



# Procedimientos

Teoría 1.B.

# ¿Hicieron la tarea?

---



## Primitivas

- Los comandos están agrupados en categorías.
- En particular “Mover a la derecha” y “Apretar botón” están dentro de la categoría **“Primitivas”**.
- “Primitivas” refiere a que son comandos propios del ejercicio, vienen incorporados con este.
- Las primitivas son las que me da el ejercicio, y no puedo inventar otras nuevas. Podemos pensarlas como “las cosas que el autómata sabe hacer”.
- En las computadoras modernas, las primitivas están relacionadas al hardware.

# A trabajar

Realizamos del libro del “Ciclo de Secundaria”, el ejercicio “El gato en la calle”, de la sección “Autómatas, Comandos, Procedimientos y Repetición”.



## El gato en la calle: ¿Qué aprendimos?



Primera aproximación



## El gato en la calle: ¿Qué aprendimos?

Volvamos a leer el enunciado:

Hacé que el gato avance un paso, se duerma, se despierte, salude y vuelva a su lugar.



## El gato en la calle: ¿Qué aprendimos?

Volvamos a leer el enunciado:

Hacé que el gato **avance un paso**, se **duerma**, se **despierte**, **salude** y **vuelva a su lugar**.

Claramente hay 5 elementos en el problema, 5 cosas que el gato debe realizar.

## El gato en la calle: ¿Qué aprendimos?

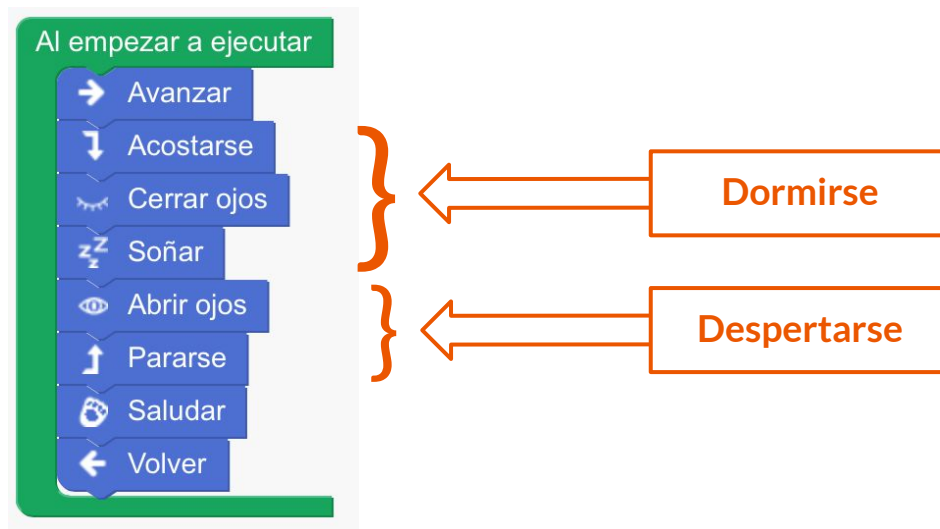
Pero sí el problema implica hacer 5 cosas, ¿Por qué nuestra solución tiene 8 elementos?





## El gato en la calle: ¿Qué aprendimos?

Podemos pensar que hay grupos de comandos que corresponden a una misma cosa a nivel conceptual:



## El gato en la calle: ¿Qué aprendimos?



Lo que queremos...

