STEAMCITY CURRICULUM

BOT BUDDY ADVENTURE

Designing a ChatBot for Accessibility

thematic: artificial intelligence and new technologies

sub-theme: citizen engagement, governance and data

Introduction

A **conversational agent**, often referred to as a **chatbot** or **virtual assistant**, is a computer program designed to engage in natural language conversations with users. It utilizes various **algorithms and language processing techniques** to understand and respond to user queries or commands, providing information, assistance, or performing tasks such as **speech recognition** to convert spoken words to text, **natural language processing** to understand user intent, **semantic analysis** to extract meaning from text, **dialog management** to maintain conversation flow or **text-to-speech synthesis** for vocal responses. To develop an **autonomous conversational agent**, it is essential to understand the different technical functions required and to design the appropriate solutions for each of them. Choosing the right combination of **speech recognition, natural language processing and text-to-speech** allows the users to interact with the conversational agent naturally and intuitively. In this protocol "**Bot Buddy Adventure**", students will explore the development of a conversational agent by identifying and implementing the key technical features. They will learn to work with speech recognition for converting voice to text, utilize natural language processing to understand user intent, and implement text-to-speech for response generation. Through hands-on experience with these technologies, students will gain practical understanding of how artificial intelligence enables natural human-computer interaction.

AI Solutions for Urban Challenges: Making Cities Smarter and More Accessible. The Bot Buddy Adventure protocol guides students through creating a conversational agent designed to enhance **urban accessibility and navigation for all citizens**. The agent combines GPS positioning, voice recognition, and AI to provide an inclusive interface for accessing essential city services, public transportation, and community resources. The conversational agent directly supports the **development of inclusive, smart cities by making urban information more accessible**. This technology can help bridge the digital divide by providing voice-based interfaces that make city services and information available to diverse populations, including those who might struggle with traditional text-based interfaces. Beyond the technical aspects of AI, this protocol encourages **meaningful discussions about urban accessibility and inclusion**. Students explore how AI-powered solutions can address various accessibility challenges in cities, such as helping visually impaired citizens navigate public spaces, supporting elderly residents in accessing municipal services, assisting non-native speakers with language barriers or providing alternative interfaces for people with different abilities. These discussions help students understand the broader social impact of AI technology and its role in creating more inclusive urban environments for all citizens.

Interdisciplinarity

Technology and engineering
Art & Design
Physics & Chemistry

Sustainable Development Goals

Overview

Protocol Structure

This protocol introduces students to the development of an autonomous conversational agent through a structured, hands-on learning experience. The process is divided into four complementary steps that build upon each other to create a comprehensive understanding of conversational AI systems.

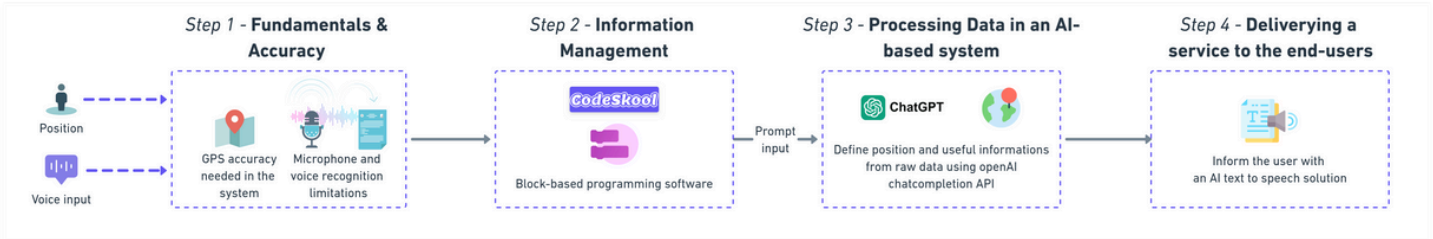
Step 1: System Fundamentals & Accuracy: This initial phase focuses on understanding the accuracy limitations of the core technologies powering conversational agents. Students explore geolocation systems and voice recognition technologies through comparative testing across different platforms and environments. By systematically documenting performance variations, they develop critical awareness of real-world constraints that will inform their design decisions, creating a scientific mindset essential for responsible AI development.

Step 2: Data Collection and Information Management: Building on their understanding of technological limitations, students implement their first functional prototype. This phase introduces the concept of variables as the "memory" of their program, teaching students how to capture and store both GPS coordinates and voice input. Through hands-on programming in CodeSkool, students create the essential data collection layer of their conversational agent, experiencing how abstract concepts translate into practical implementation.

Step 3 - Data Processing in an AI-based system: This phase transforms raw data into meaningful information through AI integration. Students explore language models and discover the art of prompt engineering—learning to craft effective instructions that guide AI responses. Through iterative testing and refinement, they convert technical coordinates into human-readable location names and develop skills for creating contextually aware prompts. This experiential learning helps students understand both the capabilities and limitations of language models.

Step 4: Delivering Accessible Information to the end-users: The final phase completes the conversational agent by implementing voice output capabilities and conversation design. Students configure text-to-speech parameters, create logical dialogue flows, and develop prompts that deliver relevant information about urban environments. This culminating activity integrates all previous components into a fully voice-interactive assistant that can engage in natural conversation, demonstrating how technology can enhance accessibility and provide valuable location-based services.

Through this progression, students develop both technical skills and critical thinking about AI systems. They move from understanding technological limitations to implementation, from basic data collection to intelligent processing, and finally to creating accessible user experiences. Each step builds naturally upon the previous one, creating a coherent learning journey that mirrors real-world AI development processes while emphasizing responsible design practices.



Getting started

Duration: The activity will take place over four sessions, each lasting around 30 to 60 minutes.

Level of difficulty:

Moderate. The activity has been structured in several stages, but does not follow a turnkey model. It needs to be appropriated by the teacher.

Material needed: Computer and mobile device (smartphone)

Online resources:

- Introduction to prompting: https://www.youtube.com/watch?v=8IQ9i_QoA3A
- Introduction to language model: https://youtu.be/K8gOvC8gvB4?si=EOvjZwRlWw9td9ww

Glossary

Keywords & Concepts	Definitions
Conversational agent	Also known as a chatbot, an intelligent conversational agent is a robotized software program that uses a database to communicate with a human in natural language, either verbally or in writing.
Semantic analysis	Semantic analysis enables computers to interpret the correct context of words or phrases with multiple meanings, which is essential for the accuracy of text-based natural language processing applications. In fact, rather than simply analyzing data, this technology goes a step further and identifies relationships between data elements.
Natural language processing	Automatic natural language processing is a multidisciplinary field involving linguistics, computer science and artificial intelligence. It aims to create tools capable of interpreting and synthesizing text for a variety of applications.
Language model	An AI system trained on large amounts of text data to understand and generate human-like language, capable of tasks like text completion, translation, and answering questions.
Prompt engineering	The practice of crafting effective input instructions for language models to obtain desired outputs, requiring careful consideration of context, clarity, and constraints.
Text-to-speech synthesis	Technology that converts written text into spoken words, using artificial intelligence to generate natural-sounding speech with appropriate rhythm and intonation.

Protocol

Step 1: System Fundamentals & Accuracy

Background and description of the problem to be solved in this step: In the Bot Buddy Adventure, students will be guided for creating an application featuring an autonomous conversational agent. In the proposed learning scenario, they will build a solution that helps people discover local leisure activities. In our increasingly connected world, accessibility to urban information remains a challenge for many citizens. Voice-based interfaces can bridge this gap, providing intuitive access to location-based services. In this first step "System Fundamentals & Accuracy," the objective will be to evaluate and understand the accuracy limitations of both geographical positioning systems and voice recognition technologies. Students will learn how to determine a user's location with appropriate precision for local information delivery, while also assessing voice interaction reliability compared to text input. This examination of accuracy in both input mechanisms will help students design a more robust, natural, and accessible experience that accounts for real-world technological constraints and optimizes performance within these limitations.

