

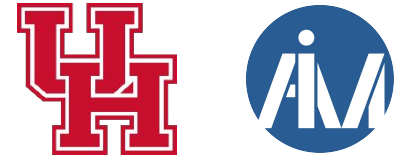


# Integrating chatGPT in a course on numerical optimization

Tian Chen

Salama Assistant Professor of Mechanical Engineering

# Context



*This is my third time teaching this senior elective / graduate course on numerical optimization in the context of mechanical engineering. The enrollment is ~40 students. This is the first instance that I sought to integrate ChatGPT in the course.*

## ***Teaching Innovation Program Grant***

*Title of Proposal: Integration of GPT-4 and 3D printing in Computational Optimization of Structures*



# Content

1. What is numerical optimization?
2. How is the course organized?
3. How is **chatGPT** integrated in the course?
4. What is **chatGPT**?
5. Example of an assignment: **Topology optimization**
6. How do we **evaluate** its effect?
7. What are the effects through **quantitative analysis**?
8. What are the **qualitative observations**?

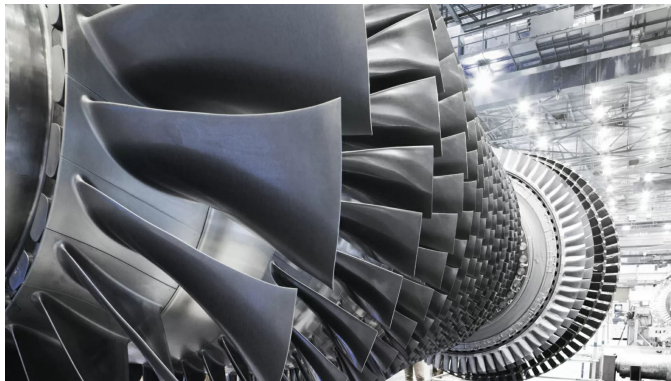
# Optimization



Optimization is a computational process used to design things. The goal is to find the best configuration of a system within given constraints. For example:

**Objective** minimizing weight, maximizing strength, and so on.

**Constraints** material properties, geometric dimensions, compliance with design codes, budget limitations, and so on.





# Course organization

Weekly lecture (1.5h)

Weekly exercise session in a computer room (1.5h)

Python coding-based exercises that can be solved in 1.5 hours

4 assignments throughout the term (20% each)

Python coding-based assignments

1 final exam (20%) – written, theoretical

## Integration of ChatGPT

1. When students ask code related questions, I would use ChatGPT in real time in class to find answer.
2. Exercises are discussed and solved using ChatGPT in class.
3. Students are reminded to use ChatGPT to assist in the assignments at home.

# ChatGPT (as generated from ChatGPT)



**Large Scale Data Training** ChatGPT has been trained on an extensive dataset that includes a significant amount of programming-related content. This dataset includes not just conversational text but also code from various sources like websites, forums (e.g., Stack Overflow), and GitHub repositories.

**Pattern Recognition and Contextual Understanding** The Transformer architecture is exceptionally good at recognizing patterns in data. When it comes to programming languages like Python, which have clear, structured syntax rules, this capability allows ChatGPT to understand and generate syntactically correct code based on the patterns it has learned during training.

**Attention Mechanism:** The attention mechanism enables ChatGPT to focus on different parts of the input when generating the output. This is particularly useful in coding, where the relevance of different parts of the code can vary greatly depending on the current function being implemented.

**Iterative Feedback and Improvement** Through interactions with users and feedback on its code generation capabilities, the response is continually updated and improved.

# Snippets to scripts to programs to code base



**prompt** write a python code to flatten a list

*i.e.*, converting

`[[1, 2], [3, 4]]`  $\square$  `[1, 2, 3, 4]`

Not the most efficient algorithm

a search in Google results in a much

better solution on Stack overflow

To flatten a list in Python, you can use recursion if the list contains nested lists of unknown depth, or a simpler approach if the nesting is only one level deep. Below is an example of a Python function that can flatten a list with any level of nesting:

python

Copy code

```
def flatten_list(nested_list):
    flat_list = []
    # Function to iterate and extract elements
    def flatten(items):
        for item in items:
            if isinstance(item, list): # Check if the element is a list
                flatten(item) # Recursively call to flatten the sublis
            else:
                flat_list.append(item) # If not a list, append directl
    flatten(nested_list)
    return flat_list

# Example usage
nested_list = [1, [2, 3], [4, [5, 6, [7, 8]]], 9]
flat_list = flatten_list(nested_list)
print(flat_list)
```

This function iterates through each element in the input list. If the element is a list itself, it calls itself recursively to flatten it. Otherwise, it appends the element to `flat_list`. This way, it can handle lists of any nesting depth.

# Snippets to **scripts** to programs to code base



**prompt** write a python code to calculate pi using only arithmetic

There are several methods to calculate the value of  $\pi$  using only arithmetic operations. One of the simplest and most famous methods is the Leibniz formula for  $\pi$ , which is an infinite series:

$$\pi = 4 \times \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots \right)$$

This formula converges very slowly, meaning you need a lot of terms to get an accurate value of  $\pi$ , but it's a good example of using only arithmetic to calculate  $\pi$ .

Here's a simple Python code snippet to calculate  $\pi$  using the Leibniz formula:

```
python Copy code

def calculate_pi(num_terms):
    pi_estimate = 0
    for i in range(num_terms):
        term = (-1) ** i / (2 * i + 1)
        pi_estimate += term
    pi_estimate *= 4
    return pi_estimate

# Number of terms to include in the approximation
num_terms = 10000
pi_approx = calculate_pi(num_terms)
print(f"Approximation of pi after {num_terms} terms: {pi_approx}")
```

Gives a good solution, more importantly, **explains this solution**

Google yields identical answer

This code defines a function `calculate_pi` that takes the number of terms to include in the approximation as an argument. It then iterates over each term, adding it to an accumulating sum. Each term of the series is calculated using the formula  $(-1)^n / (2n + 1)$ , where  $n$  is the current term number (starting from 0). After summing the series, the function multiplies the sum by 4 to estimate  $\pi$ .

Keep in mind that to get a very accurate approximation of  $\pi$ , you would need to sum a very large number of terms, which can be computationally expensive.



# Snippets to scripts to **programs** to code base



**prompt** write a python function that solves system of linear equations

$$Ax = b$$

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

```
import numpy as np

def solve_linear_system(A, b):
    """
    Solves a system of linear equations Ax = b for x.
    Parameters
    - A: 2D array representing the coefficient matrix
    - b: 1D array representing the right-hand side vector
    Returns
    - x: 1D array representing the solution vector
    """
    # Check if A is a square matrix
    if A.shape[0] != A.shape[1]:
        raise ValueError("Matrix A is not square")
    # Solve the system using NumPy
    x = np.linalg.solve(A, b)
    return x
```

This function works as follows:

**Forward Elimination:** It transforms the augmented matrix  $[A|b]$  into an upper triangular form. If you're not allowed to use NumPy, you can solve a system of linear equations using Python's built-in capabilities. One approach is to implement the Gaussian Elimination method. This method involves substituting the system's coefficients back into the equations to be solved starting from the bottom row.

**Back Substitution:** This involves substituting the values of the variables found in the previous step back into the equations to find the remaining variables.

Here's a Python function that solves a system of linear equations using NumPy:

```
python
A = [[3, 2, -1], [2, -2, 4], [-1, 0.5, -1]]
b = [1, -2, 0]
x = gaussian_solve(A, b)
print(x)
```

This will print the solution vector  $x$ .

Keep in mind that this function might occur (e.g., when solving a system of linear equations) which is an efficient NumPy function that returns the solution vector  $x$ .

