- Visualizar y analizar datos
- Modelar fenómenos físicos
- Monitorear

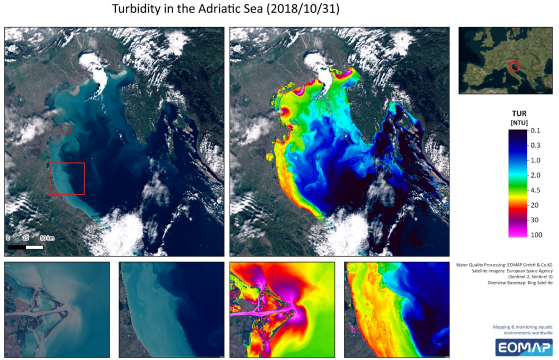


$$\frac{d\vec{v}}{dt} = -\frac{1}{\rho}\vec{\nabla}p - \vec{g} + \vec{F}_{fric} - 2\vec{\Omega}\times\vec{v}$$

Biological/chemical
principles

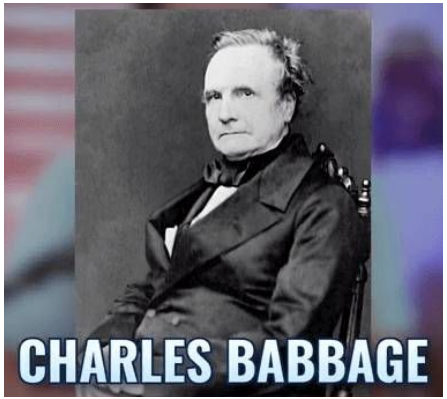
$$\vec{F} = M\vec{a}$$

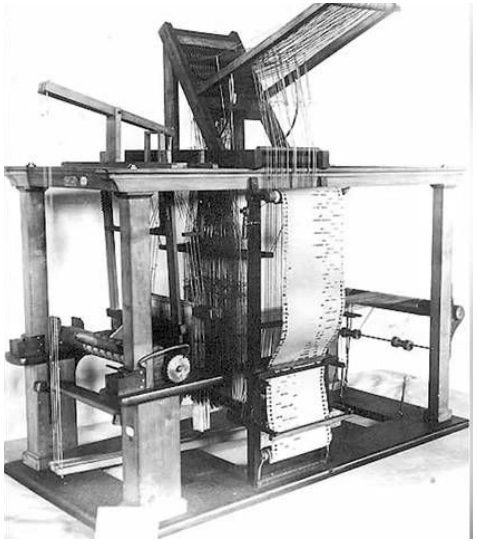
e.g. momentum conservation



Imaginen un mundo sin dispositivos electrónicos …
Por ahí de los años 1780-18..

Ada Byron condesa de Lovelace



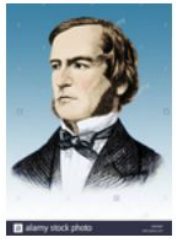


La máquina programable debe ser universal. Es decir, que no debe estar limitada a resolver problemas matemáticos sino que debía resolver cualquier problema que se le pida siempre que se le den las instrucciones. Podría aplicarse a temas como la poesía, la música y otros. Un siglo después, Alan Turing retomaría esta idea.

Las computadoras no pueden pensar (ni podrán), pues solo ejecutan las órdenes que les son entregadas. Un siglo después, Alan Turing tuvo una opinión diferente, pues creía que la inteligencia artificial era posible.

Linea del tiempo De la Electrónica

1800



1854

George Boole publicó: Las Leyes del Pensamiento, donde dió a conocer la álgebra que lleva su nombre.

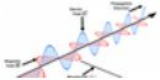
1883

Edison , observó el efécto termoiónico.

Nikola Tesla en 1897 construyó el primer trasmisor de radio.

1887

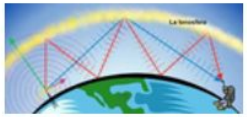
Hertz , observó que al circular una corriente alterna por un conductor, se crean ondas electromagnéticas a su alrededor.