Learning Objectives: Understand the fundamentals of geolocation and GPS. Understand the accuracy limitations and fundamental principles of geolocation systems and GPS technology. Evaluate the reliability of speech recognition solutions across different environments and user conditions. Develop problem-solving skills by working on projects that involve geolocation and speech recognition. Learn to identify potential problems and iterate on the design to improve the performance and efficiency of a system.

Conceptualisation

In this first step, students explore the foundations of location-based services and voice interfaces, which together form the **essential input mechanisms for an accessible urban navigation assistant**.

The project aims to help all users navigate urban environments, with special attention to accessibility needs. For this assistant to function effectively, it must master two fundamental capabilities: **knowing where the user is located and understanding what the user is asking**. These capabilities represent the input foundation upon which all subsequent features will be built.

Research Question 1: Location Awareness

How can we accurately determine a user's physical location in an urban environment to provide relevant, localized information?

Hypothesis: By using GPS technology, we can create a system that reliably determines user location within urban environments with sufficient accuracy to provide relevant local information.

Key Concepts:

- **GPS Fundamentals**: GPS operates through a network of satellites that transmit signals containing precise time information. Receivers calculate their position by measuring the time it takes signals to arrive from multiple satellites (triangulation).
- **GPS Accuracy Factors**: Signal quality varies based on atmospheric conditions, surrounding buildings, and receiver quality, with typical smartphone accuracy ranging from several meters in optimal conditions to less precise measurements in challenging urban environments.
- **Geospatial Context**: Understanding how raw coordinates relate to meaningful locations like streets, neighborhoods, and points of interest in a city.

Research Question 2: Voice Accessibility

How can we ensure that our interface accepts voice commands and effectively converts them into queries a system can process?

Hypothesis: By implementing robust guidelines for our voice recognition system, we can create a voice interface that reliably understands user requests across various speaking patterns, environmental conditions, and accents.

Key concepts:

- **Speech Recognition Processing**: Voice input passes through four key stages: audio capture, feature extraction (identifying phonemes), acoustic modeling (mapping sounds to words), and language modeling (determining likely word sequences).
- **Accessibility Considerations**: How voice interfaces remove barriers for users with different abilities and needs, including visual impairments, motor limitations, or reading difficulties.
- **Environmental Challenges**: Understanding how background noise, accents, and speech patterns affect recognition accuracy and developing strategies to improve reliability.

By exploring these two questions in parallel, students gain a deeper understanding of each technology while seeing how they complement each other in creating an accessible, location-aware service. The investigation activities will allow students to test their hypotheses through hands-on experimentation with both location services and voice recognition components.

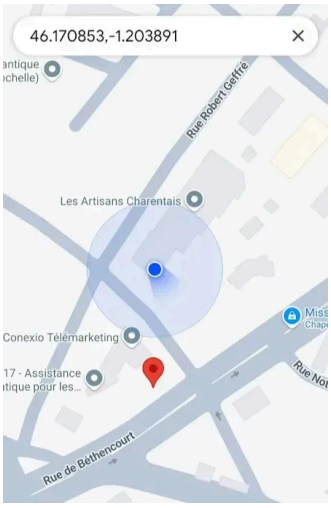
Students Investigation

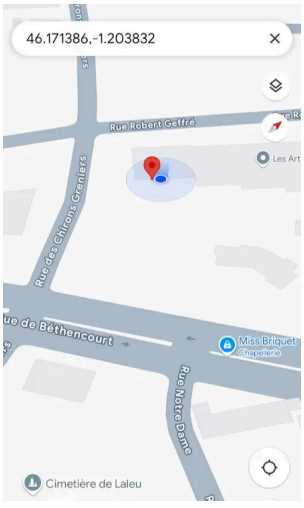
This investigation phase guides students through hands-on exploration of GPS and voice recognition technologies, the fundamental mechanisms for the Bot Buddy Urban Assistant.

Exploring GPS Capabilities

In this first activity, students will explore how to integrate GPS functionality into their virtual assistant by comparing different geolocation methods and considering the level of accuracy needed for their project. Have students test several GPS solutions to understand the capabilities and limitations of location services, regardless of the specific implementation method.

Make them access their device's built-in GPS through their map application (Google Maps, Apple Maps) to understand how geographic coordinates work in practice. Have them record their current latitude and longitude while in the classroom. When students compare their results, they will notice small variations despite being in the same location.





Have them experiment with alternative GPS methods like online mapping services, dedicated GPS apps, and web applications that use the browser's Geolocation API. Ask them to compare the location results they get across different devices. Students can fill the following table with their measurements and observe how location accuracy varies across different devices and technologies. This comparison will help them understand the reliability considerations for their virtual assistant.

Device Type	Latitude	Longitude	Variation from Reference
Smartphone GPS			
Tablet GPS			
Laptop Browser Location			
Dedicated GPS Device			

For the "**Variation from Reference**" column, students can choose one measurement as a reference point (perhaps the most precise device available) and calculate how much other measurements differ from it. This provides a concrete way to understand accuracy variations.

Guide a discussion about these variations, focusing on their implications for a location-based conversational agent:

- Does a virtual assistant helping people discover local activities need pinpoint accuracy (within 1-2 meters) or just general neighborhood awareness?
- How does the required precision depend on the specific use case? For example, finding the nearest restaurant might need only approximate location, while providing turn-by-turn directions requires greater accuracy.
- What environmental factors seem to affect GPS accuracy? (Building materials, weather, urban density)
- How might these accuracy limitations influence the design of your virtual assistant?

Have students conclude by determining the level of accuracy needed for their local assistant and documenting potential challenges they might need to address.

Exploring Voice Recognition Limitations

Similarly, for voice recognition, students could experiment with different voice assistants (Siri, Google Assistant, Alexa…), browser-based speech recognition, dedicated dictation software, or even voice typing features in word processors. The important learning comes from observing how these systems perform under different conditions, not from the specific coding environment used. To perform their test, make them create a test set of sentences related to urban navigation that their assistant might need to understand, such as: "Where is the nearest park?", "How do I get to the museum?", "Are there any cafés nearby?", "What time does the cinema close?".

Experiment with different speaking conditions: speaking clearly versus naturally, different distances from the microphone, with and without background noise, with different speakers (classmates with varied voices/accents).

Analyze the results to identify patterns: Which types of phrases are recognized most accurately, which words or sounds are frequently misinterpreted, what speaking conditions produce the best results? Record results in a simple table showing:

Device	Spoken Sentence	Tone used	System Sentence Recognised	Accuracy
Specify the system used for the test	*Enter the exact phrase you spoke*	*Note if you spoke slowly, quickly, with an accent*	*Write what the system heard*	*Rate accuracy (Good/Fair/Poor)*
Siri				
Hey Google				
Alexa				
Smart Speaker				
Cortana				
Dictation software				

After collecting data, guide students in analyzing the results to identify patterns: Which types of sentences are recognized most accurately across systems? Which words or sounds are frequently misinterpreted? What speaking conditions produce the best and worst results? Do certain systems perform better for specific types of phrases? How do accents or speech patterns affect recognition accuracy?

Lead a discussion about the implications of these findings for designing voice interfaces:

- How might you design your assistant to minimize misunderstandings?
- What strategies could help users speak in ways the system can understand?
- How could you handle uncertain recognition results?
- What backup methods might help when voice recognition fails?

This testing activity helps students understand that voice recognition technology has inherent limitations they will need to address when designing their virtual assistant. By identifying common recognition errors across platforms, they can develop strategies to improve reliability, such as designing confirmation steps for ambiguous commands or providing visual feedback about what the system understood.

