Datasheet - Programming & Sensors

Create a chatbot on CodeSkool

Materials and tools needed

CodeSkool block-based programming software (https://ide.codeskool.cc/). CodeSkool is particularly well-suited for this project because it combines Open AI integration with phone sensor extensions and text-to-speech capabilities.

There are several alternatives to CodeSkool. Scratch offers geolocation extensions that provide similar functionality in a block-based environment. MIT App Inventor includes built-in location components specifically designed for mobile app development. For classes with more advanced programming experience, Python libraries like geocoder or geopy offer sophisticated location services that can be integrated into larger applications.

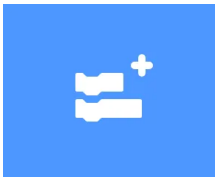
Coding and understanding your program

1 Enter CodeSkool and store GPS information using variables

To implement the GPS functionality, start by creating a new project and accessing the editor in CodeSkool.

Before accessing any GPS data, establish memory locations in the program where this information will be stored. Start by going to the "Variables" section of the Blocks palette and clicking "Create Variable." Create a variable named "Latitude" to store the user's north-south position. This variable will act as a dedicated container that remembers this specific information throughout the program. Then, return to the "Variables" section and create a second variable named "Longitude" to store the east-west position. Together, these two variables will provide a complete geographic position that persists in memory.

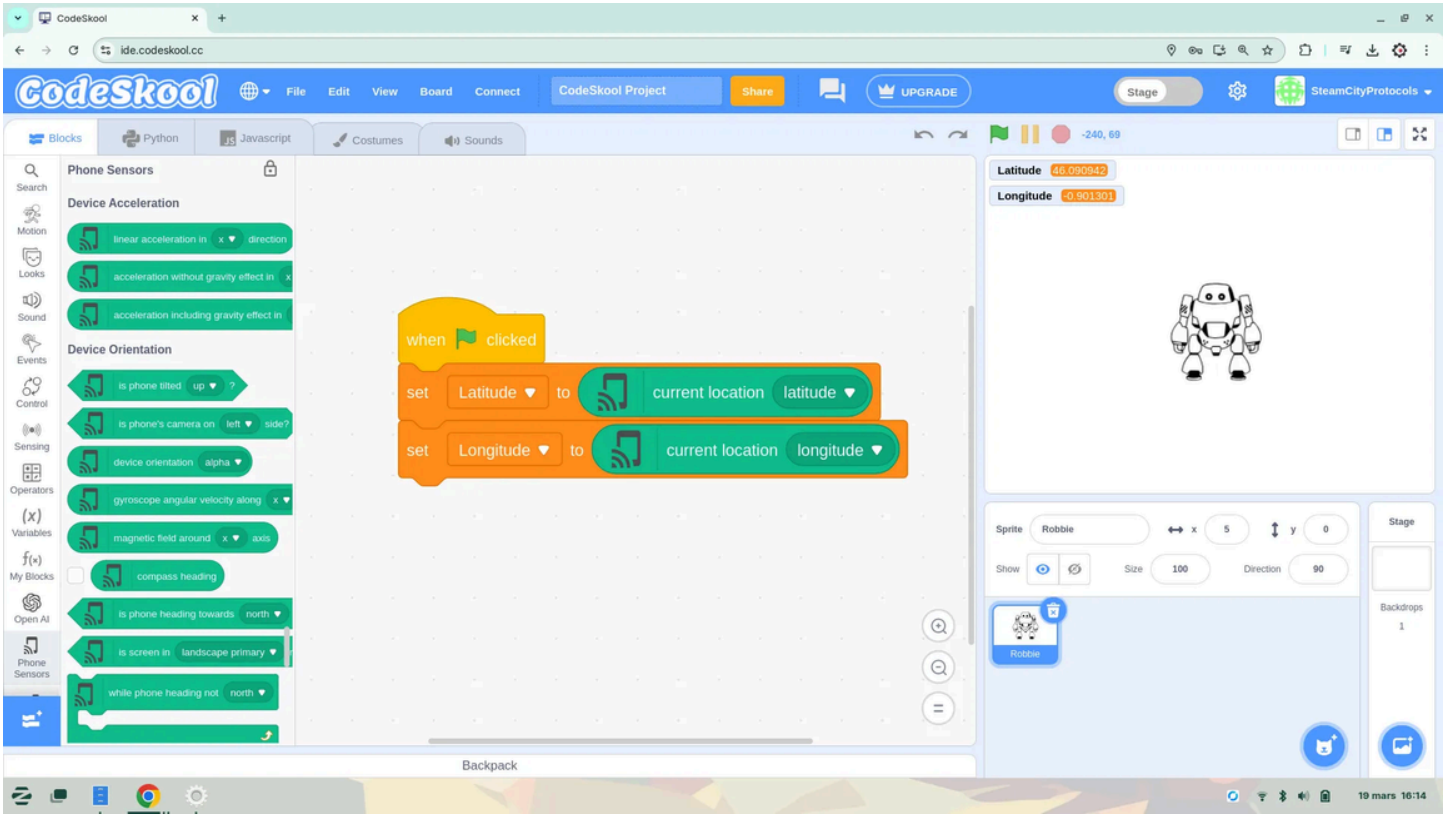
Now that the memory containers for the location data are created, access the device's GPS capabilities to populate these variables with real coordinates. From the main interface, access the "Extensions" button located at the very bottom of the toolbar. In the extensions library, search for "Phone sensors" and add this extension to the project to access the GPS functionality.



