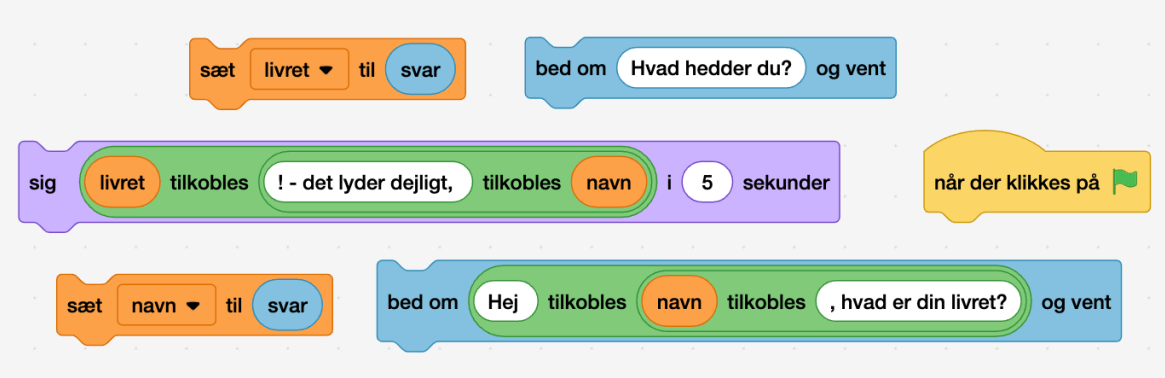
PARSONS PUZZLE

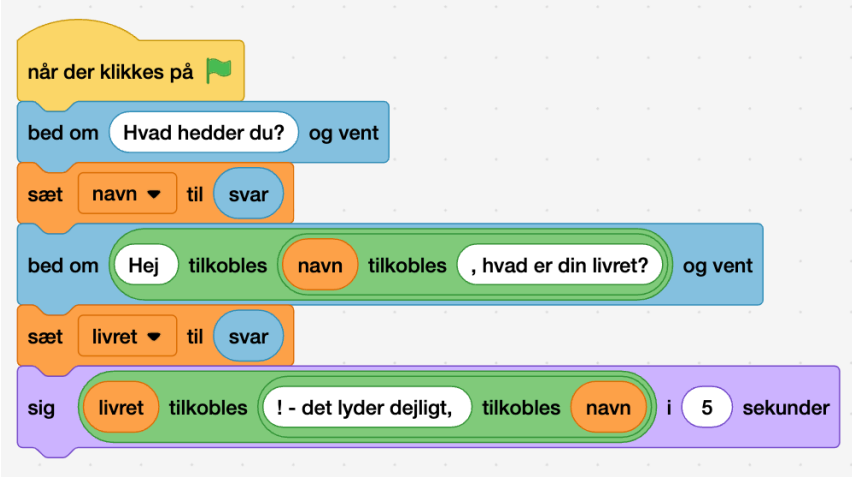
Tirsdag 27. januar 2026

PARSONS PUZZLE

Parsons Puzzle er et puslespil, hvor eleven får udleveret alle nødvendige kodelinjer til at kunne løse et konkret problem. Men kodelinjerne er ikke placeret i den rigtige rækkefølge, og det er så elevens opgave at placere dem korrekt.

Kaldes også Parsons Problem





PARSONS PUZZLE

```
Lav den rigtige rækkefølge af nedenstående kodelinjer,
så der byttes om på værdierne af variablene a og b
```

In [1]:
```
a = 5
b = 7
a = b
temp = a
b = temp
```

PARSONS PUZZLE

```
Lav den rigtige rækkefølge af nedenstående kodelinjer,
så der byttes om på værdierne af variablene a og b
```

In [1]:
```
a = 5
b = 7
a = b
temp = a
b = temp
```

In [2]:
```
a, b
```

Out[2]:
```
(7, 7)
```

PARSONS PUZZLE

```
Lav den rigtige rækkefølge af nedenstående kodelinjer,
så der byttes om på værdierne af variablene a og b
```

```
In [1]: a = 5
        b = 7
        a = b
        temp = a
        b = temp
```

```
In [2]: a, b
```

```
Out[2]: (7, 7)
```

```
In [3]: a = 5
        b = 7
        temp = a
        a = b
        b = temp
```

```
In [4]: a, b
```

```
Out[4]: (7, 5)
```

