



# Repetición Condicional

Teoría 4.A.

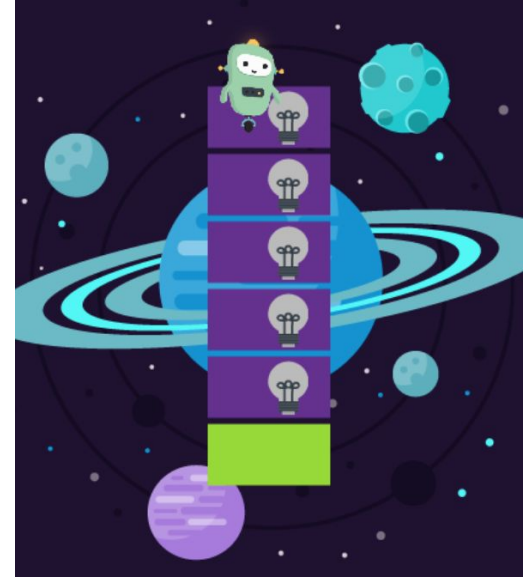
# A trabajar

Realizamos del libro del “Ciclo de Secundaria”, el ejercicio “Super Tito 1” de la sección “Repetición Condicional”.

---

## Super Tito 1: ¿Qué aprendimos?

¿Qué diferencia hay en este problema con respecto a los que veníamos trabajando hasta ahora?

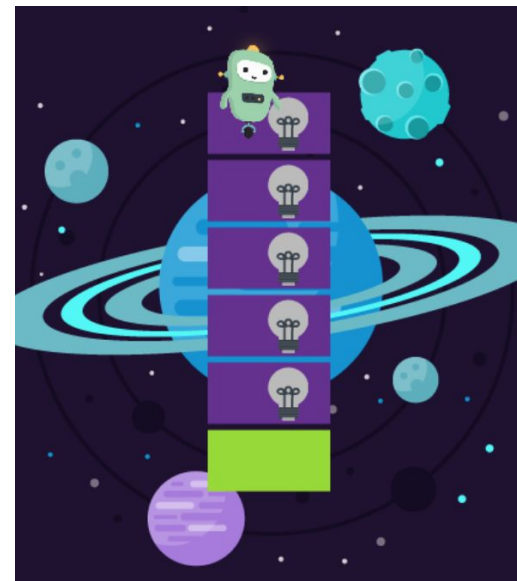


## Super Tito 1: ¿Qué aprendimos?

Ahora el escenario cambia, pero de forma distinta a lo que sucedía hasta ahora. Antes, la cantidad de lugares era fija, y sólo cambiaba el hecho de que podía o no haber un determinado elemento una ubicación particular.

Ahora la cantidad de lugares del escenario es variable.

Nuevamente, es imposible solucionar este problema con las herramientas al momento. Hay que repetir para llegar al final del camino, pero ¿Cuántas veces repetir sí el escenario cambia y no sabemos exactamente dónde terminará cada vez el camino?

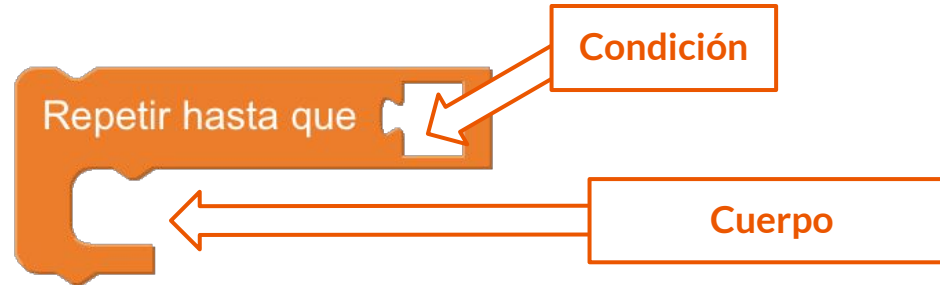




## Repetición Condicional

- La **repetición condicional** es otra forma de **repetición**.
- Se suma entonces a la repetición simple y a la alternativa como una **estructura de control del flujo**.
- No es una nueva forma de organizar el código, porque sigue siendo una repetición.
- A diferencia de la repetición simple, **la cantidad de veces a repetir no es fija**, sino que depende de una condición.
- Se repite **hasta que cierta condición se cumpla**, es decir, hasta que sea verdadera.
- Es un **comando compuesto**, que espera una **condición** y un **cuerpo**.

## Repetición Condicional





## Repetición Condicional

Lo más difícil de comprender de esta estructura es cuándo se evalúa la condición.

La condición se evaluará múltiples veces:

1. Al llegar por primera vez al punto donde se encuentra la repetición condicional, la máquina evaluará la condición una primera vez. Si es falsa, ejecutará el cuerpo (si es verdadero no ejecutará el cuerpo y seguirá con el comando siguiente a la repetición)
2. Luego volverá a evaluar la condición. Si vuelve a ser falsa, volverá a ejecutar el cuerpo (si es verdadera, continuará con el comando siguiente)
3. Seguirá realizando 2 hasta que la condición sea verdadera.



# Repetición Condicional

¿Cuántas veces se ejecuta el cuerpo?



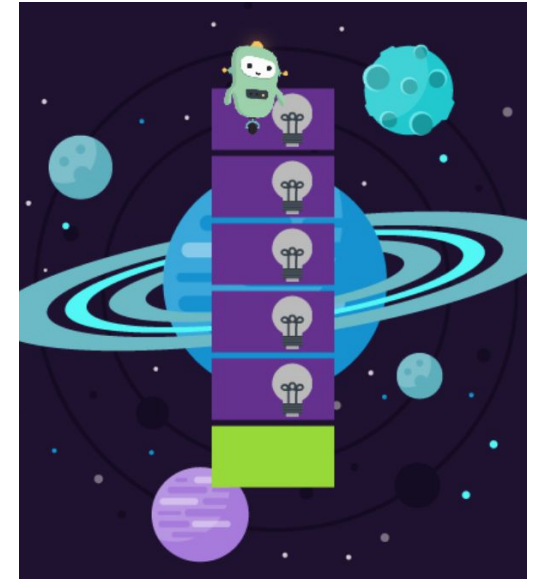
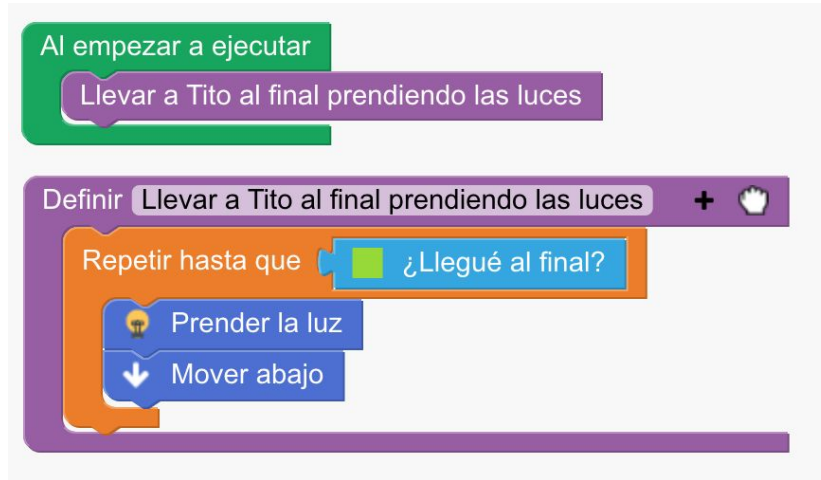


# Repetición Condicional

¿Cuántas veces se ejecuta el cuerpo?

