**CodeCruxGPT – Report**

**Name of the Bot**

CodeCruxGPT

**Description of the Bot**

CodeCruxGPT is a discord bot designed to answer questions or queries related to programming or general computer science. It mainly focuses on 3 categories: answering fundamental programming concepts, debugging code, and providing practice coding questions to the user. The bot makes use of OpenAI completion API and chat API to analyze and classify user queries and generate responses to their questions. The bot focuses only on programming related questions and does not answer to other questions.

In order to chat with the bot on discord, **you need to mention the bot before typing the question.** The bot also has a ‘.help’ command that provides more information about the bot.

**Invitation Link**

https://discord.com/oauth2/authorize?client\_id=1308260405360984194&permissions=274877908992&integration\_type=0&scope=bot

**API Calls**

1. Completion API

This API call is used in the classify\_utterance() function which is used to classify the user query using Prompt Engineering techniques into categories like:

* fundamental\_questions
* debug\_code
* questions\_request
* other\_topic

I used the model “gpt-3.5-turbo-instruct” to classify the input. The model generates the response on which category the input belongs to. Other parameter settings are: max\_tokens=20 and temperature=0.5. The output is then used to generate appropriate response for the user.

1. Chat API

This API call is used in the query\_response() function to generates an appropriate response for user based on the classification provided by the Completion API call. The model used here is the “gpt-4o” model with max\_token limit of 500 and temperature of 0.7. The model takes in the previous 5 conversation history as the context for dialog management.

**Prompt Engineering**

1. Classify Utterance

The prompt is constructed to classify the input into different categories (programming concepts, debugging, practice questions, or other topics). I have followed the Prompting with Context technique to classify the input. A small description is given for each category to make the bot understand which category the input belongs to. If the bot couldn’t match to any of the first 3 categories, it is classified as other\_topic and is given an appropriate response.

1. Query Response

Prompt for each category is given conditionally:

* For **fundamental questions**, the prompt is given to respond with examples and to give beginner-friendly responses.
* For **debugging code**, the prompts are given to offer a step-by-step instruction on identifying and fixing errors.
* For **question requests**, the model is instructed to create coding exercises with hints if necessary.
* For **other topics**, the model politely informs the user that the query is not programming-related, but the bot is available to help with programming questions. If it’s a greeting message, it is handled appropriately.

These approaches allow the bot to tailor responses that match the query’s context and intent. Here also Prompting with Context technique is used.

**Test Cases**

**Prompt:** “hello”

**Response:** Hello! How can I assist you with your programming or computer science questions today?

**Score:** Pass

**Prompt:** "Explain the concept of recursion in Python"

**Response:** Hello! I'd be happy to explain the concept of recursion in Python. Recursion is a programming technique where a function calls itself in order to solve a problem. Think of it like looking at two mirrors facing each other: you see an infinite reflection that goes on and on. In programming, recursion breaks down complex problems into smaller, more manageable ones. Here's a simple analogy: Imagine you have a set of Russian nesting dolls. To open the smallest doll, you first need to open the larger ones that contain it. Similarly, in recursion, you keep breaking the problem down until you reach the simplest form (the base case), and then you work your way back up to solve the larger problem. Let's look at a basic example in Python: calculating the factorial of a number. The factorial of a number n (written as n!) is the product of all positive integers less than or equal to n. For example, the factorial of 5 (5!) is 5 × 4 × 3 × 2 × 1 = 120. Here's how you can write a recursive function to calculate the factorial:

def factorial(n):

if n == 1: # This is the base case

return 1

else:

return n \* factorial(n - 1) # The function calls itself

# Example usage

print(factorial(5)) # Output: 120

In this code:

* The base case is when n equals 1. At this point, we know that the factorial of 1 is simply 1, so we return 1.
* If n is not 1, the function calls itself with n - 1. This breaks the problem down, getting closer to the base case.
* Each recursive call pushes the problem closer to the simplest form, and once the base case is reached, the function returns up the call stack, multiplying the results to get the final answer.

One important thing to remember with recursion is to always have a base case. Without it, the function would call itself indefinitely, leading to a stack overflow error. I hope this explanation helps! If you have any more questions or need further clarification, feel free to ask.