After adding the extension, find GPS-related blocks in the new category that appears in the block palette. Look for the block subcategory called "GPS location" in which to find the "current location" block for longitude and latitude.

From now on, all the blocks and variables needed to create the program are identified, through simple steps:

1. To trigger the data collection process, add a "When button is clicked" event block to the workspace. This will serve as a starting point for capturing the user's location.
2. From the "Variables" section, drag two "set [variable] to [value]" blocks and connect them to the button event. These blocks are the mechanism by which information passes from the sensors to the program's memory.
3. Configure the first block to define "Latitude" and the second to define "Longitude." The variable selection demonstrates that the program now has specific memory locations ready to hold different types of information.
4. Connect the appropriate "current position" blocks from the Phone Sensors extension to each variable value. Select "latitude" for the first variable and "longitude" for the second. When these blocks execute, they will capture sensor readings and store them in the program's memory.
5. Click the green flag in the test panel to verify that the variables are successfully storing information.

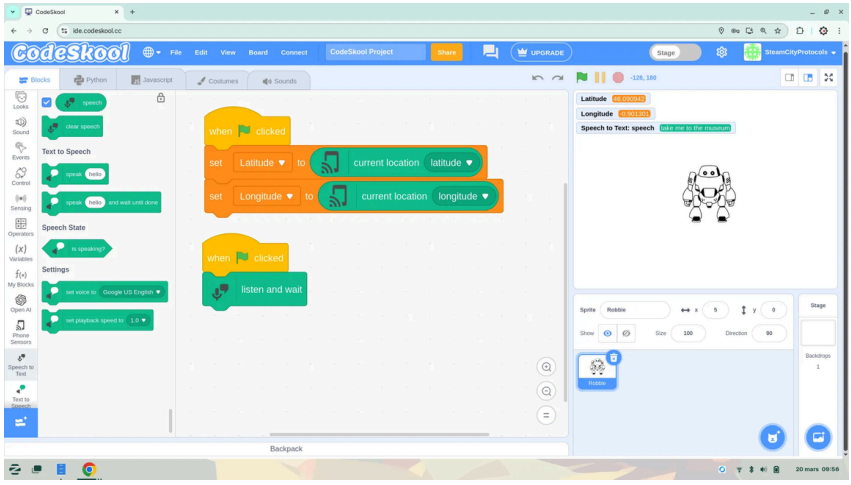


This program will work either on the computer using the machine's location or via a mobile device using the Phone Sensors extension.