Las que sean necesarias para que se cumpla la condición (podría ser ninguna).

## Super Tito 1: ¿Qué aprendimos?





# Momento de dudas o consultas

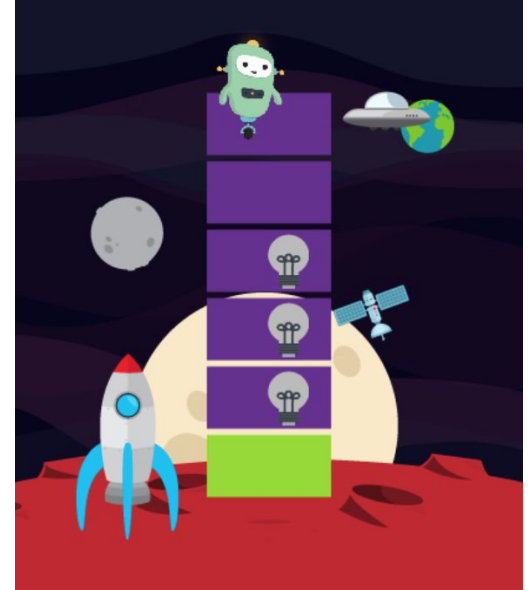
# A trabajar

Realizamos del libro del “Ciclo de Secundaria”, los ejercicios “Super Tito 2” de la sección “Repetición Condicional”.



## Super Tito 2: ¿Qué aprendimos?

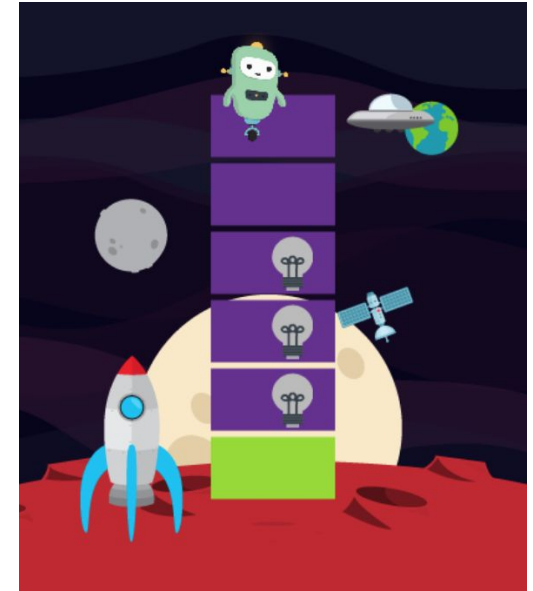
Hay que combinar repetición condicional y alternativa condicional.



## Super Tito 2: ¿Qué aprendimos?

Hay que combinar repetición condicional y alternativa condicional.

¡¡Ojo, no confundir las condiciones!!

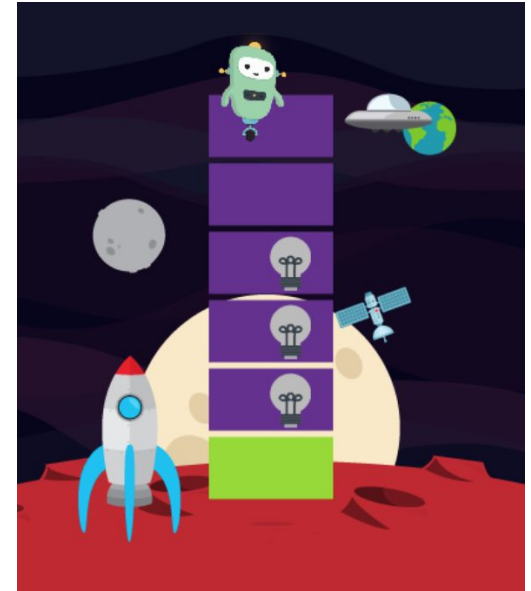


## Super Tito 2: ¿Qué aprendimos?

Hay que combinar repetición condicional y alternativa condicional.

¡¡Ojo, no confundir las condiciones!!

¿Cómo evitar errores?



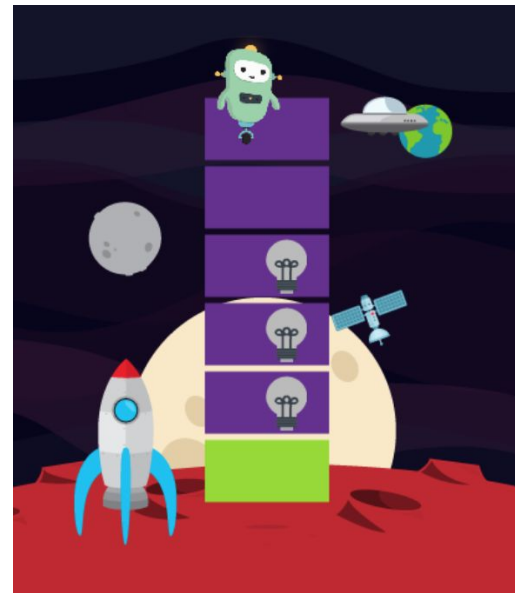
## Super Tito 2: ¿Qué aprendimos?

Hay que combinar repetición condicional y alternativa condicional.

¡¡Ojo, no confundir las condiciones!!

¿Cómo evitar errores?

