



Using LLMs to Generate Personalized Assignments and Explanations

Dr. Barbara Ericson

Assistant Professor at the School of Information and the College of Engineering

barbarer@umich.edu

The Need for Student Support

Many students struggle when developing cognitive skills

- Math
- Science
- Programming

Failure, especially early when learning a new skill, reduces self-efficacy



Practice is Crucial for Learning

- Must be successful and include timely feedback
- Interactive > Constructive > Active > Passive

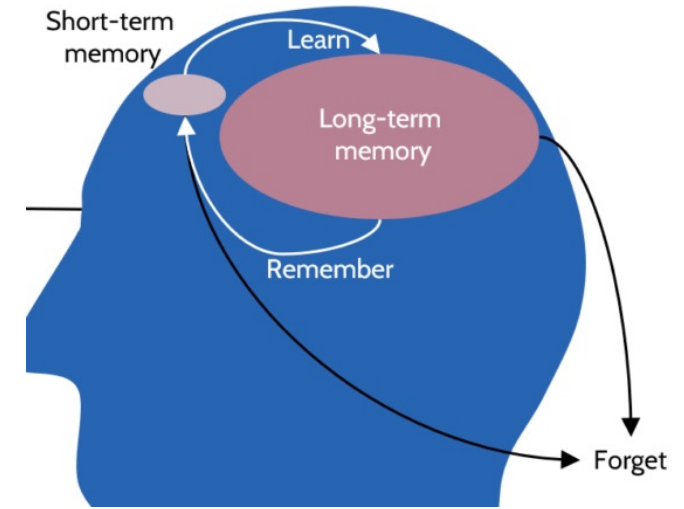
Chi and Wylie, The ICAP framework: Linking cognitive engagement to active learning outcomes, 2014

- Spaced practice is better for long term learning
- Desirable difficulties can reduce learning in the short-term
 - But increase long-term learning

Cognitive Load Theory

Humans process new information in working memory

- Limited capacity
- Whole tasks can impede learning
- Completion tasks can reduce cognitive load
- Worked examples plus practice best
- Avoid expertise reversal effect



Sweller, *Cognitive load during problem solving: Effects on learning*, 1988

Sweller, van Merriënboer, Paas, *Cognitive architecture and instructional design: 20 years later*, 2019

Parsons Problems

Create a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title, len`. For example, `print(s)` when `s = Song('Respect',150)` would print "Respect, 150".

