

Using Generative AI to Create Personalised Parsons Problems and Explanations:

<http://tinyurl.com/ericsonJan2024>

Dr. Barbara (Barb) Ericson

Assistant Professor

School of Information

University of Michigan

barbarer@umich.edu

Xinying Hou

PhD Student

School of Information

University of Michigan

xyhou@umich.edu

Barb's History

- BS in CS from Wayne State Un in 83
- Was a software engineer
 - General Motors Research Labs
 - Bell Communications Research
- MS in CS and Eng from Un of Michigan in 86
- Director of Outreach at Georgia Tech (GT)
- PhD from GT in 2018 – HCC



Research Areas

- Active learning
 - Interactive ebooks
 - Parsons problems
 - Peer Instruction
 - POGIL
- Social justice
- Near-peer mentoring programs
- Curriculum development
 - Media Computation
 - MOOC

Sisters Rise Up



Active Learning

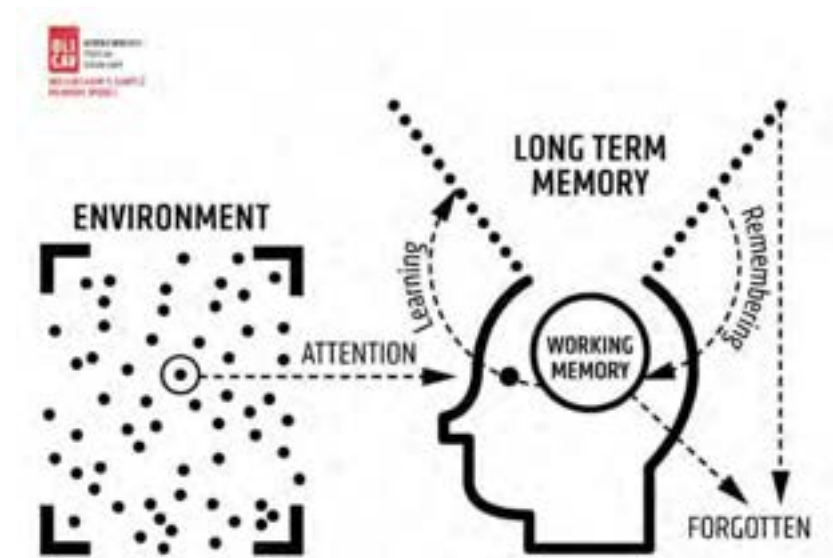
- ICAP theory of Cognitive Engagement
 - Interactive > constructive > active > passive
- Interactive – peer discussion
- Constructive – creating something new
- Active – moving something
- Passive – typical lecture

Chi, Michelene and Wylie, Ruth 2014



Cognitive Load Theory

- Developed by *John Sweller* in the 1980s
- Reduce cognitive load to free up working memory to improve learning
 - Completion task better than whole task - Van Merriënboer and De Croock. 1992.
 - Cognitive load is based on prior knowledge
- Predicts effects: worked example effect
 - Examples with interleaved practice best - Trafton & Reiser, 1993



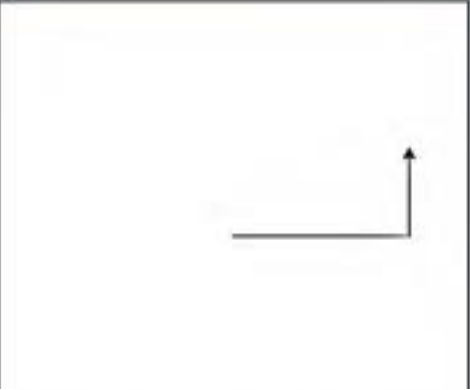
Interactive Ebooks

- Educational psychology principles
 - Worked examples plus practice with immediate feedback
 - Timed exams
 - Spaced practice tool
 - Lower cognitive load practice problems – Parsons
- My ebooks
 - CS Awesome - over 30,000 reg users
 - Python for Everybody - Interactive

Click the Run button to see what this code does. The drawing will happen below the code.

Save & Run Original - 1 of 1 Audio Tour Share Code

```
1 from turtle import * # use the turtle library
2 space = Screen() # create a Screen object named space
3 alex = Turtle() # create a Turtle object named alex
4 alex.forward(150) # ask alex to move forward by 150 units
5 alex.left(90) # ask alex to turn by 90 degrees
6 alex.forward(75) # ask alex to move forward by 75 units
7
```



csp-5-4-2: Which way does a turtle (object of the Turtle class) face when it is first created?

A. North
B. South
C. East
D. West

Check Me Compare me

Activity: 17.1.2 Multiple Choice (Turtle_Init_Dir_mc_q1)

Runestone Academy

- Founder
 - *Democratizing textbooks for the 21st century*
 - Started with one ebook in 2011
- 30+ free ebooks for computing and math
 - Over
- <https://tinyurl.com/bdz2fhzy>



Brad Miller



Timed Exams

- Instructors can create auto graded timed exams
- Can have MCQs, fill in the blank, mixed-up code (Parsons) problems, and write code problems with unit tests
 - Can compile code
- Students don't get feedback on their answers during the exam



Questions

Time Remaining 79:51

Warning: You will not be able to continue the exam if you close this tab, close the window, or navigate away from this page! Make sure you click the Finish Exam button when you are done to submit your work!