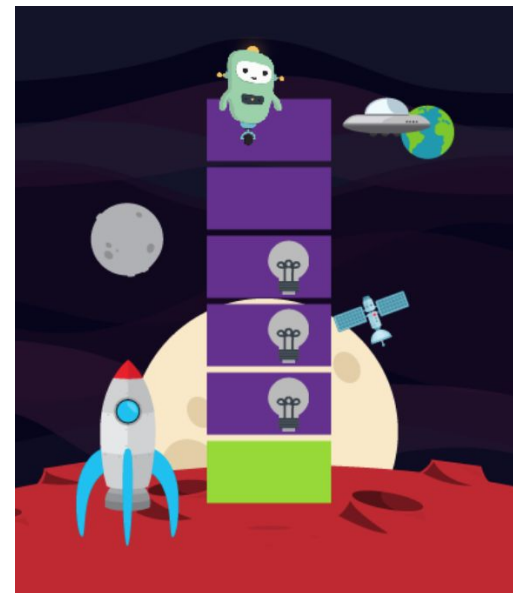
Fácil, dividir en subtareas usando procedimientos, con nombres claros.



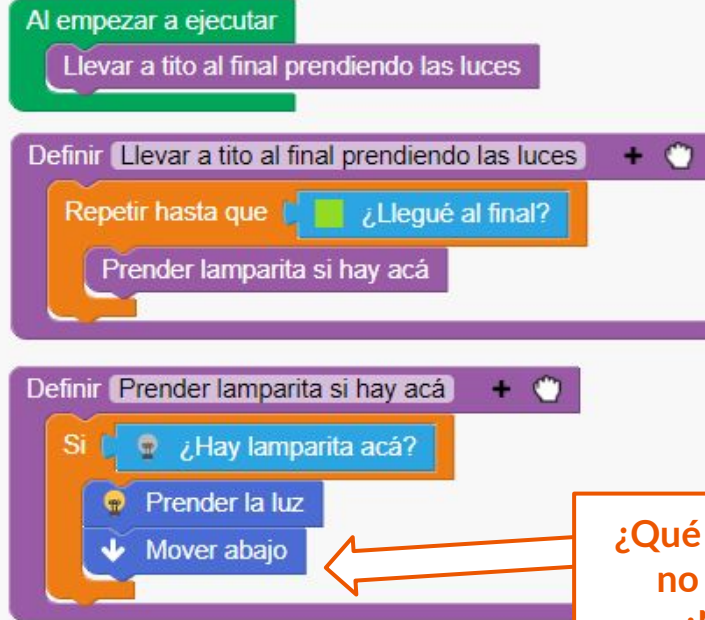


## Super Tito 2: ¿Qué aprendimos?

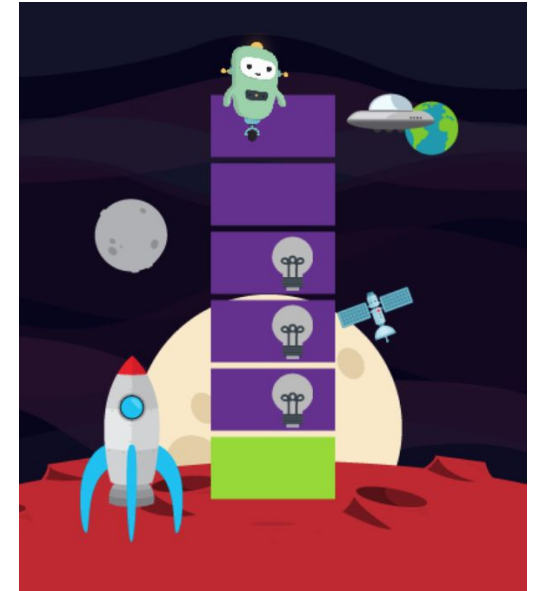
Veamos un error común :



## Super Tito 2: ¿Qué aprendimos?

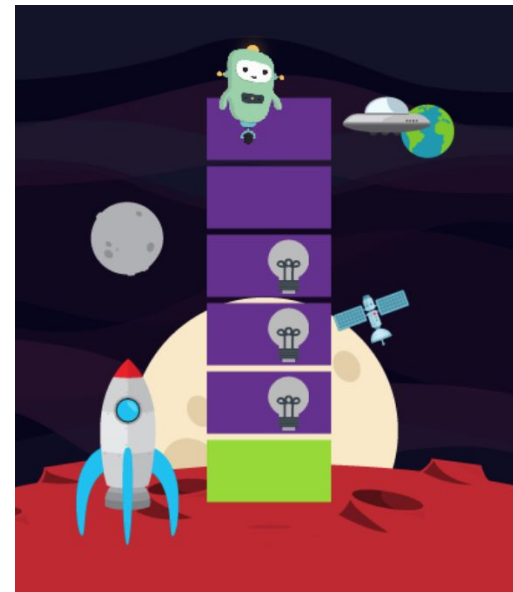


¿Qué hace Tito cuando  
no hay lamparita?  
¡No se mueve!



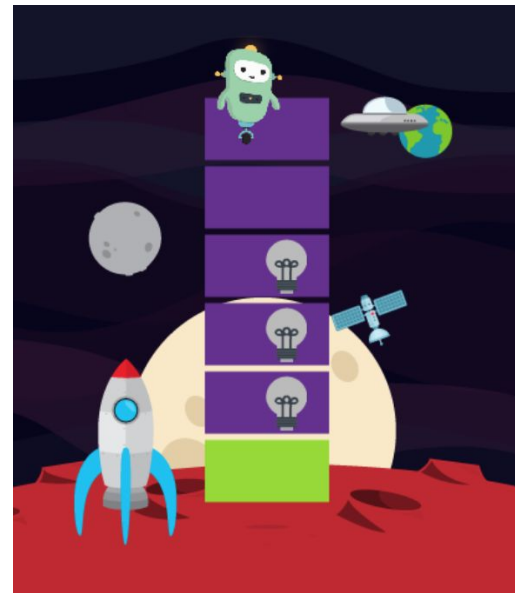
## Fallos por no terminación

- Cuando no hay una luz, el programa “se cuelga”.
- El problema es que la máquina pregunta por la condición de la repetición, y como no se cumple, ejecuta el cuerpo. Pero el cuerpo no hace nada, porque no hay luz. Por tanto, se termina el cuerpo, y se vuelve a preguntar por la condición, que otra vez estará sin cumplirse.
- Este es otro motivo por el cual un programa puede fallar.