1900

1902

Fleming, utilizó el diodo termoiónico para la detección de las señales de radio.

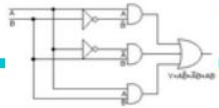


1907

Lee De Forest inventó el triodo.

se inventó la célula fotoeléctricas, el cuál fue explicado en 1905 por Einstein.

1938



Claude Shannon en 1938, demostró que las operaciones booleanas elementales se podían representar mediante circuitos eléctricos.

1947

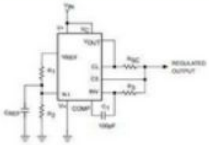
Bardeen, Brattain y Shockley, en 1947 descubrieron el transistor.

1950

1962

Kilby , inventó el circuito integrado.

Fairchild Semiconductor produce el primer circuito integrado regulador de voltaje lineal el uA723.



1968

1982

La empresa Philips fabrica el primer Compact Disc en Hannover (Alemania), desarrollado en forma conjunta por Philips y Sony.

1959

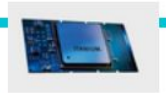
Fue lanzado el Telstar 1 primer satélite de comunicaciones de uso comercial.

2000

2001

1995, lanzan la primer versión del estándar DVD.

Toshiba lazo el primer reproductor de HD-DVD en Japón



Se desarrolla el microprocesador Itanium

2006

2007

invención del iphone.

2021

ENIAC (1936 y 1938)

Hardware y software, cuerpo y alma

Sistema operativo

Windows

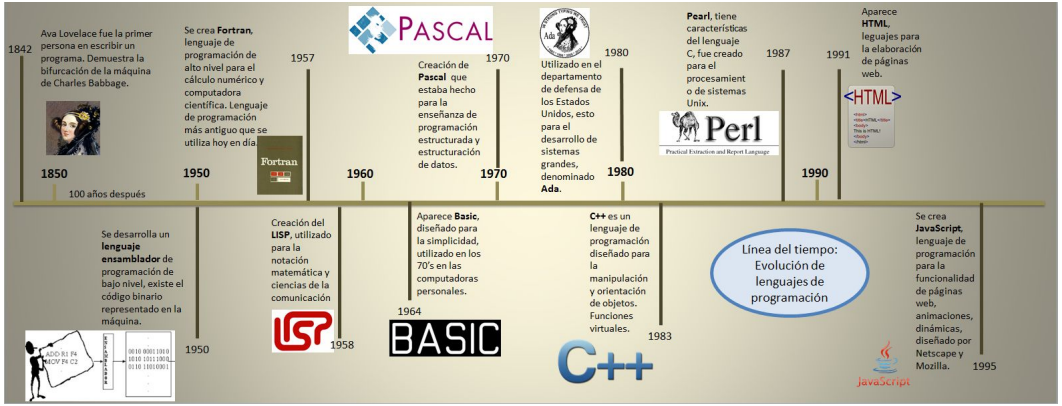
Linux

Ubuntu

Mac/OS

IOS

Android


1842
Ava Lovelace fue la primer persona en escribir un programa. Demuestra la bifurcación de la máquina de Charles Babbage.
1850
100 años después
1950
Se desarrolla un lenguaje ensamblador de programación de bajo nivel, existe el código binario representado en la máquina.
1950
Se crea Fortran, lenguaje de programación de alto nivel para el cálculo numérico y computadora científica. Lenguaje de programación más antiguo que se utiliza hoy en día.
1957
Fortran
1958
Creación del LISP, utilizado para la notación matemática y ciencias de la comunicación
1960
PASCAL
1964
Aparece Basic, diseñado para la simplicidad, utilizado en los 70's en las computadoras personales.
BASIC
1970
Creación de Pascal que estaba hecho para la enseñanza de programación estructurada y estructuración de datos.
1970
1980
Utilizado en el departamento de defensa de los Estados Unidos, esto para el desarrollo de sistemas grandes, denominado Ada.
1980
C++ es un lenguaje de programación diseñado para la manipulación y orientación de objetos. Funciones virtuales.
1983
C++
1987
Pearl, tiene características del lenguaje C, fue creado para el procesamiento de sistemas Unix.
Perl
Practical Extraction and Report Language
1990
1991
Aparece HTML, leguajes para la elaboración de páginas web.
<HTML>
1995
Se crea JavaScript, lenguaje de programación para la funcionalidad de páginas web, animaciones, dinámicas, diseñado por Netscape y Mozilla.
JavaScript
Línea del tiempo: Evolución de lenguajes de programación