## El gato en la calle: ¿Qué aprendimos?



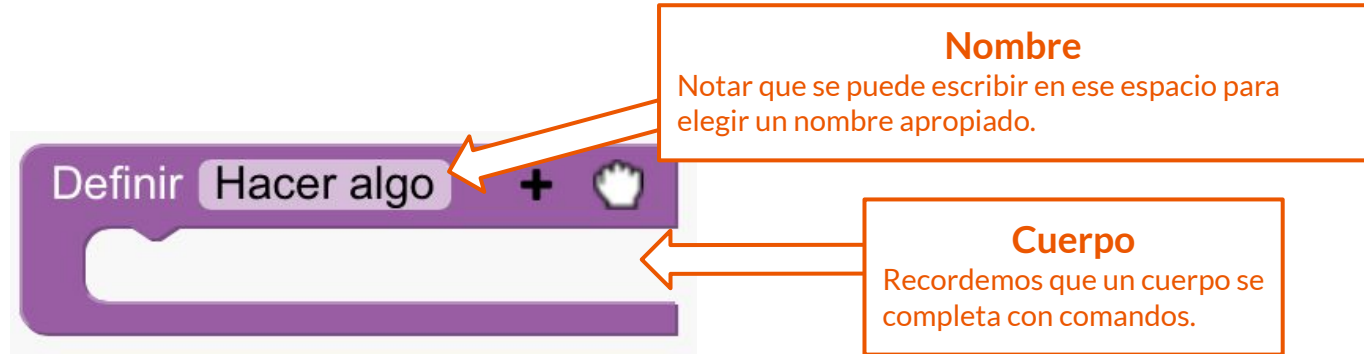
Podemos lograr esto gracias a los procedimientos



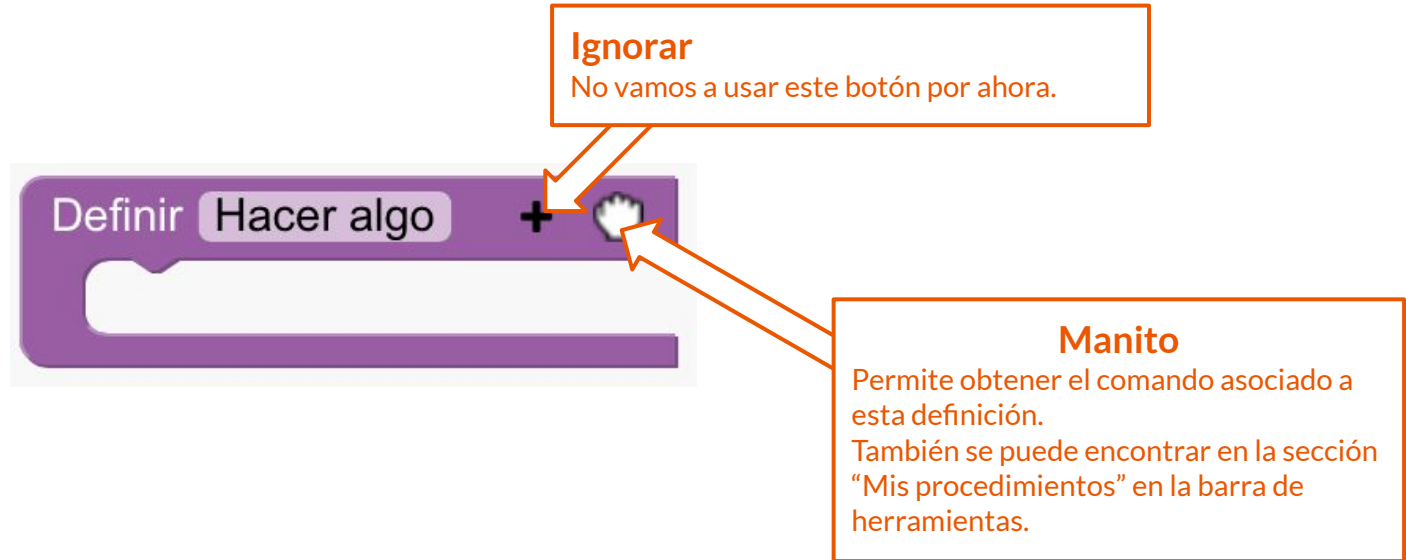
## Procedimientos

- Los procedimientos son una herramienta del lenguaje que permite al programador definir nuevos comandos.
- Un procedimiento tiene un nombre y un cuerpo.
- El procedimiento se define, por un lado, y luego se invoca (se usa como un comando más)