**Score:** Pass

|  |
| --- |
| **Prompt:** "What are the different data types in Python?" |

|  |
| --- |
|  |

**Response:** Hello! I'd be happy to explain the different data types in Python. Data types are essentially the categories of data that tell Python how to interpret and manipulate the data. Let's go through the most common ones:

1. Integers (int):
   * These are whole numbers without a fractional part, like -3, 0, 7, or 100.
   * Example:

age = 25

1. Floating Point Numbers (float):
   * These are numbers with a decimal point, like 3.14, 0.001, or -98.6.
   * Example:

pi = 3.14

1. Strings (str):
   * These are sequences of characters, used to represent text, like "Hello, world!" or "Python is fun!".
   * Example:

greeting = "Hello, world!"

1. Booleans (bool):
   * These represent truth values and can be either True or False.
   * Example:

is\_sunny = True

1. Lists (list):
   * These are ordered, mutable collections of items, which can be of mixed data types. Items are enclosed in square brackets.
   * Example:

fruits = ["apple", "banana", "cherry"]

1. Tuples (tuple):
   * These are ordered, immutable collections, meaning once they are created, they cannot be changed. Items are enclosed in parentheses.
   * Example:

coordinates = (10.0, 20.0)

1. Dictionaries (dict):
   * These are collections of key-value pairs, where each key is unique. They are unordered and mutable.
   * Example:

student = {"name": "Alice", "age": 21}

1. Sets (set):
   * These are unordered collections of unique items, meaning duplicates are not allowed. They are mutable.
   * Example:

unique\_numbers = {1, 2, 3, 4}

1. NoneType (None):
   * This represents

**Score:** Pass

**Since the responses are too big, I am not adding the exact responses for some of the test cases here**

**Prompt:** "Can you help me debug my Python code?" [Code is given]

**Response:** Step-by-step debugging response is given

**Score:** Pass

**Prompt:** "Tell me about binary trees."

**Response:** Explanation about binary trees is given

**Score:** Pass

**Prompt:** "Tell me a programming joke"

**Response:** A joke is given but did not tell that it is not directly linked to programming or computer science.

**Score:** Partial Pass

**Prompt:** "Give me a Python challenge for practice"

Response: Sure! Here's a Python challenge that will help you practice working with data types and control structures: Challenge: Build a Simple Todo List Application Create a command-line todo list application in Python. The application should allow users to perform the following actions:

1. Add a Task: Add a new task to the todo list with a title and description.
2. View All Tasks: Display all tasks with their titles, descriptions, and completion status.
3. Mark a Task as Completed: Mark a specific task as completed by its index or title.
4. Delete a Task: Remove a task from the list by its index or title.
5. Exit: Quit the application.

Requirements:

* Use a list to store tasks.
* Each task should be a dictionary with keys like title, description, and completed.
* Use functions to encapsulate different functionalities like adding, viewing, marking, and deleting tasks.
* Implement a simple text-based menu to interact with the user.

Hints:

* You can use a while loop to keep the application running until the user decides to exit.
* Consider using input validation to handle cases where the user might provide invalid input.
* You might want to use list comprehensions or loops to display and manage tasks.

This challenge will help you improve your understanding of lists, dictionaries, loops, and functions in Python. Feel free to ask for further guidance or assistance if you encounter any issues while working on this challenge!

**Score:** Pass

**Prompt:** "Can you help me with my computer science homework?"

**Response:** Provided the solution for the homework

**Score:** Pass

**Prompt:** Suggest me a good movie to watch now.

**Response:** Movie is not suggested and responds that it is not related to programming.

**Score:** Pass

**Prompt:** Give the name of a good game to play now.

**Response:** Name of the game is not given and responds that it is not related to programming.

**Score:** Pass

**Prompt:** "What's the difference between a list and a tuple in Python?"

**Response:** Clear explanation of the differences is given with examples

**Score:** Pass

**Prompt:** "What is your favorite programming language?"

**Response:** As an AI, I don't have personal preferences or feelings, but I can tell you that each programming language has its strengths and is suited for different tasks. For example:

* Python is great for beginners and has a wide range of applications, from web development to data science.
* JavaScript is essential for web development, especially for creating interactive websites.
* Java is widely used in enterprise environments and Android app development.
* C++ is popular in system/software development and game programming due to its performance.
* Ruby is known for its elegant syntax and is often used in web development with the Ruby on Rails framework.