Conclusion & Further Reflexion

- **Knowledge Acquired:** In this first step of the Bot Buddy Adventure, students have established a critical foundation for their conversational agent by thoroughly examining the accuracy and limitations of both GPS technology and voice recognition systems. They have conducted systematic testing of existing technologies to understand their real-world performance characteristics. Students have explored how GPS coordinates vary across different devices and in different environments, collecting empirical data that reveals the inherent imprecision in location services. These comparative tests have given them a concrete understanding of location accuracy as a spectrum rather than a binary property, and how environmental factors like building materials and urban density influence positioning reliability. Similarly, their methodical testing of various voice recognition platforms has revealed the significant impact of speaking patterns, background noise, accent variations, and other factors on recognition accuracy. By documenting specific recognition errors and patterns of misinterpretation, students have developed a nuanced understanding of the challenges in creating reliable voice interfaces. This exploratory approach has helped students identify the specific accuracy requirements for their own urban navigation assistant, allowing them to make informed design decisions based on empirical evidence rather than assumptions.
- **Classroom Implementation Insights:** This step works effectively when combining physical exploration with comparative analysis. Making students testing GPS in various environments makes abstract concepts tangible, while structured testing protocols develop scientific inquiry skills. The collaborative nature of data collection creates natural opportunities for peer learning. The comparative testing leads to thoughtful discussions about technology limitations, helping students develop critical thinking about tools they'll eventually use. For assessment, focus on students' ability to draw evidence-based conclusions and articulate how these findings will influence their design decisions.
- **Key Learning Outcome:** Through this step, students have learned to design systematic testing protocols that reveal technological strengths and weaknesses. They have developed critical analysis skills by comparing performance across different platforms and conditions. Students now understand that input accuracy exists on a spectrum, requiring systems to account for inevitable imprecisions. They have begun developing an engineering mindset that acknowledges constraints while seeking optimal solutions, a perspective that will inform all subsequent development work in their conversational agent project.

To deepen students' engagement with the concepts covered in this step, consider exploring these questions through discussion or reflective writing:

1. How does the physical environment affect GPS accuracy, and what implications does this have for location-based services?
2. What patterns emerge in speech recognition performance across different conditions? Consider factors like background noise, accents, and speech patterns, and discuss potential design solutions to improve accessibility.
3. How can GPS and voice recognition technologies work together effectively while maintaining user privacy? What security considerations should be addressed when collecting and storing this sensitive data?
4. Who benefits most from combining GPS and voice interfaces, and how can we ensure these technologies remain accessible to diverse user groups?

Step 2: Data Collection and Information Management

Background and description of the problem to be solved in this step: Thanks to step 1, students are able to understand the accuracy limitations of GPS and voice recognition technologies. Now in step 2, they will implement a basic system to collect these two fundamental data inputs. Students will apply their knowledge by creating a functional prototype in CodeSkool that captures location information and voice input while accounting for the real-world limitations they have identified. This implementation will form the essential data collection layer of their Bot Buddy Urban Assistant, enabling them to store information before processing it in later steps.

Learning Objectives: Implement basic GPS functionality to retrieve and store user location coordinates. Create a simple voice recognition interface that converts speech to text. Understand variable types and their roles in storing information. Implement data processing workflows that connect location awareness with user queries. Develop technical skills in working with device sensors and external APIs.

Conceptualisation

In this second step, students move from understanding to implementation, creating a functional prototype that collects the two fundamental inputs for their conversational agent. Building on their exploration of technology limitations from Step 1, they will now create a working system that actually captures user location and voice input.

Research Question 1: Location Data Collection

How can we implement a system that effectively captures and stores a user's physical location for later usages?

Hypothesis: By using CodeSkool's Phone Sensors extension and implementing appropriate variables, students can create a system that captures and stores location coordinates with sufficient reliability for an urban navigation assistant.

Key Concepts:

- **Variables**: Programming constructs that serve as containers for data storage, similar to labeled boxes in computer memory. Variables enable programs to temporarily store and recall information like coordinates, measurements, or text in RAM (Random Access Memory). Just as human short-term memory holds information temporarily, RAM stores variable data only while the computer is powered on, unlike permanent storage on a hard drive.
- **Sensor Access**: Methods for accessing a device's GPS capabilities through programming interfaces, including handling potential unavailability.
- **User Feedback**: Techniques for communicating location data and its reliability to users, helping them understand potential limitations.

Research Question 2: Voice Input Collection

How can we implement a system that effectively captures voice input and converts it to text while accounting for recognition limitations?

Hypothesis: By using CodeSkool's Speech to Text extension and implementing appropriate feedback mechanisms, students can create a system that reliably captures and stores user queries in a format suitable for further processing.

Key Concepts:

- **Speech Capture**: Methods for activating and processing speech input through programming interfaces, including voice command recognition and transcription.
- **Text Storage & Analysis**: Approaches for storing transcribed speech in variables and identifying key words or phrases to determine user intent. For example, searching for specific terms like "where," "how," or "what" can help categorize the type of request.

- **User Request Processing**: Techniques for analyzing and categorizing user queries by identifying command types (e.g., location requests, information queries) and extracting relevant keywords for appropriate response generation.
- **Recognition Feedback**: Techniques for confirming understanding, storing relevant information in variables for future use, and handling potential recognition errors through verification prompts.

By implementing these two data collection mechanisms, students create the essential foundation for their conversational agent. The ability to capture and store both location and voice input enables all subsequent processing, analysis, and response generation. These components form the critical interface between the physical world and the digital assistant, determining what information is available for helping users navigate their urban environment.

Students Investigation

In this step, students will implement the core data collection capabilities needed for their Bot Buddy Urban Assistant. In our example, we have been using the CodeSkool (https://ide.codeskool.cc/) block-based programming software. CodeSkool is particularly well-suited for this project because it combines Open AI integration with phone sensor extensions and text-to-speech capabilities.

Several alternatives exist 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 classrooms with more advanced programming experience, Python libraries like geocoder or geopy offer sophisticated location services that can be integrated into larger applications.

Enter CodeSkool and Store GPS Information Using Variables

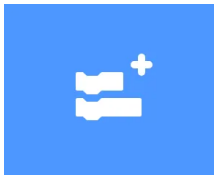
The ability to store and remember information is one of the most fundamental concepts in programming. Variables act as the "**memory**" of your program, allowing it to **retain important data** like user location that can be used throughout different parts of your application.

In this first activity, students will learn how to **create variables to store GPS coordinates**, giving their conversational agent the ability to "**remember**" where the user is located.

- To implement the GPS functionality, students should start by **creating a new project and access the editor** in CodeSkool.

- Before accessing any GPS data, they need to establish **memory locations** in their program where this information will be stored. They should begin by going to the "**Variables**" section in the block palette and click "**Make a Variable**." Students should create a variable named "**Latitude**" to store the north-south position of the user. This variable will act as a dedicated container that remembers this specific piece of information throughout their program. Then, they should 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.

