**Python 3 Artificial Intelligence – Individual Task 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Course** | **ENGR 13300** | **Semester** | *Fall 2025* |
| **Assignment Name** | *Py3 individual task 1* | **Section** | *007* |
| **Student Name** | Mark Sheng | **List collaborators if any (Name, Purdue login)** |  |
| **Student Purdue login** | Sheng65@purdue.edu |

***PART 2: What is ChatGPT and how can you use it to learn to code?***

Read the articles linked below (PART 1) and answer the following questions. Answers should be in paragraph format and fully answer all parts of each question. You may use other sources to inform your answers as long as you provide citations.

[Article 1](https://www.ibm.com/think/topics/large-language-models)

[Article 2](https://www.cnet.com/tech/services-and-software/what-is-chatgpt-everything-you-need-to-know-about-the-ai-chatbot/)

[Article 3](https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2023.123618)

1. In your own words, what is ChatGPT and how does it generate responses to given prompts?

|  |
| --- |
| ChatGPT is a large language model, or a type of neural network that attempts to predict the next word in a sequence of words, given a database of existing words and correlation values that a human has assigned in that database. ChatGPT will then search for the word with the highest correlation to its intended result and select the word that it is most “confident” in to produce complete a sentence. When generating large amounts of text, ChatGPT typically starts from a human-created template for generalized introductions, and feeds that determined phrase into its input, searching for words in its database that correlate with the sentiment of the user’s request the most. However, the inherent non-discrete nature of languages means that LLM’s such as ChatGPT cannot reach 100% computational confidence, as evidenced by latest GPT models utilizing 10,000 NVidia a100 GPU’s and currently upwards of 8.4 yottaflops of total compute power for training, yet still unable to eliminate error and uncertainty in language prediction and generation. |

1. What is a “language model”? How is it different from a search engine?

|  |
| --- |
| A language model is a type of artificial intelligence which attempts to “complete” missing parts of sentences based on the sentiment detected from existing portions of the sentences. This detected sentiment could come from a user input or previous outputs of the model and is then used to predict the best outcome for future words. However, it is difficult to train artificial intelligence to output correct grammar structures, with conventional reinforcement learning introducing too many reward vectors for successful development of grammar structures through emergent behavior. Thus, language models often use human-selected grammar structures as baseline behavior to expand upon, thereby resulting in extreme adherence to grammar structures such as triplets, lists, “not just x, but y” structuring in generated sentences, observed frequently in many commercial models such as ChatGPT, Google Gemini, DeepSeek, Llama, and others. |

1. What types of tasks does ChatGPT perform well at? What are some tasks that it struggles with?

|  |
| --- |
| Not that long ago, OpenAI, the company behind ChatGPT, had been involved in a scandal regarding plagiarism of artwork and text using its web crawlers to obtain and expand its training data on realistic scenarios and human works outside of its database. While this event did call into question the ethical concerns of generative ai, it also highlighted the ability for models with extremely high computing power to scour the web for new data, often obtaining such data moments after being released to the web. This allows language models such as ChatGPT to reinforce its existing database knowledge with information from the internet, an almost unlimited database. However, this almost unlimited information is also its largest drawback. Ask ChatGPT to provide advice on a field that has been littered with misinformation, and the artificial intelligence will struggle to discern between real and false information, often defaulting to the answers and keywords that appear most in its searches, commonly being those such misinforming ideas. With the rise of “vibe coding”, or using strictly AI prompts to create programs, the quality of such generated products can quickly decline, and various unexpected bugs may arise in a program strictly created by AI models. Such a situation had already occurred during the launch of Google’s *Gemini* generative AI model, with the decision to use social media platform *reddit*, infamous for widespread misinformation and politically/racially motivated speech, as a primary source of training data. |

1. What are **three ways** ChatGPT can help you learn python?

|  |
| --- |
| While ChatGPT is not fully accurate in its results and is not a direct replacement for lack of programming knowledge, it becomes a useful tool to programmers looking to create code that is within their expertise. ChatGPT’s powerful computational abilities allow for it to search the internet rapidly for sources of information that the user could then enter and search for reliable information or provide structural helper functions that a programmer could use as a “skeleton” to build their workflow and customized functions.  ChatGPT may not yield consistent results when attempting to generate entire workflows, but when working with *user defined structures*, ChatGPT is able to create simple helper functions that assist in the moderation of datatypes stored in those structures  As is common with most computer processes, ChatGPT excels at simple tasks where short generation tasks frequently interrupted by human regulation prevents inherent errors in its prediction algorithm to accumulate and compound. ChatGPT can also help in training proofreading and error-checking of programs, as practicing spotting potential edge cases and flaws within fully AI-generated code can transition over to enhanced ability to spot those same edge cases and flaws within code written by other humans. |

1. What are **two specific risks** of relying too heavily on ChatGPT for writing python code?

|  |
| --- |
| As mentioned before with the risk of accumulating errors, when human intervention is not frequent enough, ChatGPT’s confidence in what the next term should be begins to decline due to compounded errors, eventually resulting in the model exhibiting behavior that could best be called “taking a guess” to output the next portion of code. This can cause completely unexpected errors in an otherwise predictable workflow.  The second risk of heavy reliance on ChatGPT is somewhat like addictive behavior. Overreliance on specific tools, even non-intelligent ones, can cause people to become inseparable from said tools. With ChatGPT being an intelligent program, overreliance becomes almost trivial to establish in users, with many even opting in for paid “pro” options to unlock stronger computational abilities instead of strengthening their own abilities. |