This code snippet would solve the system of equations represented by matrix  $A$  and vector  $b$ , printing the solution vector  $x$ .

```
bash
pip install numpy
```

Then, you can use the following Python function:

```
python
import numpy as np

def solve_linear_system(A, b):
    """
    Solves a system of linear equations Ax = b for x.
    """
```

```
def linSolve(A,b):
    ...
    return x
```

# Snippets to scripts to programs to **code base**



**prompt** program in python a truss optimization code based on linear programming

Gives an example of one simplified structure

and claims

*This example does not include the detailed mechanics calculations needed for a full optimization. Those steps would significantly complicate the script and require a deeper understanding of structural analysis principles.*

# Assignment 4: Topology optimization



## 8 tasks

Task 1 – Derive the element stiffness matrix

Task 2 – Derive the element DOF Matrix

Task 3 – Code a finite element solver

Task 4 – Implement the filter function

Task 5 – Code optimality criteria

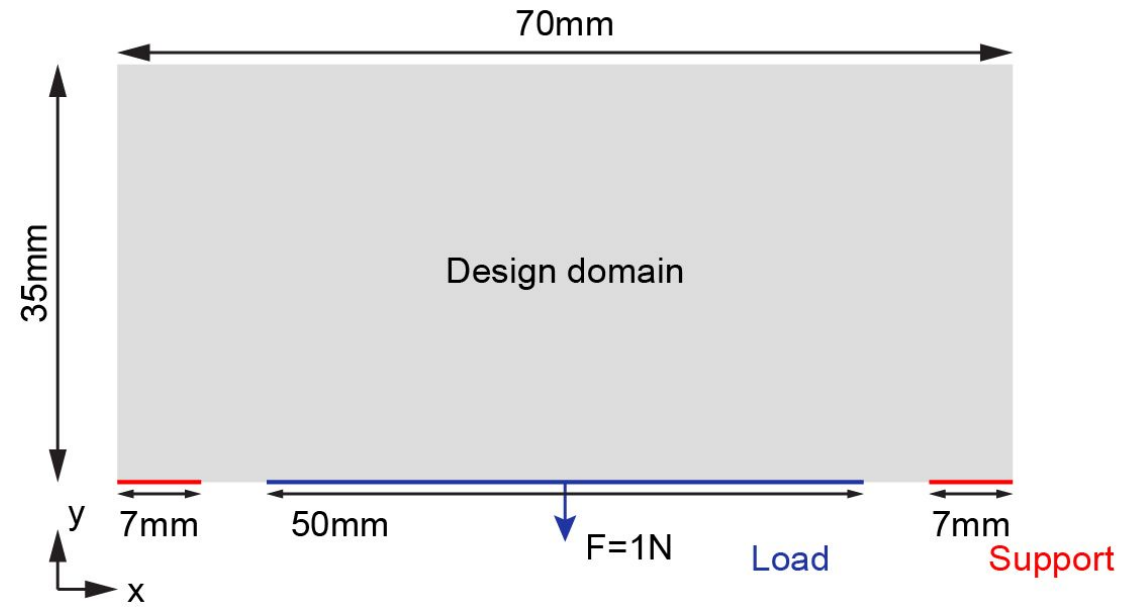
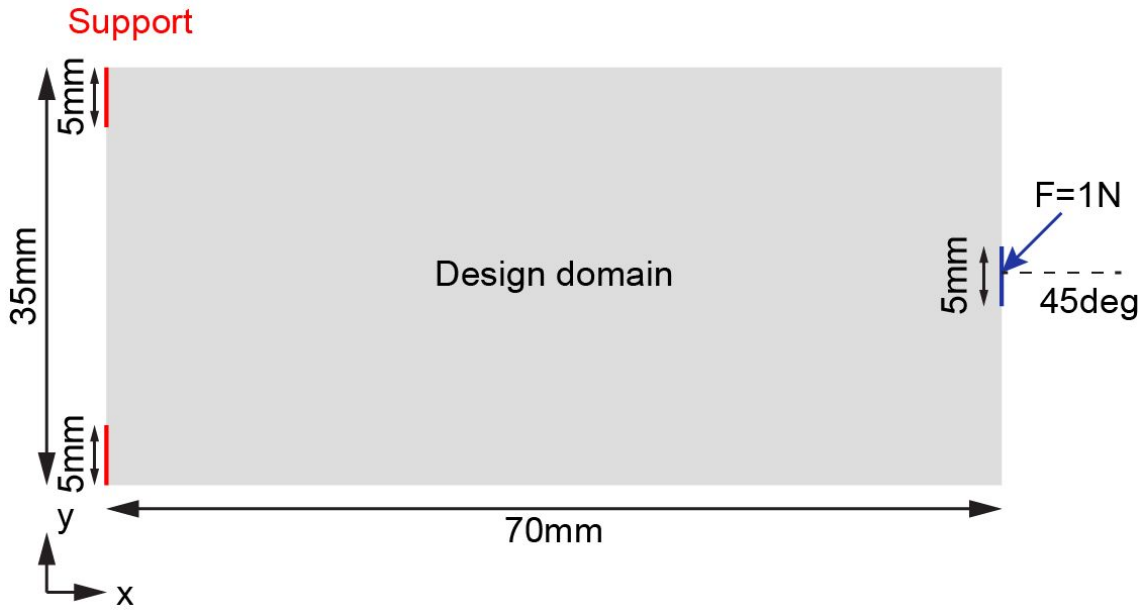
Task 6 – Implement the SIMP algorithm

Task 7 – Design structures

## Optional

Task 8 – 3D print and test them

# Example problems



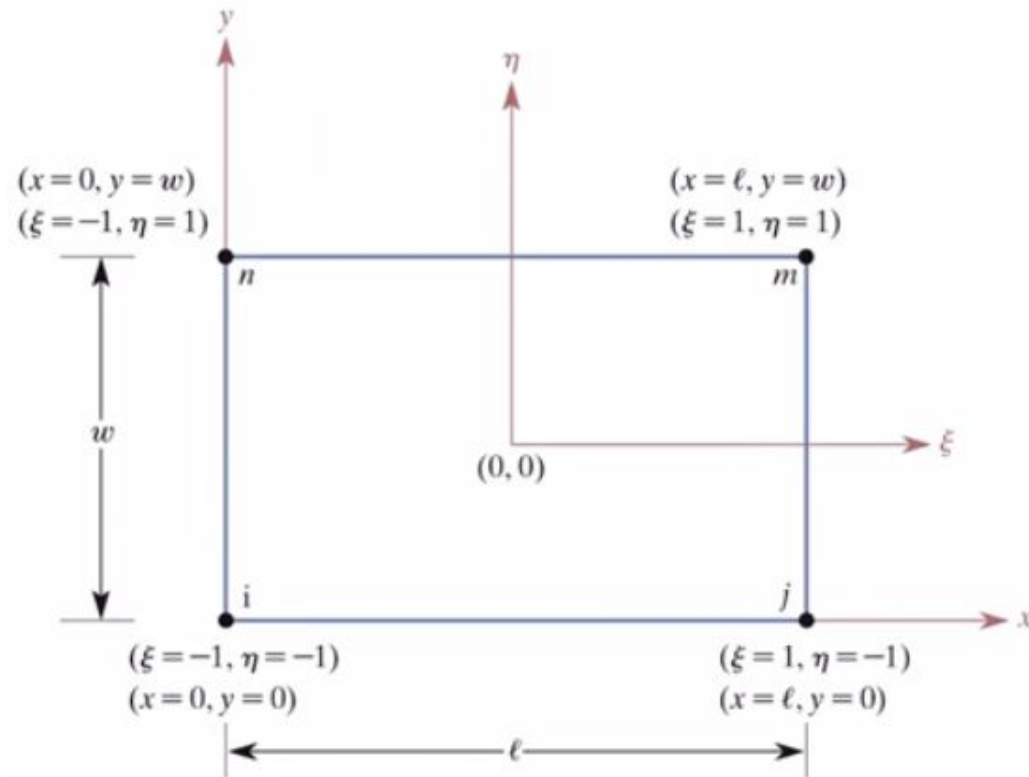
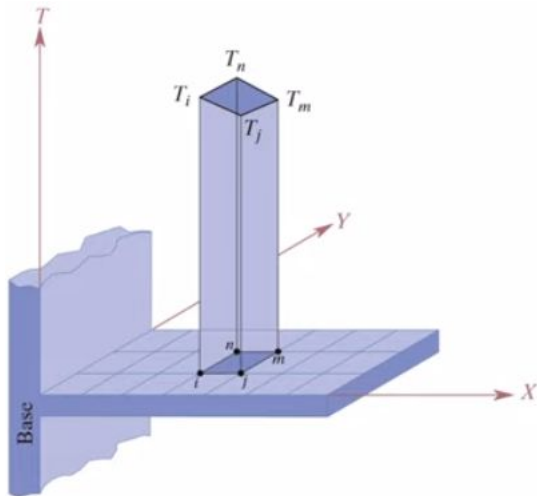
# Task 1: Derive the element stiffness matrix



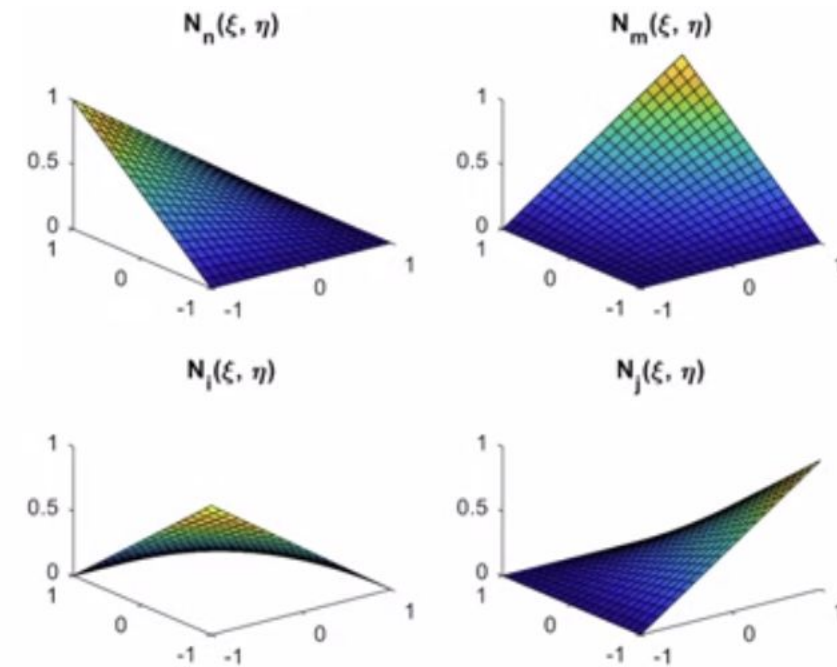
4-noded

$$u = f(x, y)$$

Quadrilateral elements (4 nodes)



