Three Levels of Programming with the Petoi Bittle Robot Dog

Overview

In this workshop, students will learn three distinct levels of programming through controlling the Petoi Bittle Robot Dog. We'll start at the lowest level of programming – **Serial Protocol**, then move to **Python Coding**, and finally explore the future of programming through **Prompt Engineering** with AI.

Each level introduces a progressively more abstracted form of controlling the robot dog, giving students a deeper understanding of how computers function, from low-level command execution to high-level AI-driven tasks.

Key Objectives

By the end of this workshop, students will:

1. **Explore Three Levels of Programming**: Understand programming from **low-level commands** (Serial Protocol) to **high-level languages** (Python), and finally, **AI-driven prompt engineering**.

2. **Develop Hands-On Coding Skills**: Write code in Python to control the Petoi Bittle X Robot Dog, learning the essentials of **AI-assisted coding** and **natural language commands**.

3. **Experience the Fusion of Hardware and Software**: See how **software programming** translates into physical actions, gaining insight into how **hardware** and **software** integrate in robotics.

4. **Build an AI Perspective**: Understand the growing role of **Generative AI** in programming and explore how **AI prompt engineering** allows high-level commands to control machines.

5. **Gain Practical Robotics Knowledge**: Learn foundational robotics concepts like **serial communication**, **sensor integration**, and **autonomous movement**, which are essential skills for future work in **hardware-software interaction**.

Materials

- The Petoi Bittle X Robot Dog. Ideally 1 per 5-6 students.
- Laptops/Desktops. 1 laptop per 5-6 students.
- The Serial Protocol documentation for the robot dog from Petoi found at https://docs.petoi.com/apis/serial-protocol. These can be printed or provided via a slide/QR code. A nicely formatted version of the commands is can be found here
- The Visual Studio Code IDE installed on the computers.
- Python installed on computers.
- The Python packages `openai`, `dotenv`, and `pyserial` installed via `pip`, which comes preinstalled with Python as documented [here]. The command below can be used from a terminal to install the relevant packages.
    - `pip install openai python-dotenv pyserial`

- The `PetoiRobot`, `SerialCommunication`, and `ardSerial` modules from Petoi, found at https://github.com/PetoiCamp/Petoi_MindPlusLib/tree/main/python/libraries.
- An OpenAI API key listed in a `.ENV` file in the same directory as the code files for the workshop. You can get one at https://platform.openai.com/api-keys. Create a file called `.ENV` in the code directory on the computer where you're running the workshop. The file should contain 1 line with the contents `OPENAI_API_KEY=YOUR_API_KEY_HERE` where you would replace `YOUR_API_KEY_HERE` with your own key.
- Optional, but a projector for demonstrating concepts, results, and an accompanying PowerPoint.

Level 1: Serial Protocol

The lowest (kind of, not really, but kind of) level of programming.

Objective

Introduce students to low-level programming by sending commands directly to the Petoi Bittle Robot Dog using Serial Protocol. This level aims to give students a hands-on experience of controlling a computer through individual, direct commands, simulating what it's like to operate at the core level of computing. While they're not actually coding in binary (1s and 0s), Serial Protocol allows them to experience what "sending commands to the computer one by one" feels like. This approach provides an understanding of foundational programming concepts and how computers operate at a low level.

Introduction

- Begin by explaining that **Serial Protocol** is one of the earliest methods for direct communication with a machine, allowing specific instructions to be sent in a straightforward manner.
- Show the Petoi Serial Protocol documentation (found here, formatted version here), which outlines the commands that the robot dog can understand and execute.
- Demonstrate a simple command to make the robot dog perform an action, such as moving or sitting.
- Emphasize that these commands operate near the lowest level of programming – close to binary – the true "language" of computers. Every other programming method builds on top of this fundamental layer, adding levels of abstraction that make programming easier but mask the direct, foundational interactions that Serial Protocol provides.

In this workshop, Serial Protocol is the closest we'll come to "talking" to the computer at its core level, making it an ideal entry point for understanding the basics of how computers interpret and respond to instructions.