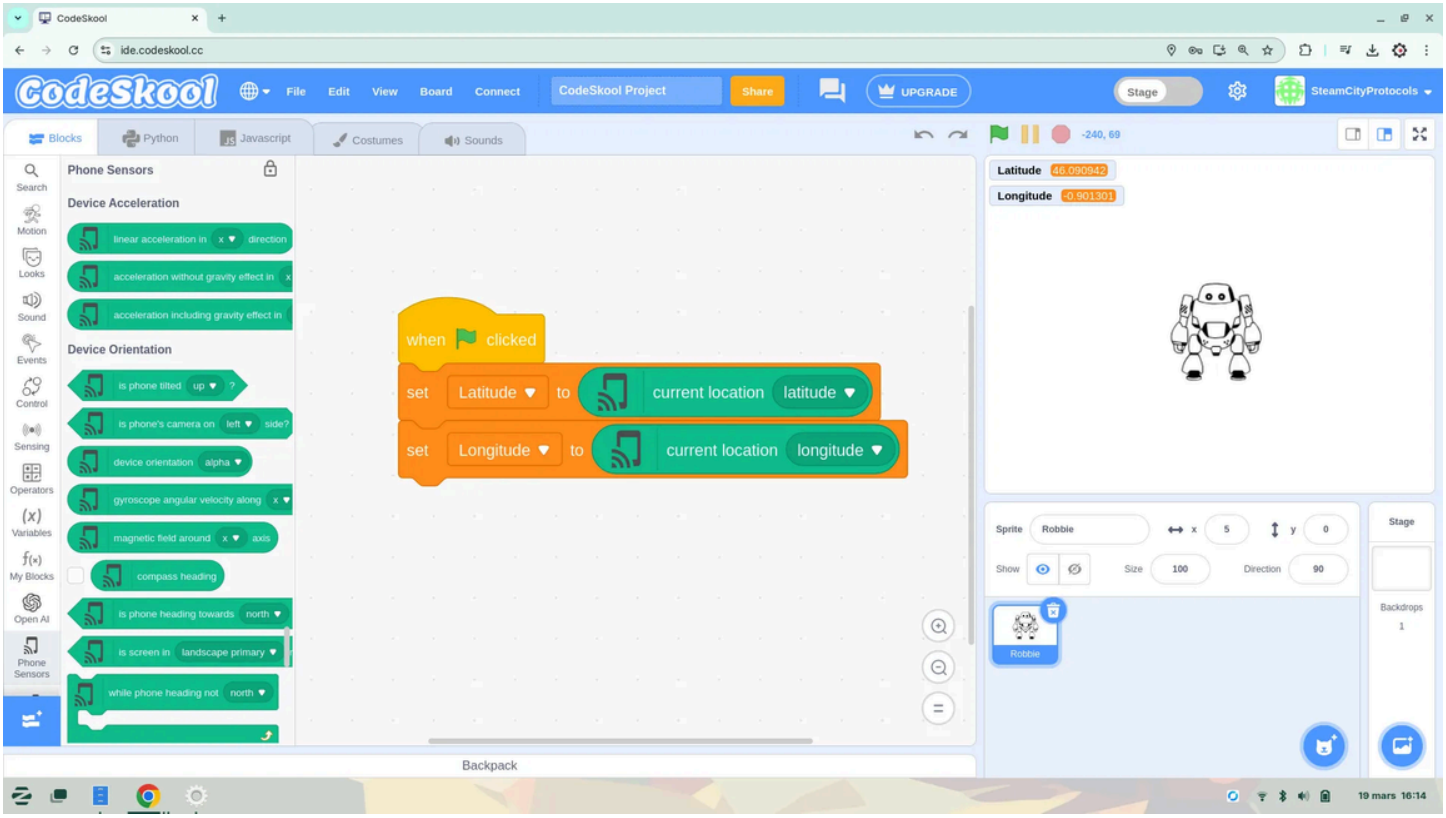
- Now that students have created the memory containers for their location data, they need to **access the device's GPS capabilities** to fill these variables with actual coordinates. From the main interface, they should access the "**Extensions**" button located at the very end of the toolbar. In the extensions library, search for "**Phone sensors**" and add this extension to their project to gain access to GPS functionality.



- After adding the extension, they can find **GPS-related blocks** in the new category that appears in their block palette. They can search for the subcategory of blocks called "**GPS location**" in which they should the block "**current location**" for both longitude and latitude.

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

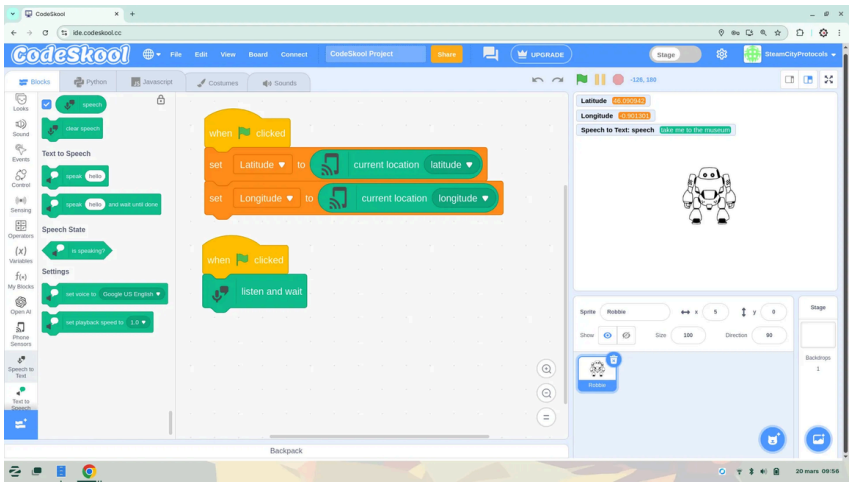
1. To trigger the data collection process, add a "**When button clicked**" event block to your workspace. This will serve as the starting point for capturing the user's location.
2. From the "**Variables**" section, drag two **"set [variable] to [value]"** blocks and connect them to your button event. These blocks are the mechanism by which information moves from sensors into your program's memory.
3. Configure the first block to set "**Latitude**" and the second to set "**Longitude**." The variable selection demonstrates that your program now has specific memory locations ready to hold different types of information.
4. Connect the appropriate "**current location**" 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 your program's memory.
5. Click on the **green flag in the testing pannel** to verify that your variables are successfully storing information.



This program will run either on the computer using the location of the machine, or through a mobile device thanks to the Phone Sensors Extension.

Implementing and Testing Voice Recognition

Based on the program created, students will need to implement the second functionnality for speech recognition. On the same editor, students should 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, they can find **speech recognition-related blocks** in the new category that appears in their block palette. They should look for the block "**Listen and wait**. For more clarity in the blocks, make the students add a second "**When button clicked**" bloc to identify the two basic functionnalities of the system.

Once students have implemented the basic voice recognition feature, they can reuse the same testing method as in step 1 to systematically evaluate its accuracy to understand its capabilities and limitations. They can reuse the test

sentences and complete their table with their own system capabilities:

Spoken Phrase	System Recognition	Accuracy
The sentence formulated by the students	*What the system transcribes from your speech*	*Rating of the understanding (Good / Fair / Poor)*
"Where is the nearest park?"		
"Take me to the museum"		
"Is there a café nearby?"		
"What's the time for cinema?"		

Conclusion & Further Reflexion

- **Knowledge Acquired:** In this second step of the Bot Buddy Adventure, students have moved from understanding to implementation, creating a functional prototype that collects the two fundamental inputs for their conversational agent: location data and voice queries. They have learned to access device sensors through CodeSkool's extensions, store information in variables, and perform basic analysis of user requests. Students have gained practical experience working with variables as storage containers for different types of information. They understand how these variables allow a program to remember user inputs and sensor readings, creating continuity in the user experience. The concept of variables as "memory" for their program provides a foundation for more complex data handling in future steps.
- **Classroom Implementation Insights:** This implementation step works best with a hands-on, exploratory approach where students can test their code frequently. Consider having students work in pairs, with one student speaking queries while the other observes how the system responds. For troubleshooting common technical issues, you can create a shared document where students can record problems they encounter and solutions they discover - that can also be used for the coming steps of the protocol. This collaborative debugging approach helps the entire class overcome obstacles more efficiently.
- **Key Learning Outcomes:** Through this step, students have developed several important skills and understandings. They have learned to access device sensors programmatically, capturing real-world data like location and voice input. They understand how to store different types of information in variables, creating a program with "memory" that can use this information across different functions.

To deepen students' engagement with the concepts covered in this step, consider exploring these questions through discussion or reflective writing:

1. How do variables enable your program to "remember" information, and what additional data might you need to store to make your conversational agent more helpful and context-aware?
2. What challenges did you encounter when implementing GPS and voice recognition, and how did your understanding from Step 1 help you address these challenges?
3. How might your query analysis be improved to better understand user intentions and handle situations where GPS or voice recognition performs poorly?
4. How does the ability to store information in variables create new possibilities for the types of assistance your conversational agent could provide?