Drag from here

1a `def __init__(title, len):`

or

1b `def __init__(self, title, len):`

2a `Class Song:`

or

2b `class Song:`

3 `self.title = title`
`self.len = len`

4a `def __str__(self):`

or

4b `def str(self):`

5a `return self.title + ", " + str(self.len)`

or

5b `return title + ", " + len`

Drop blocks here

Solution

2b `class Song:`

1b `def __init__(self, title, len):`

3 `self.title = title`
`self.len = len`

4a `def __str__(self):`

5a `return self.title + ", " + str(self.len)`

Check

Reset

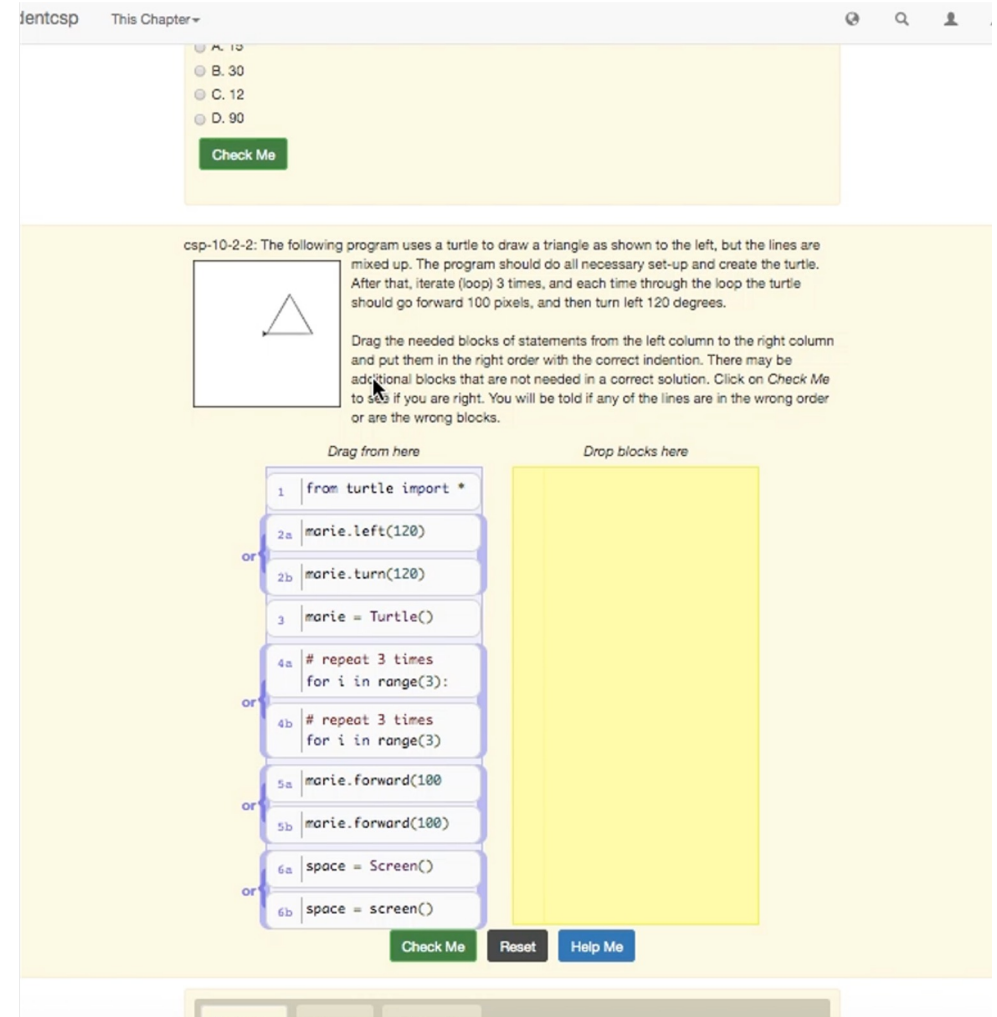
Help me

Parsons (Classes_Basic_Song_wd3_pp)

- Completion problems
- Usually have lower cognitive load than writing code
- Many variants
 - Adaptive
 - Proof Blocks
 - Faded
 - Micro

Adaptive Parsons Problems

- ▶ Intra-problem
 - ▶ If the learner is struggling to solve the current problem
 - ▶ Remove distractors
 - ▶ Provide Indentation
 - ▶ Combine Blocks




The screenshot shows a web-based interface for a Parsons problem. At the top, there's a header with "fentcsp" and "This Chapter". Below that, a list of options (A, B, C, D) is shown, with a "Check Me" button. The main problem area is titled "csp-10-2-2" and contains a description: "The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 100 pixels, and then turn left 120 degrees." To the left of the text is a small diagram of a triangle. Below the text, there are two columns: "Drag from here" and "Drop blocks here". The "Drag from here" column contains several code blocks, some of which are already placed in the "Drop blocks here" column. The blocks are: 1. "from turtle import *", 2a. "marie.left(120)", 2b. "marie.turn(120)", 3. "marie = Turtle()", 4a. "# repeat 3 times for i in range(3):", 4b. "# repeat 3 times for i in range(3)", 5a. "marie.forward(100)", 5b. "marie.forward(100)", 6a. "space = Screen()", 6b. "space = screen()". At the bottom, there are "Check Me", "Reset", and "Help Me" buttons.

Adaptive Parsons Problems

► Inter-problem

If solved the last one easily make the next one harder

csp-10-2-2: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 100 pixels, and then turn left 120 degrees.



Drag the needed blocks of statements from the left column to the right column and put them in the right order with the correct indentation. There may be additional blocks that are not needed in a correct solution. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

Drag from here

```

1 | marie = Turtle()
2 | space = Screen()
3 | space = screen()
4 | marie.turn(120)
5 | marie.left(120)
6 | from turtle import *
7 | # repeat 3 times
  | for i in range(3):
8 | # repeat 3 times
  | for i in range(3):
9 | marie.forward(100)
10| marie.forward(100)

```


Drop blocks here

Don't show distractors as paired and use all distractors

Check Me Reset Help Me

If many attempts then remove some distractors and pair them with the correct code

csp-10-2-2: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 100 pixels, and then turn left 120 degrees.



Drag the needed blocks of statements from the left column to the right column and put them in the right order with the correct indentation. There may be additional blocks that are not needed in a correct solution. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

Drag from here

```

1 | marie = Turtle()
2 | from turtle import *
3 | # repeat 3 times
  | for i in range(3):
4 | marie.forward(100)
5a| marie.turn(120)
5b| marie.left(120)
6a| space = Screen()
6b| space = screen()

```

Drop blocks here

Remove some distractors and pair the rest with the correct code

Check Me Reset Help Me

Micro Parsons – Zihan Wu

- Assemble fragments into order to create a statement

Please write a SELECT statement to retrieve the column `english` from table `grades` .