2 Implement and test speech recognition

Based on the created program, implement the second functionality for voice recognition.

On the same editor, return to the "Extensions" button at the end of the toolbar and search for "Speech to Text" in the extensions library to add it to the project.



After adding the extension, find speech recognition-related blocks in the new category that appears in the block palette. Look for the "Listen and Wait" block.

For clarity in the blocks, add a second block "When button is clicked" to identify the two basic functionalities of the system.

Once the basic speech recognition functionality is implemented, reuse the same testing method as in step 1 to systematically evaluate its accuracy to understand its capabilities and limitations. Reuse the test sentences and complete the table with the system's capabilities:

Pronounced sentence	System recognition	Precision
The sentence formulated by the students	*What the system transcribes from your speech*	*Assessment of understanding (Good / Fair / Poor)*
“Where is the nearest park?”		
“Take me to the museum”		
" Is there a café nearby? "		
"What time is it for the movies?"		

3 Convert location to city

Implement the first AI integration by transforming raw GPS coordinates into a meaningful city name. This feature serves an important purpose: it converts technical data (digital coordinates) into human-readable information that is more intuitive and useful for both the system and its users. This conversion will also serve as the basis for more complex prompts in subsequent activities, as including the city name in future prompts will help the AI provide more contextually relevant information.

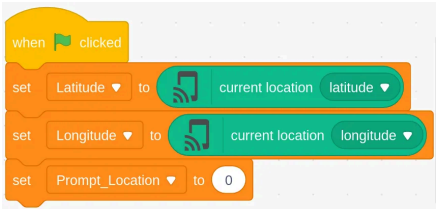
1. Prepare the variables

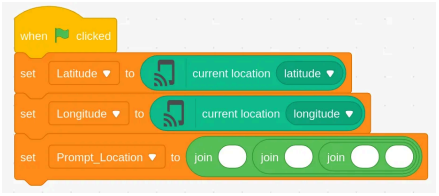
To implement this coordinate-to-city conversion in CodeSkool, prepare the program using various variables that can then be activated as needed. First, create a new variable called "Location" that will store the city name corresponding to the GPS coordinates and a second one called "Prompt_Location," which will contain the structure of the question to ask the AI model.

2. Create the prompt structure

The "Prompt_Location" variable should allow you to transform the latitude and longitude variables already collected by the program into city names. To clearly structure this prompt:

- In the same "When button is clicked" event block where the GPS coordinates are retrieved, add a "set [LocationPrompt] to [value]" block which will be used to define the prompt structure.
- Next, define the structure of the prompt in the [value] field. Start by formulating the request naturally by offering prompts out loud. This could be, for example, "Tell me which city is located at this latitude and longitude." Once you have formulated several ideas, test them in the program.
- To combine text and variables into a complete prompt, use the "join [value] [value]" blocks in the "Operators" section. Because two variables (latitude and longitude) must be integrated with text, use nested join blocks:



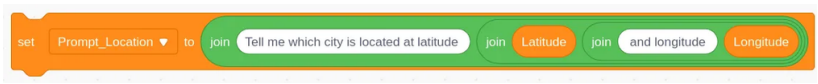


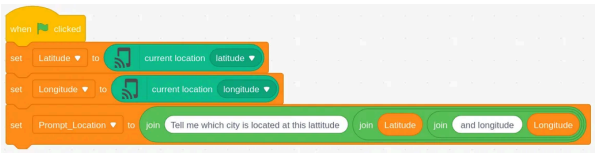
> [Tell me which city is at this latitude] + [variable "Latitude"] + [and longitude] + [variable "Longitude"]

This requires three "group [value] [value]" blocks nested together. This nesting structure is necessary because join blocks can only combine two elements at a time. By nesting them, construct a complete sentence that incorporates multiple variables and text segments:

- First join block: Value 1: The text "Tell me what city is at latitude" (note the space at the end) / Value 2: The second join block