# Task 1: Derive the element stiffness matrix



$$u(\xi, \eta) = [N_1(\xi, \eta), N_2(\xi, \eta), N_3(\xi, \eta), N_4(\xi, \eta)] \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}$$

$$v(\xi, \eta) = [N_1(\xi, \eta), N_2(\xi, \eta), N_3(\xi, \eta), N_4(\xi, \eta)] \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix}$$

$$\begin{Bmatrix} u(\xi, \eta) \\ v(\xi, \eta) \end{Bmatrix} = \begin{vmatrix} N_1(\xi, \eta) & 0 & N_2(\xi, \eta) & 0 & N_3(\xi, \eta) & 0 & N_4(\xi, \eta) & 0 \\ 0 & N_1(\xi, \eta) & 0 & N_2(\xi, \eta) & 0 & N_3(\xi, \eta) & 0 & N_4(\xi, \eta) \end{vmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix}$$

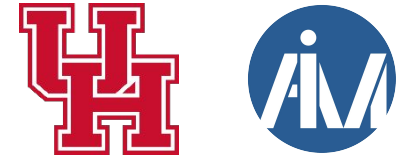
# Task 1: Derive the element stiffness matrix



$$\begin{aligned}
 \text{Displacement field: } \begin{Bmatrix} u(\xi, \eta) \\ v(\xi, \eta) \end{Bmatrix} &= \begin{bmatrix} N_1(\xi, \eta) & 0 & N_2(\xi, \eta) & 0 & N_3(\xi, \eta) & 0 & N_4(\xi, \eta) & 0 \\ 0 & N_1(\xi, \eta) & 0 & N_2(\xi, \eta) & 0 & N_3(\xi, \eta) & 0 & N_4(\xi, \eta) \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} \\
 \begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} &= \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix} \\
 \begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} &= \begin{bmatrix} \frac{\partial N_1(\xi, \eta)}{\partial x} & 0 & \frac{\partial N_2(\xi, \eta)}{\partial x} & 0 & \frac{\partial N_3(\xi, \eta)}{\partial x} & 0 & \frac{\partial N_4(\xi, \eta)}{\partial x} & 0 \\ 0 & \frac{\partial N_1(\xi, \eta)}{\partial y} & 0 & \frac{\partial N_2(\xi, \eta)}{\partial y} & 0 & \frac{\partial N_3(\xi, \eta)}{\partial y} & 0 & \frac{\partial N_4(\xi, \eta)}{\partial y} \\ \frac{\partial N_1(\xi, \eta)}{\partial y} & \frac{\partial N_1(\xi, \eta)}{\partial x} & \frac{\partial N_2(\xi, \eta)}{\partial y} & \frac{\partial N_2(\xi, \eta)}{\partial x} & \frac{\partial N_3(\xi, \eta)}{\partial y} & \frac{\partial N_3(\xi, \eta)}{\partial x} & \frac{\partial N_4(\xi, \eta)}{\partial y} & \frac{\partial N_4(\xi, \eta)}{\partial x} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix}
 \end{aligned}$$

$$\boldsymbol{\epsilon} = \mathbf{B} \mathbf{u}$$

# Task 1: Derive the element stiffness matrix



$$U = \frac{1}{2} \int_V \boldsymbol{\sigma} \boldsymbol{\varepsilon} dV$$

$$U = \frac{1}{2} \int_V \boldsymbol{\varepsilon} \mathbf{C} \boldsymbol{\varepsilon} dV$$

$$U = \frac{1}{2} \int_V \mathbf{u} \mathbf{B}^T \mathbf{C} \mathbf{B} \mathbf{u} dV$$

$$U = \frac{1}{2} \int_V \mathbf{u} \mathbf{K} \mathbf{u} dV$$

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV$$

$\mathbf{C}$  is the elasticity tensor [3x3]

$\mathbf{B}$  is the shape function [3x8]

$\boldsymbol{\sigma}$  is the stress tensor

$\boldsymbol{\varepsilon}$  is the strain tensor

**Plane stress**  $t \ll w, l$

variation in stress across thickness is negligible

$$\mathbf{C} = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1 - \nu}{2} \end{bmatrix}$$

**Plane strain**  $t \gg w, l$

variation in strain across thickness is negligible

$$\mathbf{C} = \frac{E}{(1 - 2\nu)(1 + \nu)} \begin{bmatrix} 1 - \nu & \nu & 0 \\ \nu & 1 - \nu & 0 \\ 0 & 0 & \frac{1 - 2\nu}{2} \end{bmatrix}$$