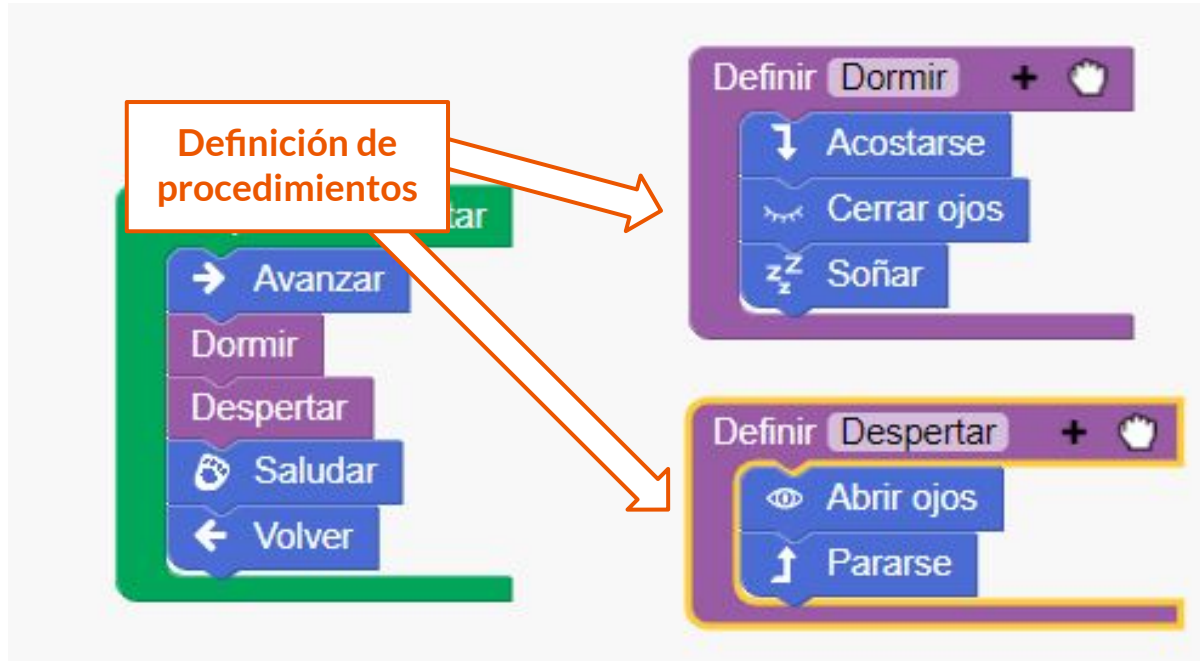
## Procedimientos



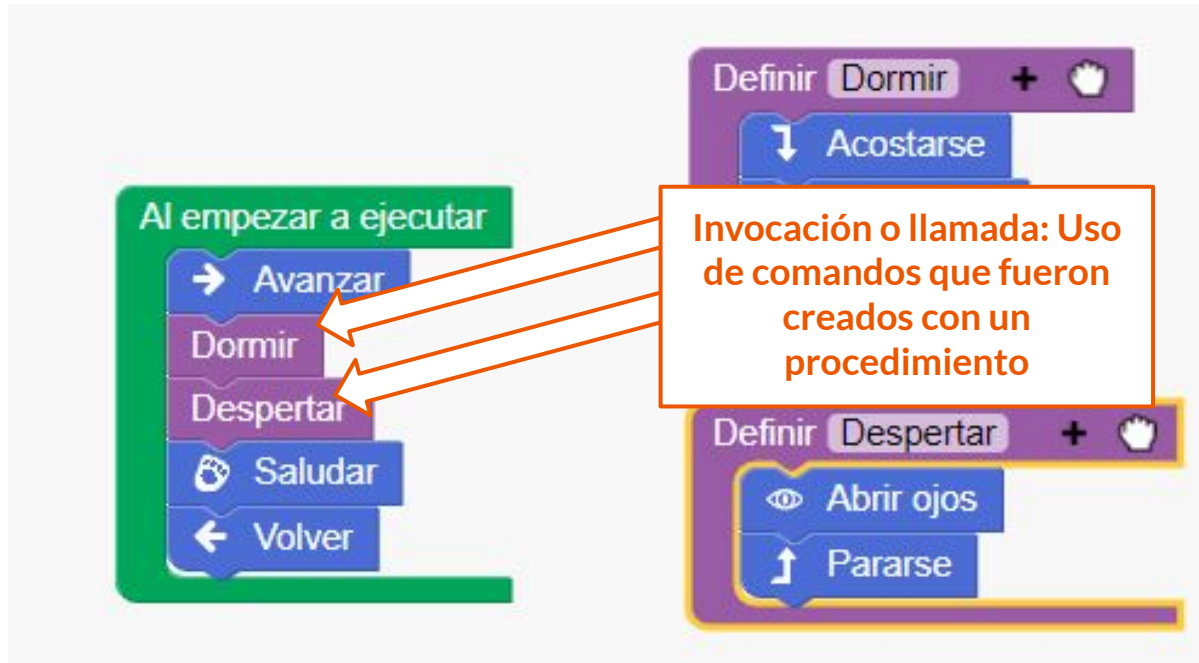
## Procedimientos



## Procedimientos



## Procedimientos







## Procedimientos

- Brindan **claridad** en nuestra solución, permitiendo comunicar claramente las ideas que el programador tenía en la cabeza al momento de realizar el código.
- Aportan **legibilidad**, haciendo que la solución se lea como una historia.
- Proveen mayor **modificabilidad**, ya que con procedimientos es más fácil modificar una solución existente para que se adapte a nuevas necesidades.
- Van a permitir realizar soluciones más simples (cada procedimiento puede ser pensado como un programa independiente del resto, pero que resuelve solo una partecita del problema total). Es decir, sirven para aportar **estructura** al código, y pensar en términos de **subtareas**.
- La definición puede ser única, pero puede invocarse muchas veces, en lugares distintos, permitiendo **reutilizar** código.
- **Reducen la cantidad de errores**, ya que el código no se duplica y es más fácil identificar puntos de fallo en caso de que los hubiera.



## Procedimientos

**¡SON “EL TEMA” DE LA MATERIA!**



## Programar es comunicar

Programar implica **comunicar**. Comunicamos la solución a la máquina (motivo por el cual tenemos que escribirla en un lenguaje de programación), quien la ejecutará para solucionar finalmente el problema.