If you're looking to learn a new language, it often depends on what you want to achieve with it. Feel free to ask for more details about any specific language or its applications!

**Score:** Pass

**Prompt:** "Give me a C++ exercise"

**Response:** A C++ exercise question is given.

**Score:** Pass

**Prompt:** Give me the code for a very simple Employee class in Java

1. **Response:** Certainly! Below is a simple implementation of an Employee class in Java. This class includes basic attributes such as name, id, and salary, along with constructors, getter methods, and a method to display the employee's details.
2. public class Employee {
3. // Attributes of the Employee class
4. private String name;
5. private int id;
6. private double salary;
7. // Constructor to initialize Employee objects
8. public Employee(String name, int id, double salary) {
9. this.name = name;
10. this.id = id;
11. this.salary = salary;
12. }
13. // Getter method for name
14. public String getName() {
15. return name;
16. }
17. // Getter method for id
18. public int getId() {
19. return id;
20. }
21. // Getter method for salary
22. public double getSalary() {
23. return salary;
24. }
25. // Method to display employee details
26. public void displayDetails() {
27. System.out.println("Employee Details:");
28. System.out.println("Name: " + name);
29. System.out.println("ID: " + id);
30. System.out.println("Salary: $" + salary);
31. }
32. // Main method for testing the Employee class
33. public static void main(String[] args) {
34. // Creating an Employee object
35. Employee emp = new Employee("Alice", 101, 75000.00);
36. // Displaying the employee's details
37. emp.displayDetails();
38. }

}

**Explanation:**

* 1. **Attributes**: The class has three private attributes: name (String), id (int), and salary (double). These represent basic properties of an employee.
  2. **Constructor**: The constructor Employee(String name, int id, double salary) initializes the attributes when a new Employee object is created.
  3. **Getter Methods**: These methods (getName, getId, getSalary) provide access to the private attributes of the class.
  4. **Display Method**: The displayDetails() method prints the employee's information to the console.
  5. **Main Method**: This is a simple test within the same class to demonstrate creating an Employee object and displaying its details. Feel free to modify or extend this class according to your needs. If you have any questions or need further assistance, please let me know!

**Score:** Pass

**Prompt:** Explain me about the Merge Sort Algorithm

**Response:** Merge Sort Algorithm is explained with examples

**Score:** Pass

**Prompt:** What is Python used for?

1. **Response:** Python is a versatile and widely-used programming language known for its readability and simplicity, which makes it a great choice for both beginners and experienced developers. Here are some of the common uses of Python:
   1. Web Development: Python is used to build server-side web applications. Frameworks like Django and Flask make it easy to develop web applications quickly and efficiently.
   2. Data Science and Analytics: Python is a favorite in the data science community for data analysis and visualization. Libraries like Pandas, NumPy, and Matplotlib are powerful tools for handling data, performing statistical analysis, and creating plots.
   3. Machine Learning and Artificial Intelligence: With libraries such as TensorFlow, Keras, and scikit-learn, Python is extensively used for building machine learning models and AI applications.
   4. Automation and Scripting: Python is excellent for automating repetitive tasks, writing scripts to manage systems, and performing batch operations.
   5. Scientific Computing: Python is used in scientific research for simulations, modeling, and computations. Libraries like SciPy and SymPy are commonly used for these purposes.
   6. Game Development: Python can be used to develop simple games and even complex ones with libraries like Pygame.
   7. Network Programming: Python provides libraries to handle network protocols, allowing developers to build network applications.
   8. Internet of Things (IoT): Python is used in IoT projects for its simplicity and powerful libraries that can interact with sensors and other hardware.
   9. Desktop Applications: Python can be used to create GUI-based desktop applications with libraries like Tkinter and PyQt.