Step 3: Data Processing in an AI-based system

Background and description of the problem to be solved in this step: Thanks to steps 1 and 2, students have already explored the accuracy limitations of location and voice technologies, and implemented a basic system that collects and stores this data using variables. Now in Step 3, they face a new challenge: how to process this raw data to generate meaningful responses. While they have latitude and longitude coordinates and transcribed voice queries, these alone do not provide the contextual information users need. In this step, students will explore how artificial intelligence, specifically language models, can transform this raw data into personalized, localized, and up-to-date information.

Learning Objectives: Understand the fundamental principles of language models and how they process information. Learn the basics of prompt engineering to effectively communicate with AI systems. Develop skills for integrating geolocation data and user queries into well-structured prompts. Implement a system that connects their data collection code with AI capabilities. Critically evaluate AI responses for accuracy and relevance to user needs.

Conceptualisation

In this third step, students move from data collection to data analysis and intelligence integration. Having established the foundational inputs of location awareness and voice understanding, they now explore how artificial intelligence can transform this raw data into meaningful, contextual information for users. Now that their assistant can capture location and voice, it must develop the **intelligence to provide relevant, helpful responses**. This AI integration represents the "brain" that will process the inputs and generate appropriate outputs.

Research Question 1: AI Integration

How can we use language models to transform raw location and query data into personalized, contextually relevant information about urban environments?

Hypothesis: By crafting well-structured prompts that incorporate both the user's location context and the specifics of their query, we can leverage language models to generate relevant and accurate responses about local activities and points of interest. Based on the analysis of historical data and user feedback, the conversational agent can also personalize its recommendations over time.

Key Concepts:

- **Language Models**: AI systems trained on vast text datasets that can understand and generate human-like text based on the inputs they receive.
- **Prompt Engineering**: The practice of crafting input texts (prompts) that effectively guide language models to produce desired outputs.
- **Contextual Understanding**: How language models interpret information based on the surrounding context provided in the prompt.

Research Question 2: Prompt Design

How can we formulate effective prompts that clearly communicate both location context and user intent to language models?

Hypothesis: By developing a structured prompt format that systematically including specific information, we can create prompts that consistently yield relevant, location-aware responses from language models.

Key Concepts:

- **Prompt Structure**: Organizing information within prompts in a clear, consistent format that language models can reliably interpret.

- **Geographic Context Integration**: Methods for incorporating location information in a way that helps the AI understand spatial relationships.
- **Intent Clarification**: Ensuring the AI correctly interprets the user's request within the geographic context.

By exploring these questions, students will develop an understanding of how artificial intelligence can bridge the gap between raw data collection and meaningful user assistance. The language model becomes the crucial processing layer that transforms coordinates and voice queries into helpful information about urban environments, creating a truly intelligent conversational agent.

Students Investigation

In this investigation phase, students will explore how to integrate AI capabilities into their Bot Buddy Urban Assistant. Building on the data collection system they created in Step 2, they will now add the intelligence layer that processes this information to generate relevant responses.

Understanding Large Language Models

Before implementing AI integration, students should develop a **basic understanding of how language models work**. Begin with a brief introduction to language models as AI systems trained on vast text datasets that can understand and generate human-like text. Have students conduct brief research on language models providing them with recommended resources such as suggested video (for instance How Language Models Work), simple articles explaining language model principles or examples of language model interactions. This foundation will help students understand why prompt engineering is important and how it affects the quality of AI responses.

First AI-Integration - Location to City Conversion

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

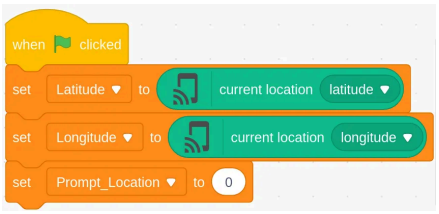
1. Preparing the variables

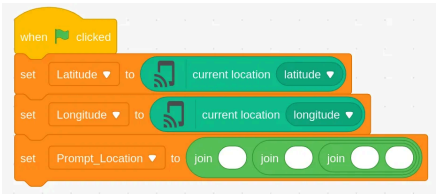
To implement this coordinate-to-city conversion in CodeSkool, we encourage students to prepare their programme using diverse **variables** they can then activate when they need. They can first create a new variable called "**Location**" that will **store the city name** corresponding to the GPS coordinates and a second one called "**Prompt_Location**", that will contain the **structure of the question to be asked to the AI model**.

2. Creating the prompt structure

The variable "**Prompt_Location**" should enable transforming the latitude and longitude variables already collected by the programme into a city name. For structuring this prompt clearly:

- In the same "**When button clicked**" event block where they retrieve the GPS coordinates, students should add a block "**set [LocationPrompt] to [value]**" that will be used to define the structure of the prompt.
- They should then define in the [**value**] area, the structure of the prompt. Start by asking to the students how they would **naturally phrase their request** by proposing prompts outloud. It can be for instance "**Tell me which city is located at this latitude and longitude**". Once they have several ideas, let's test them in the programme.
- To combine text and variables into a complete prompt, students will need to use the "**join [value] [value]**" blocks from the "Operators" section. Because we need to integrate two variables (latitude and longitude) with text, students will need to use nested join blocks:

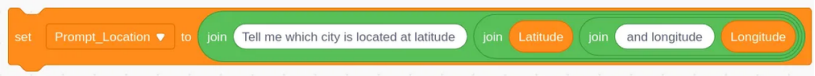


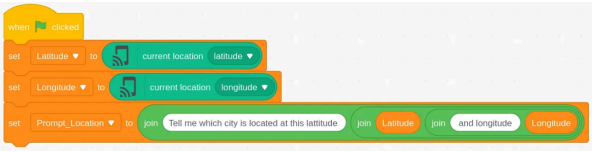


> **[Tell me which city is located at this lattitude] + [variable "Latitude"] + [and longitude] + [variable "Longitude"]**

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

- **First Join Block:** Value 1: The text "Tell me which city is located at latitude " (note the space at the end) / Value 2: The second join block
- **Second Join Block:** Value 1: The "Latitude" variable / Value 2: The third join block
- **Third Join Block:** Value 1: The text " and longitude " (note the spaces before and after) / Value 2: The "Longitude" variable

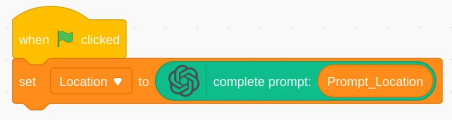


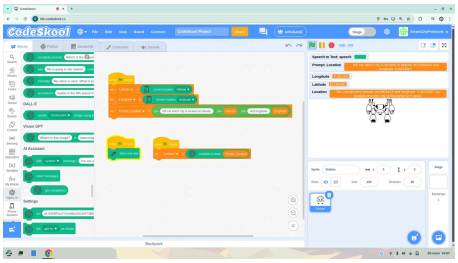


3. Implementing the AI Integration

Let the students test their prompt. They should first create a new "**When button is clicked**" block for clarity and organization in their program. This separates the data collection functionality from the AI processing. In this block, they should:

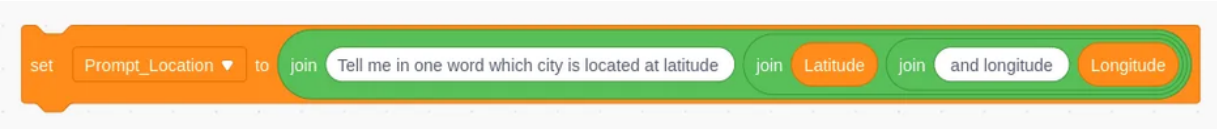
- Add a "**set [variable] to [value]**" block
- Set the variable to "**Location**"
- Return to the "**Extensions**" tab and add the "**Open AI**" extension to their project
- Find the "**complete prompt: [prompt]**" block in the Open AI extension
- Place this block **as the value** for "set Location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