1. Give an example of a situation where using ChatGPT would be **helpful** to learning python and one example of a situation where ChatGPT would be **harmful** to learning python.

|  |
| --- |
| ChatGPT can be beneficial when practicing working in a group, as its somewhat unpredictable nature allows for a pseudo-simulation of what a group project situation may become. Thus, ChatGPT becomes a training tool to practice working in a group without the pressure of failing, as a user can always restart a prompting session when communicating with ChatGPT, an option not available with human partners.  However, ChatGPT can also hinder development of further programming skills when used outside this training technique and instead used to build code for projects with real stakes and shareholders. This can often lead to the gradual reliance on ChatGPT more and more, until users begin to trust a computer program more than themselves. |

***PART 3: Debugging Python Code: Human vs AI***

An engineer is working on creating a script that can calculate the maximum bending stress on a beam under a point load at the center. The formulas the engineer is working with to calculate the maximum bending stress are listed below. The following python script does not work as intended by the engineer.

Stress Formulas:

Variables:

* F = applied force (N)
* L = span length (mm)
* b = beam width (mm)
* h = beam height (mm)

Broken Python Code:

def beam\_stress(F, L, b, h):

    # Force must be positive

    if F <= 0:

        print("Force must be positive")

        return 0

    # Calculate bending moment

    M = F \* (L / 4)

    # Moment of inertia

    I = b \* (h \*\* 3) / 12

    # Distance to outer fiber

    c = h / 2

    # Stress formula

    I = 0.075

    stress = (M \* c) / I

    # Return Stress value

    return stress

def main():

# Inputs

    Force = input("Input applied force (N): ")

    Length = float(input("Input span length (mm): "))

    Width = float(input("Input beam width (mm): "))

    Height = float(input("Input beam height (mm): "))

    # Check user inputs

    if Force > 0 or Length > 0 or Width > 0 or Height > 0:

        print("Inputs are valid")

    else:

        print("Invalid input values")

    # Calculate Stress Value

    stress\_value = beam\_stress(Force, Length, Width, Height)

    # Convert Stress Value to Pa

    stress\_value = stress\_value / 1000 \* 2

    # Print Output

    print("Beam stress:", stress\_value, "Pa")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. List three possible debugging steps you could take to determine what parts of the code are not functioning properly. Be descriptive in how you would perform each of these steps.

|  |
| --- |
| 1. check indenting: python, unlike c/c++, does not use bracketing “{}” to differentiate portions of code within a statement from those outside  2. add print statements to debug all variables, and check for correct datatypes in these debug statements by calling the correct variable type. (eg. %f for float, %d for integer, %s for string) If type errors are thrown, it means there is something wrong with that variable  3. check all paths in non-void functions end in a return statement, and check ALL POSSIBLE CASES WHERE GLOBAL VARIABLES COULD BE MODIFIED |

2. Identify four bugs (incorrect statements) in the code presented above based on the debugging steps you listed in question 1. For each bug, list the line number, what about the statement is incorrect, and one possible correction. This does not need to be in a paragraph format; you may use bullets or a list.

|  |
| --- |
| Bug 1. Variable I in function beam\_stress is overwritten on line 14 to always equal to 0.075  Bug 2. Input into variable Force on line 21 is not typecasted into a type, may cause downstream errors  Bug 3. stress\_value variable is not pre-defined before it is being used. This may not cause immediate errors but can frequently result in memory allocation bugs causing the variable to be assigned to seemingly random numbers  Bug 4. Algorithm edge case: since or statements are used instead of and statements when checking for input validity, only 1 of 4 inputs needs to be valid to pass the check, instead of all 4 needing to be valid. |

3. How could you have used ChatGPT to help you debug this code or your own code in the future? What type of prompts would be most useful?

|  |
| --- |
| I would be careful of dropping the entire code file into ChatGPT, as this generates more tokens than necessary. Additionally, avoiding overly general prompts such as “please debug my code” will avoid generating more tokens than necessary. Instead, dropping in small, localized samples of code, such as an algorithm of function, would result in more accurate and efficient debugging using AI tools |

4. How would asking ChatGPT for help here potentially hurt your understanding? What steps could you take to prevent these misunderstandings from happening?

|  |
| --- |
| Using overly general prompts will cause ChatGPT to use more compute tokens than necessary to complete its task and may not yield the intended results. Additionally, AI models like ChatGPT are programmed to always possess a positive sentiment, preferring to complete a task poorly as opposed to not attempting that task at all. This can result in the “debugged” program being provided possessing different bugs instead. To avoid this, make prompts specific and use the minimal amount of processing power to understand its sentiment and goal, as well as giving the AI samples of code that are suspected to contain bugs instead of the entire program. |

Using tools like ChatGPT is allowed **only if explicitly permitted**. For this assignment, you are expected to complete the coding task in Part 3 without AI assistance. In future assignments, you may use tools like ChatGPT **only to supplement your understanding**, not to complete the work for you.