Check Me

Reset

Drag or click the blocks below to form your code:

english

FROM

grades

SELECT

WHERE

Your code (click on a block to remove it):

Research on Parsons Problems

- Most learners find them useful
- Significantly faster to solve than writing the equivalent code
 - With similar learning gains from pre to post
- Significantly more learners complete them vs write code problems in voluntary practice
- Can be used to learn common algorithms
- Nearly twice as likely to correctly solve adaptive than non-adaptive

Issues with Parsons Problems

Some learners would rather write the code themselves

- Especially those with more prior experience
- View writing code as more authentic

Some can solve them (especially adaptive)

- But not understand the solution

Giving Students A Choice

If given a Parsons problem

- Can switch to the equivalent write code
- Grade whichever is left on the page

Toggle Question: Parsons Mixed-Up Code - mult_class_point_and_triangle_practice_pp

Given a `Point` class that has a `distance` object method that takes another point and returns the distance between the two points, create a `Triangle` class with an `__init__` method that takes three points. Also write the `perimeter` method in the `Triangle` class to return the sum of the lengths of the sides of the triangle. Use the `distance` method in the `Point` class to calculate the distance between two points.

Drag from here Drop blocks here

```
1 class Triangle:
2     return total
```

```
3a def __init__(p1, p2, p3):
3b def __init__(self, p1, p2, p3):
```

```
4a total = distance(self.p1, self.p2)
4b total = self.p1.distance(self.p2)
```

```
5 self.p1 = p1
   self.p2 = p2
   self.p3 = p3
```

```
6a total += self.p2.distance(self.p3)
   total += self.p3.distance(self.p1)
6b total += distance(self.p2,
   self.p3)
   total += distance(self.p3,
   self.p1)
```

```
7a def perimeter():
7b def perimeter(self):
```

Parsons (mult_class_point_and_triangle_practice_pp)

Toggle Question: Active Write Code - mult_class_point_and_triangle_practice_ac

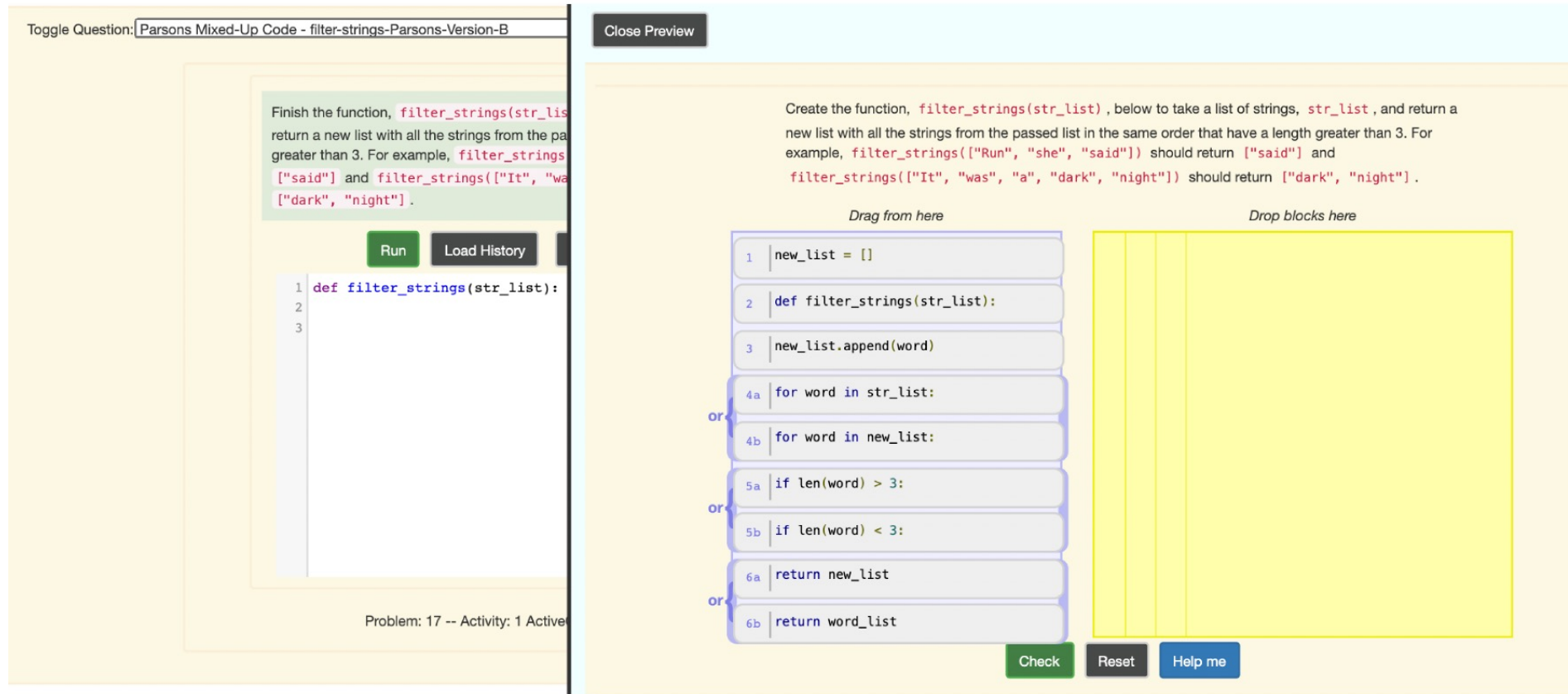
Given a `Point` class below with a `distance` object method that takes another point and returns the distance between the two points, create a `Triangle` class with an `__init__` method that takes three points. Also write the `perimeter` method in the `Triangle` class to return the sum of the lengths of the sides of the triangle. Use the `distance` method in the `Point` class to calculate the distance between two points.

```
1 import math
2 class Point:
3
4     def __init__(self, initX, initY):
5         """ Create a new point at the given coordinates. """
6         self.x = initX
7         self.y = initY
8
9     def distanceFromOrigin(self):
10        return ((self.x ** 2) + (self.y ** 2)) ** 0.5
11
12    def __str__(self):
13        return f"x = {self.x}, y = {self.y}"
14
15    def halfway(self, target):
16        mx = (self.x + target.x) / 2
17        my = (self.y + target.y) / 2
18        return Point(mx, my)
19
20    def distance(self, other):
21        mx = (self.x - other.x) ** 2
22        my = (self.y - other.y) ** 2
23        return math.sqrt(mx + my)
24
25 class Triangle:
26
27
```