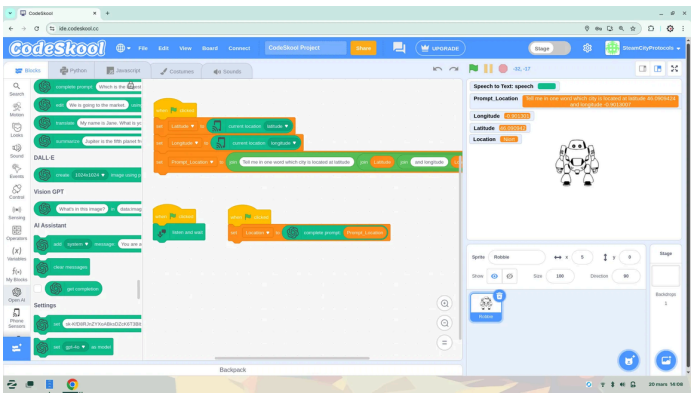


Make students click on the green flag to test their programme. They will observe that the AI typically provides verbose responses like "The city located at latitude 48.8566 and longitude 2.3522 is Paris, France. Paris is the capital and most populous city of France." This verbosity, while informative, may not be ideal for their conversational agent that needs just the city name as a component for subsequent prompts.

Students should now refine their prompt to request only the specific information needed. They can modify Prompt_Location to be more precise: **“Tell me in one word which city is located at latitude [Latitude variable] and longitude [Longitude variable]”**.



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



Understanding AI Limitations in Location Identification When testing on computers rather than mobile devices, students might notice that the locations returned are not always accurate or precise. This presents an excellent teachable moment about AI and geolocation limitations. Desktop browsers often provide approximate locations based on IP addresses or network information rather than GPS, resulting in less accurate coordinates. The AI model might then identify nearby cities or metropolitan areas rather than exact locations. Additionally, some coordinates may fall in areas between cities or in regions where the AI model has less training data. In these cases, the model might provide the nearest recognized location, a region name instead of a city, or occasionally even make an incorrect determination. Have students compare results from desktop testing versus mobile testing to understand these differences. This helps them appreciate both the technological limitations (browser geolocation vs. GPS) and AI limitations (training data coverage, geographic granularity). These observations reinforce the importance of critical thinking when working with AI systems and the need to verify important information, especially when precise location data is crucial for application functionality.

Conclusion & Further Reflexion

- **Knowledge Acquired:** In this third step, students have moved from basic data collection to intelligent data processing using AI. They have learned to transform raw coordinates into meaningful location names through language models, experiencing both the strengths and limitations of AI systems. Students have discovered key strengths of language models: their ability to analyze context, leverage diverse training data, generate basic reasoning, and provide creative solutions to problems. They have also encountered limitations such as potential biases, computational demands, and occasional "hallucinations" or inaccuracies. Most importantly, students have developed practical skills in prompt engineering - learning that effective communication with AI requires specificity, clarity, and structure. They have seen how different prompt formulations yield dramatically different results, reinforcing the importance of careful design in AI interactions.
- **Classroom Implementation Insights:** This step can be implemented as a progressive exploration, starting with simple AI interactions before building to more complex implementations. The iterative approach to prompt refinement naturally encourages experimentation and critical thinking.Group di scussions comparing different prompt formulations and their results can be particularly valuable. For assessment, focus on students' reasoning about their prompt design choices rather than just functionality, as this demonstrates deeper understanding of the concepts involved.
- **Key Learning Outcomes:** Through this investigation, students have learned that communication with AI systems requires structured approaches even when using natural language. They have experienced the impact of iterative design by revising prompts to improve responses, and begun developing critical evaluation skills for AI outputs. These skills prepare students not just for the next steps in developing their conversational agent, but for thoughtful engagement with AI technologies in many future contexts.

To deepen students' engagement with the concepts covered in this step, consider exploring these guiding questions through discussion or reflective writing:

1. **Prompt Design Evolution**: How did your initial prompt differ from your final optimized prompt? What specific changes did you make, and how did each change affect the AI's response? If you were designing a prompt to get information about local attractions near the identified city, how would you structure it to ensure you receive useful, specific recommendations?
2. **AI Integration Benefits**: In what ways does adding AI capabilities transform what your Bot Buddy can do compared to traditional programming approaches alone? What types of questions can your assistant now answer that would have been difficult or impossible to program manually? How might this combination of GPS data and AI response change how users interact with location-based services?
3. **Ethical Considerations**: What privacy concerns might arise when an application collects location data and processes it through AI systems? How might AI-powered location services affect different communities or

neighborhoods differently? What responsibilities do developers have when creating systems that provide recommendations about places to visit or avoid in a city?

4. **Technical Limitations and Solutions**: What challenges did you observe when testing your location-to-city conversion? How might your conversational agent handle situations where GPS signals are weak, coordinates are imprecise, or the AI provides incorrect location information? What backup systems or verification methods could make your assistant more reliable in real-world conditions?

Step 4: Delivering Accessible Information to the end-users

Background and description of the problem to be solved in this step: In the previous steps, students have successfully built a solution that can automatically determine user location thanks to voice queries, generating relevant responses using AI. They can reuse this code to create a real conversational agent that can communicate its responses through voice rather than just text. In the current context, an increasing number of people are using voice assistants to interact with devices and obtain information. This includes driving assistance applications, call response systems, personal assistants such as Siri or Google Assistant, and many others. One of the major challenges of these systems is to provide a smooth and natural user experience through high-quality voice synthesis. This final step will focus on integrating text-to-speech capabilities, completing the full interaction loop of the Bot Buddy Urban Assistant.

Learning Objectives: Understand the fundamental principles of voice synthesis, including the techniques and models used to generate artificial speech. Learn how to implement text-to-speech functionality in CodeSkool. Explore the challenges and issues related to the quality of voice synthesis, such as naturalness, intelligibility, adaptability, and personalization. Develop skills for configuring voice parameters to optimize user experience. Create a complete conversational agent that collects, processes, and delivers information through natural speech.

Conceptualisation

In this fourth and final step, students complete their conversational agent by adding voice output capabilities and additional prompts to deliver a real service to the users. This transforms their assistant from a text-based system to a fully voice-interactive application that both listens to users and speaks responses, creating a more natural and accessible experience.

Research Question 1: Voice Synthesis Implementation

How can we effectively convert text-based AI responses into speech output that sounds natural and is easily understood by users?

Hypothesis: By implementing text-to-speech functionality with appropriate configuration of voice parameters, students can create a voice system that effectively communicates AI-generated information with naturalness.

Key Concepts:

- **Text-to-Speech Synthesis**: Technology that converts written text into spoken voice output using algorithms that generate artificial speech.
- **Voice Parameters**: Configurable aspects of synthesized speech including voice selection, speaking rate, pitch, and volume.
- **Speech Naturalness**: Factors that influence how human-like synthetic speech sounds to listeners.

Research Question 2: Creating a Cohesive Voice-Interactive Experience

How can we integrate voice input, AI processing, and voice output to create a consistent, reliable conversational agent that delivers valuable location-based information?

Hypothesis: By designing structured conversation flows and context-aware prompts, students can create a cohesive voice-interactive system that reliably delivers relevant information about urban environments through natural conversation.

Key Concepts:

- **Conversation Design**: Principles for creating logical, intuitive dialogue flows that guide users through interactions.

- **Context Preservation**: Methods for maintaining location awareness and conversation history throughout interactions.
- **Information Delivery**: Techniques for presenting complex information through voice in accessible ways.

By addressing these questions, students will create a fully accessible conversational agent that completes the input-processing-output cycle through natural voice interaction. This final component transforms their project from a technical demonstration into a practical, user-centered application with real-world utility.