LENGUAJES DE PROGRAMACIÓN
1801 / 1936
EL GERMEN DE LA INFORMÁTICA Y LA PROGRAMACIÓN, ENTRE TELAS
1801 — Telar de Jacquard.
1839 — Máquina analítica de Charles Babbage.
1839 — Aportes de Ada Lovelace a la máquina analítica.
1840 — Babbage y Lovelace: inicio de la informática.
1936 — Máquina de Turing.
1940 / 1965
LOS PRIMEROS LENGUAJES DE PROGRAMACIÓN
1945 — Plankalkül.
1949 — Lenguaje ensamblador.
1957 — FORTRAN.
1959 — COBOL.
1964 — BASIC.
1970 / 1990
LOS LENGUAJES DE PROGRAMACIÓN QUE ESTABLECIERON PARADIGMAS
1970 — Pascal.
1972 — Smalltalk, SQL, Prolog y C.
1982 — C++.
1983 — Objective-C.
1990 — Haskell.
1990 / 2015
LENGUAJES DE PROGRAMACIÓN QUE HAN CONSTRUIDO LA WEB
1991 — Python, nuestro preferido.
1992 — Ruby.
1995 — Java, PHP y JavaScript.
2000 — C#.
2003 — Scala.
2005 — Scratch.
2009 — Go.
2011 — Dart.
2012 — Typescript y Kotlin.
2013 — Swift.
2014 — Rust.
4Geeks Academy

¿Qué es un lenguaje de programación?

“En términos generales, un *lenguaje de programación* es una herramienta que permite desarrollar *software* o programas para computadora. Los lenguajes de programación son empleados para diseñar e implementar programas encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora. Lo anterior se logra mediante la creación e implementación de algoritmos de precisión que se utilizan como una forma de comunicación humana con la computadora.”

https://repositorio-uapa.cuaieed.unam.mx/repositorio/moodle/pluginfile.php/2655/mod_resource/content/1/UAPA-Lenguajes-Programacion/index.html

Tipos de lenguajes de programación

- **De máquina:** Es el sistema de códigos interpretable directamente por un circuito microprogramable, como el microprocesador de una computadora.

- **Lenguajes de bajo nivel:** proporciona poca o ninguna abstracción del microprocesador de una computadora.

- **Lenguaje de alto nivel:** su estructura semántica es muy similar a la forma como escriben los humanos, lo que permite codificar los algoritmos de manera más natural, en lugar de codificarlos en el lenguaje binario


1
Primera generación:
Lenguajes máquina y ensambladores.
2
Segunda generación:
Primeros lenguajes de alto nivel imperativo (Fortran, Cobol).
3
Tercera generación:
Lenguajes de alto nivel imperativo. Son los más utilizados en la actualidad (Algol 8, PL/I, Pascal, Modula).
4
Cuarta generación:
Orientados básicamente a las aplicaciones de gestión y manejo de bases de datos (Natural, SQL).
5
Quinta generación:
Orientados a la inteligencia artificial y al procesamiento de los lenguajes naturales (LISP, Prolog).

Lenguajes de alto nivel

Fue hecho por Guido van Rossum en 1990. En la actualidad se desarrolla como un proyecto de código abierto administrado por la *Python Software Foundation*. La última versión estable del lenguaje es la 3.13.1 (3 dic 2024).