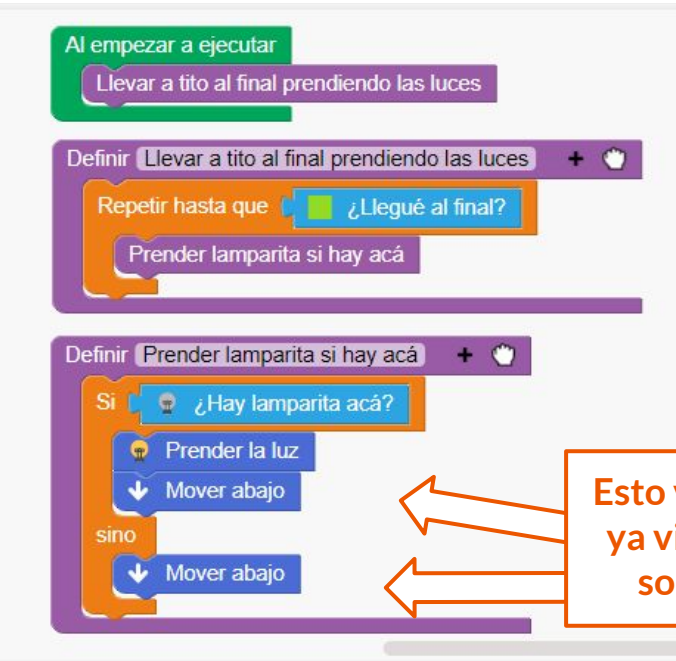
## Fallos por no terminación

- Para evitar los fallos por no terminación, hay que asegurarse que dentro una repetición condicional siempre se ejecute una acción que haga que nos acerquemos a la condición de corte.
- Sí tenemos que llegar a cierta ubicación, siempre deberíamos movernos hacia esa ubicación, para asegurar que eventualmente vamos a llegar.



## Super Tito 2: ¿Qué aprendimos?

Una segunda aproximación

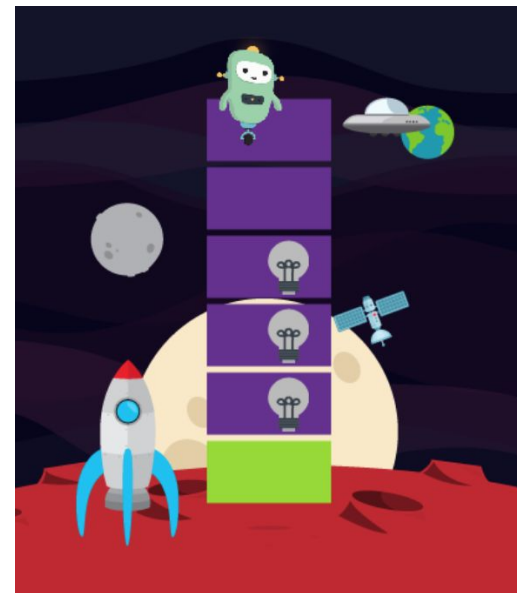


The image shows a Scratch script with the following blocks:

- Al empezar a ejecutar** (When green flag clicked):
  - Llevar a tito al final prendiendo las luces (Take Tito to the end by turning on the lights)
- Definir** (Define function):
  - Llevar a tito al final prendiendo las luces** (Take Tito to the end by turning on the lights):
    - Repetir hasta que** (Repeat until) loop:
      - Condition: ¿Llegué al final? (Did I reach the end?)
      - Action: Prender lamparita si hay acá (Turn on the light if there is one)
  - Prender lamparita si hay acá** (Turn on the light if there is one):
    - Si** (If) statement:
      - Condition: ¿Hay lamparita acá? (Is there a light here?)
      - Then branch:
        - Prender la luz (Turn on the light)
        - Mover abajo (Move down)
      - Else branch:
        - Mover abajo (Move down)

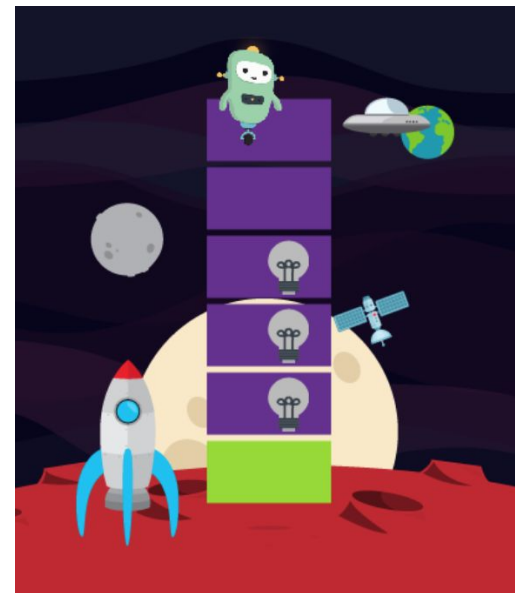
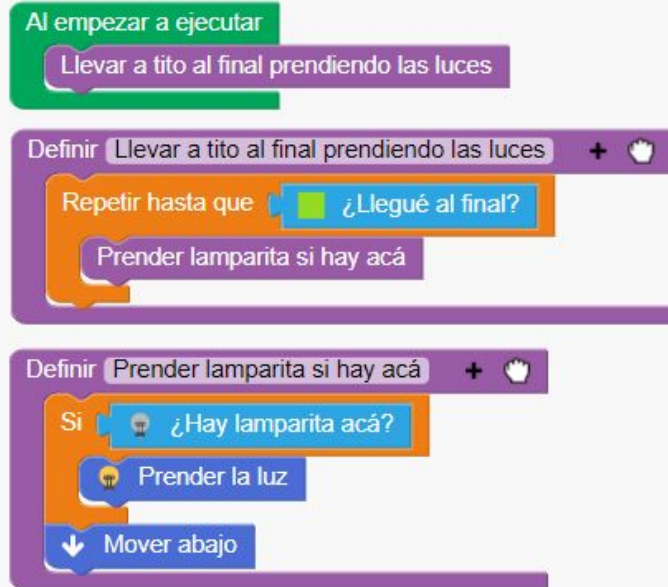
Three orange arrows point from the text box to the 'Prender la luz' block, the 'Mover abajo' block in the 'Si' branch, and the 'Mover abajo' block in the 'sino' branch.

Esto va a funcionar, pero ya vimos que no es una solución adecuada.



## Super Tito 2: ¿Qué aprendimos?

Una solución más adecuada.

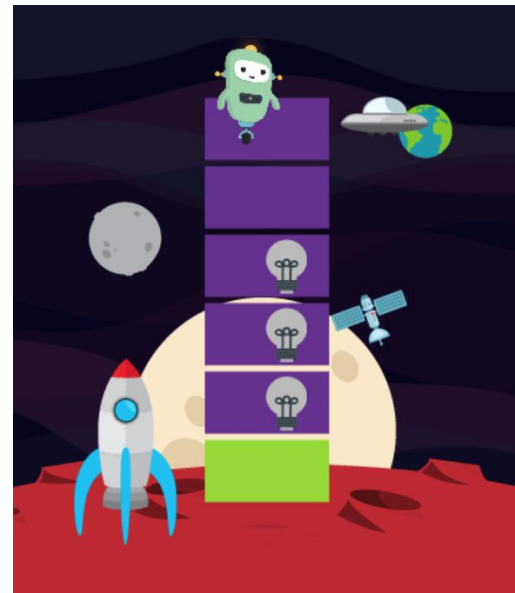


## Fallos por no terminación

Sí seguimos las prácticas que venimos trabajando (no anidar bloques y separar el problema mediante procedimientos), los riesgos disminuyen significativamente.