Pero los programadores no trabajan solos en la mayoría de los casos. También tenemos que comunicar a las personas, para que otros entiendan nuestras soluciones, y que las ideas y cosas que pensamos al solucionar un problema sean claras. Si comunicamos solo en español, las personas entienden, pero la máquina no.

Es importante entender esta dualidad y tenerla siempre presente.

Así, hay soluciones que son correctas, pero no adecuadas porque no comunican correctamente a las personas.

## Procedimientos



Solución adecuada, las ideas se comunican claramente.



Solución inadecuada, funciona, pero no se entiende.

## Procedimientos



Notar que la solución inadecuada es más corta, pero esto no la hace correcta. No buscamos la solución con menos bloques, sino la que es más clara en expresar las ideas.



## Procedimientos

Vamos a hablar entonces de “**buenas prácticas**” en la programación cada vez que hagamos referencia a aquellas pequeñas cosas que podemos hacer de manera más adecuada para aportar claridad a nuestro programa y facilitar así la comunicación de cómo éste resuelve el problema.

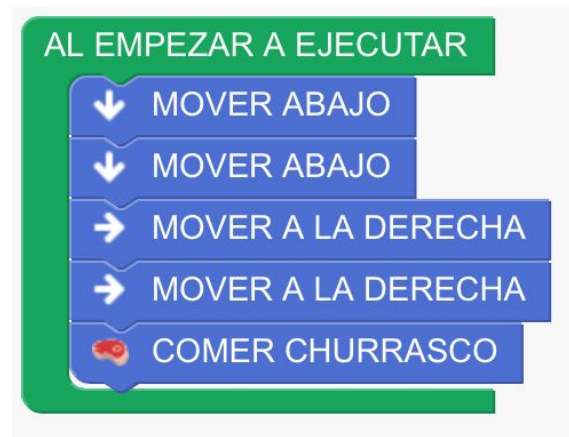
Dicho de otro modo, no solo debemos preocuparnos porque el programa funcione, sino que también deberemos estar atentos y hacer buen uso de ciertas prácticas como por ejemplo la que hoy aprendimos: hacer una buena subdivisión de tareas utilizando procedimientos que resuelvan partes concretas de nuestro problema.