CAMPOS DE APLICACIÓN DE PYTHON

Python es el lenguaje más usado, es fácil de aprender y tiene muchos campos de aplicación. **¿Qué esperas para aprenderlo?**

SEGURIDAD INFORMÁTICA

Programa scripts que ejecuten pruebas automáticas para detectar vulnerabilidades.

DESARROLLO WEB

Crea apps web con frameworks como Django, Flask, Pyramid, etc.

TESTING Y QA

Automatiza tests de código y de funcionalidades.

BIG DATA Y DATA SCIENCE

Extrae, procesa, almacena (ETL) y analiza grandes cantidades de datos.

VIDEOJUEGOS

Crea videojuegos con los frameworks: PyGame, PyOpenGl, etc.

MACHINE LEARNING

Escribe modelos de machine learning con librerías como SciKit, SciPy, etc.

Comienza a estudiar Python en EDteam y #NoTeDetengas

ed.team/cursos/python

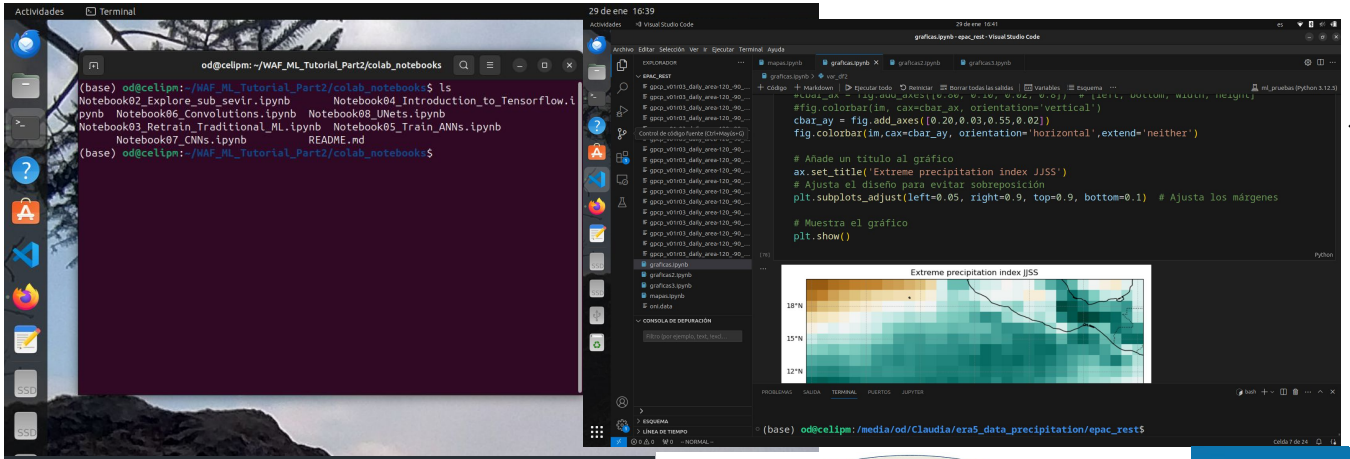
EDteam

Lo que vamos a necesitar:

GitHub

jupyter

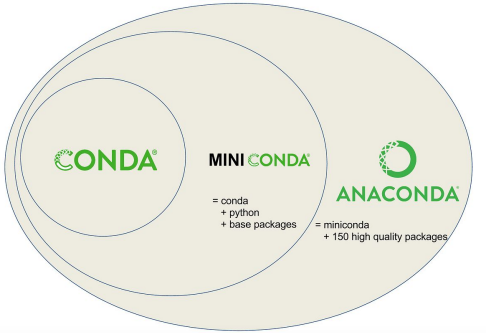
Otras cosas que deben saber que existen:



Visual Studio Code

Vim

YouTube



Conceptos básicos de programación

¿Qué es un algoritmo ?

¿Qué es un código?

¿Qué es un pseudocódigo?

¿Qué es una variable?