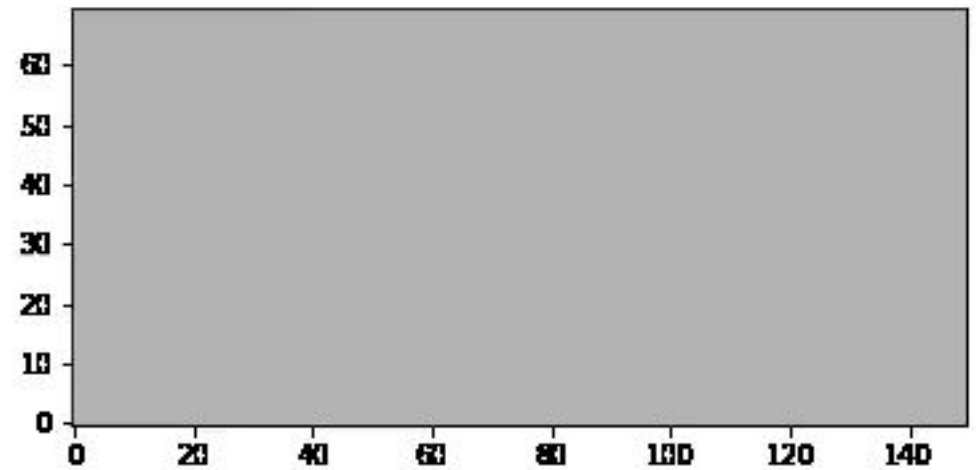
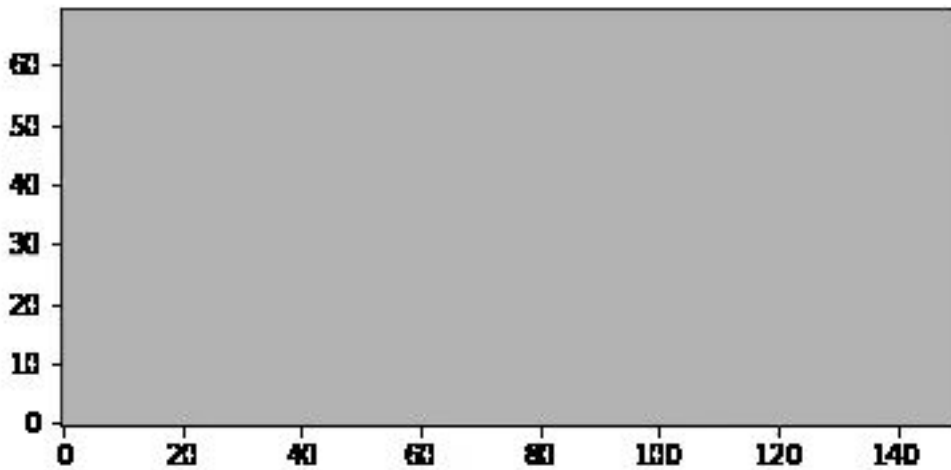
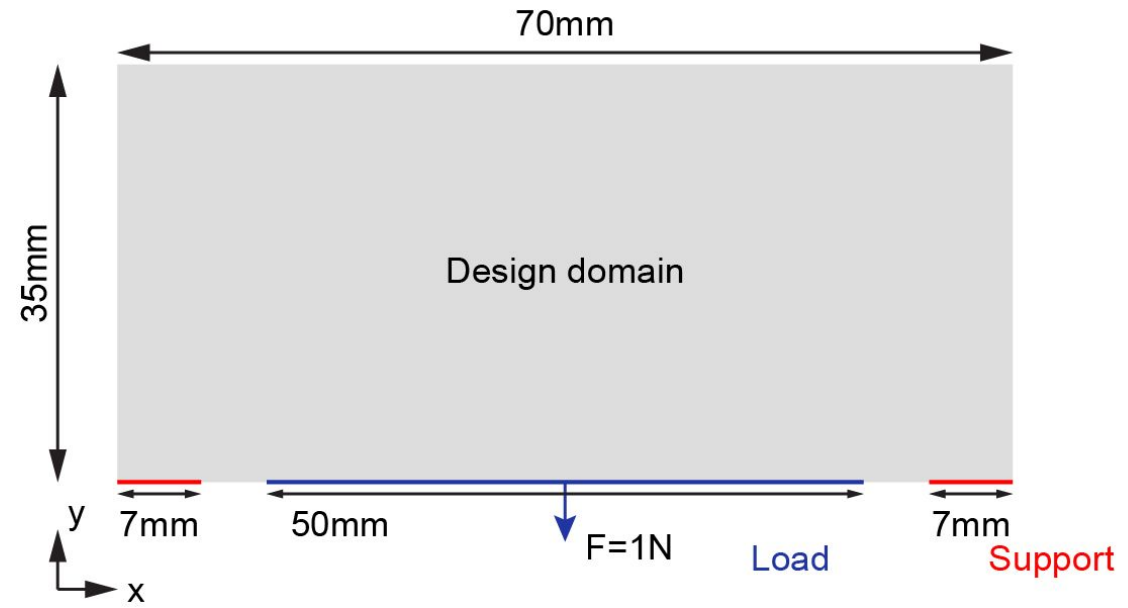
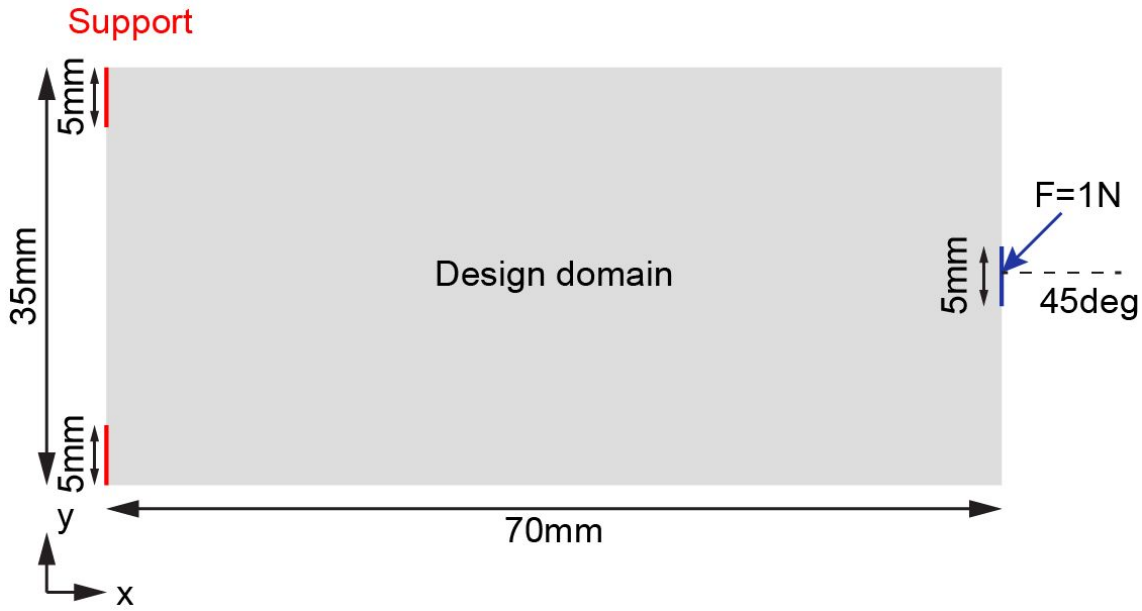


# Task 1: one expected outcome

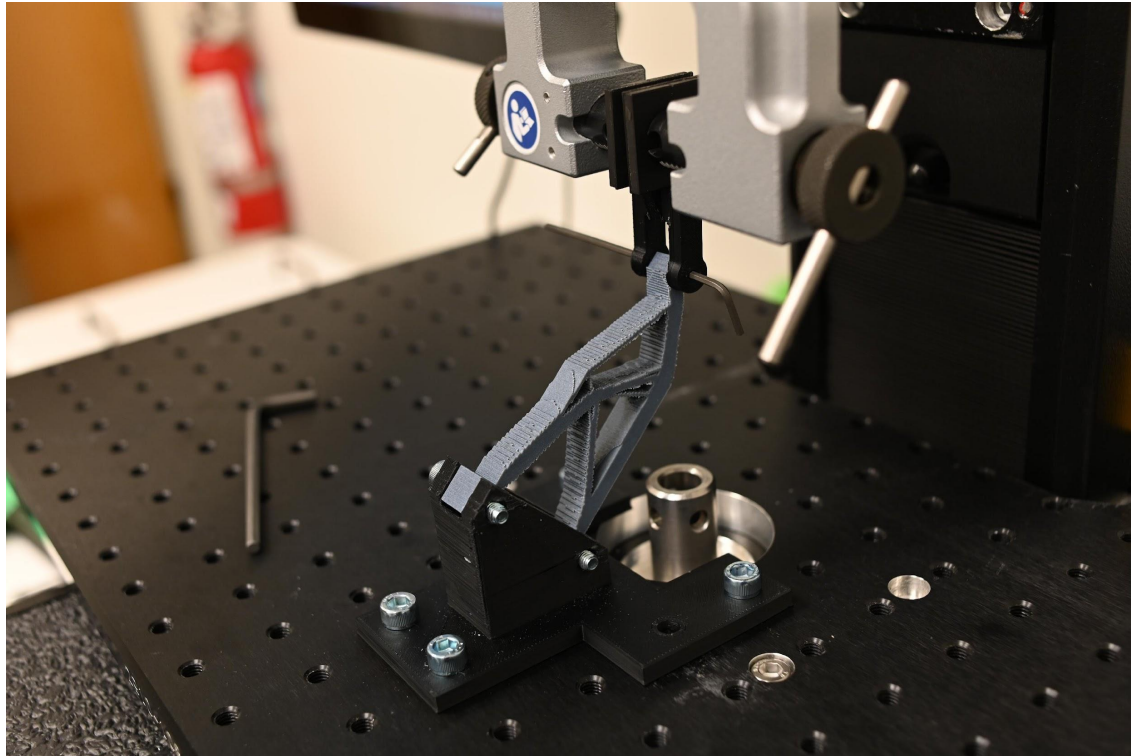


0.49450549	0.17857143	-0.3021978	-0.0137363	-0.2472528	-0.1785714	0.05494505	0.01373626
0.17857143	0.49450549	0.01373626	0.05494505	-0.1785714	-0.2472528	-0.0137363	-0.3021978
-0.3021978	0.01373626	0.49450549	-0.1785714	0.05494505	-0.0137363	-0.2472528	0.17857143
-0.0137363	0.05494505	-0.1785714	0.49450549	0.01373626	-0.3021978	0.17857143	-0.2472528
-0.2472528	-0.1785714	0.05494505	0.01373626	0.49450549	0.17857143	-0.3021978	-0.0137363
-0.1785714	-0.2472528	-0.0137363	-0.3021978	0.17857143	0.49450549	0.01373626	0.05494505
0.055	-0.0137363	-0.2472528	0.17857143	-0.3021978	0.01373626	0.49450549	-0.1785714
0.014	-0.3021978	0.17857143	-0.2472528	-0.0137363	0.05494505	-0.1785714	0.49450549