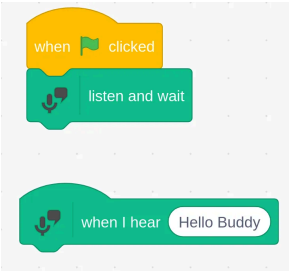
Students Investigation

In this investigation phase, students will implement voice output capabilities and create a conversational flow for their Bot Buddy Urban Assistant, making it a truly interactive voice-based experience.

Creating a Voice-Activated Conversational Agent

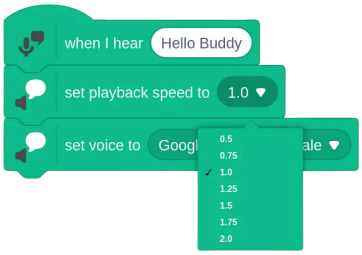
Students will set up the basic structure of their voice-activated assistant by implementing a trigger phrase and welcome message.

To implement this in CodeSkool, students should ensure that the "**Speech to Text**" and "**Text to Speech**" extensions are added to the project. In the "**Speech to Text**" blocks, they can select "**When I hear [value]**" **trigger block** to activate the conversational agent. Hence, everytime the green flag will be clicked, thanks to the "**Listen and Wait**" block added at the begining of the protocol, the conversational agent will react automatically if it hears the trigger sentence (in the example "Hello Buddy".



Once done, students need to configure the **voice parameters** for the conversational agent. Optimizing voice parameters is essential for **natural and effective communication**. The right settings for **speed and voice type** significantly impact how well users understand and engage with the conversational agent. Hence, two blocks should be used to define these parameters available directly in the "Text to Speech" section, in the settings category:

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



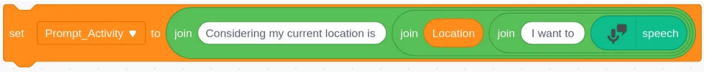
Designing Conversation Flows with Prompts

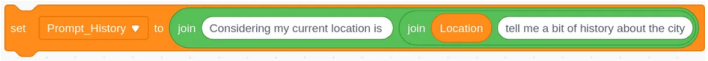
After setting up the voice, students create structured conversation flows to guide user interactions with their assistant. We provide guidance for creating a logical block structure, though this serves as just one example. Students are encouraged to experiment with different organizational approaches, 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 previously done for location. Our agent needs to:

- **Propose activities nearby**: we have created the variable "**Prompt_Activity**" using 3 join blocks as: **"Considering my current location is [Location variable], I want to [speech]"**

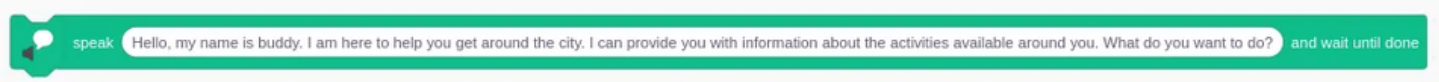
- **Provide historical information about the current city**: we have created the variable "**Prompt_History**" using 2 join blocks as: **"Considering my current location is [Location variable], tell me a bit of history about the city"**



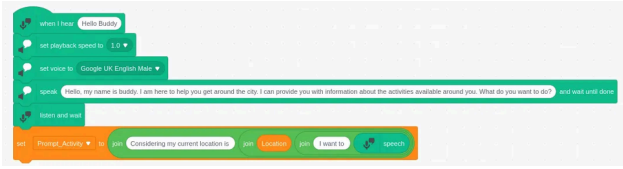


Implement a simple dialogue tree that responds to **speech inputs**:

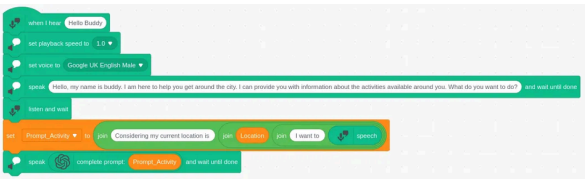
- Once the user says the trigger phrase "Hello Buddy", Buddy responds with a first proposal about nearby activities using the following message (using a text to speech block): **“Hello, my name is buddy. I am here to help you get around the city. I can provide you with information about the activities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



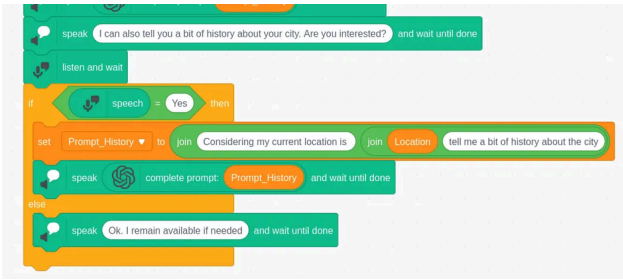
- After Buddy finishes speaking, we use a "listen and wait" block that allows the user to specify their desired activity through the "**set Prompt_Activity**" block.



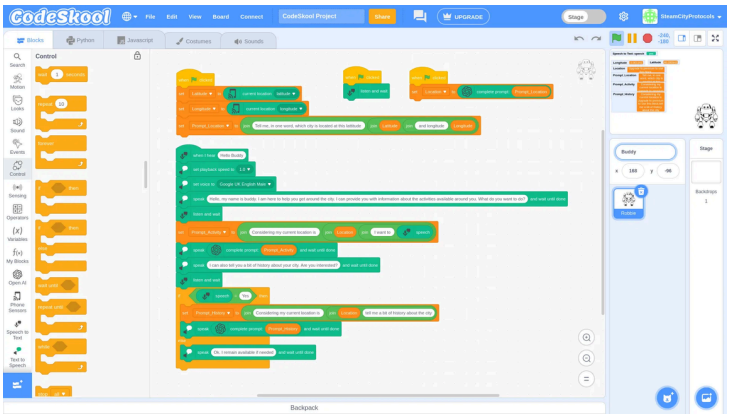
- Based on the user's response, such as "**eat pizza**", Buddy uses a **combination of "speak" and Open AI "complete prompt" blocks** to provide a detailed answer to the request.



- After this discussion concludes, Buddy transitions to offering information about the city's history. The interaction continues with Buddy asking "**I can also tell you a bit of history about your city. Are you interested?**".



- Unlike the activity interaction which used an **open-ended question**, here we implement a **closed-ended question** requiring a **"if [ ] then" control block**. The user can only respond with yes or no, triggering different responses from the bot.
  - If the user says **yes**, Buddy presents the city history.
  - If the user says **no**, Buddy politely ends the conversation: "**Ok. I remain available if needed**"



Here aside is our complete example that builds upon all the programming activities completed in previous steps available at this link: https://share.codeskool.cc/cqZu

This investigation demonstrates that creating a basic conversational agent is relatively straightforward with the right tools and approach. However, the key challenge lies in **designing a logical and engaging conversation flow** that provides value to users. While our example uses a **simple, linear structure with predetermined responses**, this represents just one possible implementation approach.

Students can exercise their creativity by developing **more sophisticated interaction patterns** that:

- **Handle multiple user input variations through expanded conditional logic**
- **Implement branching conversation paths based on user preferences**
- **Create more natural dialogue flows with follow-up questions**

- **Include error handling for unexpected user responses**

By experimenting with different conversation structures and expanded if-then logic, students can create increasingly sophisticated and responsive agents while learning valuable lessons about user interaction design and conversation flow management.