Overall, Python's versatility, ease of learning, and a large community with a wealth of libraries make it a powerful tool for a wide range of applications. If you have a specific area you're interested in, I can provide more detailed information or resources related to Python's use in that field!

**Score:** Pass

**Prompt:** Can you provide me 1 theoretical and 1 coding question for my Java midterm

**Response:** 1 theoretical and 1 coding question in Java is given

**Score:** Pass

**Prompt:** Who is the father of computers

**Response:** The title "Father of Computers" is often attributed to Charles Babbage. He was an English mathematician, philosopher, inventor, and mechanical engineer who is best known for conceptualizing the first mechanical computer. Babbage designed two machines called the Difference Engine and the Analytical Engine in the 19th century. The Analytical Engine, in particular, had features that are similar to modern computers, such as the use of punched cards for input and a form of memory. Although Babbage was never able to complete a fully functional machine in his lifetime, his designs laid the groundwork for future developments in computing. If you have any more questions about computer history or related topics, feel free to ask!

**Score:** Pass

**Prompt:** Who is known as the godfather of AI

**Response:** The title "godfather of AI" is often attributed to several prominent figures in the field of artificial intelligence, but it is most commonly associated with Geoffrey Hinton, Yann LeCun, and Yoshua Bengio. These three researchers are renowned for their pioneering work in deep learning, a subset of AI that has significantly advanced the capabilities of machine learning systems. Their contributions have been instrumental in the development of neural networks and have paved the way for many modern AI applications. If you have more questions about AI or programming, feel free to ask!

**Score:** Pass

**Prompt:** what are the applications of Stack data structure

1. **Response:** Stacks are a fundamental data structure in computer science, and they are used in various applications due to their simple yet powerful Last In, First Out (LIFO) property. Let's break down some common applications of stacks to help you understand where and how they are used:
   1. Expression Evaluation and Syntax Parsing:
      * Stacks are used in evaluating expressions, particularly those in postfix (Reverse Polish Notation) form. They help in parsing expressions, which is essential in compilers and calculators.
      * For example, when converting infix expressions (like A + B) to postfix expressions (like AB+), stacks are used to handle the operators and parentheses.
   2. Backtracking:
      * Stacks are useful in algorithms that require backtracking, such as solving mazes or puzzles like Sudoku.
      * Imagine you're navigating a maze and you reach a dead end. You can use a stack to "backtrack" to the last decision point and try a different path.
   3. Function Call Management:
      * When a function is called in a program, the current state (like local variables and the point to return to after the function call) is stored on a stack. This is known as the call stack.
      * It allows the program to return to the correct state once the function execution is complete.
   4. Undo Mechanisms:
      * Applications like text editors use stacks to implement the undo feature. Each action (like typing a character) is pushed onto a stack, and when you perform an undo, the last action is popped off the stack.
   5. Browser Navigation:
      * Web browsers use stacks to manage the back and forward navigation. Each visited page is pushed onto a stack, and when you click the back button, the current page is popped off to reveal the previous one.
   6. Memory Management:
      * Stacks are used in memory management systems to keep track of memory allocation and deallocation, especially in languages with manual memory management like C and C++.

These are just a few examples, but they highlight the versatility and importance of stacks in both simple and complex computing tasks. If you have any more questions or need further clarification, feel free to ask!

**Score:** Pass

**Performance**

CodeCruxGPT performs well in responding with helpful and accurate responses to programming queries. It provides clear explanations for fundamental programming concepts, assists in debugging code, and generates practice questions tailored to the user's needs. The bot is responsive and quick in generating output, as long as the user input is clear and within the scope of programming.

If I were to continue improving it, I would focus on:

* Fine-tuning the models with more detailed programming examples to enhance its ability to create varied and challenging questions.
* Adding support for additional languages or more advanced programming topics.
* Also, make the dialog management better to consider long and advanced conversations.

**RESOURCES**

* <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
* <https://cookbook.openai.com/articles/related_resources>
* https://stackoverflow.com/questions/74711107/openai-api-continuing-conversation-in-a-dialogue

**KEVIN BINU THOTTUMKAL**

**Student Id: 000884769**