# Momento de dudas o consultas

## Repaso

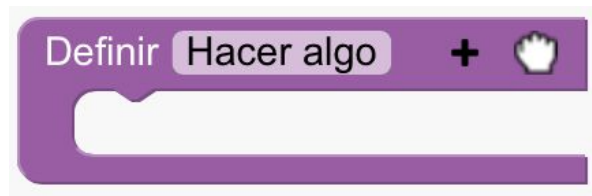
- Un **programa** es una **descripción** de la **solución** a un **problema computacional**.
- Un **problema computacional** es aquel que puede expresarse como una **transformación de estado**.
- En PilasBloques lo expresamos mediante bloques que se encastran entre sí, para expresar un cambio de estado 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**.
- 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**).



# ¿Ya no hay más ejercicios?

---



# Límites de la herramienta actual

- PilasBloques es una excelente herramienta para el aprendizaje de la programación inicial. Pero, la cantidad de ejercicios que hay es limitada. A veces queremos más ejercicios.
- Existen otras herramientas para aprendizaje, pero cambiar de entorno y acostumbrarse al mismo lleva un tiempo y esfuerzo que no es deseable invertir.
- La computadora está buena, pero su dependencia extrema es peligrosa. ¿Estás pensando los ejercicios o simplemente encajando bloques hasta que funciona?



# Las ventajas de programar en papel

- No dependemos de ninguna herramienta, y podemos tener tantos ejercicios como nuestra imaginación (y la propuesta didáctica) lo permita.
- Nos permite trabajar comprensión de enunciados más complejos, a veces sin el componente gráfico que brinda la computadora, lo cual es muy importante que desarrollen como estudiantes universitarios.
- Elimina la dependencia de la computadora, y nos obliga a pensar sí o sí sobre lo que estamos haciendo, ya que no podemos sacar las cosas por “prueba y error”.



# La dualidad de la programación

Cuando programamos, existe siempre una especie de paradoja que nos debe dominar.

El programa debe poder ser ejecutado por una computadora, por lo que la computadora subyacente es muy importante. A su vez, la computadora subyacente no es relevante (caja negra) y queremos programas que puedan funcionar en cualquier computadora, por lo que lo más importante es que el programa cumpla otras características (plantee adecuadamente la estrategia de solución, sea claro y legible, utilice las herramientas del lenguaje y las primitivas de forma apropiada, etc.).

Cuando trabajamos en máquina corremos el riesgo de enfocarnos mucho en que las cosas “funcionen” sin prestar atención a la parte verdaderamente importante, la de usar bien los conceptos que venimos trabajando.

Al trabajar en papel, se elimina el factor de pensar en que funcione, y nos fuerza a pensar en usar bien los conceptos.



# Pero debería funcionar...

- El hecho de que no vaya a ser ejecutado no significa que podamos hacer cualquier cosa.
- La idea es escribir programas de forma tal que, si estuviéramos ejecutando en una computadora, ese programa debería funcionar.
- Es decir, la idea de ejecución mediante secuencia y repetición, cuándo un programa va a fallar, etc. se debe tener en cuenta, a pesar de que no haya una computadora que vaya a ejecutar las cosas.

# Una sintaxis para el papel

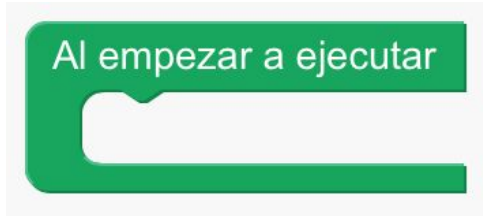


# ¿Cómo escribimos un programa en papel?

- Dibujar bloques no es práctico, por lo que necesitamos una forma de escribir las cosas de forma sencilla, marcando claramente la jerarquía de los elementos.
- Por jerarquía nos referimos a “qué bloques están dentro de qué otros bloques”, es decir, como están anidados.
- Además, debemos dejar en claro los bloques que son parte del lenguaje, como la repetición simple, el punto de entrada, o las definiciones de procedimientos.
- Vamos a usar la idea de “palabras clave”, cosas que queremos escribir todos de la misma forma, porque tienen una semántica puntual bien clara, ej. “al empezar a ejecutar”