< Prev Next >

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Flag Question

Q-1: What will the following code print?

```
list1 = [0, 1, 2]
list2 = ["a", "b", "c"]
list1.append(list2)
print(list1)
```

A. [0, 1, 2, ["a", "b", "c"]]
B. [0, 1, 2, "a", "b", "c"]
C. [0, 1, 2]
D. The code will not run

Spaced-Practice Tool

- Spaced practice
 - Displays previous questions from an interactive ebook
 - Based on user's performance
 - Algorithm predicts when likely to forget a topic
 - Can earn points for answering X questions correctly over Y days
- Each hour of using the practice tool was associated with > 1% increase in the final exam grade

Iman Yeckehzaare



rec-5-44: What is the value of the following expression:

$$16 - 2 * 5 // 3 + 1$$

*A. 14
 B. 24
 C. 3
 D. 13.667

Check Me Compare me

Using parentheses, the expression is evaluated as (2*5) first, then (10 // 3), then (16-3), and then (13+1).

Done! Ask me another question!

I want to postpone this to tomorrow! Ask me another question.

What is a Parsons Problem?

- Provides mixed up code to solve a problem
 - Put the blocks in order
- Can have distractor blocks
 - Common syntax or semantic errors
- Can require indentation

Drag the blocks from the left and put them in the correct order on the right to define a function `print_greeting` that asks for your name and prints "Hello Name". Then define a `main` function that calls `print_greeting`. Be sure to also call the `main` function. Note that you will have to indent the lines that are in the body of each function. Click the Check button to check your solution.

Drag from here

Drop blocks here

```
1a print_greeting
1b print_greeting()
2 name = input("What is your name?")
3a def print_greeting():
3b def print_greeting()
4 main()
5 print("Hello " + name)
6a Def main():
6b def main():
```

```
3a def print_greeting():
2 name = input("What is your name?")
5 print("Hello " + name)
6b def main():
1b print_greeting()
4 main()
```


Check Reset Help me

Parsons (func_pp_print_name_v2)

Prior Research on Parsons Problems

- Distractors increase the difficulty
 - Stuart Garner 2007
 - Harms, Chen, and Kelleher 2016
 - Denny, Luxton-Reilly, and Simon 2008
- Pairing the correct and distractor blocks reduces the difficulty
 - Denny, Luxton-Reilly, and Simon 2008
- Providing indentation reduces the difficulty
 - Denny, Luxton-Reilly, and Simon 2008
 - Ihantola and Karavirta 2011
- Less blocks makes the problem easier
 - Denny, Luxton-Reilly, and Simon 2008

csp-10-4-1: The following program uses the stamp method to create a line of turtle shapes as shown to the left, but the lines are mixed up. The program should do all necessary set-up, create the turtle, set the shape to "turtle", and pick up the pen. Then the turtle should repeat the following three times: go forward 50 pixels and leave a copy of the turtle at the current position.



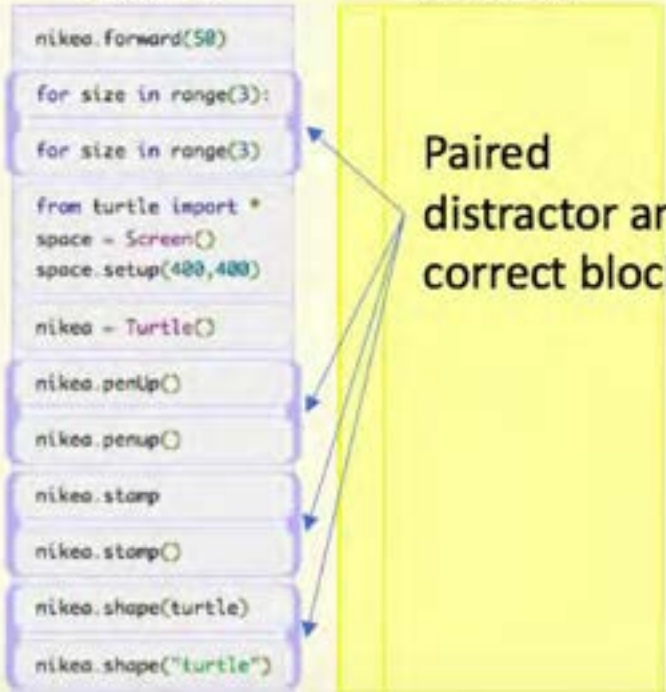
Drag the needed blocks of statements from the left column to the right column and put them in the right order with the correct indentation. Click on Check Me to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