- Second join block: Value 1: The variable “Latitude” / Value 2: The third join block
- Third join block: Value 1: The text "and longitude" (note the spaces before and after) / Value 2: The variable "Longitude"

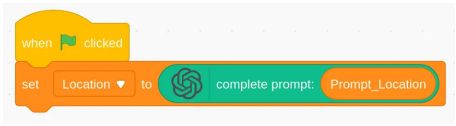


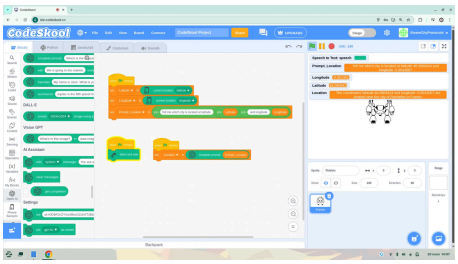


3. Implement AI integration

Test the prompt. First, create a new "When button is clicked" block for clarity and organization in the program. This separates the data collection functionality from the AI processing. In this block:

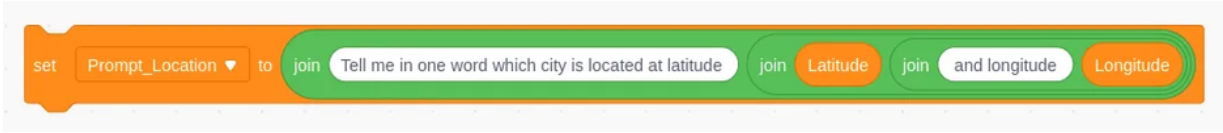
- Add a “set [variable] to [value]” block
- Set the variable to “Location”
- Return to the “Extensions” tab and add the “Open AI” extension to their project
- Find the "complete prompt: [prompt]" block in the Open AI extension
- Set this block as the value to "set Location to [value]"
- Replace the default "prompt" value with their "Prompt_Location" variable



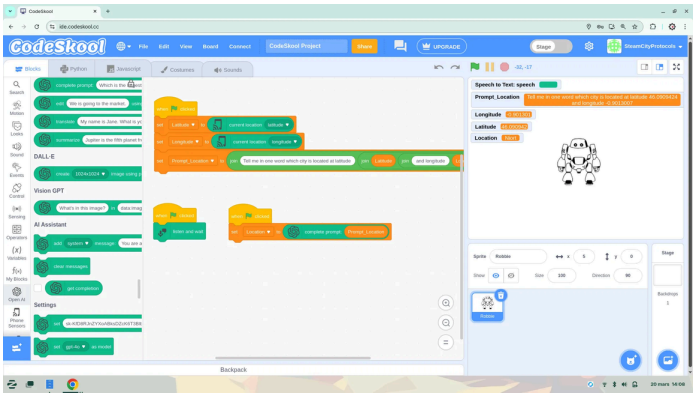


Click the green flag to test the program. Observe that the AI typically provides verbose responses like "The city located at latitude 48.8566 and longitude 2.3522 is Paris, France. Paris is the capital and most populous city of France." This verbosity, while informative, may not be ideal for the chatbot that just needs the city name as a component for subsequent prompts.

Now refine the prompt to ask for only the specific information needed. Modify Prompt_Location to be more specific: "Tell me in one word what city is at latitude [Latitude variable] and longitude [Longitude variable]."



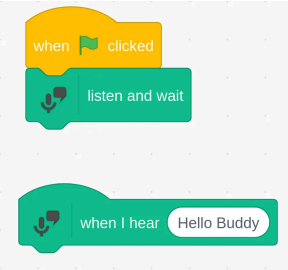
By testing and refining the prompt, learn an important principle of prompt engineering: be specific about both the requested information and the desired response format. This iterative process demonstrates how prompts can be optimized to produce precisely the necessary information, teaching that effective interaction with AI often requires multiple refinements to achieve the desired results.