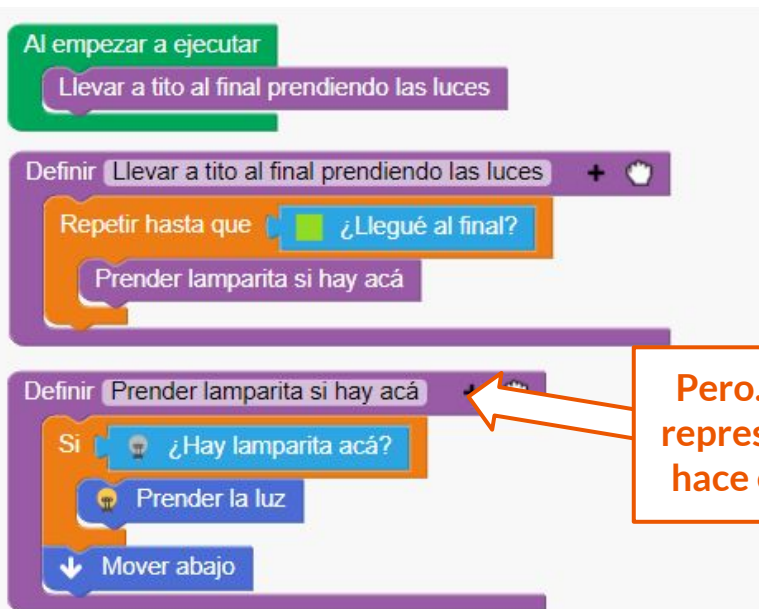
En particular, la mayoría de los problemas que usan repetición condicional pueden plantearse en términos de dos partes bien claras, que van en el cuerpo de la repetición, y deben realizarse hasta que se cumpla una condición:

- Una acción que debe ser realizada en cada ubicación (esa acción puede ser algo condicionado, por ej. “prender una lamparita, sí hay una”.)
- Pasar a la siguiente ubicación donde debe realizarse la acción.

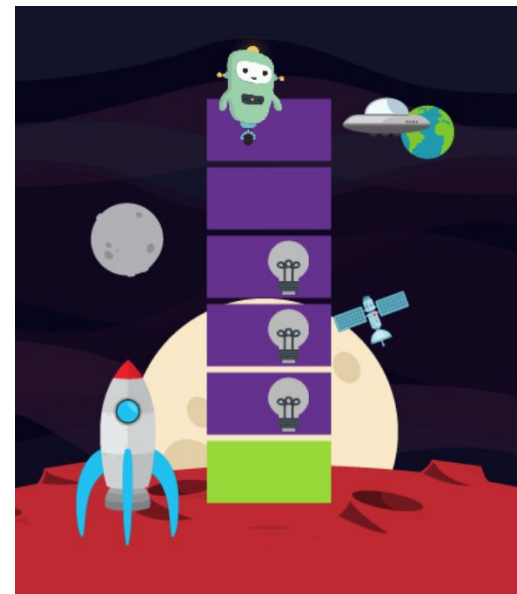


## Fallos por no terminación

P.D. Sí separamos bien en subtareas, dividiendo el “procesar” del “avanzar”, como vimos cuando charlamos de recorridos, entonces este error no debería ocurrirnos nunca.



Pero...este nombre ¿es representativo de lo que hace el procedimiento?





## Super Tito 2: ¿Qué aprendimos?

Una excelente solución. El hacer la subtarea permite ver claramente la acción y el pasar al siguiente y ayuda a prevenir el error por no terminación.

Al empezar a ejecutar

Llevar a tito al final prendiendo las luces

Definir Llevar a tito al final prendiendo las luces

Repetir hasta que ¿Llegué al final?

Prender lamparita si hay acá

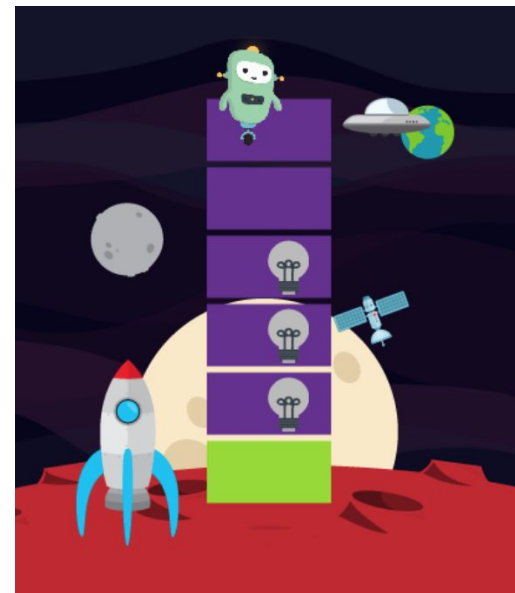
Mover abajo

Definir Prender lamparita si hay acá

Si ¿Hay lamparita acá?

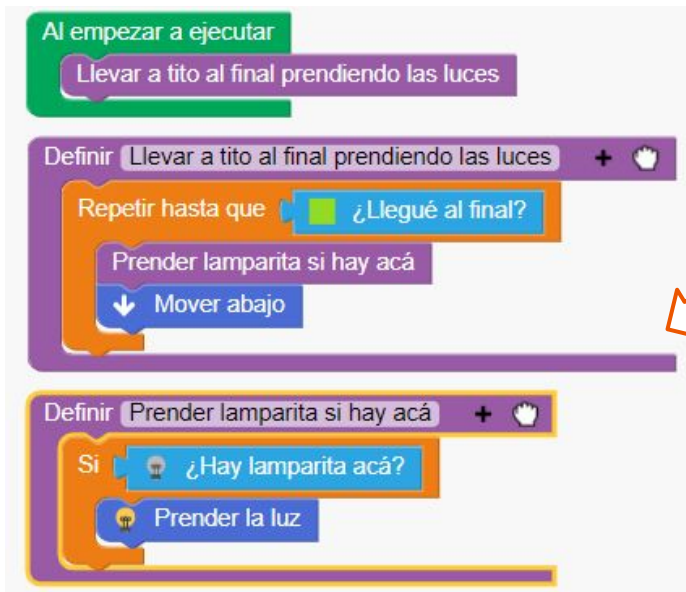
Prender la luz

¡SEPREMOS EN SUBTAREAS!