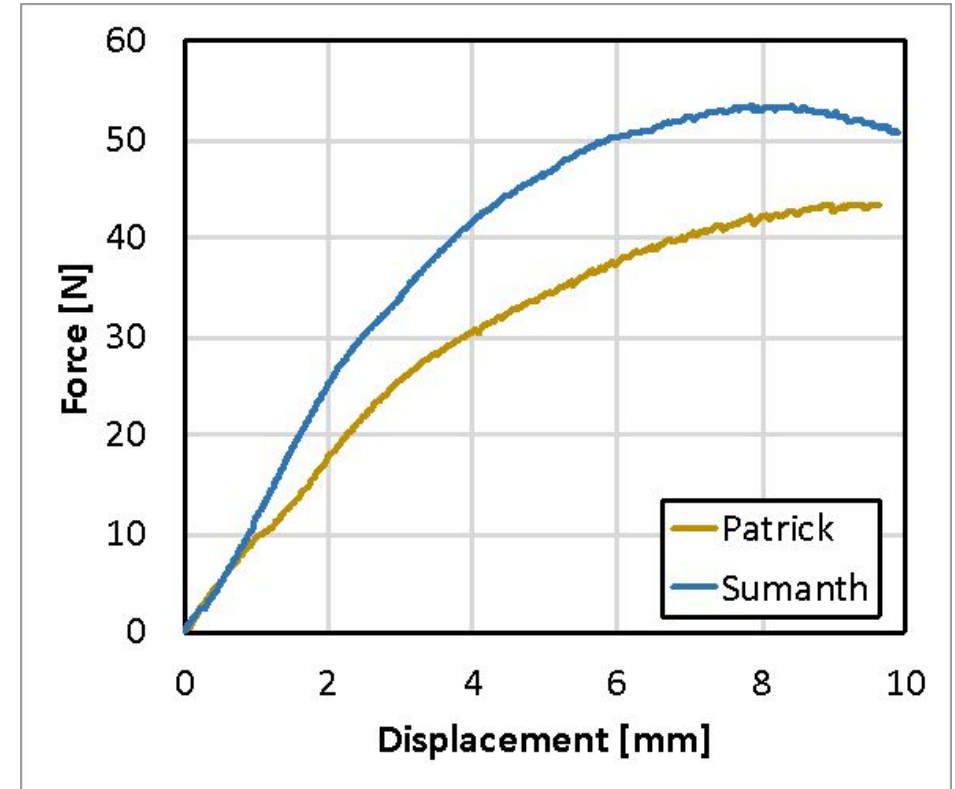
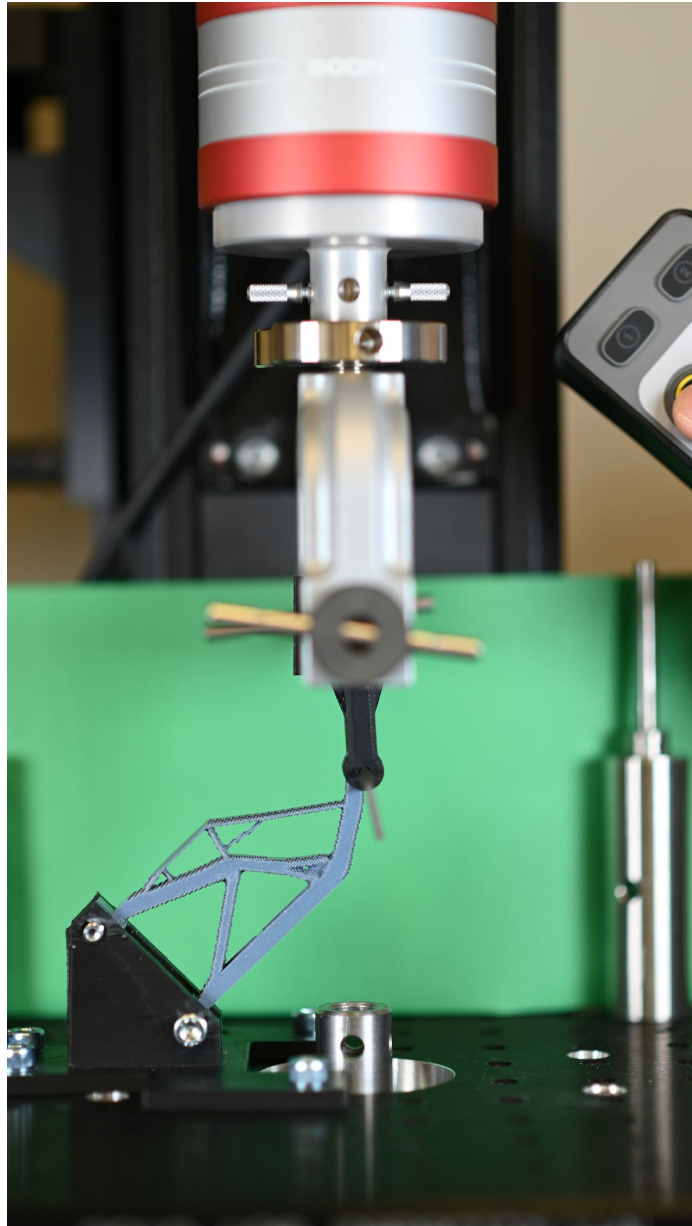
# Student outcomes



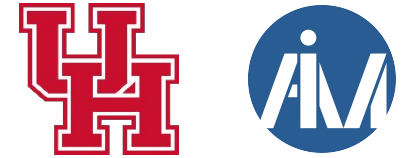
# Student outcomes



# Student outcomes



# Surveys



Two surveys are conducted (independently from course evaluation) to assess the impact of integrating ChatGPT within the course.

Survey #1: set of 12 questions scored from 1 to 5

Survey #2: set of same 12 questions + 4 long form questions



**Dr. Sissy Wong**

**Title:** Associate Professor & Science Education Specialization Lead

**Department:** Curriculum & Instruction

**Program:** Science Education

**Office Number:** 444 FH

**Phone:** 713-743-1690

**Fax:** 713-743-4990

**Email:** [sswong3@cougarnet.uh.edu](mailto:sswong3@cougarnet.uh.edu)

**CV:** [My Vita](#)

# Surveys



## Logistics

- Google form shared via a link
- Survey #1 – Given during lecture #1 and completed prior to lecture #2
- Survey #2 – Given at the conclusion of the course
- 33 students completed both the pre- and post- surveys
- 11 students completed only the pre-survey
- Quantitative results are statistically analyzed using code provided by GPT-4
- Qualitative results are summarized using GPT-4

# Pre- and Post- Survey questions: set 1



1. How would you rate your first-hand experience with any Artificial Intelligence (AI)-based tools? **Awful**  **Superb**
2. How familiar are you with the AI-based tool, Chat Generative Pre-trained Transformer (ChatGPT)? **Not familiar**  **Very familiar**
3. How often do you use ChatGPT? **Never**  **Very frequently**
4. How often do you use ChatGPT for academic purposes (course work, research, etc.)? **Never**  **Very Frequently**
5. How useful do you think ChatGPT is in assisting engineering students to grasp class content? **Not useful --> Very useful**
6. How proficient are you in using ChatGPT for learning purposes? **Not proficient --> Very proficient**

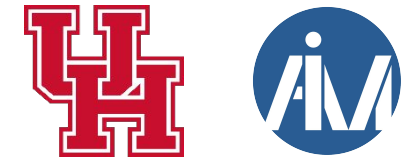
# Pre- and Post- Survey questions: set 2



7. Do you agree with this statement: ChatGPT promotes problem-solving skills.  
**Strongly disagree**  **Strongly agree**
8. Do you agree with this statement: ChatGPT can code functional computer programs.  
**Strongly disagree --> Strongly agree**
9. How would you rate your current understanding of computer programming? **No competence --> High competence**
10. How confident are you in your ability to code a functional computer program? **Not confident --> Very confident**
11. Do you agree with this statement: ChatGPT increases your interest in computer programming. **Strongly disagree --> Strongly agree**
12. Do you agree with this statement: ChatGPT gives you more confidence in your ability to code a functional computer program. **Strongly disagree --> Strongly agree**



# Questionnaire summary



Question	Pre	Post	t	P-value
1 your first-hand experience AI tools	3.33 (1.13)	3.94 (0.75)	2.796	0.867%
2 familiarity with ChatGPT	3.30 (1.17)	3.97 (0.98)	3.319	0.226%
3 frequency of use of ChatGPT	2.77 (1.00)	3.76 (1.00)	3.884	0.0480%
4 frequency of ChatGPT for academic	2.74 (1.14)	3.76 (0.94)	3.550	0.122%
5 usefulness of ChatGPT in learning	3.67 (1.11)	4.27 (0.80)	2.683	1.15%
6 proficiency of ChatGPT for learning	3.05 (1.27)	3.67 (0.99)	2.349	2.52%
7 ChatGPT promotes problem-solving skills	3.47 (1.20)	3.76 (1.06)	1.377	17.8%
8 ChatGPT can code computer programs	3.79 (0.89)	3.76 (0.94)	0.273	78.6%
9 current understanding of programming	2.74 (0.90)	3.39 (0.83)	2.593	1.42%
10 confidence of ability to code	2.56 (1.12)	3.33 (0.99)	2.429	2.10%
11 ChatGPT increases interest in programming	3.77 (1.09)	4.06 (0.97)	1.071	29.2%
12 ChatGPT improve confidence in ability to code	3.91 (1.11)	4.24 (0.90)	2.273	2.99%

# Pre- and Post- Survey questions: set 1



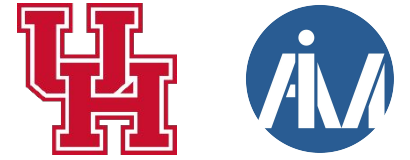
## **Most significant**

Confidence, proficiency, frequency

## **Not significant**

1. ChatGPT can code functional computer programs. (3.79  $\square$  3.76)
2. ChatGPT increases your interest in computer programming. (3.77  $\square$  4.06)
3. ChatGPT promotes problem-solving skills. (3.47  $\square$  3.76)

# Post-survey long-form questions



- What computer programming concepts did you understand better through the use of ChatGPT?
- What were the most effective ways you used ChatGPT to find answers to your queries related to computer programming?
- Were there any difficulties you faced while using ChatGPT for this course? If so, please specify.
- How did your perceptions about ChatGPT change throughout the course?