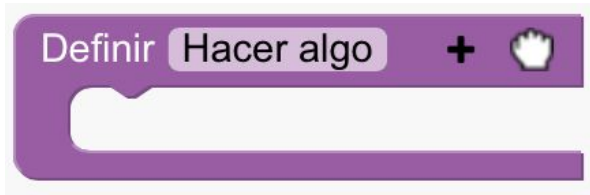
## Palabras claves



Al empezar a ejecutar

| <cuerpo>

## Palabras claves



**Definir** <nombre del procedimiento>

| <cuerpo>



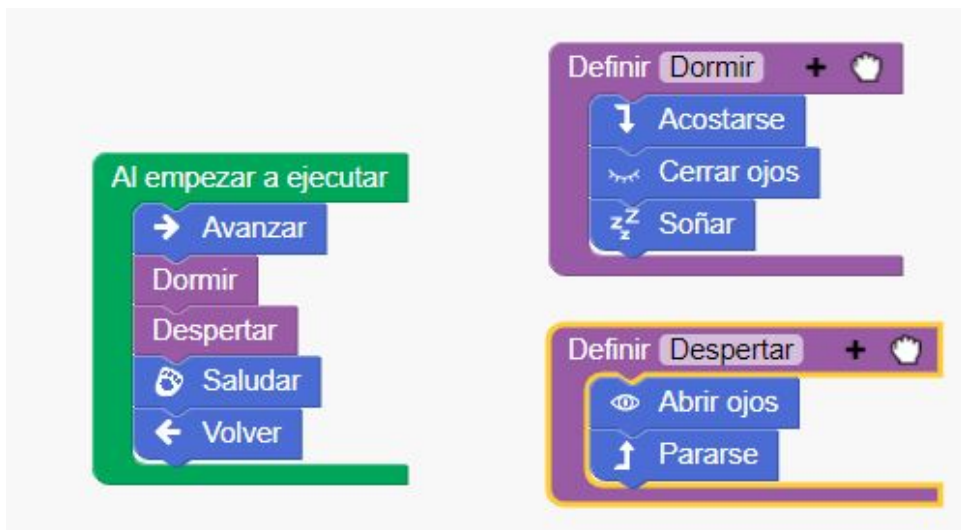
## Comandos y primitivas



Mover a la derecha

Es decir, escribimos el nombre, tal cual fue dado en las primitivas o pusimos en la definición de un procedimiento que realizamos.

## Ejemplo completo



### Al empezar a ejecutar

- Avanzar
- Dormir
- Despertar
- Saludar
- Volver

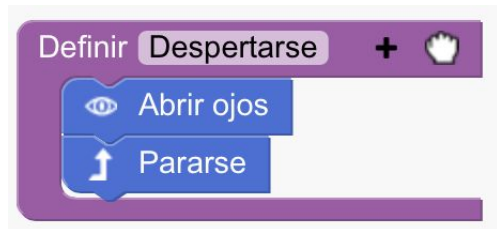
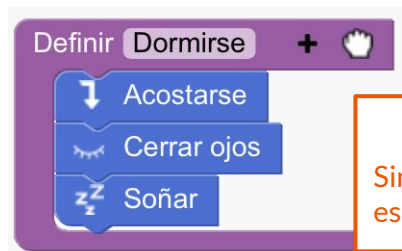
### Definir Dormir