PARSONS PUZZLE

```
Lav den rigtige rækkefølge af nedenstående kodelinjer,
så der byttes om på værdierne af variablene a og b
```

```
In [1]: a = 5
        b = 7
        a = b
        temp = a
        b = temp
```

```
In [2]: a, b
Out[2]: (7, 7)
```

```
In [3]: a = 5
        b = 7
        temp = a
        a = b
        b = temp
```

```
In [4]: a, b
Out[4]: (7, 5)
```

```
In [5]: a = 5
        b = 7
        a, b = b, a
```

PARSONS PUZZLE

```
Lav den rigtige rækkefølge af nedenstående kodelinjer,
så der byttes om på værdierne af variablene a og b
```

```
In [1]: a = 5
        b = 7
        a = b
        temp = a
        b = temp
```

```
In [2]: a, b
Out[2]: (7, 7)
```

```
In [3]: a = 5
        b = 7
        temp = a
        a = b
        b = temp
```

```
In [4]: a, b
Out[4]: (7, 5)
```

```
In [5]: a = 5
        b = 7
        a, b = b, a
```

```
In [6]: a, b
Out[6]: (7, 5)
```

PARSONS PUZZLE

Sæt brikkerne korrekt sammen, så der dannes en SQL-sætning, der udvælger alle venner, der hedder Ole

Venner	=	SELECT	'Ole'
FROM	WHERE	*	Navn

PARSONS PUZZLE

Sæt brikkerne korrekt sammen, så der dannes en SQL-sætning, der udvælger alle venner, der hedder Ole

Venner | = | SELECT | 'Ole'

FROM | WHERE | * | Navn

SELECT | * | FROM | Venner | WHERE | Navn | = | 'Ole'

PARSONS PUZZLE

MATEMATIK

Sæt de enkelte kasser i en korrekt rækkefølge, så der løses en ligning

$x^2 = 16$

$x \cdot x = 2 \cdot 8$

$x = 4 \quad \text{eller} \quad x = -4$

$x = \sqrt{16} \quad \text{eller} \quad x = -\sqrt{16}$

$\frac{x}{2} = \frac{8}{x}$

PARSONS PUZZLE

MATEMATIK

Sæt de enkelte kasser i en korrekt rækkefølge, så der løses en ligning

$$\frac{x}{2} = \frac{8}{x}$$

$$x \cdot x = 2 \cdot 8$$

$$x^2 = 16$$

$$x = \sqrt{16} \quad \text{eller} \quad x = -\sqrt{16}$$

$$x = 4 \quad \text{eller} \quad x = -4$$

PARSONS PUZZLE

Example of js-parsons turtle graphics assignment

Construct a program by drag&dropping and reordering lines from the top to the bottom. The constructed program should draw a triangle like shown below. Click ?? to select the correct value for that position.

Drag from here

```
forward(100)
ENDREPEAT
REPEAT ?? TIMES
left(120)
```

Construct your solution here

Model Drawing

Your Code Drawing

Reset | Feedback

https:// js-parsons.github.io/

PARSONS PUZZLE

Example of js-parsons turtle graphics assignment

Construct a program by drag&dropping and reordering lines from the top to the bottom. The constructed program should draw a triangle like shown below. Click ?? to select the correct value for that position.