Para crear un código es necesario seguir unos pasos

1. Saber qué queremos que haga el código
2. Escribir el pseudocódigo:
    a. Establecer las instrucciones por medio de lógica
3. Abrir la terminal, colab, notebook...
4. ...
5. ...
6. ...
7. ...
8. Llorar...

Para crear un código es necesario seguir unos pasos

1. Saber qué queremos que haga el código
2. Escribir el pseudocódigo:
    a. Establecer las instrucciones por medio de lógica
3. Abrir la terminal, colab, notebook...
4. Comenzar con el algoritmo

¿Qué es el pseudocódigo?

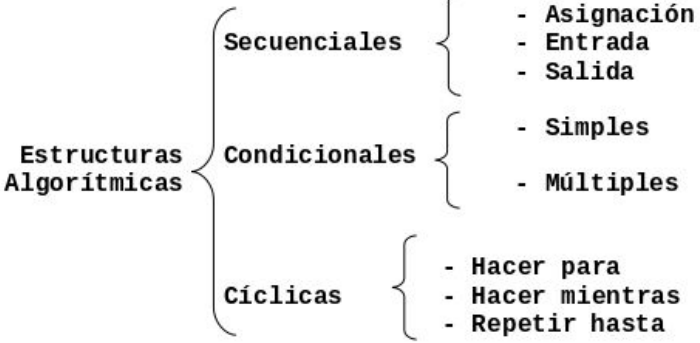
El pseudocódigo es una forma de representar código, como algoritmos, funciones y otros procesos, utilizando una combinación de lenguaje natural y elementos similares al lenguaje de programación.

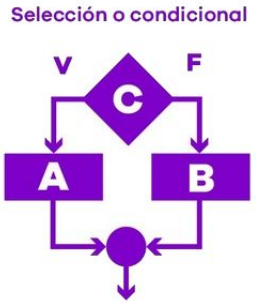
Se llama «pseudocódigo» porque no es realmente ejecutable. En cambio, es una forma de que los humanos comprendan y planifiquen la lógica de la programación — describir los pasos de un programa de forma que sea fácil de entender para los humanos, sin dejar de ser lo suficientemente detallado como para convertirse rápidamente en un lenguaje de programación específico.

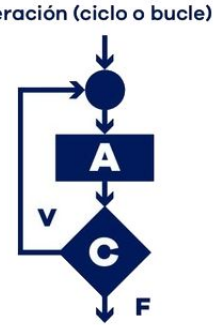
Para crear un código es necesario seguir unos pasos

1. Saber qué queremos que haga el código
2. Escribir el pseudocódigo:
    a. Establecer las instrucciones por medio de lógica

Un algoritmo es ...



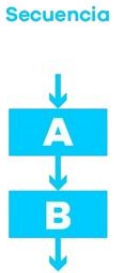


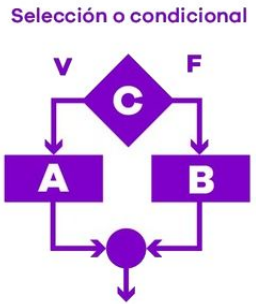


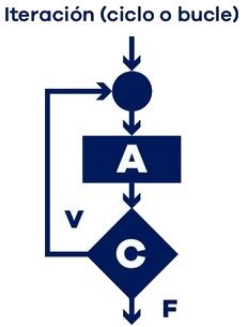
Algo importante: **las variables y ... las sentencias...**

Para el pseudocódigo hay varias formas de organizarse ...

```
Inicio
    Mostrar  'mensaje'
    leer  variable
    Repetir
        instrucción
    Hasta
    si condición
        acción
    sino  alternativa
Fin
```







Para el pseudocódigo...