- Acostarse
- Cerrar ojos
- Soñar

### Definir Despertar

- Abrir ojos
- Pararse

## Ejemplo completo



Al empezar a ejecutar

**¡Usar las palabras clave!**

Sino es muy difícil entender qué se está queriendo escribir



Al empezar a ejecutar

Avanzar  
Dormirse  
Despertarse  
Saludar  
Volver

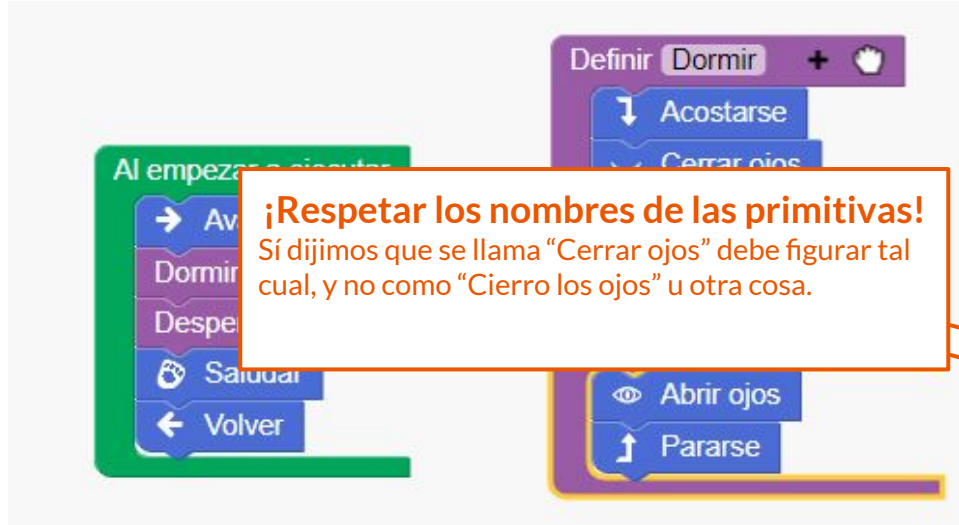
Definir Dormirse

Acostarse  
Cerrar ojos  
Soñar

Definir Despertarse

Abrir ojos  
Pararse

## Ejemplo completo



No respetar el nombre de las primitivas o de los procedimientos cuando estamos escribiendo en la computadora se conoce como “error de sintaxis”

Al empezar a ejecutar

- Avanzar
- Dormir
- Despertar
- Saludar
- Volver

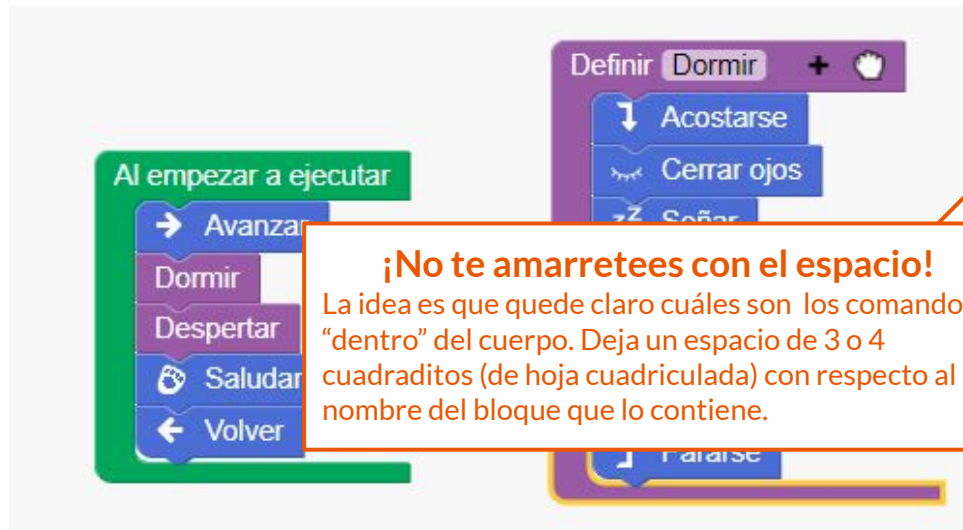
Definir Dormir

- Acostarse
- Cerrar ojos
- Soñar

Definir Despertar

- Abrir ojos
- Pararse

## Ejemplo completo



Al empezar a ejecutar

- Avanzar
- Dormir
- Despertar
- Saludar
- Volver

Definir Dormir

- Acostarse
- Cerrar ojos
- Soñar

Definir Despertar

- Abrir ojos
- Pararse

## Ejemplo en hoja cuadriculada

Podes usar imprenta, cursiva o lo que quieras, pero pensá que la idea más adelante es poder mostrarle a los profes el código, por lo que, escribas como escribas, tratá de que sea claro y prolijo.