Drag from here

```
nikea.forward(50)
for size in range(3):
for size in range(3)
from turtle import *
space = Screen()
space.setup(400,400)
nikea = Turtle()
nikea.penUp()
nikea.penup()
nikea.stamp
nikea.stamp()
nikea.shape(turtle)
nikea.shape("turtle")
```

Drop blocks here

Paired distractor and correct blocks



My Team's Research on Parsons Problems

- More people try to solve them than nearby multiple-choice questions
 - *Some people struggle and some never solve them*
- Parsons problems are *significantly faster* than fixing and writing code
 - *Equivalent learning gains* from pretest to posttest
- People are nearly *twice as likely to solve* adaptive Parsons
 - Most people prefer the adaptive
- Most students (75+%) find them *useful for learning programming*
 - Some would rather write the equivalent code
- Parsons problems *are not faster to solve if the solution is unusual*

Adaptive Parsons Problems

- Intra-problem
 - If the learner is struggling to solve the current problem
 - Remove Distractors
 - Combine Blocks

The following program uses a turtle to draw a triangle as shown below, 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* 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

Drop blocks here

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

Check Reset Help me

Parsons (turtle_2_2_Triangle)

Question Answer


Adaptive Parsons Problems

- Inter-problem

If user solved the last one easily, make the next one harder

If many attempts, 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 | turtle = Turtle()
2 | space = Screen()
3 | space = screen()
4 | turtle.turn(120)
5 | turtle.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 | turtle.forward(100)
10| turtle.forward(100)
```

Drop blocks here

Don't show distractors as paired and use all distractors

Check Me Reset Help Me

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

Drop blocks here

Remove some distractors and pair the rest with the correct code

Check Me Reset Help Me

Toggle Parsons Problems

- Users can solve either the Parsons problem
 - Or toggle to solve the equivalent write code problem with unit tests
- Grades whichever one the user leaves selected

Toggle Question: Parsons Mixed-Up Code - Classes, Basic, Airport.py

Not yet graded

Create a class `Airport` with an `__init__` method that takes a `name` and `code` as strings and initializes these attributes in the current object. Then define the `__str__` method to return the `name code`. For example, `print(a)` when `a = Airport("Detroit", "DTW")` would print `Detroit DTW`.

Drag from here

```
1 def __init__(self, name, code):
2 def __init__(name, code):
3 return name + " " + code
4 return self.name + " " + self.code
5 class Airport:
6 def Airport:
7 self.name = name
8 self.code = code
9 def __str__(self):
10 def str(self):
```

Drop blocks here

```
1 class Airport:
2 def __init__(self, name, code):
3     self.name = name
4     self.code = code
5 def __str__(self):
6     return self.name + " " + self.code
```

Check Reset Help me

Problem: 1 -- Parsons (Classes, Basic, Airport.py)

Write a class `Airport` with an `__init__` method that takes a `name` and `code` as strings and initializes these attributes in the current object. Then define the `__str__` method to return the `name code`. For example, `print(a)` when `a = Airport("Detroit", "DTW")` would print `Detroit DTW`.

Run 3/20/2022, 2:30:20 PM - 3 of 3 Show in CodeLens Share Code

```
1 class Airport:
2     def __init__(self, name, code):
3         self.name = name
4         self.code = code
5     def __str__(self):
6         return f"{self.name} {self.code}"
7
8 a = Airport("Detroit", "DTW")
9 print(a)
10
11
```

Detroit DTW

Result	Actual Value	Expected Value	Notes
Pass	'Detroit DTW'	'Detroit DTW'	testing __str__ for Detroit
Pass	'Atlanta ATL'	'Atlanta ATL'	testing __str__ for Atlanta

You passed: 100.0% of the tests

Micro Parsons Problems for Regex and SQL

Construct a regex that matches both `cat` and `cats`.

Drag or click the blocks below to form your code:

`s` `cat` `?` `*`

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

`cat` `s` `?`

Activity: 12.11.1 MicroParsons (hp-optional_cats)

Create a student table with an id of type integer (INT), and a name of type string (TEXT).

Drag or click the blocks below to form your code:

`"student"` `"name": TEXT` `(` `"id": INT,` `CREATE` `)`

`"id" INT,` `TABLE` `"name" TEXT`

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

`CREATE` `TABLE` `"student"` `(` `"id" INT,` `"name" TEXT`

Activity: 1 MicroParsons (mparsons-sql-pract-table)

Micro Parsons Problems Research

- Micro Parsons condition had a significantly higher completion rate than text-entry
 - and higher learning gains on regex symbols
- Tested two types of feedback with SQL
 - Block based
 - Execution based
- Students who solved SQL micro Parsons in the *blocks-based condition* had *significantly higher learning gain* than in text-entry condition

Zihan Wu



Parsons as Scaffolding While Writing Code

- When writing code
 - Can pop-up a Parsons problem
- Must still solve the write code problem
- Need to analyze log file data to see what percentage of students use this feature

The image displays two screenshots of a Parsons problem interface. The left screenshot shows a progress bar with five items, each with a percentage (0.0%, 0.0%, 1.18.7%, 1.18.7%, 0.0%) and a 'Run' button. Below the progress bar is a code editor with a partially written function:

```
1 def has22(nums):  
2  
3
```

 The right screenshot shows a 'Close Preview' button, a problem description, a 'Drag from here' section with code blocks, a 'Drop blocks here' section, and 'Check', 'Reset', and 'Help me' buttons. The problem description reads: 'Create the function `has22(nums)` below to return `True` if there are at least two items in the list `nums` that are adjacent and both equal to `2`, otherwise return `False`. For example, return `True` for `has22([1, 2, 2])` since there are two adjacent items equal to `2` (at index 1 and 2) and `False` for `has22([2, 1, 2])` since the `2`'s are not adjacent.'

Can Parsons problems support struggling students with completing write-code problems?

An equivalent adaptive
Parsons problem

Toggle Question: Parsons Mixed-Up Code - Classes_Basic_Song_pp

Write a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number. Then define the `__str__` method to return the `title`, `len`. For example

Original - 1 of 1

```
1
2 s = Song('Respect',150)
3 print(s)
4
5
```

Activity: 4 ActiveCod

Write-code Problem

Close Preview

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

Drop blocks here

```
1 self.title = title
  self.len = len
or
2a def __str__():
2b def __str__(self):
or
3a class Song:
or
3b class Song:
or
4a def __init__(self, title, len):
or
4b def __init__(title, len):
or
5a return title + ", " + len
or
5b return self.title + ", " + str(self.len)
```

Parsons (Classes_Basic_Song_pp)

Most Students Find it Helpful

1 - Reduce difficulty and completion time

2 - Learn problem-solving strategies

3 - Syntax reminder

4 - Prompt for deeper thinking

Some Still Struggle to Solve the Parsons Problem



"I was thinking differently on how to approach it, so it's just the Parsons problem (solution) didn't seem reasonable to me." (P9)

"Because a couple of times I used the wrong choice, so that wasn't very helpful. And that kind of led me astray a little bit. " (P3)

Can Parsons support **align** with student's approach?

Can Parsons support provide **targeted** distractors?

Students with Low CS Self-efficacy Levels

Practice writing code with
Parsons problem or independently

▲ Higher Practice Performance

▲ Higher Problem-solving Efficiency

When practice writing code
with Parsons problem

More likely to **solve** the Parsons
problem

Can Parsons support be more
concise in a personalized way?

2022-2023:

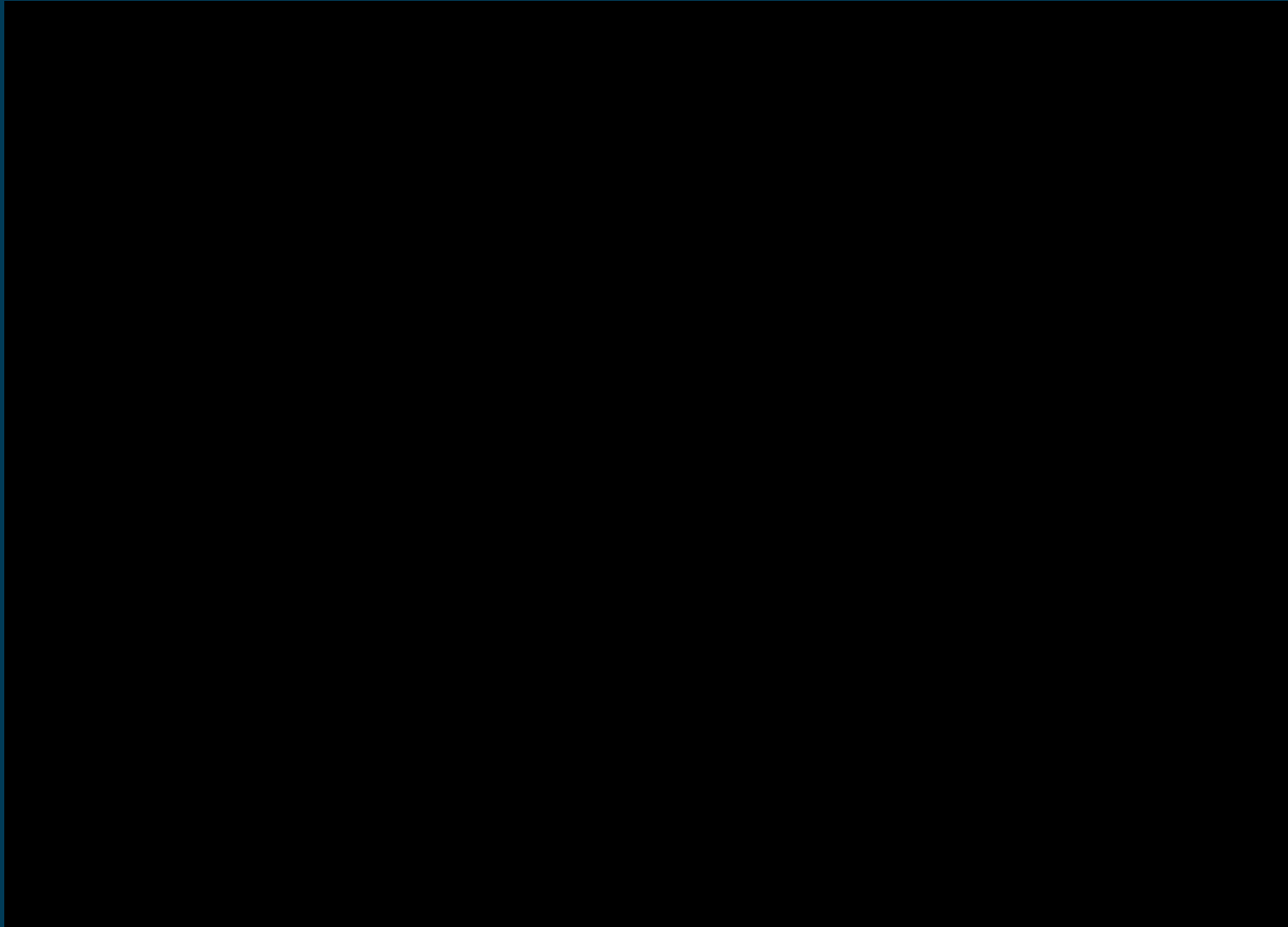
GitHub Copilot, Open AI ChatGPT,

Google Bard, Anthropic Claude ...

Generative AI tools can generate code solutions

They are now in CS classrooms

**I don't know how to solve the problem.
So I'll just use ChatGPT and submit the work!**



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.
- If the list `nums` has less than two numbers in it return `True`.

Example Input

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

Click to regenerate a personalized Parsons problem

Save & Run Original - 1 of 1 Regenerate Help

