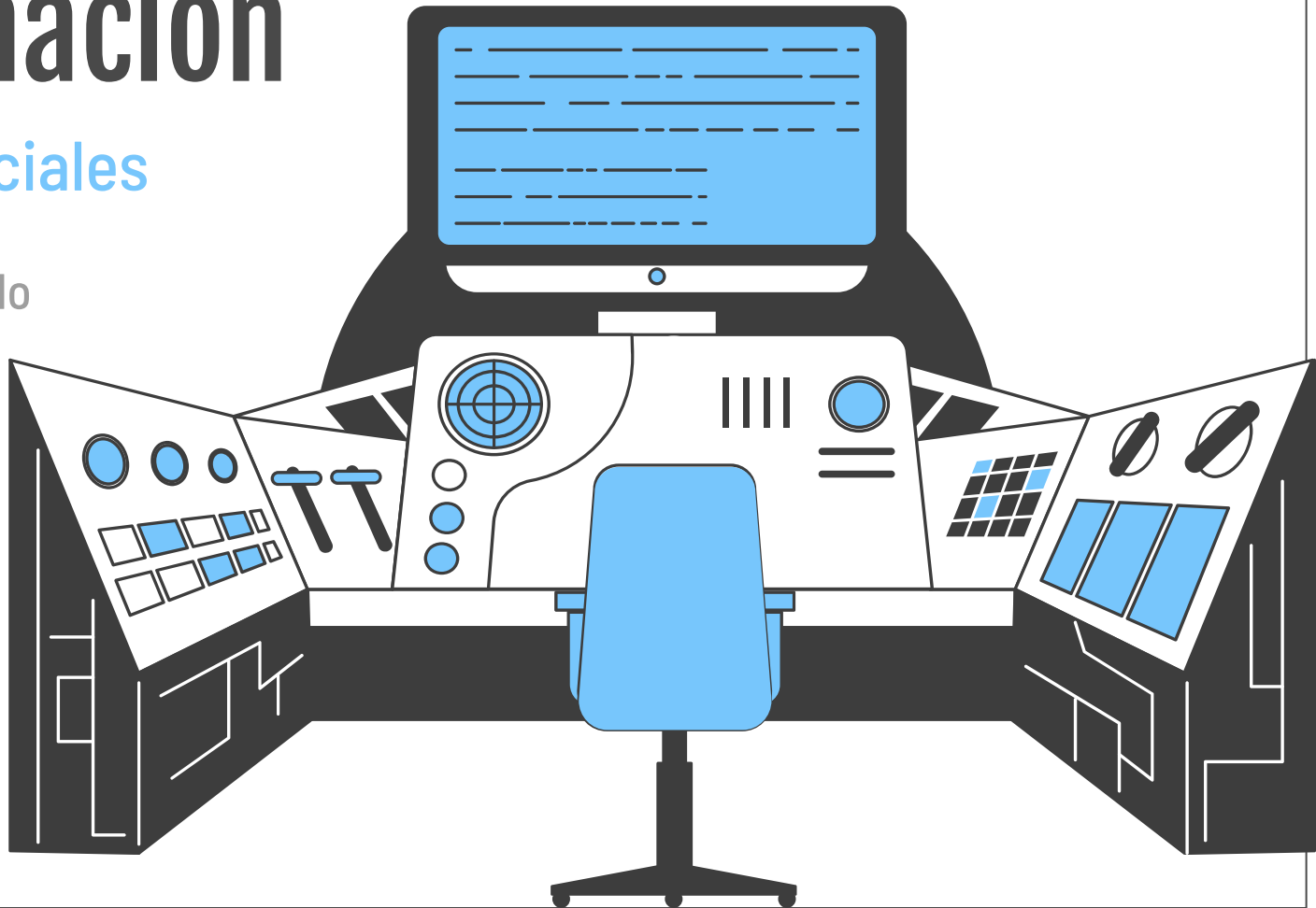
Programación

Conceptos Iniciales

Joaquín Badillo



¿Qué es programar?

¿Programar un horario?

El programa de una escuela...

¿Programar una computadora?

Joaquín Badillo

¿Qué tienen en común estos usos?

Programar es **PLANEAR**

Programar una computadora es crear un «plan» para resolver un problema

Joaquín Badillo

Algoritmos

El plan que le damos a una computadora tiene características especiales...

Recibe una *entrada*
(Esa entrada puede ser vacía, un valor o muchos valores)

Contiene las instrucciones para transformar la *entrada* en la *salida*.