Hands-On Activity

For details on sending serial protocol commands to the Petoi Bittle using Arduino, see https://docs.petoi.com/arduino-ide/serial-monitor. For details on sending serial protocol commands using Python (such as via a terminal in VS Code), see https://docs.petoi.com/apis/python-api.

1. **Demo**: At the front of the room, demo sending a command via Serial Protocol to the robot dog, such as making it sit down or stand up.
2. **Task**: Now, let the students explore. Provide each group with copies of the Serial Protocol documentation and ask them to control the robot dog themselves. A part of this exercise is getting them familiar with reading and understanding technical documentation, which is a fundamental part of

programming. For an extra challenge, do not provide students with the command required for the action described, let them find it in the documentation. With the robot dog, they should perform the following actions:
  - Make the dog sit down. Command `ksit`.
  - Make the dog stand up. Command `kbalance`.
  - Make the dog hug. Command `khg`.
  - Make the dog hi-five. Command `kfiv`.
  - Make the dog dig. Command `kdg`.
  - Make the dog kick. Command `kkc`.
  - Make the dog do push ups. Command `kpu`.
  - Make the dog do push ups with 1 hand. Command `kpu1`.
  - Make the dog cheer. Command `kchr`.
  - Make the dog jump. Command `kjmp`.
  - Make the dog moon walk. Command `kmw`.
  - Make the dog box. Command `kbx`.
  - Make the dog handstand. Command `khds`.
  - Make the dog rest. Command `krest`.

Code you can run in VS Code to activate a terminal where Serial Protocol commands can be typed in can be found at the link below. It is suggested this file is placed on the machine the students are using and run from VS Code, allowing them to type in their commands in the terminal window.

Note: copy the entire `code` folder under which the file linked below sits to the computer where you'll be running the workshop. There are reference files within it that each level uses for robot dog control.

Level 1 | Serial Protocol Code

Talking Points

Share these points while students are working through the Serial Protocol activity to deepen their understanding of low-level programming:

- Explain that **this is programming at its most fundamental level** – the way computers were programmed before the advent of human-readable languages. This hands-on activity allows students to understand how early programmers interacted directly with hardware.
- Discuss the **role of mathematical and logical precision** required at this level. Working with low-level commands demands exact values and sequences, underscoring the importance of math and logic in foundational programming.
- Highlight how certain industries, like **NASA, SpaceX, NVIDIA, AMD, Intel**, and fields such as **quantum computing, operating system development (Windows, Linux), and hardware engineering**, still depend on low-level programming. This level of control is critical in applications that require precise hardware management and performance. It's like driving a manual transmission in a high-performance car: every action requires direct input and understanding.
- Emphasize that while most modern programming abstracts away these details, **understanding this level is essential** for truly grasping how computers process instructions and manage tasks. It offers invaluable insight into how software and hardware work together at the core level.

Discussion

- Ask students to reflect on how manual this process feels. Was it easy or hard to understand the commands?
- Introduce the idea that there are easier ways to program, but the **foundation** of all computing lies here.

Level 2: High-Level Programming Languages

This is where we're mostly hanging out today in the world of software engineering.

Objective

Transition from Serial Protocol commands to writing code in Python, a **high-level programming language** that is more readable and accessible. Students will perform the same actions they did in Level 1 but will use Python to simplify and structure the commands, demonstrating how high-level languages streamline programming tasks.