4 Create a voice-activated chatbot

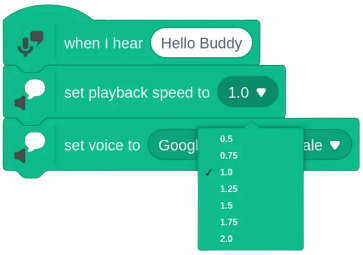
Establish the basic structure of the voice-activated assistant by implementing a trigger phrase and a greeting message.

To implement this in CodeSkool, make sure the "Text to speech" and "Text to speech" extensions are added to the project. In the "Text to speech" blocks, select the "When I hear [value]" trigger block to activate the chatbot. This way, every time the green flag is clicked, thanks to the "Listen and wait" block added at the beginning of the protocol, the chatbot will automatically react if it hears the trigger phrase (in the example "Hello Buddy").



Once this is done, configure the voice settings for the chatbot. Optimizing voice settings is essential for natural and effective communication. The right settings for speed and voice type significantly impact how users understand and interact with the chatbot. Therefore, two blocks must be used to define these settings, available directly in the "Text to Speech" section, in the settings category:

- “set playback speed to [value]”: This block controls how fast the synthetic voice speaks. The value typically ranges from 0 (very slow) to 2 (very fast), with 1 being normal speed
- “choose voice [value]”: This block allows you to select different voice options for the text-to-speech output. Different voices may vary in gender, age, or accent characteristics
-



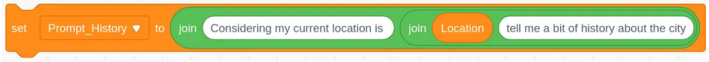
5 Designing conversation flows with prompts

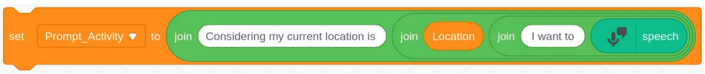
After configuring the voice, create structured conversation flows to guide user interactions with the assistant. This logical block structure serves as an example. Experiment with different organizational approaches, learning from how the assistant responds to build better code and prompts.

Example - Hello Buddy!

Create variables to store different AI prompts for specific topics, as done previously for localization. Our agent must:

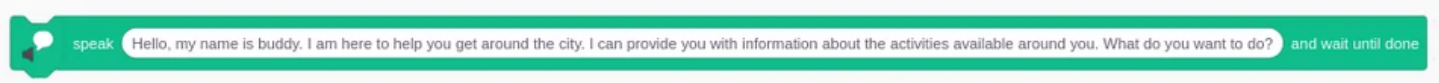
- Suggest nearby activities: We created the variable “Prompt_Activity” using 3 join blocks like: “Considering my current location is [Location variable], I want to [speech]”
- Provide historical information about the current city: We created the variable “Prompt_History” using 2 join blocks like: “Considering my current location is [Location variable], tell me a little bit about the history of the city”



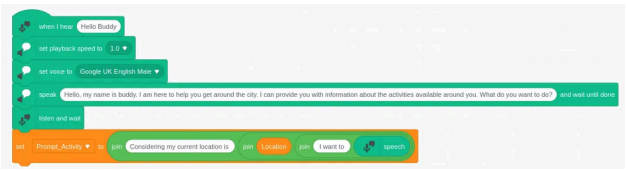


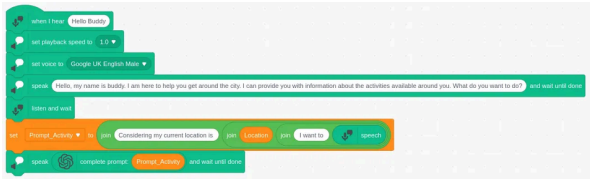
Implement a simple dialog tree that responds to voice input:

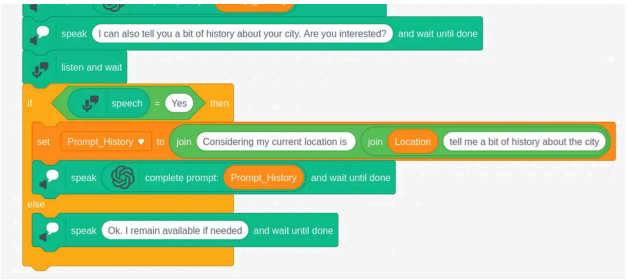
- Once the user says the trigger phrase “Hello Buddy,” Buddy responds with a first suggestion about nearby activities using the following message (with a text-to-speech block): “Hello, my name is Buddy. I’m here to help you get around the city. I can provide you with information about the activities available around you. What would you like to do?”