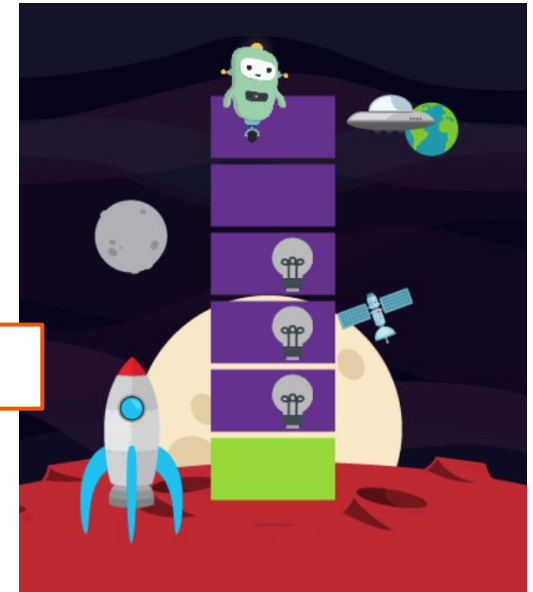


## Super Tito 2: ¿Qué aprendimos?

Una excelente solución. El hacer la subtaska permite ver claramente la acción y el pasar al siguiente y ayuda a prevenir el error por no terminación.

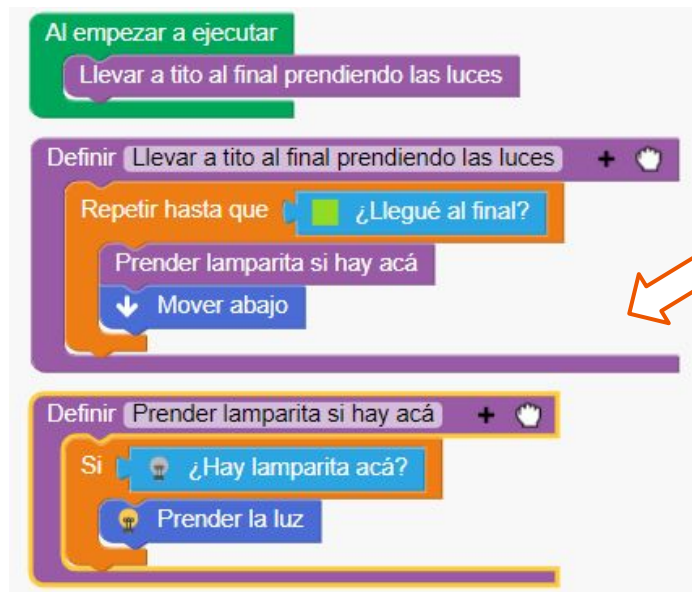


¿Qué forma tiene?



## Super Tito 2: ¿Qué aprendimos?

Una excelente solución. El hacer la subtaska permite ver claramente la acción y el pasar al siguiente y ayuda a prevenir el error por no terminación.



¡Es un recorrido!, ¡Claro que sí!



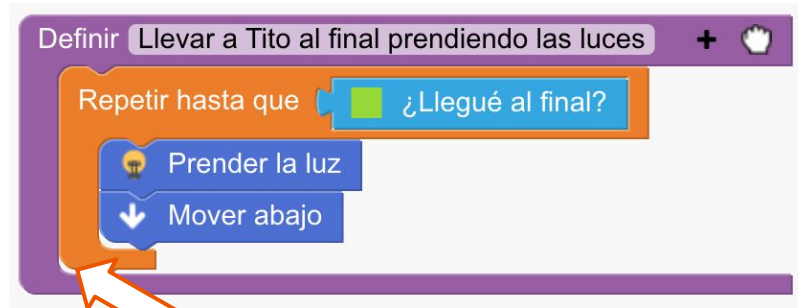


# Momento de dudas o consultas

# Condiciones antes, después y dentro de la repetición

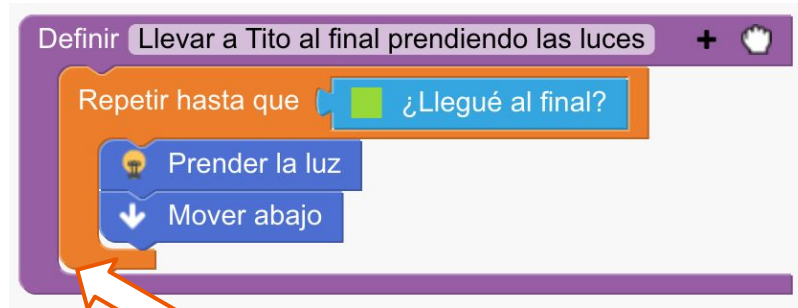
---

## Dentro y fuera de la repetición



¿Puede pasar que llegue  
a este punto del código y  
NO esté en el final?

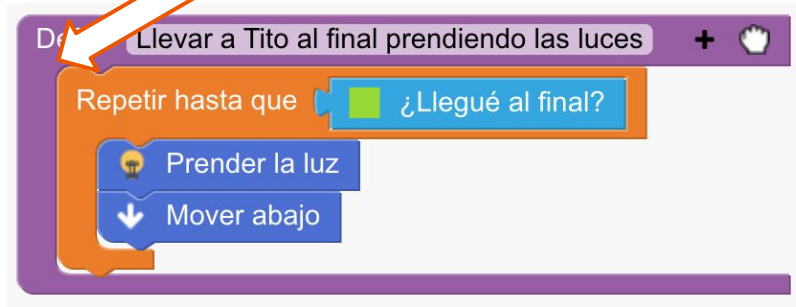
## Dentro y fuera de la repetición



**NO. Si no hubiéramos llegado al final no habríamos terminado de repetir**

## Dentro y fuera de la repetición

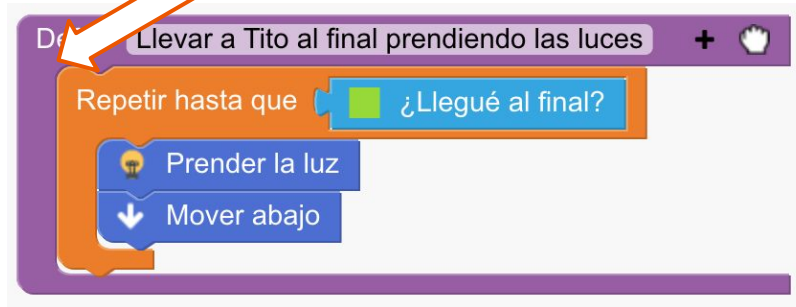
¿Puede pasar que al  
arrancar ya esté en el final?