Introduction

- Start by explaining **modern programming** and how high-level languages like Python make coding easier and more intuitive by providing human-readable syntax.
- Introduce Microsoft's Visual Studio Code (VS Code) as an Integrated Development Environment (IDE) widely used in professional software development. Explain how IDEs like VS Code offer tools and features for writing, testing, and debugging code, specifically with Python in this context.
- Mention that many software engineers today use IDEs for writing code across different applications and devices.
- Explain the **compilation and interpretation process** – how even though Python is a high-level language, it must still be translated into a form the computer can understand, similar to how serial protocol commands are interpreted by the robot.
    - **Python's Process**: When Python code runs, it's first converted into **bytecode**, an intermediate form between source code and machine code. This bytecode is then processed by the **Python Virtual Machine (PVM)**, which ultimately instructs the computer on what to do.
    - **Examples of Interpreted Languages**:
        - **Python**: Translated into bytecode, then interpreted by the PVM at runtime.
        - **JavaScript**: Directly interpreted in environments like web browsers or Node.js, where code is executed line by line.
        - **Ruby**: Interpreted in runtime environments commonly used in web development.
        - **PHP**: Primarily interpreted for server-side scripting, often used in web applications.
    - **Hybrid Languages** (both compiled and interpreted):
        - **Java**: Compiled into **bytecode**, which is then either interpreted or **Just-in-Time (JIT) compiled** by the **Java Virtual Machine (JVM)**, allowing for both flexibility and speed.
        - **C#**: Compiled into **Intermediate Language (IL)**, which is then interpreted by the **Common Language Runtime (CLR)**, offering similar flexibility as Java.
    - **Difference between Hybrid Languages and Fully Compiled Languages**: Hybrid languages offer a blend of interpreted flexibility and compiled performance by converting code into an intermediate format (like bytecode) rather than directly into machine code (as with fully compiled languages like **C** or **C++**). They rely on interpreters or **JIT compilers** to execute bytecode at runtime, balancing cross-platform compatibility with optimized performance.

This level demonstrates how Python, as a high-level language, abstracts the low-level complexity, making it accessible to more programmers and enabling rapid development while still providing the computational power needed to control devices like the Petoi Bittle Robot Dog.

Hands-On Activity

1. **Task**: Ask students to write Python code to make the robot dog:
   - `Make the dog sit down then wait 1 second.`
   - `Make the dog stand up then wait 1 second.`
   - `Make the dog hug then wait 1 second.`
   - `Make the dog hi-five then wait 1 second.`
   - `Make the dog dig then wait 1 second.`
   - `Make the dog kick then wait 1 second.`
   - `Make the dog do push ups then wait 3 seconds.`
   - `Make the dog do push ups with 1 hand then wait 3 seconds.`
   - `Make the dog cheer then wait 1 second.`
   - `Make the dog jump then wait 3 seconds.`
   - `Make the dog moon walk then wait 1 second.`
   - `Make the dog box then wait 1 second.`
   - `Make the dog handstand then wait 3 seconds.`
   - `Make the dog rest then wait 1 second.`
2. **Highlight the Power of High-Level Programming Languages**: Once the basic commands are done, encourage them to extend the task:
   - Make the commands **loop** a fixed number of times.
   - Add some **data structures** such as a list of commands over which `sendSkillStr` is called, as opposed to writing the function each time it needs to be invoked.

The starter code to provide the students can be found at the link below.

Note: copy the entire `code` folder under which the file linked below sits to the computer where you'll be running the workshop. There are reference files within it that each level uses for robot dog control.

Level 2 | High-Level Programming Languages Code.

Talking Points

Introduce these while the students are coding.

- Emphasize that this is where most **software development** happens today. In high level languages like Python.
- Mention that coding in languages like Python allows us to **add more complexity** - loops, variables, conditions - to what we did in Serial Protocol.
- Discuss how engineers at companies like **Microsoft, Google, Apple, Meta, and Amazon** use these kinds of languages daily to build apps, websites, and software. This is the skill set in-demand today, and as we'll explore in the next activity, remains relevant for the future with AI.

Discussion

- Ask students to compare this experience with Level 1. Which was easier? What are the benefits of using a high-level language like Python?
- Guide them to see how **Python abstracts** many of the complexities we dealt with at the lower level, making programming faster and more efficient.

Level 3: AI Prompt Engineering

Writing natural language and letting AI figure out what you mean to generate code. Programming through natural language.

Objective

Introduce students to **Prompt Engineering**, where natural language commands are used to control machines via AI. This is the most advanced level of programming (as it relates to this workshop, not as a general statement), where interactions become more intuitive and accessible. Students will experience how AI interprets human language to perform tasks, representing the future of programming.