# Long-form questions



**What computer programming concepts did you understand better through the use of ChatGPT?**

**Optimization Techniques:** Around 15-25% of the respondents referred to optimization, both in general terms and specific to Python. This includes optimizing code for performance and resource usage.

**Understanding and Implementing Algorithms:** About 10-20% of the responses indicated that ChatGPT helped in understanding algorithms, including specific methods like Steepest Descent or Golden Search.

**Debugging and Error Correction:** Approximately 10-20% of users mentioned improvements in debugging skills and error correction, suggesting that ChatGPT was a useful tool for troubleshooting in programming.

**Data Structures and Variables:** Around 10-15% of responses focused on understanding data structures and variables, such as arrays, symbolic variables, and numpy arrays.

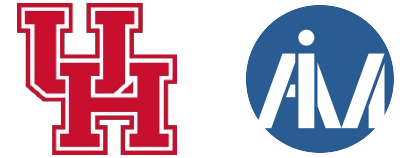
**Loop Constructs and Control Structures:** Loops and other control structures were mentioned in about 10-15% of the responses, highlighting ChatGPT's role in clarifying the use and formation of loops and functions.

**Syntax and Basic Concepts:** Syntax and basic programming concepts were cited in approximately 10-20% of the responses, indicating that ChatGPT was helpful for beginners and those solidifying their foundational knowledge.

**Pseudo Code and Problem Solving:** A smaller portion, about 5-10%, mentioned using ChatGPT to develop pseudo code or to assist in the initial stages of problem-solving.

**Overall Programming Understanding:** A few responses (around 5-10%) suggested a broader impact of ChatGPT in improving overall understanding of programming languages and how they work.

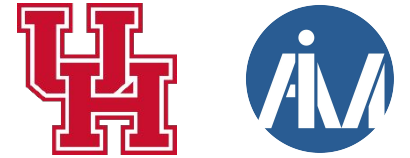
# Long-form questions



**What computer programming concepts did you understand better through the use of ChatGPT?**

1. Optimization Techniques ~15-25%
2. Understanding and Implementing Algorithms ~10-20%
3. Debugging and Error Correction ~10-20%
4. Loop Constructs and Control Structures ~10-15%
5. Syntax and Basic Concepts ~10-20%
6. Pseudo Code and Problem Solving ~5-10%
7. Overall Programming Understanding ~5-10%

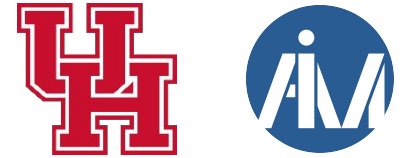
# Long-form questions



**What were the most effective ways you used ChatGPT to find answers to your queries related to computer programming?**

1. Code Debugging and Error Correction ~30-40%
2. Creating and Refining Code Outlines ~20-30%
3. Providing Detailed Context and Information ~15-25%
4. Asking Specific and Precise Questions ~10-20%
5. Understanding and Learning Concepts ~10-15%
6. Dividing Problems into Smaller Parts ~5-10%
7. Query Formulation Skills ~5-10%
8. Syntax and Function Explanation ~5-10%

# Long-form questions



**Were there any difficulties you faced while using ChatGPT for this course? If so, please specify.**

1. Inaccurate or Limited Responses: ~25-35%
2. **Requirement for Detailed Context and Specificity:** ~20-30%
3. Difficulties with Code Formatting and Structure: ~15-20%
4. Server Accessibility and Reliability Issues: ~10-15%
5. Understanding of Complex or Advanced Code: ~10-15%
6. Looping in Problem-Solving: ~5-10%
7. Gap in Course-Specific Knowledge: ~5-10%

# Long-form questions



**How did your perceptions about ChatGPT change throughout the course?**

1. Increased Appreciation and Utility: ~40-50%
2. Recognition of Limitations and Errors: ~20-30%
3. Unchanged Perceptions for Experienced Users: ~10-15%
4. Realization of ChatGPT's Coding Capabilities: ~15-20%
5. ChatGPT as a Complementary Tool: ~10-15%
6. Concerns about AI and Job Impact: ~5-10%
7. Improvements in Interacting with ChatGPT: ~5-10%
8. Expanded Understanding of Applications: ~5-10%

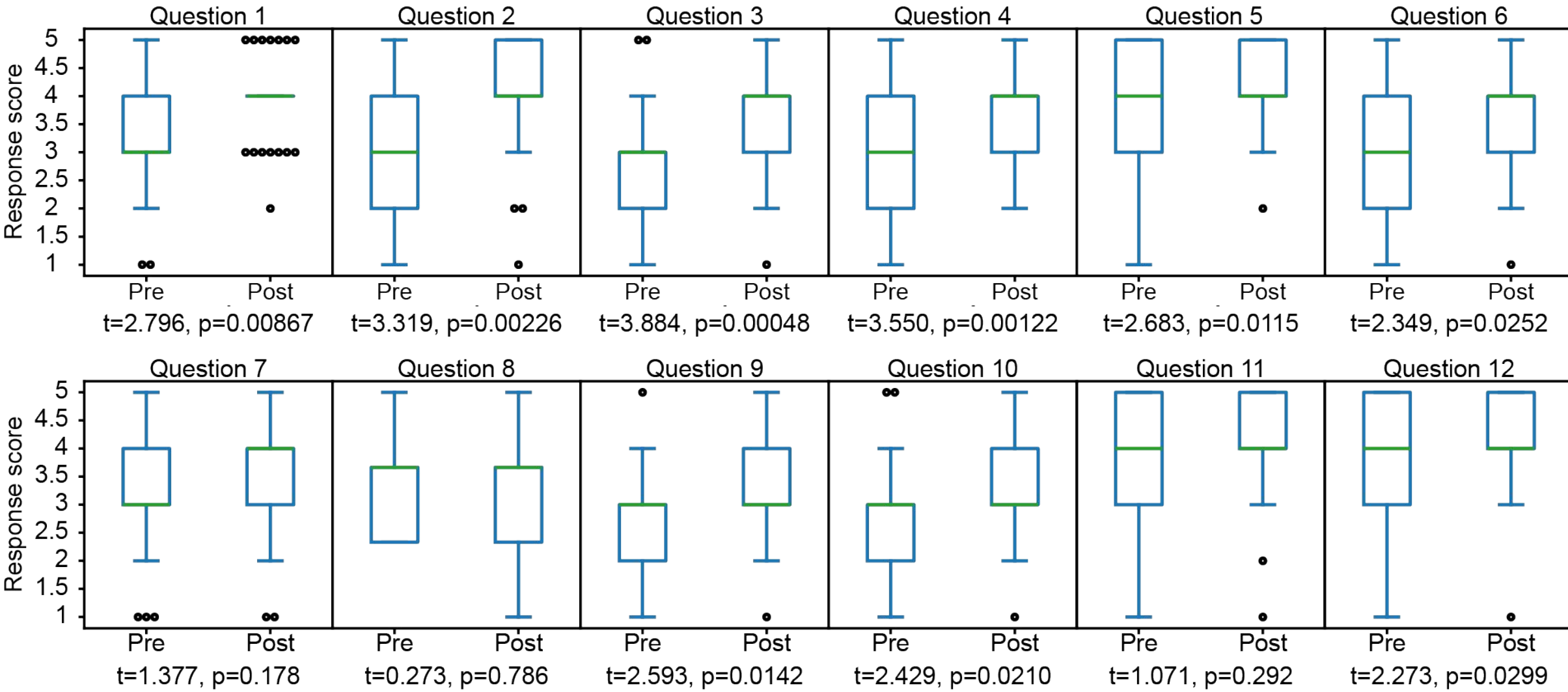


# Conclusion and thank you



- ChatGPT gave students more confidence and ability in their programming skills
- This exercise improved students' proficiency in using ChatGPT
- ChatGPT cannot code complete programs
- Rather, it is useful for debugging and providing outlines