Drag from here

Model Drawing

Construct your solution here

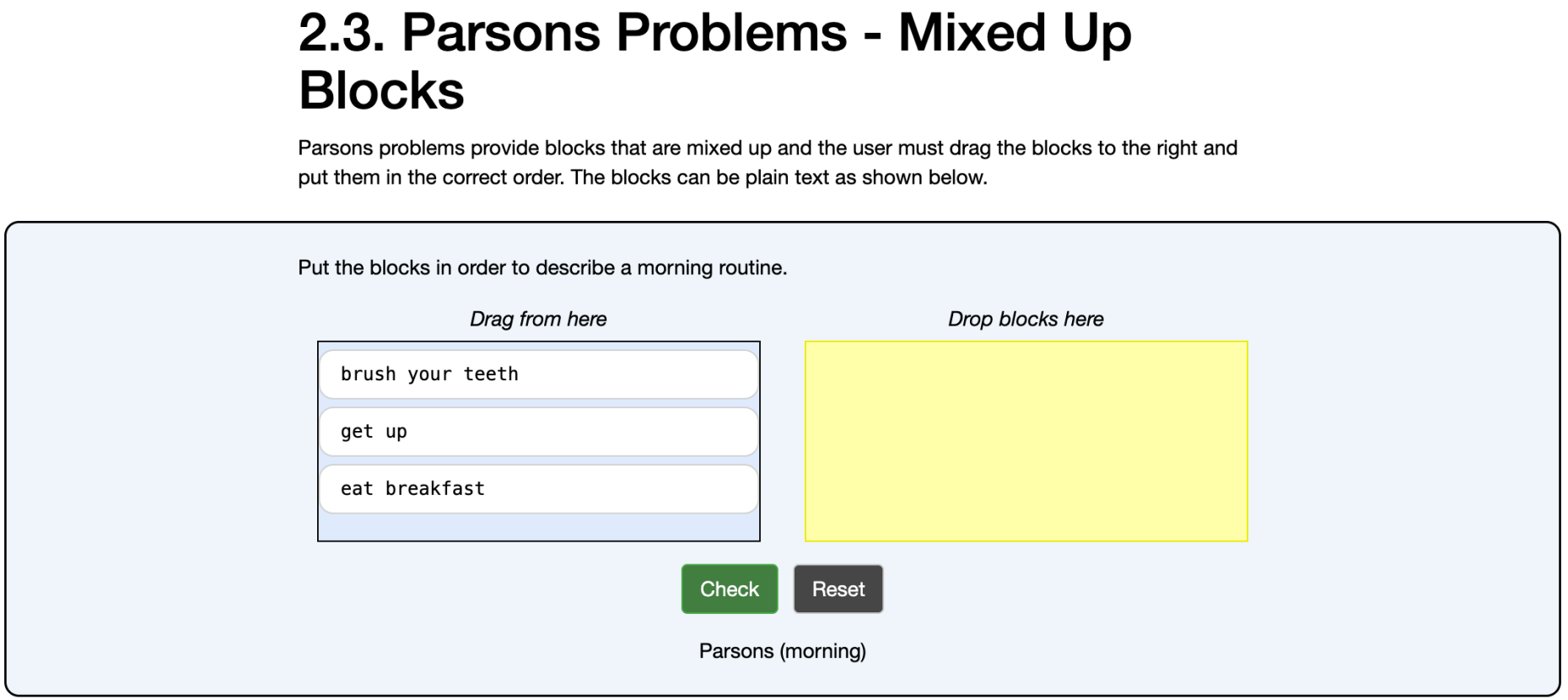
```
REPEAT 3 TIMES
    forward(100)
    left(120)
ENDREPEAT
```

Your Code Drawing

Reset | Feedback

https:// js-parsons.github.io/

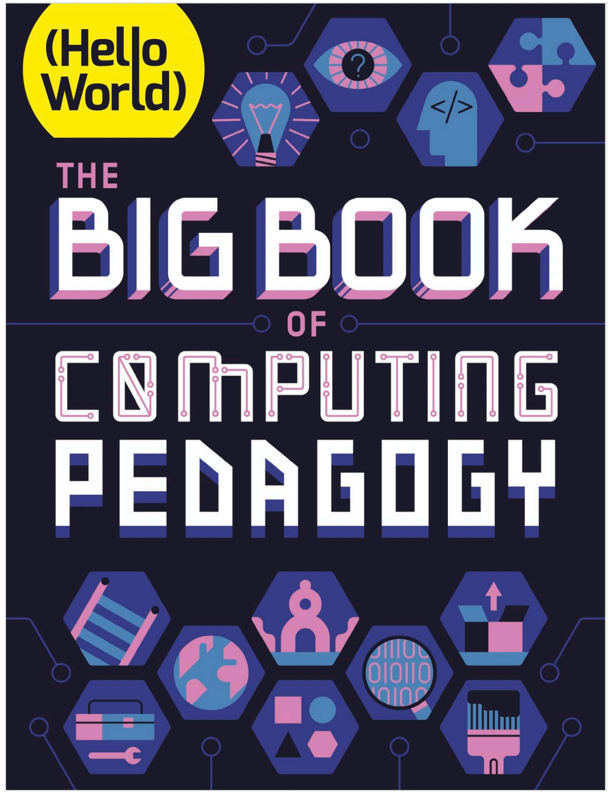
PARSONS PUZZLE



https:// runestone.academy/ns/books/published/overview/Assessments/__parsons.html

PARSONS PUZZLE

TheBigBook.pdf side 80 – 81



RESEARCH

PARSON'S PROBLEMS

Educators can foster program comprehension by using Parson's Problems to reorganise jumbled lines of code

An important precursor to learning how to write computer programs is having the necessary program comprehension to interpret the function and structure of existing programs. One tool that can help learners develop program comprehension is Parson's Problems. Here, we explore Parson's Problems, their benefits, the concept of distractors, and how to write a Parson's Problem task.