```
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
```

Write-code box

Close Help

Personalized Parsons problem

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

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

Expected Output

```
True  
True
```

The area and number of blocks that need to be placed

Drag from here

```
1 if len(nums) < 2:  
2     return True  
3a if nums[i] > nums[i+1]:  
3b if nums[i] > nums[i+1]:  
4a for i in range(len(nums)):  
4b for i in range(len(nums)-1):
```

Drop blocks here

```
def is_ascending(nums): 4 blocks  
return False  
return True
```

4 blocks are missing here

Pre-placed static blocks are displayed as dark green.

Check to see the timely block-based feedback

Check Reset

Scenario 1: Receive a fully movable Parsons problem with "combine block" feature and no distractors

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

P1 Generate movable Parsons blocks based on a correct solution

The screenshot shows a Parsons problem interface. On the left, under the heading "Drag from here", there are seven 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

On the right, under the heading "Drop blocks here", there is a large yellow rectangular area. At the bottom of the interface, there are three buttons: "Check" (green), "Reset" (dark grey), and "Combine Blocks" (blue). A callout box with an orange border and a red circle containing the letter 'k' points to the "Combine Blocks" button, with the text "click to combine two blocks into one block".

Scenario 2: Receive a partially movable Parsons problem with paired distractors from student errors

The diagram illustrates a Parsons problem interface for a Python function. On the left, the source code is shown with line numbers 1 to 10. Lines 2 and 3 are highlighted in red, and lines 4 and 5 are highlighted in green. A red dashed box labeled 'P2' points to lines 2 and 3, with the text 'Use incorrect lines to create paired distractors'. A green dashed box labeled 'P3' points to lines 4 and 5, with the text 'Use correct lines to create static Parsons blocks'. In the center, a 'Drag from here' area contains code blocks: a light blue block with 'if len(nums) < 2: return True', a red block with 'if nums[i] > nums[i+1]:', a light blue block with 'for i in range(len(nums)):', and another red block with 'for i in range(len(nums)-1):'. On the right, a 'Drop blocks here' area shows the target code structure. The first block is 'def is_ascending(nums):', followed by a gap labeled '4 blocks are missing here', then 'return False', and finally 'return True'. A yellow shaded area is below the 'return True' block. At the bottom right, there are 'Check' and 'Reset' buttons.