Produce una *salida*
(Esa salida puede ser vacía, un valor o muchos valores)

Una computadora realiza cálculos rápidamente, pero por sí sola no es *inteligente*.

Las instrucciones que le podemos pedir son operaciones básicas* o secuencias bien definidas** de estas operaciones.

---

* Son como "calculadoras": saben hacer aritmética pero también operaciones lógicas
Tienen la ventaja de poder registrar/almacenar datos en *memoria*

** Finitas con pasos bien especificados

Joaquín Badillo

¿Por qué nos importan los algoritmos?

Velocidad

Una computadora puede hacer ciertas operaciones a velocidades que ninguna persona podría. **Nos gusta la inmediatez**.

Seguridad

El avance tecnológico se puede usar como herramienta... pero también como arma. Es necesario utilizar estos avances para **protegernos**.

Utilidad

Los procesos nos permiten **transformar** datos brutos en información. El *software* nos ayuda en una gran variedad de áreas de estudio (yo argumentaría que está presente en todas, aunque se vale que cuestionen esta postura)

Resolución de Problemas

Pensar algorítmicamente te ayuda a resolver problemas con los puntos anteriores.

Manos a la obra

Joaquín Badillo

¿Lógica?

$$¿p \implies q \implies r?$$

Elemental

Joaquín Badillo

0

¡La lógica (bivalente*) es muy relevante en la computación!

Por ahora hablaremos de lógica cuando nos importe evaluar si un enunciado es *verdadero* o *falso*

¿Pero cómo puede saber una computadora que es verdadero? -_-

¡Con condiciones! Las verdades vienen de checar si se cumple (o no se cumple) una condición que la computadora puede evaluar. Con estas condiciones podemos además hacer que se ejecute código distinto dependiendo de cuál sea el caso.

1

* Bivalente: 2 valores
La lógica bivalente tiene 2 "valores" de *verdad*: **Verdadero** o **Falso**

Joaquín Badillo

0

Vamos a ver cómo una computadora puede checar **equivalencias**

La computadora tiene representaciones para los datos y una computadora no va a confundir el valor de dos números.

Por ejemplo si almaceno el número 42 y le pido ese dato a la computadora después, me va a devolver 42... sería muy raro que de repente me dijera que ese 42 ahora vale 3.

1

Técnicamente podemos cambiar el significado de esa representación
Concepto: *Casting*

0

Representaciones

Para la computadora almacenar datos es básicamente contar, solo que este conteo es medio especial, la computadora lo puede hacer prendiendo y apagando *cosas*.

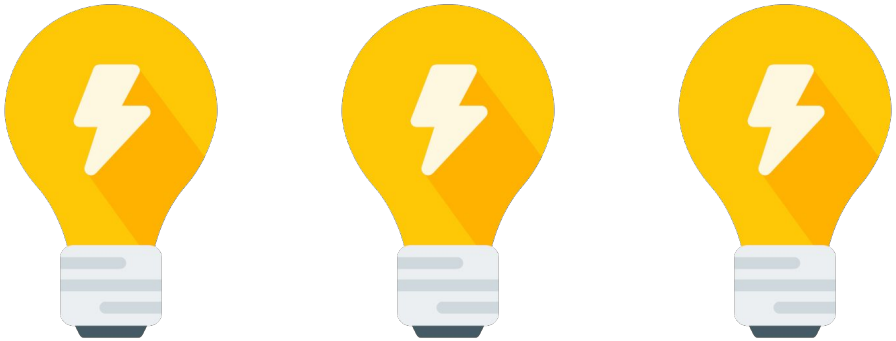
(Sí, literal un valor en la memoria de la computadora tiene sentido con tan solo contar... para datos más "complejos" como los *Strings*, hay trucos con dicha representación).

1

Joaquín Badillo

0

Contar del 0 al 7



1

Joaquín Badillo

0

Con el ejercicio anterior ya es más entendible como representar un número *natural*[*] en una computadora.

Para tener enteros negativos usamos otro foquito adicional para el signo y los valores negativos los contamos "al revés" (esto es irrelevante para los fines de este curso y se hace así para que las restas funcionen jajaja).

Para tener números decimales usamos algo como notación científica y tenemos "2 conteos", una parte para el coeficiente y otra para el exponente. Para incluir negativos también guardamos un foquito para el signo.

1

* Los números naturales son los enteros positivos y el cero:
$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$

Joaquín Badillo

0

Ahorita no nos importa tanto ser específicos con las representaciones que tiene la computadora, solo quiero dejarles claro que la computadora tiene una forma de “contar”.