# Questionnaire summary

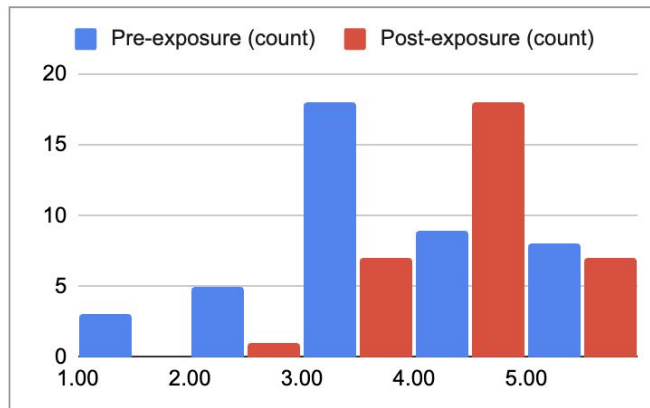


# Detailed results, Q1-4



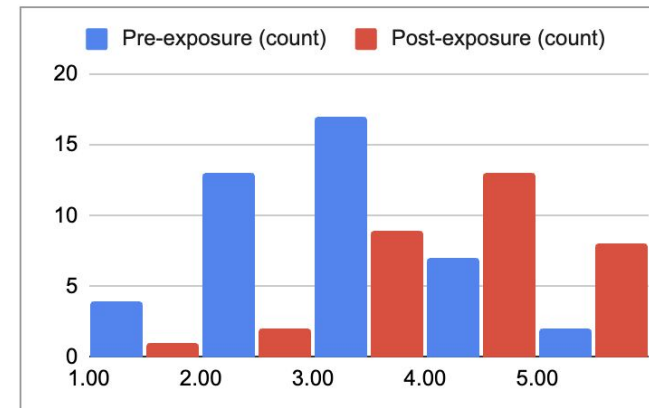
**Question 1**  
**How would you rate your first-hand experience with any Artificial Intelligence (AI)-based tools?**

Pre-exposure Mean 3.33 Std 1.13  
 Post-exposure Mean 3.94 Std 0.75



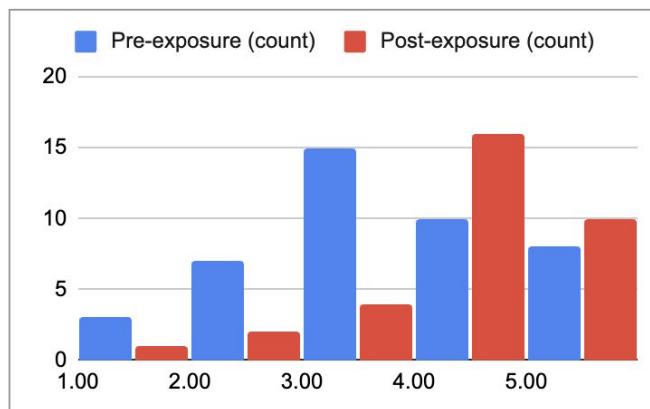
**Question 3**  
**How often do you use ChatGPT?**

Pre-exposure Mean 2.77 Std 1  
 Post-exposure Mean 3.76 Std 1



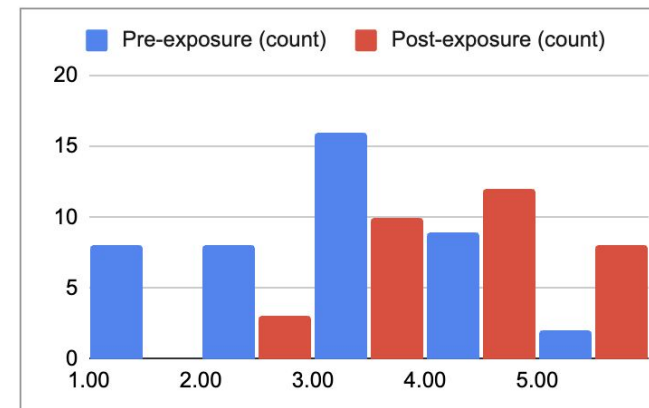
**Question 2**  
**How familiar are you with the AI-based tool, Chat Generative Pre-trained Transformer (ChatGPT)?**

Pre-exposure Mean 3.3 Std 1.17  
 Post-exposure Mean 3.97 Std 0.98



**Question 4**  
**How often do you use ChatGPT for academic purposes (course work, research, etc.)?**

Pre-exposure Mean 2.74 Std 1.14  
 Post-exposure Mean 3.76 Std 0.94

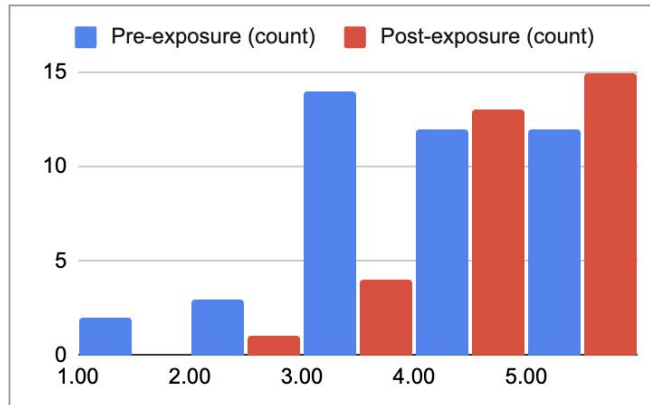


# Detailed results, Q5-8



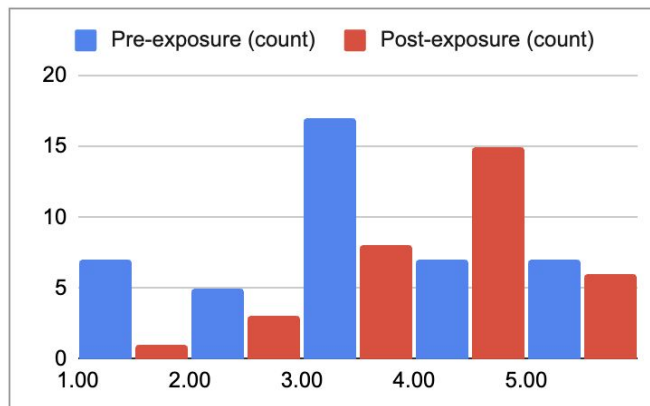
**Question 5**  
**How useful do you think ChatGPT is in assisting engineering students to grasp class content?**

Pre-exposure Mean 3.67 Std 1.11  
 Post-exposure Mean 4.27 Std 0.8



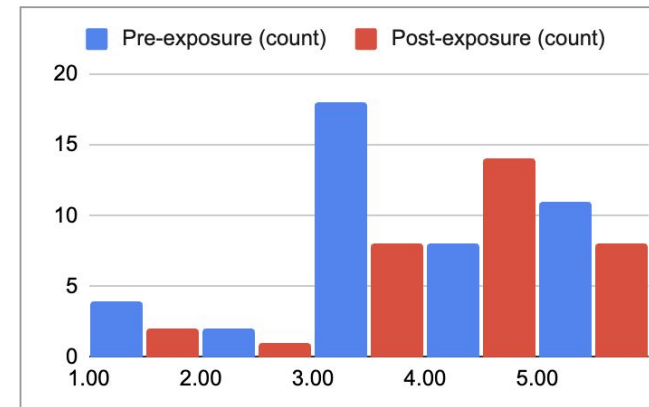
**Question 6**  
**How proficient are you in using ChatGPT for learning purposes?**

Pre-exposure Mean 3.05 Std 1.27  
 Post-exposure Mean 3.67 Std 0.99



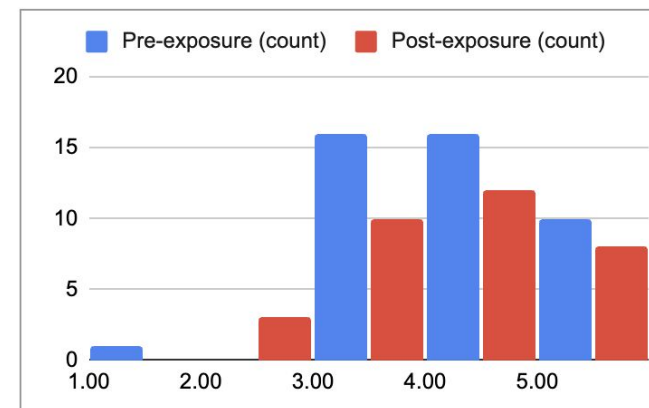
**Question 7**  
**Do you agree with this statement: ChatGPT promotes problem-solving skills.**