```
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
```

P2 Use incorrect lines to create paired distractors

P3 Use correct lines to create static Parsons blocks

Drag from here

```
if len(nums) < 2:
    return True
if nums[i] > nums[i+1]:
    if nums[i] > nums[i+1]:
for i in range(len(nums)):
for i in range(len(nums)-1):
```

Drop blocks here

```
def is_ascending(nums):
4 blocks are missing here
return False
return True
```

Check Reset

Scenario 3: Receive a partially movable Parsons problem with paired distractors from AI and student errors

The image shows a Parsons problem interface for a Python function. The source code on the left is as follows:

```
1 def is_ascending(nums):
2     if len(nums) < 2:
3         return True
4     i = 0
5     while i < len(nums):
6         if nums[i] > nums[i+1]:
7             return False
8             i += 1
9     return True
10
```

The interface includes a "Drag from here" area with the following blocks:

- 1a: `i = 0` (green)
- 1b: `while i < len(nums) - 1:` (green)
- 2a: `while i < len(nums):` (red)
- 2b: `if (nums[i] <= nums[i+1]):` (red)
- 3a: `if nums[i] > nums[i+1]:` (green)
- 3b: `if nums[i] > nums[i+1]:` (green)

The "Drop blocks here" area contains the following blocks:

- `def is_ascending(nums):` (green)
- `if len(nums) < 2:` (green)
- `return True` (green)
- 2 blocks are missing here (yellow)
- `return False` (green)
- `i += 1` (green)
- `return True` (green)

Annotations and actions:

- A red dashed box: "Use the incorrect line to create a paired distractor" (points to line 5).
- A blue dashed box: "P4 Convert a correct line into a distractor generated by LLM." (points to line 6).
- A green dashed box: "6 static Parsons blocks from correct lines" (points to the 'Drop blocks here' area).
- Buttons: "Check" and "Reset".

Block Relative Placement Guidance

Timely Block-based Feedback

Regenerate the Help

The area and number of blocks that need to be placed

Drop blocks here

The screenshot shows a code editor with a function definition: `def is_ascending(nums):`. A dashed red box highlights the opening curly brace and the first parameter `nums`. To the right of this box, a dark grey tooltip displays "4 blocks" with a downward arrow. Below the code, a white feedback message states "4 blocks are missing here". A yellow highlight covers the entire function block. A red circle labeled "B" is in the top right corner, with a red arrow pointing to the tooltip and another red arrow pointing to the feedback message. The text "Drop blocks here" is positioned above the function block.

**Block Relative
Placement Guidance**

**Timely Block-based
Feedback**

Regenerate the Help

Drag from here

```
4a if nums[i] > nums[i+1]
```

```
3a for i in range(len(nums)-1):
```

Drop blocks here

```
def is_ascending(nums): 4 blocks
```

```
2 if len(nums) < 2:
```

```
1 return True
```

```
3b for i in range(len(nums)):
```

```
4b if nums[i] > nums[i+1]:
```

```
return False
```

```
return True
```

the incorrect block is highlighted in red

Check Reset

Highlighted blocks in your answer are wrong or are in the wrong order. This can be fixed by moving, removing, or replacing highlighted blocks.

**Block Relative
Placement Guidance**

**Timely Block-based
Feedback**

Regenerate the Help

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.
- If the list `nums` has less than two numbers in it return `True`.

Example Input

```
is_ascending(  
is_ascending(
```

Click to regenerate a
personalized Parsons problem



Save & Run

Original - 1 of 1

Regenerate Help

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

In 2023 summer, with ...

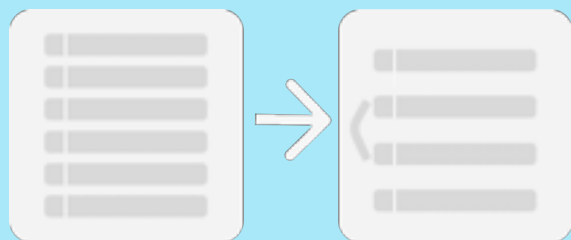
- **800 former student buggy code: Can CodeTailor provide high-quality personalized Parsons puzzles?**
- **18 novice programming learners: Can CodeTailor support a more engaging and educational write-code experience than the typical AI-generated solution?**

😊 CodeTailor can

100

Error-Free Solution Generation

Closely align with the incorrect student code compared to a common solution.

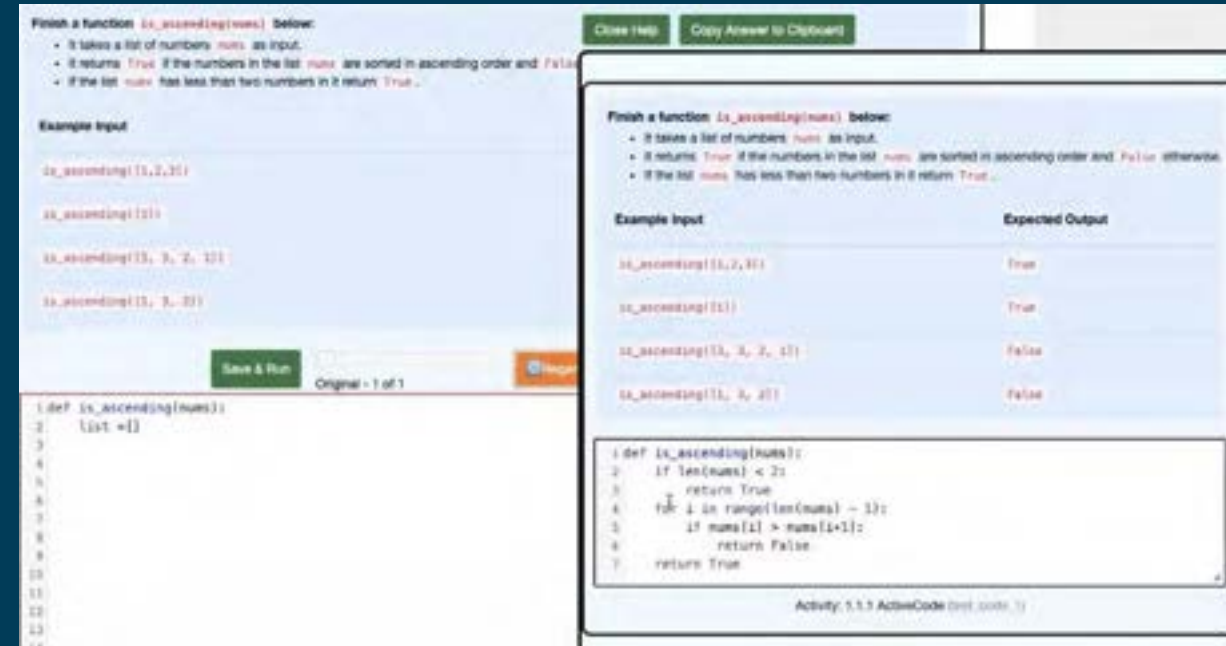


Delivery of More Concise Parsons Problems

😊 Students ... (CodeTailor / An AI-generated solution)

Report higher engagement
with CodeTailor

Recalled and applied more new
components from the practice
to the posttest



The screenshot displays the CodeTailor interface for a coding problem. The problem statement is: "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`." The interface includes a "Save & Run" button and a "Copy Answer to Clipboard" button. The solution code is shown in a code editor:

```
1 def is_ascending(nums):
2     list = []
3
4
5
6
7
8
9
10
11
12
13
```

The expected output is shown in a table:

Example Input	Expected Output
<code>is_ascending([1,2,3])</code>	<code>True</code>
<code>is_ascending([1])</code>	<code>True</code>
<code>is_ascending([3, 2, 1])</code>	<code>False</code>
<code>is_ascending([1, 3, 2])</code>	<code>False</code>

The solution code is:

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

😊 Students ... (CodeTailor / An AI-generated solution)

More
interactive

Right amount
of effort

Think of the
solution-building
process

**88% preferred CodeTailor
more for learning**

Boost confidence

Diagnose original
thinking

Students with CodeTailor

**Can solve, but not fully
understand the solved solution**

To address this challenge ...

Subgoal Breakdown

- 1: Check the Length of the Input List
- 2: Handle the Initial Condition
- 3: Iterate through the List
- 4: Compare Adjacent Elements
- 5: Return False on First Decreasing Pair
- 6: Return True If No Decreasing Pair is Found

Reset

Subgoal guidance explanation

Drag from here

```
1a for i in range(len(nums)):  
4a   if nums[i] > nums[i+1]
```

Drop blocks here

```
def is_ascending(nums):  
3   if len(nums) < 2:  
       return True  
   for i in range(len(nums) - 1):  
       return True
```

It checks if the length of the **nums** list is less than 2. If so, the code block following this line (indented below it) will be executed.

block-level explanation

The "-1" is used here to ensure that the loop only iterates up to the second-to-last element, avoiding an "out of range" error when comparing the last two elements.

atom-level explanation

Check Reset

Perfect! It took you 3 tries to solve this. Click Reset to try to solve it in one attempt.

In 2023 fall, to address this challenge ...

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.
- If the list `nums` has less than two numbers in it return `True`.

Example Input

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

```
1 def is_ascending(nums):  
2     for i in range(len(nums)-1):  
3  
4  
5
```

Close Help

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

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

Expected Output