Introduction

- Explain that **Prompt Engineering** is a rapidly growing area where people use **natural language** to interact directly with AI systems. Rather than writing specific code, we can use plain language to communicate with the machine, and the AI interprets these commands to carry out tasks.
- Provide a real-world analogy: If you ask a friend to get you a glass of water, you don't need to break down each step - they intuitively understand what you want. AI works similarly now. By interpreting context and intention, it "understands" what you mean without needing explicit programming instructions for each step.
- Emphasize that **AI does not inherently know what you want**; its "understanding" is rooted in complex programming created by people. This is what makes prompt engineering unique—it leverages AI's advanced interpretation capabilities, but **human knowledge and programming remain essential** in guiding and shaping what the AI can do. More simply put, drive home that people **still need to understand how to code** to make use of AI at this level. This workshop is an example, where the AI portion required a software engineer to write code facilitating interaction between the robot dog and the AI system (large language model).

Hands-On Activity

1. **Demo**: Show the students how to write prompts (natural language inputs into the terminal) that, through AI translation into code, control the robot dog. For example, "Make the robot walk forward."
2. **Task**: Let students create their own prompts:
   - Ask them to prompt the robot dog to perform various actions like walking, jumping, or sitting. Prompts should be delivered in the format `Make the robot dog do...` or `Make it do..` or some variations of that.
   - Have them complete the same set of tasks they issued via serial protocol and Python programming, but now using natural language. Namely, the following.
     - `Make the dog sit down.`
     - `Make the dog stand up.`
     - `Make the dog hug.`

        - `Make the dog hi-five.`
        - `Make the dog dig.`
        - `Make the dog kick.`
        - `Make the dog do push ups.`
        - `Make the dog do push ups with 1 hand.`
        - `Make the dog cheer.`
        - `Make the dog jump.`
        - `Make the dog moon walk.`
        - `Make the dog box.`
        - `Make the dog handstand.`
        - `Make the dog rest.`
    - Additionally, have the students try out the flip features of the robot dog.
        - `Make the dog do a front flip`.
        - `Make the dog do a back flip`.
    - Encourage creativity: `Make it dance` or `Make it go crazy`.
    - If a prompt results in no action, explain that the code behind the system hasn't been programmed to understand that command. Emphasize that to make this cool thing work, understanding how to code is essential. Highlight AI-assisted programming, similar to Jarvis used by Tony Stark in Iron Man.
3. **Explain the AI's Role**: After they try out prompts, explain that **AI acts as their assistant**. While it makes things easier, it's only because **we programmed it** to understand those commands.

Provide the code in the file linked below, which translates natural language commands to Python to control the Petoi Bittle Robot Dog. Have students run the program in an IDE (Visual Studio Code). It should result in a live console (terminal) window where they can type natural language commands and see the robot's reactions.

Ensure an OpenAI API key is present in a `.ENV` file in the same directory as the Python script. You can get an API key from OpenAI at https://platform.openai.com/api-keys. The file should contain one line: `OPENAI_API_KEY=YOUR_KEY_HERE` replacing `YOUR_KEY_HERE` with your key.

Note: copy the entire `code` folder under which the file linked below sits to the computer where you'll be running the workshop. There are reference files within it that each level uses for robot dog control.

Level 3 | AI Prompt Engineering Code

Talking Points

Deliver these while the students are experimenting.

- Prompt Engineering allows us to **program without writing code**, but it's built on a deep understanding of how programming and machines work.
- Discuss how **AI is an assistant, not a replacement** for human intelligence.
- Highlight how we are still in control – the human knowledge behind coding is essential to teach AI.
- Make the connection that Serial Protocol is like driving stick shift, High-Level Programming Languages is like driving automatic, and AI-Assisted Programming is like Tesla's driver assistance tech that makes driving way easier, but still requires human oversight.

Discussion

- Ask the students what they think about the future of programming – will AI replace coding, or just make it easier?
- Help them see the big picture: **All three levels** of programming are different ways of controlling the same machine, but each level abstracts complexity and makes the process easier.

Conclusion

Wrap up the workshop by reflecting on the journey:

- **Level 1**: They controlled the robot dog at the lowest level using Serial Protocol.
- **Level 2**: They wrote Python code, showing how much easier modern programming can be.
- **Level 3**: They used natural language and AI, showing the future of programming.

Remind students that while **AI is powerful**, human intelligence and coding skills remain foundational. Encourage them to continue exploring programming, AI, and technology.