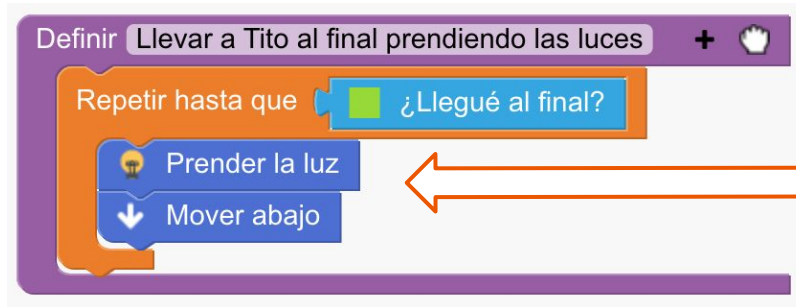


## Dentro y fuera de la repetición

Claro que sí, en ese caso nunca se entra a la repetición.

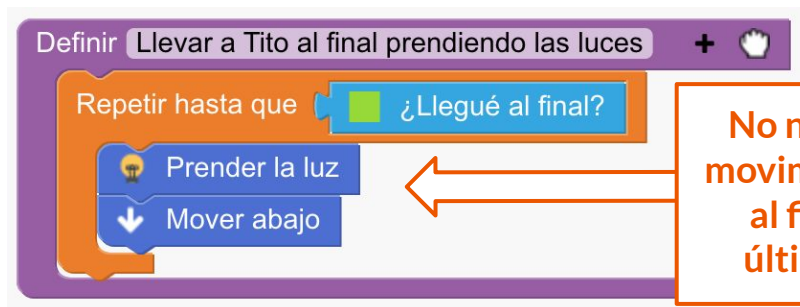


## Dentro y fuera de la repetición



¿Puede asegurarse qué  
NO estoy en el final  
dentro del cuerpo?

## Dentro y fuera de la repetición



No necesariamente. Luego del movimiento puedo haber llegado al final (de hecho, pasa en el último ciclo de la repetición)

# Recorridos

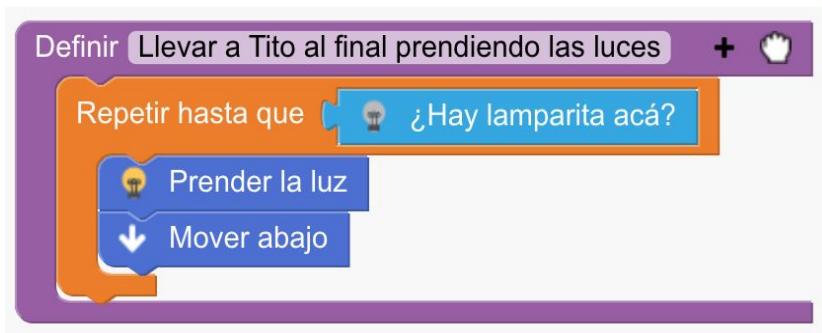
---

## Repetición condicional y recorridos

- Hablamos de “**Recorridos simples**”, porque usábamos **repetición simple**.
- Ahora podemos hablar de “**recorridos**” propiamente dichos.
- La idea conceptual es la misma, el recorrido va a tener:
  - Una repetición principal (simple o condicional)
  - Una subtarea (puede ya venir como primitiva) que “procesa” el elemento actual. Puede darse el caso de que en algunos recorridos no haga falta procesar nada en cada elemento. Incluso puede pasar que el “procesado” sea condicional (a veces proceso, a veces no), por lo que usaremos alternativa condicional (ej. prender una luz, solo sí hay una).
  - Una subtarea (puede ya venir como primitiva) que “avanza” al próximo elemento en el recorrido.
  - Puede incluir (o no) casos de borde, tanto antes como después.

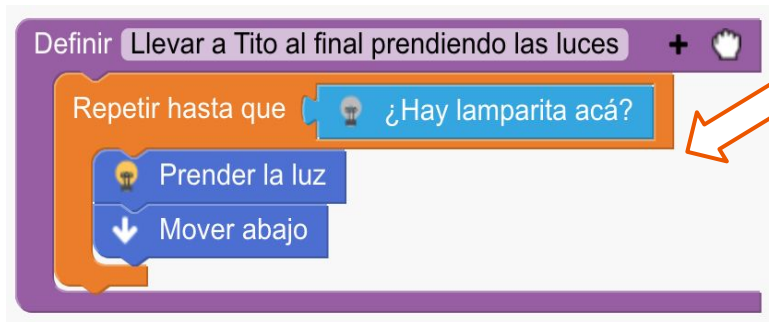
## Repetición condicional y recorridos

- Un error común:  
Al tener que procesar de forma condicional, confundir la condición de “hasta cuándo recorrer” con la de “cuándo procesar”. Ej.



## Repetición condicional y recorridos

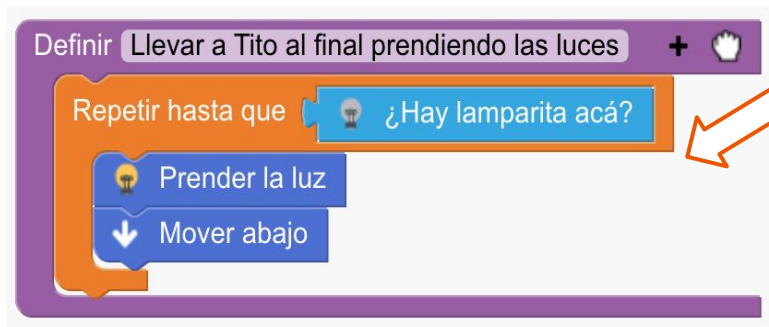
- Un error común:  
Al tener que procesar de forma condicional, confundir la condición de “hasta cuándo recorrer” con la de “cuándo procesar”. Ej.



¡Si en la primera ubicación hay  
lamparita, termina ahí!

## Repetición condicional y recorridos

- Un error común:  
Al tener que procesar de forma condicional, confundir la condición de “hasta cuándo recorrer” con la de “cuándo procesar”. Ej.