Pre-exposure Mean 3.47 Std 1.2  
 Post-exposure Mean 3.76 Std 1.06



**Question 8**  
**Do you agree with this statement: ChatGPT can code functional computer programs.**

Pre-exposure Mean 3.79 Std 0.89  
 Post-exposure Mean 3.76 Std 0.94



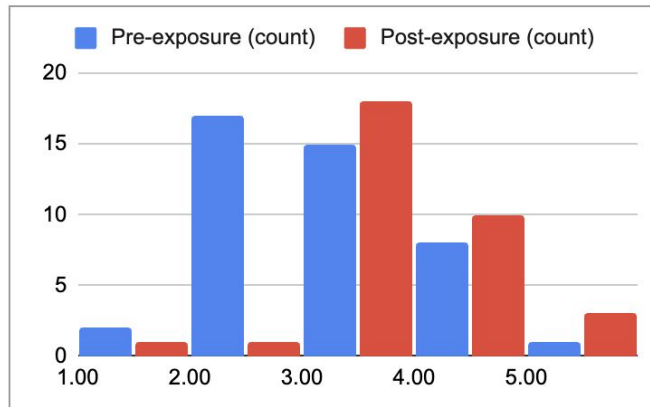
# Detailed results, Q9-12



**Question 9**

**How would you rate your current understanding of computer programming?**

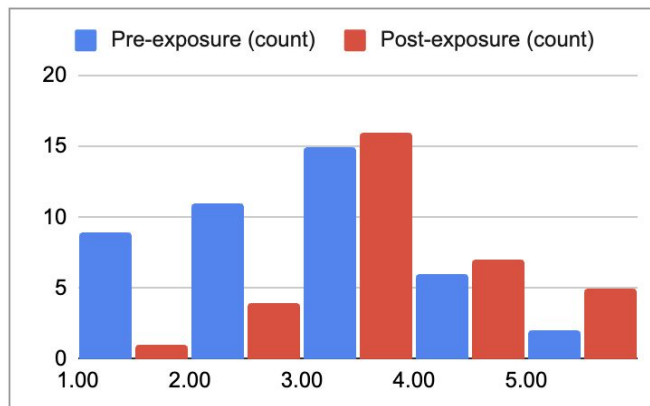
Pre-exposure Mean 2.74 Std 0.9  
 Post-exposure Mean 3.39 Std 0.83



**Question 10**

**How confident are you in your ability to code a functional computer program?**

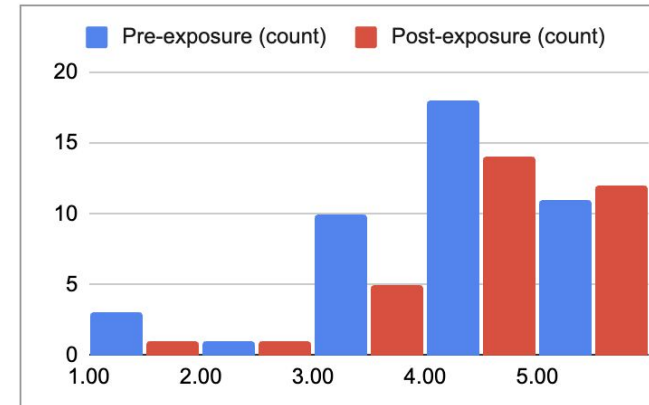
Pre-exposure Mean 2.56 Std 1.12  
 Post-exposure Mean 3.33 Std 0.99



**Question 11**

**Do you agree with this statement: ChatGPT increases your interest in computer programming.**

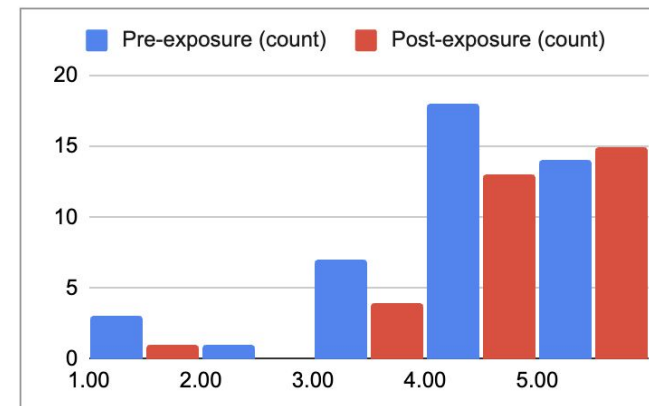
Pre-exposure Mean 3.77 Std 1.09  
 Post-exposure Mean 4.06 Std 0.97



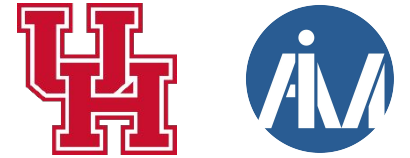
**Question 12**

**Do you agree with this statement: ChatGPT gives you more confidence in your ability to code a functional computer program?**

Pre-exposure Mean 3.91 Std 1.11  
 Post-exposure Mean 4.24 Std 0.9



# Long-form questions



**What were the most effective ways you used ChatGPT to find answers to your queries related to computer programming?**

**Code Debugging and Error Correction:** A large proportion of responses (approximately 30-40%) highlighted using ChatGPT for debugging code and error correction. This involved providing specific code snippets and error messages to ChatGPT for troubleshooting.

**Creating and Refining Code Outlines:** Around 20-30% of the responses indicated using ChatGPT to create initial code outlines and then iteratively refining them to achieve functional code.

**Providing Detailed Context and Information:** About 15-25% of users emphasized the importance of feeding ChatGPT detailed information about the problem, including the context and specific requirements, for effective assistance.

**Asking Specific and Precise Questions:** Approximately 10-20% of the responses suggested that being specific and precise in the queries posed to ChatGPT was crucial for obtaining useful answers.

**Understanding and Learning Concepts:** Around 10-15% of users used ChatGPT to understand broader programming concepts and the context of what they were trying to achieve, indicating its role in educational assistance.

**Dividing Problems into Smaller Parts:** Approximately 5-10% of the responses mentioned breaking down larger problems into smaller, more manageable parts before querying ChatGPT.

**Query Formulation Skills:** A smaller proportion of responses (around 5-10%) highlighted that the process of formulating effective queries itself was an important skill gained from using ChatGPT.

**Syntax and Function Explanation:** About 5-10% of users specifically mentioned using ChatGPT to understand syntax and the usage of specific functions in programming languages like Python.

# Long-form questions



Were there any difficulties you faced while using ChatGPT for this course? If so, please specify.

**Inaccurate or Limited Responses:** A significant number of respondents (approximately 25-35%) mentioned issues with ChatGPT providing inaccurate, limited, or incorrect responses. This included difficulties with the tool providing code that contained errors or was not applicable to the specific context.

**Requirement for Detailed Context and Specificity:** Around 20-30% of the users highlighted the need for providing detailed context and being very specific in prompts to get useful answers from ChatGPT. This suggests a challenge in formulating queries effectively.

**Difficulties with Code Formatting and Structure:** Approximately 15-20% of responses indicated difficulties in getting ChatGPT to provide code in a specific format or structure required for their assignments, suggesting a gap in ChatGPT's ability to tailor code to specific academic requirements.

**Server Accessibility and Reliability Issues:** About 10-15% of users experienced issues related to server reliability and accessibility, such as ChatGPT being down or not accepting new users.

**Understanding of Complex or Advanced Code:** Around 10-15% of the responses pointed to difficulties in using ChatGPT for complex or advanced coding problems, indicating a limitation in its capability to handle more sophisticated programming tasks.

**Looping in Problem-Solving:** A smaller proportion of responses (around 5-10%) mentioned that ChatGPT sometimes got caught in loops, alternating between a limited set of solutions, especially when trying to debug code.

**Gap in Course-Specific Knowledge:** A few responses (approximately 5-10%) indicated that there were discrepancies between the course content and ChatGPT's model, leading to challenges in bridging the knowledge gap.

# Long-form questions



**How did your perceptions about ChatGPT change throughout the course?**

**Increased Appreciation and Utility:** A large number of respondents (around 40-50%) expressed an increased appreciation for ChatGPT's usefulness, particularly in coding and understanding how to extract specific answers.

**Recognition of Limitations and Errors:** Approximately 20-30% of the users acknowledged that while ChatGPT is helpful, it also has limitations and can produce errors, indicating a more nuanced understanding of the tool.

**Unchanged Perceptions for Experienced Users:** About 10-15% of the responses indicated that their perceptions of ChatGPT did not change significantly, likely because they were already familiar with its capabilities.

**Realization of ChatGPT's Coding Capabilities:** Around 15-20% of the respondents, who were initially unaware of ChatGPT's coding capabilities, reported a positive change in perception regarding its use in programming.

**ChatGPT as a Complementary Tool:** Approximately 10-15% of users came to view ChatGPT as a complementary tool, useful for assistance but not a standalone solution.

**Concerns about AI and Job Impact:** A smaller proportion (about 5-10%) expressed concerns about AI's impact on jobs and the need for careful usage, reflecting a cautious outlook towards technological advancement.

**Improvements in Interacting with ChatGPT:** Around 5-10% of users mentioned that they learned how to interact more effectively with ChatGPT, such as providing clear guidelines and precise questions.

**Expanded Understanding of Applications:** Some responses (around 5-10%) indicated that users discovered new applications of ChatGPT beyond their initial expectations, such as its use in different academic areas.