```
True  
True  
False
```

Subgoal Breakdown

Drop blocks here

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

Check Reset

Perfect! It took you 3 tries to solve this. Click Reset to try to solve it in one attempt.

To start, it employs `if` and `len` to verify if the list contains `few than or two` elements, indicating an inherently sorted list. For lists with more than one element, it utilizes a `for` loop through the elements. Employing `range`, `len(nums)`, and `nums[i]`, it compares each number with the `following element`. The variable `i` is used as `an index` within a `for` loop to traverse through the list `nums`. If any number is greater than its `previous element`, the function concludes using `return False` within the loop. If the loop goes `following element` numbers without finding any "mismatched" pair `previous element` the list's ascending order and uses `return True`.

A self-explanation question to reflect on the Parsons solution

Peer Instruction

- Created by Eric Mazur
 - Display a hard multiple-choice question
 - Students vote individually
 - Discuss with peers
 - Vote again
- More students succeed
 - But clickers are problematic
 - Hard to find good questions
- We have added existing questions to ebooks
- Added new interfaces
 - In lecture text-chat
 - After lecture pseudo text-chat

The image shows two screenshots of the Peer Instruction interface. The top screenshot is the 'Peer Instruction Dashboard' with navigation tabs for Student Progress, Admin, Grading, Assignments, Practice, and Help/Documentation. It features a 'Group Size' dropdown set to 2 and several control buttons: 'Stop Vote 1', 'Enable Discussion', 'Start Vote 2', 'Stop Vote 2', 'Next Question', and 'Start Over'. A question is displayed: 'Q-1: What is returned from 3 % 4?' with radio button options A. 0, B. 1, C. 2, D. 3, and E. 4. A 'Check Me' button is at the bottom. On the right, it shows 'Number of Answers 0' and a 'Hide/Show Graph' button.

The bottom screenshot is titled 'Peer Instruction Question (After Class)'. It provides instructions: '1. Answer the question as best you can.', '2. Then, in the space provided write a justification for your answer.', '3. Read the dialog between two of your peers on why they answered the question the way they did.', and '4. Answer the question again. Even if you are not changing your answer from the first time.' The question is: 'Q-1: Which of the following finds all the links with a class of csa-1?' with options A. `links = soup.find('a', class = 'csa-1')`, B. `links = soup.find_all('a', class = 'csa-1')`, C. `links = soup.find('a', class_ = 'csa-1')`, and D. `links = soup.find_all('a', class_ = 'csa-1')`. Option D is selected. A 'Check Me' button is present. Below the question, a confirmation message says 'D. find all will return all the link tags'. To the right, there is a text area for justification with the text 'Links are the 'a' tag and the class is 'csa-1'' and a 'Submit' button. Below that, a 'A discussion for you to consider' section shows a dialog between two users: 'User 1: answered D' and 'User 2: said: find_all finds all the links and class_ is the right tag' and 'User 3: said: The class needs to have an underscore and the question requests that we find all links, so find_all is necessary.' A red note at the bottom says: 'Please Answer the question again. Even if you do not wish to change your answer. After answering click the button to go on to the next question.' A 'Next Question' button is at the bottom right.

Guided Inquiry Learning in Lecture

<https://tinyurl.com/bddz8zx3>

- Work in groups
 - One person fills in answers and shares with others
- Activities expose students to the concepts to be learned
- Students record what they learned and any questions in padlet
- Barb goes over questions from the padlet

5.29. Group Work: Functions and Lists

It is best to use a POGIL approach with the following. In POGIL students work in groups on activities and each member has an assigned role. For more information see <https://cspogil.org/Home>.

Note

If you work in a group, have only one member of the group fill in the answers on this page. You will be able to share your answers with the group at the bottom of the page.

Learning Objectives

Students will know and be able to do the following.

Content Objectives:

- Use positive and negative indices to access elements of a list.
- Identify the purpose of common list methods and common methods that take lists as parameters
- Use the slice operator to copy parts of a list.

Process Objectives:

- Predict the output of code with lists (Information Processing)
- Write code using the slice operator (Assessment)

5.29.1. List Indexing

A list holds items in order and you can get the value at an index, just like you can with strings.

Q-1: What is the last thing that will be printed when the code below runs?

Check me

Compare me

Social Justice in Computing

Aadarsh Padiyath

- Semi-structured interviews with 11 students
 - 63% say computing lacks communal applications
- Most students wanted more homework/projects with a social justice context
 - Real data, direct action, in-class discussion
- Use satirical Programming to critique computing artifacts



MOOC Based on Python for Everybody

- Four courses will be on Coursera in late Feb
- Audience
 - teachers
 - students
 - conversational programmers
- Not lecture!
 - Open-ended and creative assignments
 - Work in interactive ebook



The screenshot shows the FutureLearn website interface. At the top, the FutureLearn logo is on the left, and navigation links for 'Subjects', 'Courses', and 'Using FutureLearn' are in the center. On the right, there are search, 'Sign in', and 'Register' buttons. Below the navigation, the breadcrumb 'Online Courses / IT & Computer Science' is visible. The course card features the University of Michigan logo, the title 'Big Ideas in Programming: Expressing Yourself with Python', and a description: 'Harness the power of Python and its programming concepts to express yourself and automate the work you do.' A 'Join course' button is present, along with the text '1,073 enrolled on this course'. To the right of the text is a large illustration of people working together to assemble a large puzzle on a yellow background.

Summary

<http://tinyurl.com/ericsonJan2024>

- Use active learning
 - Free and interactive ebooks
 - Peer Instruction
 - Low cognitive load practice (Parsons problems)
 - Process oriented guided inquiry learning (POGIL)
- Use AI with guardrails to help students succeed
- Use AI to generate or personalize problems
- Incorporate social justice in computing

Runestone Ebooks