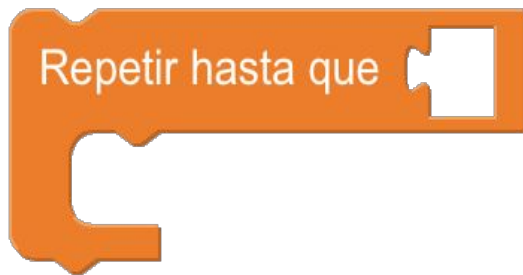
**Incluso si no hay en la primera, termina cuando encuentre una, aunque queden un montón de ubicaciones para ver.**



# Sintaxis en papel

---

## En papel: nuevas palabras claves y formas



Repetir hasta que <condición>

<cuerpo>

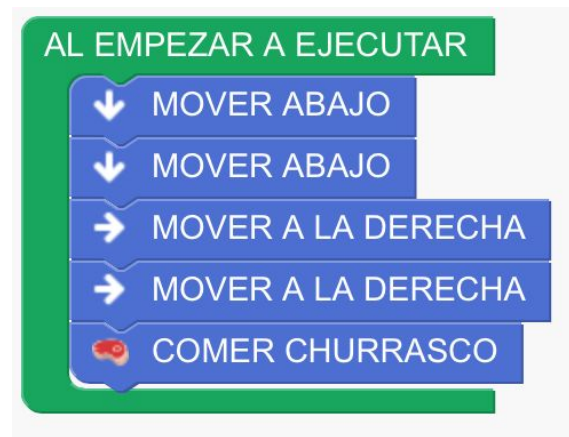
|



# Momento de dudas o consultas

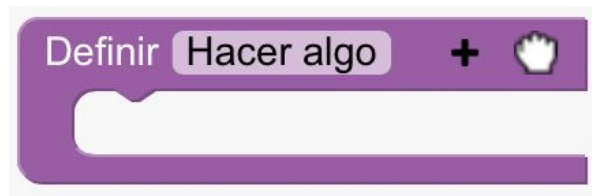
## Repaso

- Un **programa** es una **descripción** de la **solución** a un **problema computacional**.
- Un **problema computacional** es aquel que puede expresarse como una **transformación de estado**.
- En PilasBloques lo expresamos mediante bloques que se encastran entre sí, para expresar un cambio de estado en el escenario.
- Todo programa tiene un **punto de entrada**.
- Los elementos fundamentales del programa son los **comandos** (descripciones de acciones).
- Los comandos se organizan en **secuencia**, y la solución se ejecuta según esa secuencia.
- Hay **infinitos** programas que solucionan un problema. Decimos que dos programas que solucionan el mismo problema son **equivalentes**.



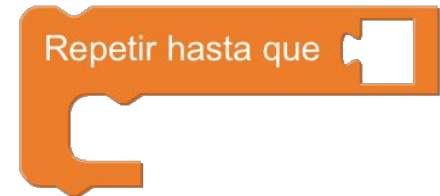
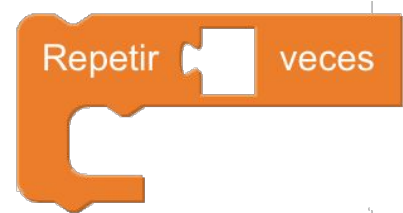
## Repaso

- Los **procedimientos** son una herramienta del lenguaje que permite definir nuevos comandos.
- Tienen un **cuerpo** y un **nombre**.
- **El nombre debe ser claro y legible, comenzar con un verbo en infinitivo y estar relacionado a su propósito.**
- La **definición** va por un lado, y el **uso (invocación o llamada)** va en el cuerpo de algún otro bloque.
- Aportan claridad, legibilidad y modificabilidad al código.
- Pueden ser **reutilizados** muchas veces.
- Permiten transmitir claramente las ideas pensadas en nuestra **estrategia**.
- Permiten separar el problema (tarea) en partes más pequeñas para su más fácil resolución (**subtareas**).



## Repaso

- La **repetición** es una herramienta del lenguaje que **permite cambiar el flujo** del programa (**estructura de control de flujo**).
- Viene en dos sabores:
  - La **repetición simple**, que permite **repetir una cantidad fija y finita** de veces. Espera una **expresión numérica** para indicar la cantidad de veces a repetir.
  - La **repetición condicional**, que permite **repetir una cantidad (a priori) indeterminada de veces**, continuando el ciclo hasta que se cumpla una **condición**. Espera una expresión que sea un **valor de verdad**.
- Permite estructurar el código de una forma distinta a la secuencia.
- Es un comando (y se puede usar junto con otros comandos en un cuerpo), pero tiene a su vez un cuerpo (es un **comando compuesto**)
- **No hay que anidar repeticiones (ni ninguna otra estructura de control).**



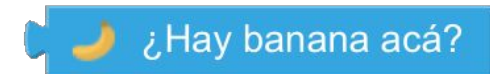
## Repaso

- La **alternativa condicional** es una herramienta del lenguaje que **permite cambiar el flujo** del programa (**estructura de control de flujo**).
- Permite **elegir** entre posibles **ramas**, según una **condición** en el estado del programa.
- Es un comando (y se puede usar junto con otros comandos en un cuerpo), pero tiene a su vez un cuerpo (es un **comando compuesto**)
- Espera una **expresión de valor de verdad** para indicar cuándo se elige un camino y cuando otro..
- **No hay que anidar alternativas (ni ninguna otra estructura de control).**



## Repaso

- Un **comando** es la descripción de una **acción**.
- Una **expresión** en la descripción de un **dato** (un **valor, información**)
- Puede ser **numéricas**, describen una cantidad, y las usamos en la repetición simple.
- Pueden ser de **valor de verdad**, se responden con verdadero o falso y las usamos en la alternativa.
- Los **sensores** son la herramienta mediante la cual obtenemos valores de verdad basados en el estado del programa (escenario)
- Podemos usar **conectivas** para combinar sensores que describan valores de verdad, para tener valores de verdad más complejos que dependen de esos sensores.





## Repaso

- La **conjunción** une dos sensores, dando verdadero sólo cuando ambos evalúan a verdadero.
- La **disyunción** une dos sensores, dando verdadero sólo cuando alguno de los dos (o ámbos) son verdaderos.
- La **negación** aplica a un único sensor, y cambia el valor de verdad del mismo, transformando el verdadero en falso y viceversa.
- Se pueden usar expresiones con varias conectivas, y se resuelven por orden, primero las negaciones que aplican solo a un sensor, luego según indiquen los paréntesis, y si no hay más paréntesis, de izquierda a derecha.
- Podemos definir nuestras propias expresiones en base a otras, para comunicar mejor.

p	$\neg p$
V	F
F	V

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F



## Recordatorio: **iiiProgramar es comunicar!!!**

- Tus programas deberían quedar claros a partir de la lectura.
- Sí leo el punto de entrada, tiene que quedar más que explicita la estrategia elegida.
- Usamos procedimientos para la claridad, legibilidad y expresar la estrategia.
- Es importantísimo elegir nombres adecuados para los procedimientos que definimos.
- No anidar estructuras. Usamos procedimientos para dividir el problema en partes pequeñas y darles nombres adecuados.
- Definimos expresiones con nombres claros a partir de otras para comunicar mejor, y dividir el problema de las expresiones, también, en partes más pequeñas.