Activity: 5 ActiveCode (mult_class_point_and_triangle_practice_ac)

Parsons as Scaffolding

Students can pop-up an equivalent Parsons problem while writing code



The image shows a Parsons problem interface. On the left, a code editor displays the following code:

```
1 def filter_strings(str_list):  
2  
3
```

Below the code editor are buttons for "Run" and "Load History".

On the right, a Parsons problem is displayed. The instructions are:

Create the function, `filter_strings(str_list)`, below to take a list of strings, `str_list`, and return a new list with all the strings from the passed list in the same order that have a length greater than 3. For example, `filter_strings(["Run", "she", "said"])` should return `["said"]` and `filter_strings(["It", "was", "a", "dark", "night"])` should return `["dark", "night"]`.

The Parsons problem consists of two columns:

- Drag from here:** A list of code blocks to be dragged into the Parsons problem. The blocks are:
 - 1 | `new_list = []`
 - 2 | `def filter_strings(str_list):`
 - 3 | `new_list.append(word)`
 - 4a | `for word in str_list:`
 - 4b | `for word in new_list:`
 - 5a | `if len(word) > 3:`
 - 5b | `if len(word) < 3:`
 - 6a | `return new_list`
 - 6b | `return word_list`
- Drop blocks here:** A large yellow box where the code blocks can be dropped.

At the bottom of the Parsons problem are buttons for "Check", "Reset", and "Help me".

Parsons as Scaffolding Results

A think-aloud study with 11 students

- Use to get started
- Help when stuck (how do I do x in Python)
- Use to debug

Between-subject study with 81 undergraduate students

- Parsons group took significantly less time to complete practice problems
- However, there was a ceiling effect on the pre-test

Parsons as Scaffolding Results

Between-subjects study with 89 undergraduates

- Parsons as scaffolding vs No scaffolding (write code)
- Students with low computing self-efficacy
 - With scaffolding had significantly higher practice performance and practice efficiency than no scaffolding
 - Still had students who didn't understand the Parsons solution
 - Parsons not helpful if it didn't match their approach