Conclusion & Further Reflexion

- **Knowledge Acquired:** In this fourth and final step of the Bot Buddy Adventure, students have transformed their project into a fully voice-interactive conversational agent. They have implemented both speech recognition for input and text-to-speech for output, creating a seamless voice experience that enhances accessibility and usability. Students have learned about voice synthesis technology and how parameters like speaking rate and voice selection affect the user experience. They have gained practical experience implementing trigger phrases, conversational flows, and context-aware responses that guide users through interactions with their assistant. By creating structured conversation prompts and response paths, students have developed an understanding of conversation design principles that go beyond simple command-response interactions. They have seen how offering choices and providing contextual information creates a more natural and helpful assistant. The integration of text-to-speech synthesis into their conversational agent has given students the opportunity to explore different knowledge domains:
  - **Text-to-Speech Technology**: Understanding the fundamental principles of converting written text into spoken language and how different parameters affect the output quality.
  - **Natural Language Processing**: Exploring how computers can understand and respond to human language in conversational contexts.
  - **Accessibility Design**: Learning how voice interfaces make technology more accessible to diverse users and how design choices can enhance or hinder accessibility.
- **Classroom Implementation Insights:** This step works best when students have multiple opportunities to test and refine their implementations. The quality of voice interaction often improves significantly with several iterations of testing and adjustment. Peer testing is particularly valuable, as different voices, accents, and speaking styles can reveal strengths and weaknesses in speech recognition that might not be apparent when tested by a single user. Similarly, having different listeners evaluate the speech output can provide diverse perspectives on voice quality and understandability. For assessment, consider having students demonstrate their conversational agents to the class, showing both the technical implementation and the user experience. Discussions about design choices and the reasoning behind specific implementations help students articulate their learning and share insights with peers.
- **Key Learning Outcomes:** Through this final investigation, students have developed several important competencies. They understand how to implement voice-activated triggers and conversational flows that guide users through interactions. They have gained experience configuring voice parameters to optimize the listening experience and designing conversation structures that feel natural and helpful. Students have also developed an appreciation for the iterative nature of designing voice interfaces, experiencing how testing and refinement are essential to creating effective voice experiences. They understand that small details in implementation—from voice speed to conversational phrasing—can significantly impact how users perceive and interact with voice-based systems. These skills and insights prepare students to think critically about the voice technologies they encounter in everyday life and to design more accessible, user-centered technologies in the future.

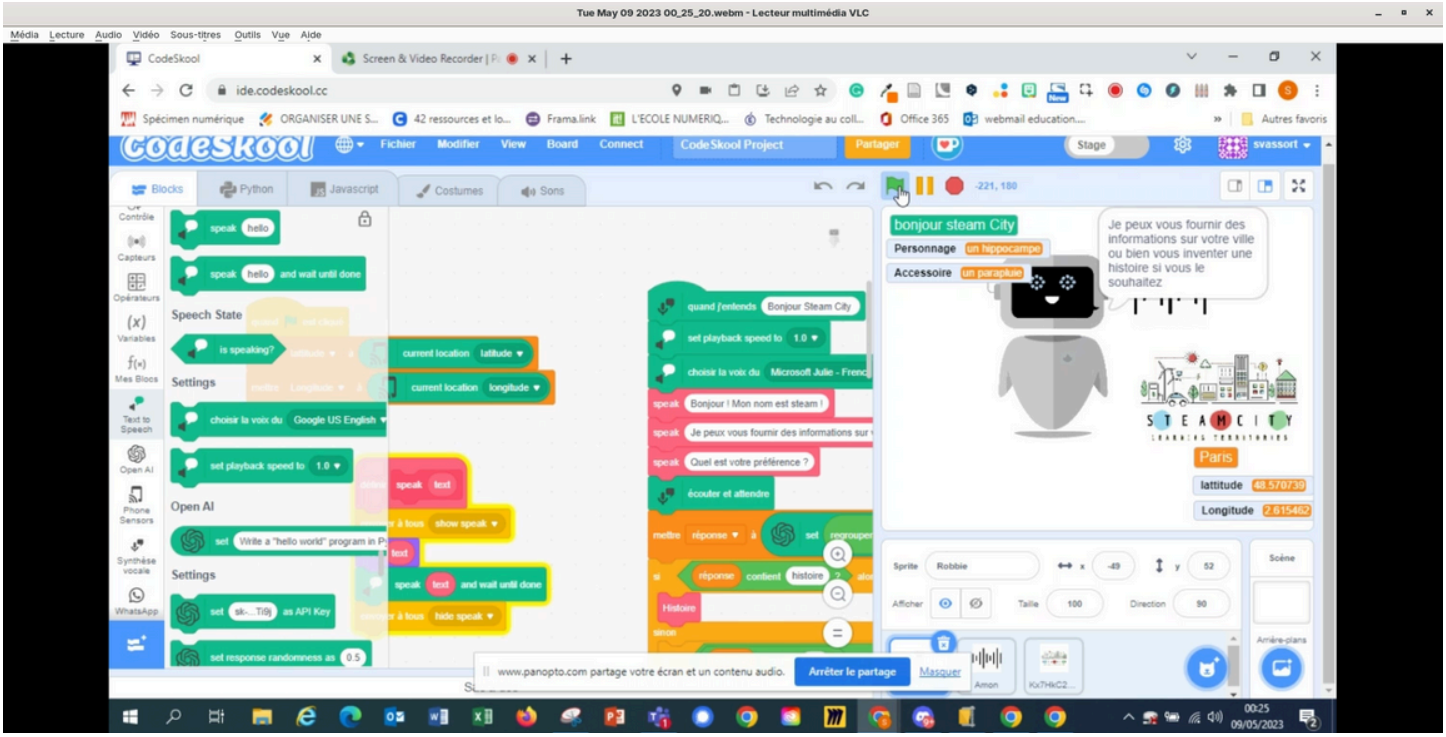
To deepen students' engagement with the concepts covered in this step, consider exploring these questions through discussion or reflective writing:

1. How does a voice-based interface change the way users interact with technology compared to traditional screen-based interfaces?
2. What challenges might arise when designing conversation flows, and how can these be overcome?
3. How might voice assistants be further personalized to meet the needs of different users?
4. What ethical considerations arise when creating systems that use increasingly natural-sounding voices?

Test another version of the program

Here is another version of the program based on the same blocks, adding functions and capabilities to tell stories using accessories and multiple variables. To open it, go to ide.codeskool.cc and select “open project from link”. You can use the **sharing 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



Real-World Inspirations

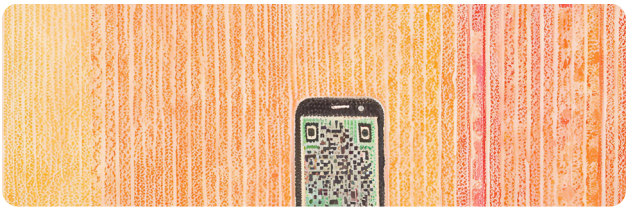
Voice-based AI assistants and accessibility tools are helping change how people interact with digital services. While still evolving, these technologies show promise in supporting people with disabilities and addressing various challenges in society. Here are some examples of how voice-enabled technologies are being used to help make solutions more inclusive and accessible.

1. Be My Eyes - Visual Assistance Enhanced with AI

Be My Eyes began as an application connecting blind and low-vision users with sighted volunteers through video calls for visual assistance. In 2023, they integrated "Be My AI," an AI assistant powered by GPT-4 that can describe images and provide immediate visual assistance without waiting for a human volunteer. Users can simply take a photo and the AI describes what is in the image, reads text, identifies products, or provides navigation guidance.

This initiative demonstrates how voice-based AI can significantly enhance independence for visually impaired individuals by providing immediate, accurate information about their surroundings. The combination of voice interaction, image recognition, and natural language generation creates an accessible interface that works with, rather than instead of, human assistance.

https://www.bemyeyes.com



2. NaviLens - Color Code Navigation System

NaviLens is an innovative system developed in Spain that uses colorful QR-like codes combined with a smartphone application to help visually impaired people navigate independently in urban environments. Unlike traditional QR codes, NaviLens markers can be detected from distances up to 12 meters away and at angles up to 160 degrees, allowing users to scan their surroundings by simply panning their smartphone.

What makes NaviLens particularly notable is its voice-based interface that reads aloud information about detected markers, providing location context, directional guidance, and detailed information about the surroundings. The system has been implemented in public transportation networks in Barcelona, Madrid, and other cities across Europe, helping visually impaired travelers navigate complex transit systems independently.

https://www.navilens.com