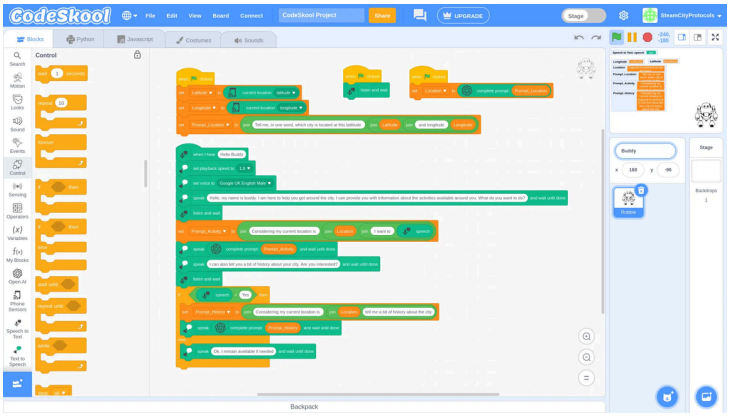


- After Buddy finishes speaking, we use a "listen and wait" block that allows the user to specify the desired activity through the "set Prompt_Activity" block.
- Based on the user's response, such as "eat pizza," Buddy uses a combination of "speak" and Open AI "complete prompt" blocks to provide a detailed response to the request.
- After this discussion ends, Buddy moves on to offer information about the city's history. The interaction continues with Buddy asking, "I can also tell you a little about the history of your city. Are you interested?"
- Unlike the activity interaction which used an open-ended question, here we implement a closed-ended question requiring an "if [ ] then" control block. The user can only answer yes or no, triggering different responses from the bot.
  - If the user says yes, Buddy presents the city's history.
  - If the user says no, Buddy politely ends the conversation: "Okay. I'm available if needed."