1. **Abre tu editor de texto:** La mayoría de las veces, el pseudocódigo se escribe en un editor de texto. Puedes elegir tu favorito y abrir un nuevo archivo.
2. **Define tu objetivo:** Determina la finalidad de tu programa o función. ¿Qué quieres que haga?
3. **Sepáralo en partes:** Divide el problema en trozos más pequeños y manejables. Esto puede ayudarte a pensar en el problema con más claridad y facilitar la organización de las piezas para que funcionen donde y cuando deban.
4. **Organízalo en pasos:** Escribe los pasos de tu programa en orden lógico. Utiliza un lenguaje natural y evita utilizar construcciones o métodos de programación específicos, como estructuras de control o conversión de tipos.
5. **Sangría en las líneas:** Utiliza la sangría para mostrar la estructura de tu código. Por ejemplo, puedes sangrar las líneas de código que pertenecen a un bucle.
6. **Pruébalo:** Prueba tu pseudocódigo para asegurarte de que es claro y lógico. Puedes hacerlo recorriéndolo verbalmente o pidiendo a otra persona que lo lea y te informe de lo que cree que debe hacer el pseudocódigo.

Un poco de lógica: (and, or, not)

Operator Name	Operator Symbol	Functionality	Example
Logical AND	and	If both the operands are true, then the condition becomes true.	X = True Y = False X and Y = False
Logical OR	or	If any of the two operands are true, then the condition becomes true.	X = True Y = False X or Y = True
Logical NOT	not	Used to reverse the logical state of its operand.	X = True Y = False not(X and Y) = True

1 = Verdadero , 0 = Falso

Negación

q	¬
1	0
0	1

$\neg p$

Conjunción

p	q	∧
1	1	1
1	0	0
0	1	0
0	0	0

$p \wedge q$

La **conjunción**, similar al castellano «...y...»; es verdadera solo cuando ambas proposiciones son verdaderas. Es equivalente al producto en el Álgebra de Boole.

«Muchos vivos merecerían la muerte y algunos que mueren merecen la vida»

Disyunción

p	q	∨
1	1	1
1	0	1
0	1	1
0	0	0

$p \vee q$

La **disyunción** es verdadera siempre y cuando sean verdaderas alguna de las variables o ambas. No se corresponde exactamente tampoco con la disyunción gramatical «...o...», pues no expresa simplemente la alternancia entre las dos opciones. Es igualmente verdadera cuando ambas proposiciones son verdaderas. Su equivalente en el Álgebra de Boole es la suma.

«¿Me deseas un buen día o quieres decir que hoy es un buen día lo quiera o no?»

Otras que no vimos en clase ...

Condicional

p	q	→
1	1	1
1	0	0
0	1	1
0	0	1

$p \rightarrow q$

El **condicional**, también llamado implicación, niega la posibilidad de que la primera variable sea cierta sin que lo sea la segunda. Corresponde a lo que vulgarmente sería «si... entonces...». Debe apuntarse que la condicionalidad no es bidireccional: p no puede concluirse a partir de q. Tampoco expresa ninguna implicación causal.

«En caso de duda, Meriadoc, sigue siempre a tu olfato»

Bicondicional

p	q	↔
1	1	1
1	0	0
0	1	0
0	0	1

$p \leftrightarrow q$

El **bicondicional** o condicional recíproco restringe su valor de verdad o bien cuando ambas variables son ciertas o cuando ambas son falsas. En lenguaje ordinario sería «...si y solo sí...». En este caso sí es bidireccional de forma que $(p \rightarrow q) \wedge (q \rightarrow p)$. Tanto el bicondicional como el condicional cumplen el principio de que, dadas unas premisas verdaderas, la conclusión nunca puede ser falsa, un principio que será trascendental cuando veamos reglas de inferencia.

«Solo tu puedes decidir qué hacer con el tiempo que se te ha dado»

La programación saca nuestro lado tóxico ...
debemos condicionar y limitar muchas cosas.

Si ocurre x **entonces** pasa y

o ocurre a **entonces** pasa b

Mientras ocurre z **se** hace k

hasta que pase i descansas

Y no olviden que hay que ser MUY EXPLÍCITOS

Hagamos un algoritmo, su función es que los alumnos salgan del salón cuando termina la clase.