1

Nota: Para los caracteres definimos un orden tal que existe un primer carácter. Las cadenas de carácteres son muchos espacios de memoria en los que almacenamos carácter por carácter..

0

Como al contar estamos identificando de forma única una cantidad, siempre y cuando respetemos una convención en nuestro conteo, sabemos que dos cantidades son iguales si tienen la misma representación.

Por ejemplo, nosotros sabemos que dos números son distintos cuando sus **dígitos** son **distintos**, o bien sabemos que son iguales cuando sus **dígitos** son **idénticos**. (Al menos cuando lo tenemos escrito en notación decimal, para una fracción tenemos nuestros trucos)

1

Joaquín Badillo

0

Al determinar si un número (en binario) es igual a otro, solo hay dos posibles estados:

Sus dígitos son iguales

Son dígitos son distintos

¡Boom!

Tenemos 2 proposiciones: pueden ser verdaderas o falsas exclusivamente

$$x = y$$

$$x \neq y$$

1

Al escribir esto en Python necesitamos usar 2 signos "=" porque uno solo ya se entiende como *asignar* (guardar en una variable).

Joaquín Badillo

0

Siguiendo el razonamiento del conteo hay una forma de determinar si un número es mayor a otro viendo o sus dígitos.

$$x > y$$

Podemos usar la proposición opuesta y checar si un número es menor que otro.

$$x < y$$

También podemos usar la relación "es mayor o igual que", así como la relación "es menor o igual que"

$$x \geq y$$

$$x \leq y$$

1

Joaquín Badillo

Proposición	Representación en Python
x es igual que y	x == y
x es distinto de y	x != y
x es menor que y	x < y
x es menor o igual a y	x <= y
x es mayor que y	x > y
x es mayor o igual a y	x >= y

Enunciados condicionales

If Si (esto es verdadero)

Entonces haces esto

Elif En cambio si (esto otro es verdadero)

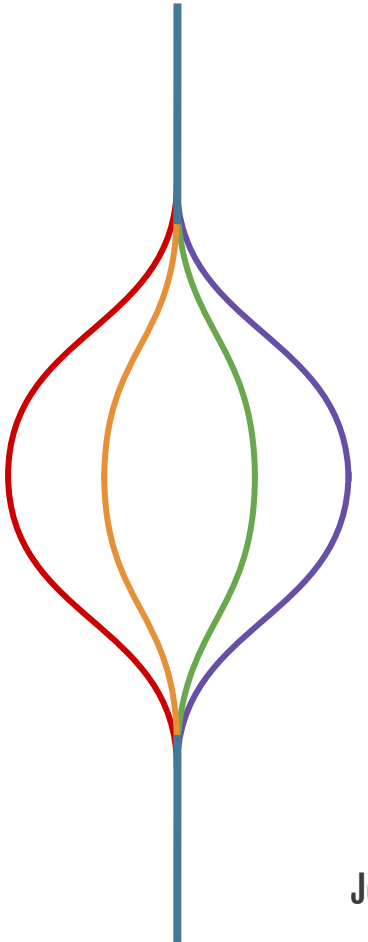
Realiza esto

⋮ En cambio si (esto otro es verdadero)

Realiza esto

Else De lo contrario

Haces esto



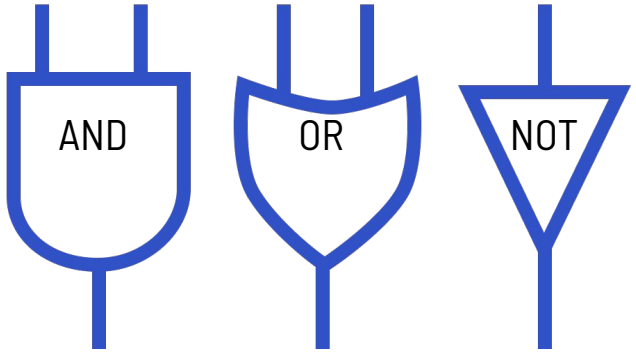
Joaquín Badillo

Operadores lógicos

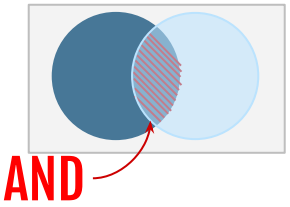
Si las proposiciones solo pueden ser verdaderas o falsas el “else if” parece ser medio aburrido y probablemente innecesario... aunque tener al menos 3 casos puede ser relevante por la *propiedad de tricotomía* de algunas relaciones como “<”.