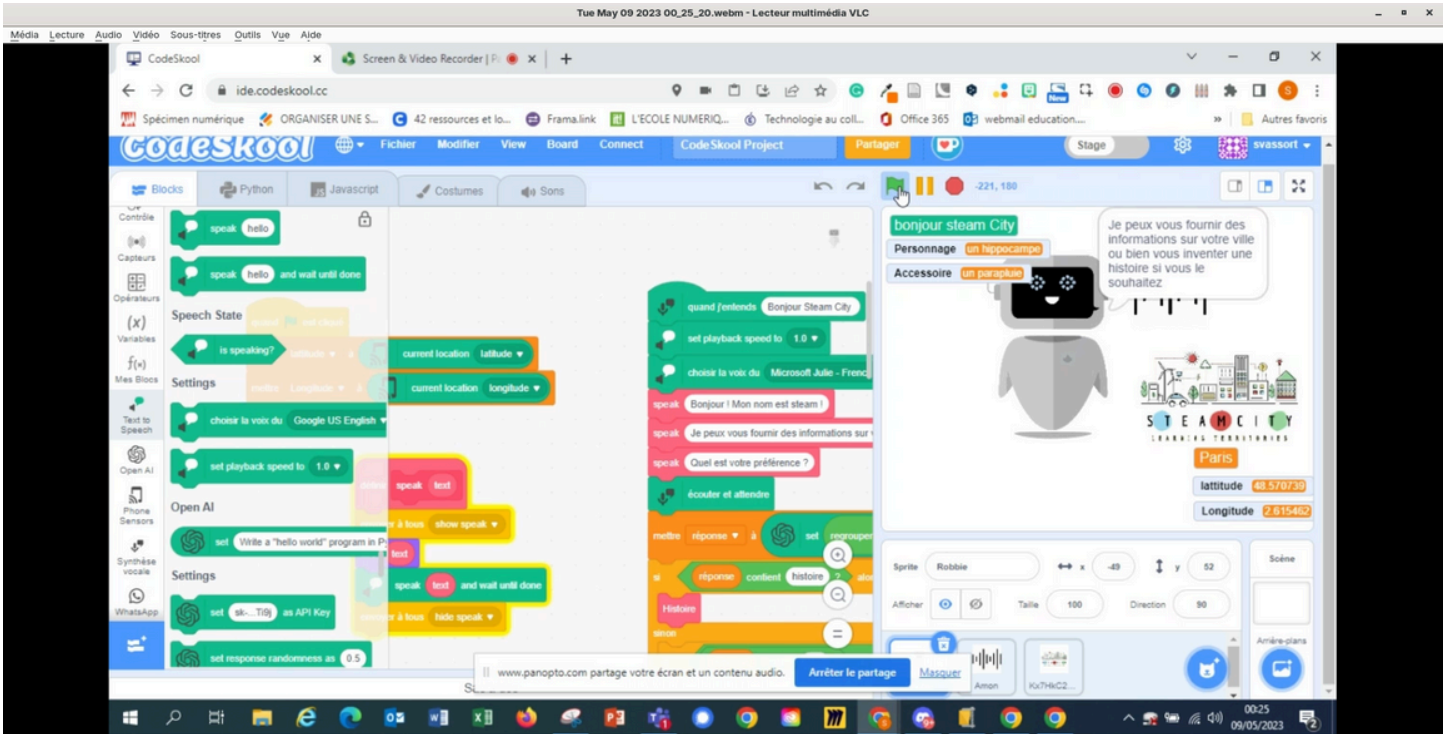
Here is our complete example which is based on all the programming activities carried out in the previous steps available at this link: https://share.codeskool.cc/cqZu

Example

Here's another version of the program based on the same blocks, adding storytelling functions and capabilities using multiple props and variables. To open it, go to ide.codeskool.cc and select "open project from link."

You can use the share link: https://share.codeskool.cc/eT4U

Here is a demonstration video (in French): https://drive.google.com/file/d/19cR5d5_sKCEtvqY__fR5_TJbMsap-Knd/view?usp=drivesdk


Tue May 09 2023 00_25_20.webm - Lecteur multimédia VLC
Média Lecture Audio Vidéo Sous-titres Outils Vue Aide
CodeSkool
Screen & Video Recorder
ide.codeskool.cc
CodeSkool
Fichier Modifier View Board Connect
CodeSkool Project
Partager
Stage
svassort
Blocks
Python
Javascript
Costumes
Sons
speak hello
speak hello and wait until done
Speech State
is speaking?
Settings
choisir la voix du Google US English
set playback speed to 1.0
Open AI
set Write a "hello world" program in P
Settings
set sk-...Tdj as API Key
set response randomness as 0.5
current location latitude
current location longitude
speak text
show speak
speak text and wait until done
hide speak
quand j'entends Bonjour Steam City
set playback speed to 1.0
choisir la voix du Microsoft Julie - Frenc
speak Bonjour ! Mon nom est steam !
speak Je peux vous fournir des informations sur
speak Quel est votre préférence ?
écouter et attendre
mettre réponse à set regrouper
si réponse contient histoire 2
Histoire
sinon
bonjour steam City
Personnage un hippocampe
Accessoire un parapluie
Je peux vous fournir des informations sur votre ville ou bien vous inventer une histoire si vous le souhaitez
Paris
lattitude 48.570730
Longitude 2.615462
Sprite Robbie
x -49
y 52
Afficher
Taille 100
Direction 90
Scène
Arrière-plans
www.panopto.com partage votre écran et un contenu audio.
Arrêter le partage
Masquer
00:25
09/05/2023