Al empezar a ejecutar

Avanzar

Dormir

Despertar

Saludar

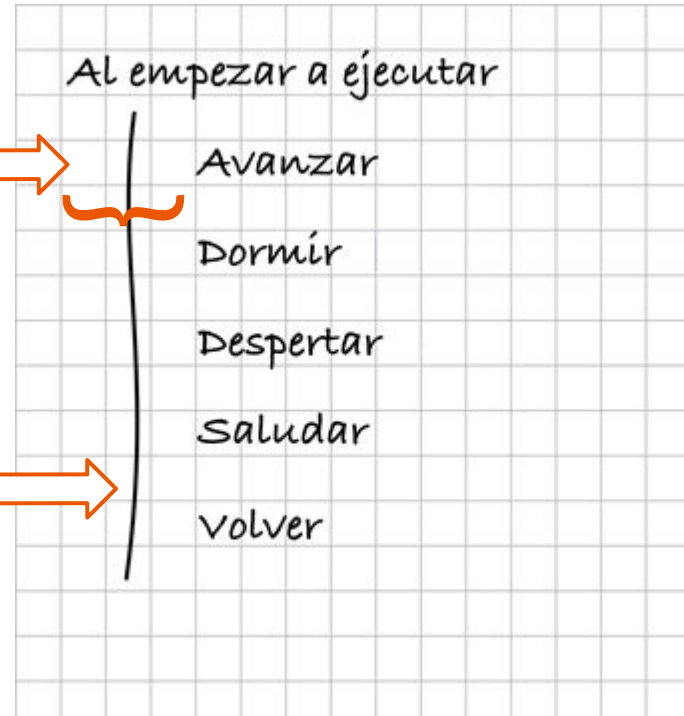
Volver



## Ejemplo en hoja cuadriculada

Acá dejamos 3 cuadraditos entre donde empieza el "Al empezar a ejecutar" y los comandos que están dentro de este.

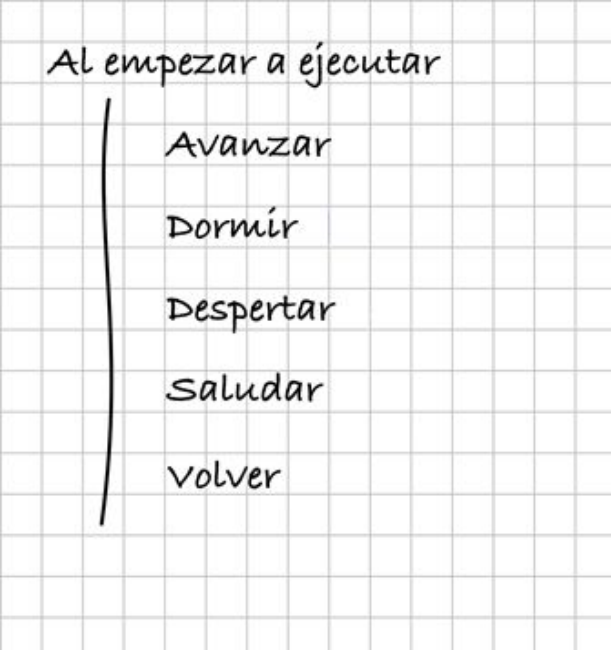
La línea está 1 cuadradito más adentro, para dejar en claro que pertenece al "Al empezar a ejecutar", y dejamos un pequeño lugar entre la línea y los comandos de adentro, para que se lea bien. Además, la línea termina un cuadradito después del último comando, para dejar bien en claro que está adentro y qué no.



## Ejemplo en hoja cuadriculada

¡¡Es importante respetar la sintaxis propuesta para que todos entendamos lo mismo!!

Desde las palabras claves, el nombre de las primitivas hasta las líneas y el espaciado.



The image shows a grid of squares with handwritten text. The text is written in a simple, slightly irregular font. A vertical line is drawn to the left of the list of actions. The text is as follows:

Al empezar a ejecutar
Avanzar
Dormir
Despertar
Saludar
Volver

# Tarea

Realizamos la guía de ejercicios en papel “P1.A. Programas y primeros procedimientos”.

**Debe estar terminada para la próxima clase.**

---



# Procedimientos

Nos vemos la próxima