



AI Learning Support for CS

<https://codehelp.app/>

Tailored to your class contexts

Bring-your-own API key with low costs (OpenAI, Gemini, etc.)

Simple to add to any course to augment existing student support

+ it's open source

SIGCSE 2026 Demo: **Focused Tutors** feature



<https://codehelp.app/demo/SIGCSE26>

To use CodeHelp with a class, **go to the homepage or contact me:**
















Mark Liffiton – Illinois Wesleyan University

mliffito@iwu.edu

Class Configuration: Focused Tutors

Focused Tutors

Instructors can design Focused Tutors for students with pre-defined learning objectives and assessment questions (e.g., to be used as reinforcement and/or low-stakes assessments following a reading or video).

↓↑	Name <small>click to edit</small>	Available	actions
⋮	Stacks and Queues 	Now 	  
⋮	Pointers 	Scheduled: Fri, Feb 27 	  
⋮	Class definitions 	Hidden 	  

[+ Create new focused tutor](#) [Copy from another course...](#)

Creating/Editing a Focused Tutor

Editing focused tutor 'Stacks and Queues' in class 'SIGCSE 2026 Demo'

Name

Stacks and Queues

What students will see when selecting a tutor.

Topic or Focus

Stacks and Queues using Python lists

What should this tutor cover?

Learning context

A college CS 2 course

Briefly describe the class or other context.

E.g.: "A college CS 1 course using Python"

Context document (optional)

You may upload a PDF or DOCX file or paste text with additional context for generating objectives and questions (e.g., lecture notes, a book chapter, or an assignment).

Text will be extracted in your browser.

Max extracted text: 50,000 words.

Choose a PDF or DOCX...

No file chosen

or

Paste text

Creating/Editing a Focused Tutor

Generate draft
objectives &
questions.

Quickly/easily edit.

Learning Objectives

Regenerate ↴

5

draft learning objectives. *This will overwrite existing objectives and questions.*

1. Explain the differences between stacks and queues in terms of data access and order.
2. Implement a stack using Python lists with push, pop, and peek operations.
3. Implement a queue using Python lists with enqueue, dequeue, and peek operations.
4. Analyze the time complexity of stack and queue operations when implemented with Python lists.

Edit objectives

Regenerate ↴

3

draft questions per objective. *This will overwrite existing questions.*

Tutor Plan

1. Explain the differences between stacks and queues in terms of data access and order.

1. How does the order in which elements are accessed differ between a stack and a queue?
2. If we add elements A, B, and C in that order to both a stack and a queue, in what order will they come out?
3. What would be more suitable for implementing Undo functionality in an application: a stack of previous actions or a queue of previous actions?

Edit questions

2. Implement a stack using Python lists with push, pop, and peek operations.

1. Given the following code snippet, what will be the output after executing all the operations?

```
stack = []
stack.append(10)
stack.append(20)
```

Chat Interface

You I think a stack. You want to undo the most recent thing, not the oldest.

Tutor Perfect! That's exactly right.

An Undo feature works best with a **stack** because you want to reverse the most recent action first — the last thing you did should be the first thing undone. This matches the LIFO behavior of a stack.

Great job on this first objective! Do you feel comfortable with the differences between stacks and queues now, or is there anything else you want to clarify before we move on?

You I feel good. Let's move on.

Tutor Awesome! Let's move on to implementing a stack using Python lists.

Before diving into code, can you tell me what you think the main operations of a stack are? What would you expect a stack to be able to do?

You

Send

Plan & Progress Widget

Tutor's plan & your progress

COMPLETED	Explain the differences between stacks and queues in terms of data access and order.
IN PROGRESS	Implement a stack using Python lists with push, pop, and peek operations.
NOT STARTED	Implement a queue using Python lists with enqueue, dequeue, and peek operations.
NOT STARTED	Analyze the time complexity of stack and queue operations when implemented with Python lists.

Instructor View

Progress and completion at a glance. Quick access to chats.

Chats

id	user	chat_started	topic	user messages	analysis
1405	mark (local)	2/18/26, 12:31 AM	Stacks and Queues using Python...	5	2 0 1 1
1404	mark (local)	2/18/26, 12:26 AM	Stacks and Queues using Python...	9	1 0 1 2

Export CSV

Search...