ActiveHint – Xinying Hou

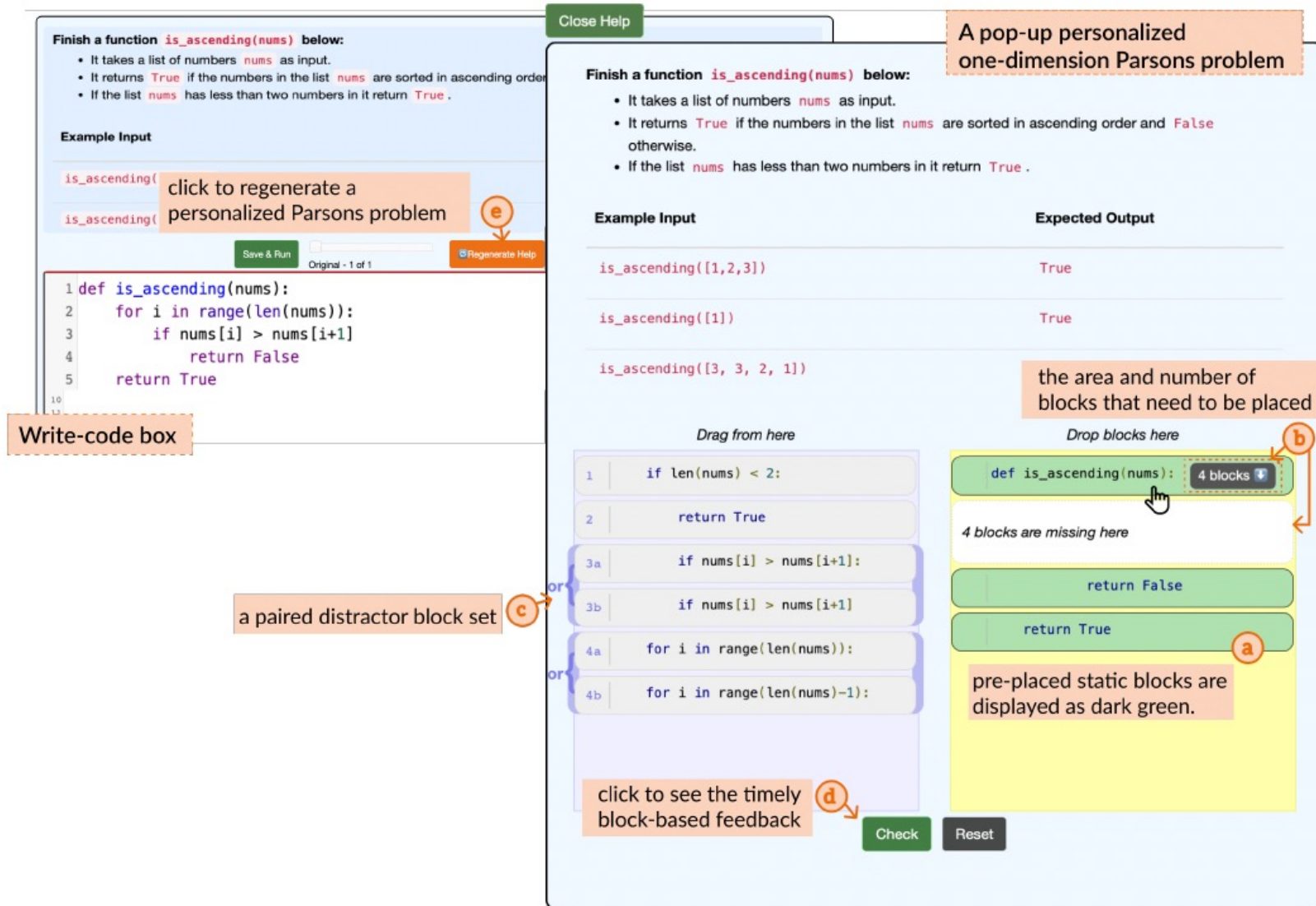
Use an LLM to generate a Parsons problem from a student's incorrect solution

1) Technical evaluation

Does ActiveHint create a more similar Parsons problem than using the most common solution?

2) Within-subjects study with 18 undergraduates. Received both generated solution and a personalized Parsons problem.

Personalized Parsons Problems



The interface is divided into two main sections: a code editor on the left and a block-based editor on the right.

Left Section (Write-code box): Contains the problem description and a code editor. The problem description asks to finish a function `is_ascending(nums)` that checks if a list of numbers is sorted in ascending order. The code editor shows a partially completed function with a `for` loop and an `if` statement. A button labeled "click to regenerate a personalized Parsons problem" is located above the code editor.