Pero tener una cantidad arbitraria de enunciados “*else if*” es útil porque dentro de un solo enunciado podemos evaluar si distintas condiciones se cumplen.

¿Cómo *juntamos* proposiciones?



Joaquín Badillo



AND

Si te dijera que leí un libro **y** no leí ese mismo libro, dirías que estoy loco. Esto se debe a que te estoy diciendo una contradicción, estoy tratando de decir que simultáneamente algo es verdadero **y** falso.

En lógica definimos una *operación* sobre proposiciones llamada la conjunción (AND). Para que algo el resultado sea verdadero ambas proposiciones deben ser verdad.

En programación tiene utilidad esta operación porque podemos hacer que algo se ejecute siempre y cuando se cumplan 2 (o más) condiciones.

Si (tu número es impar y sus dígitos suman 3)
    Entonces tu número es divisible entre 3, pero no es divisible entre 6

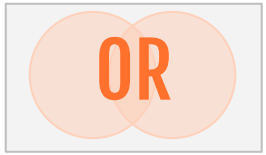
Algunos ejemplos con aplicaciones más evidentes:

Si (este usuario existe y su contraseña coincide)
    Inicia sesión

Si (le gustó esta serie y le gustó esta peli)
    Recomienda esta peli

Si (el mes es febrero y el año es bisiesto)
    El 29 es el último día del mes

Si (el usuario escoge pago con tarjeta y en línea)
    Dirige al usuario a la página de transacciones



Joaquín Badillo

OR

Otra forma en la que se pueden juntar proposiciones es con la disyunción lógica (OR). Esta operación se pregunta si se cumple al menos una de las proposiciones dadas: ¿es esto o esto verdadero?

Aunque en nuestro uso cotidiano del nexo "o" no consideramos que se cumplan ambas proposiciones, en lógica si se llegan a cumplir ambas seguimos considerando el enunciado como verdadero. Si te digo algo como: no recuerdo que desayuné... comí uvas o piña.

En el caso de que haya comido uvas y piña te habría dicho la verdad (desde el punto de vista lógico)

Ejemplos aplicados

Joaquín Badillo

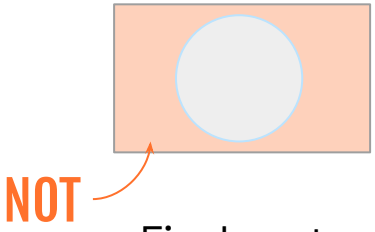
Si (olvidaste tu correo u olvidaste tu contraseña)
    Mandar correo de recuperación

Si (le gusta Zoé o le gusta Café Tacvba)
    Recomiéndale rock alternativo

Si (el método de pago es tarjeta de crédito o tarjeta de débito)
    Solicitar los datos de la tarjeta para el pago

Si (tu escoges la clase de guerrera o asesina)
    Recibe una daga inicialmente

Joaquín Badillo



NOT

Finalmente se tiene el operador "lleva la contraria"* (NOT). Este solo toma una proposición y regresa el valor de verdad opuesto.

Ejemplo ilustrativo:

➔ ¿Zeus es un perro?
➢ No

La proposición "Zeus es un perro" es falsa.
Mientras que decir "**no es el caso** que Zeus sea un perro" es verdadero.

* En español se llama negación jajaja

Ejemplos aplicados

Si (no es el caso que la contraseña ingresada es correcta)
    Enviar un correo para informar que intentaron entrar a su cuenta

Si (no habías programado antes)
    Espero que te esté gustando aprender :)

De lo contrario
    Espero que puedas ver los temas con una nueva perspectiva ;D

Resumen

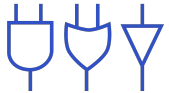
Joaquín Badillo

Programar es escribir un “plan” detallado para que dada una entrada se tenga una salida deseada.

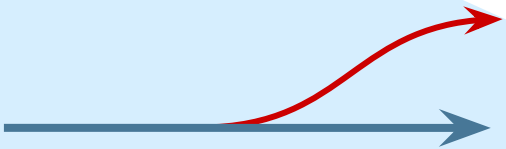
Las computadoras pueden realizar aritmética  $(+, -, \times, \div)$

Las computadoras tienen representaciones para datos que están fundamentadas en contar (en binario)

Las computadoras pueden evaluar expresiones lógicas 1 0

Podemos operar lógicamente sobre proposiciones 

Podemos crear “casos” para ejecutar instrucciones distintas



¿Dudas?

Atribución

Ilustraciones de Stories
Plantilla inspirada en *Technology Consulting* de *Slidesgo*

Joaquín Badillo