Right Section (Block-based editor): Contains the same problem description, a table of example inputs and expected outputs, and a block-based editor. The table lists three inputs: `is_ascending([1,2,3])` (True), `is_ascending([1])` (True), and `is_ascending([3, 3, 2, 1])` (False). The block-based editor has a "Drag from here" area with four pairs of blocks (3a/3b, 4a/4b) and a "Drop blocks here" area. The "Drop blocks here" area shows a function definition block with a dropdown menu set to "4 blocks", followed by two return blocks. A message "4 blocks are missing here" is displayed. A button labeled "click to see the timely block-based feedback" is located below the block-based editor.

Annotations: Several callouts are present: "A pop-up personalized one-dimension Parsons problem" points to the right section; "the area and number of blocks that need to be placed" points to the "Drop blocks here" area; "pre-placed static blocks are displayed as dark green." points to the return blocks; "a paired distractor block set" points to the 3a/3b and 4a/4b pairs; "Write-code box" points to the code editor; "click to regenerate a personalized Parsons problem" points to the button above the code editor; "click to see the timely block-based feedback" points to the button below the block-based editor.

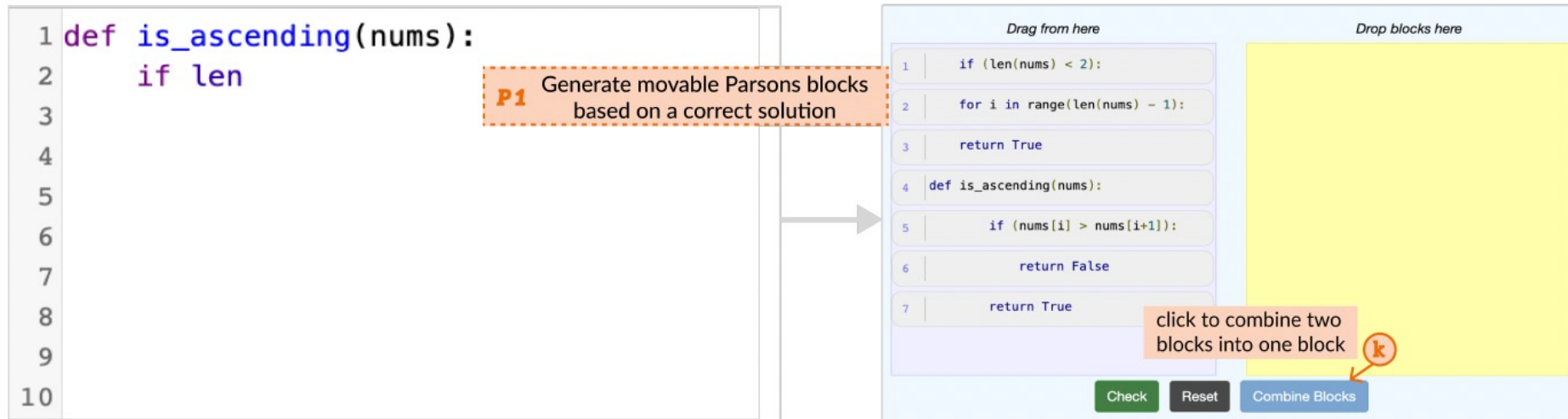
Use student's incorrect solution and an LLM to generate a personalized Parsons problem

Use student's incorrect code as distractors

User Has Not Written Anything/Much

- Generate Parsons from most common solution

A: Receive a fully movable Parsons problem that includes the combine block feature and no distractors



The image shows a Parsons problem interface. On the left, a code editor contains the following Python code:

```
1 def is_ascending(nums):  
2     if len  
3  
4  
5  
6  
7  
8  
9  
10
```

An orange callout box labeled "P1" points to the code editor with the text: "Generate movable Parsons blocks based on a correct solution".

On the right, the Parsons problem interface is shown. It has two main areas: "Drag from here" and "Drop blocks here". The "Drag from here" area contains the following code blocks:

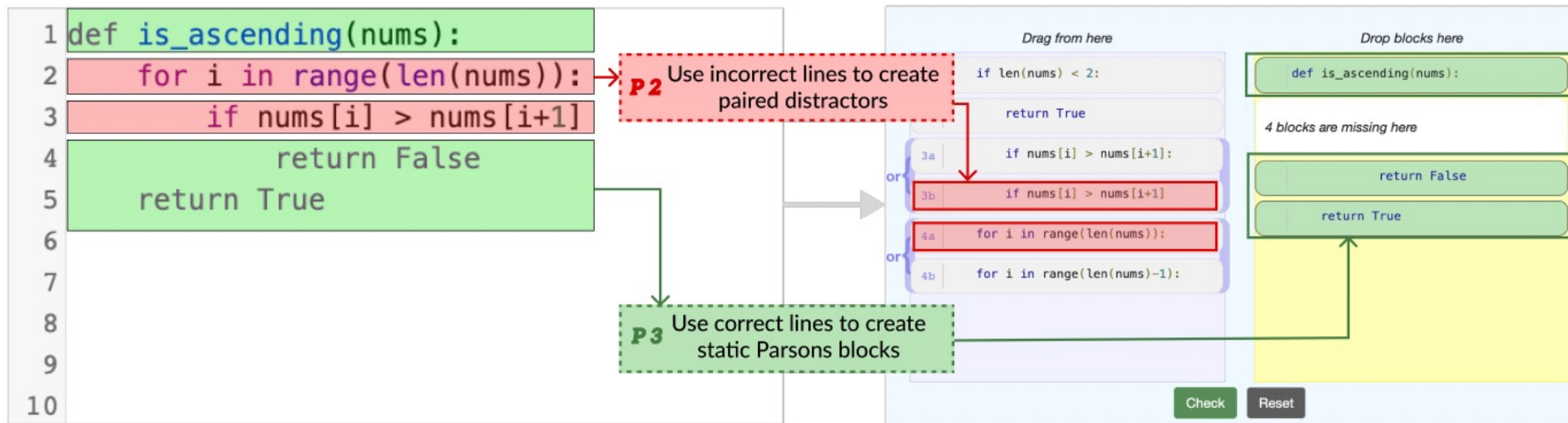
```
1 | if (len(nums) < 2):  
2 | for i in range(len(nums) - 1):  
3 | return True  
4 | def is_ascending(nums):  
5 |     if (nums[i] > nums[i+1]):  
6 |         return False  
7 |     return True
```

The "Drop blocks here" area is a large yellow rectangle. At the bottom of the interface, there are three buttons: "Check", "Reset", and "Combine Blocks". An orange callout box with a key icon points to the "Combine Blocks" button with the text: "click to combine two blocks into one block".

User Has Written Incorrect Code

- Generate distractors from the incorrect parts

B: Receive a partially movable Parsons problem with paired distractors containing student incorrect lines.



The diagram illustrates a Parsons problem interface. On the left, a code editor shows the following code:

```
1 def is_ascending(nums):
2   for i in range(len(nums)):
3     if nums[i] > nums[i+1]
4     return False
5     return True
6
7
8
9
10
```

Annotations on the code editor:

- A red dashed box labeled **P2** highlights lines 2 and 3 with the text: "Use incorrect lines to create paired distractors".
- A green dashed box labeled **P3** highlights lines 4 and 5 with the text: "Use correct lines to create static Parsons blocks".



The interface is divided into two main sections:

- Drag from here:** A light blue area containing code blocks. It includes a "return True" block, an "if len(nums) < 2:" block, and two "if nums[i] > nums[i+1]:" blocks labeled 3a and 3b. Below these are two "for i in range(len(nums)):" blocks labeled 4a and 4b. A red arrow points from the **P2** annotation to block 3b.
- Drop blocks here:** A light yellow area where blocks are placed. It contains a "def is_ascending(nums):" block, a "return False" block, and a "return True" block. A text label says "4 blocks are missing here". A green arrow points from the **P3** annotation to the "return True" block.

At the bottom right of the interface are "Check" and "Reset" buttons.

Getting Help

Loading help... You're capable of achieving great things. We will help you get there!

 a help loading bar

Finish a function `is_ascending(nums)` below:

- It takes a list of numbers `nums` as input.
- It returns `True` if the numbers in the list `nums` are sorted in ascending order and `False` otherwise.
- If the list `nums` has less than two numbers in it return `True`.

Example Input

Expected Output

`is_ascending([1,2,3])`

`True`

`is_ascending([1])`

`True`

`is_ascending([3, 3, 2, 1])`

`False`

Save & Run

Original - 1 of 1

Help



```
1 def is_ascending(nums):
2     for i in range(len(nums)):
3         if nums[i] > nums[i+1]
4             return False
```

Fig. 2. The help loading bar of ActiveHint includes a spinning loader and an encouragement sentence

Generation Process

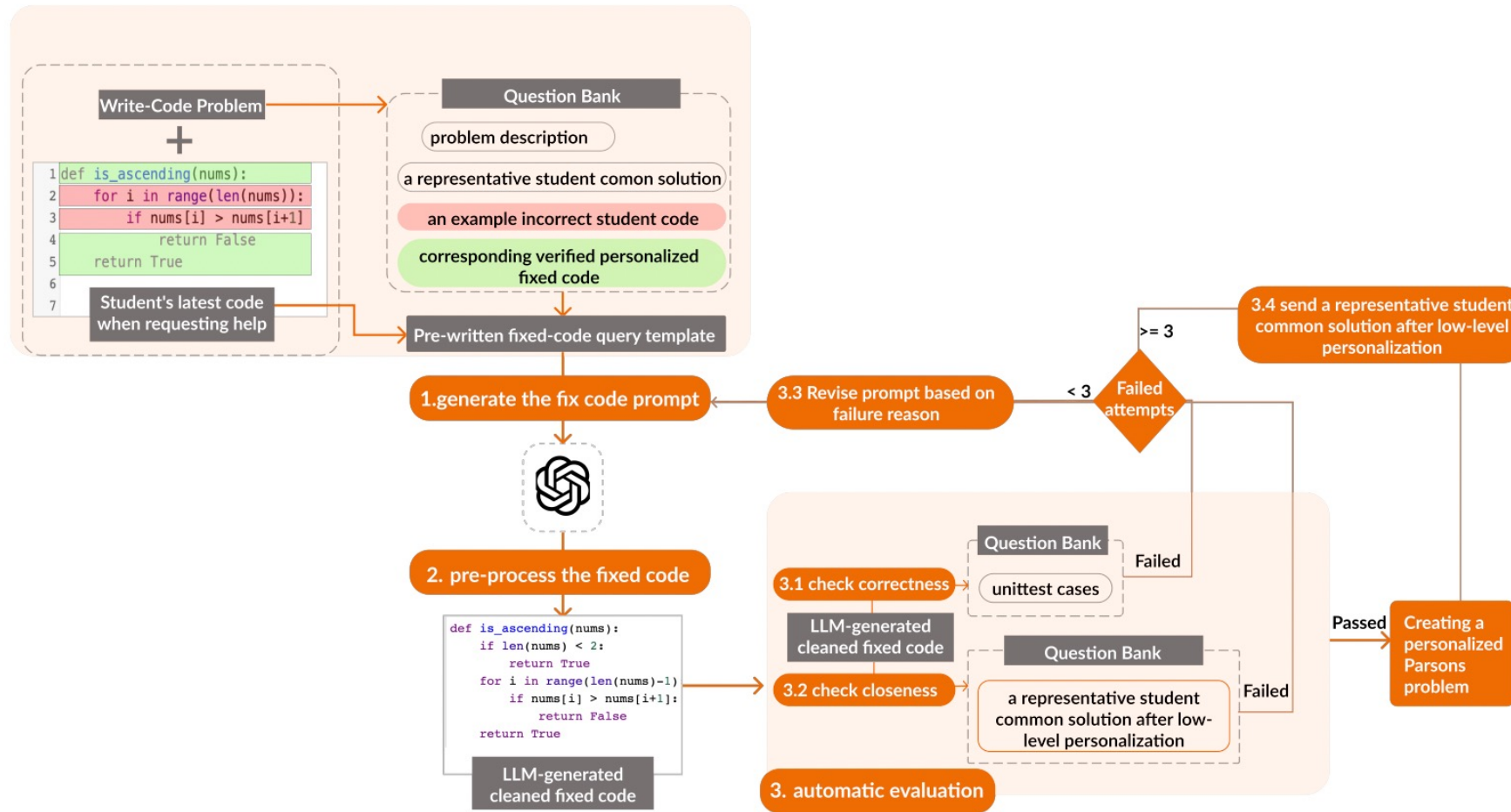


Fig. 5. The ActiveHint's personalized correct solution generation backend

Results

- ActiveHint generates Parsons problems that are significantly similar to the student's incorrect code
 - Vs the most common student solution
- Most students prefer receiving a personalized Parsons problem vs just the correct code
 - Some still need an explanation
- Will do an A/B test in Jan 2024
 - Personalized Parsons vs Generated Code

Future Work

- Create fully adaptable Parsons problems
 - Incorporate subgoal labels
 - To help transition from the problem description to code
 - Add different levels of LLM generated explanation
 - High level
 - Statement level
 - Symbol level
- Generate Personalized Micro Parsons for a subgoal
- Try in other fields

Recommendations

- Create tools that leverage LLMs
 - Leverage existing student data
 - Create personalized practice with different levels of explanations
 - Guarantee correct solutions
- Find ways to encourage students to learn
 - Not just copy/paste from LLMs
- Keep students in the Zone of Proximal Development