SUMMARY

Parson's Problems support learners by:
- Developing their understanding of how the program is executed (their 'notional machine' - see page 69)
- Reducing cognitive load
- Focusing on blocks of code, rather than on syntax
- Providing all the correct code in an engaging challenge
- Promoting dialogue and discussion about code

Benefits of Parson's Problems:
- Constrain the logic
- Avoid common syntax errors that can be barriers to learning to code
- Model good programming practices
- Provide the potential for immediate feedback
- Make it easier to identify common misconceptions
- Increase engagement of learners

Advice for writing Parson's Problems:
- Share problems with only a single solution
- Allow learners to manipulate actual code blocks
- Provide a clear description of the problem
- Clearly show the desired logic
- Share multiple similar problems over time

What is a Parson's Problem?

A Parson's Problem is a task in which learners are given all of the blocks or lines of code needed to solve a problem. The lines of code, though, have been jumbled so that they are no longer in the correct order. Learners are asked to reorganise the code into the correct order to perform a specific task.

The short example in **Figure 1** shows some jumbled lines of code (in Python and Scratch), and sets out the task that needs to be completed. Why not see if you can solve the problems in the example?

Parson's Problems can be applied to both text- and block-based programming and can vary in difficulty, to accommodate learners' existing understanding. For example, when you feel that learners are ready, you could provide them with lines of code and ask them to work out the indentation themselves (known as 2D Parson's Problems).

There are many ways in which Parson's Problems can be presented to learners. They make for excellent offline or paper-based activities that could be done individually, in pairs, or in small groups. You may choose to create problems directly in the development environment, to allow learners to immediately test their solutions. Alternatively, there are online tools such as js-parsons that allow you to create your own interactive problems (**helloworld.cc/jsparsons**).

Parson's Problems can be used to support formative assessment, as classroom discussion following the activity plays an important part in learners' development. Immediate feedback also avoids any misconceptions being committed to long-term memory.

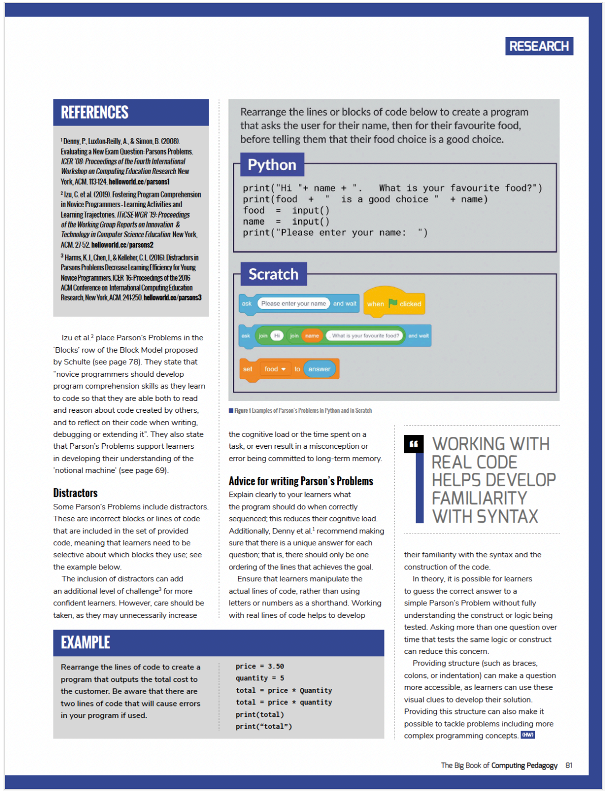
Benefits

The main benefit of Parson's Problems is that the learner is focusing on the structure and logic of blocks of code, rather than on the syntax of individual text elements (the 'atoms' of the Block Model). The process reduces the cognitive load experienced by learners, allowing them to practise sequencing and problem-solving with code. This experience is particularly helpful in the early stages of learning to program, when learners may be easily frustrated and put off by repeated unsuccessful attempts to solve a problem. Parson's Problems also expose learners to logic and syntax that they may not be fully familiar with.

Denny et al.[1] suggest that learners' solutions to a Parson's Problem "make clear what students don't know (specifically in both syntax and logic)". These solutions can allow for an easier analysis of the common errors that learners make, whereas "the open-ended nature of code-writing questions makes identifying such errors difficult". For example, when using a Parson's Problem, we can be sure that an error was not caused by a typing mistake.

Parson's Problems can promote some higher-order thinking in learners than simple code tracing (reading code and identifying its purpose or output). Parson's Problems can act as a stepping stone between the lowest and highest categories — being able to read and interpret code, and being able to write original code, which involves evaluation and creation (the highest categories in Bloom's taxonomy).

80 The Big Book of Computing Pedagogy



RESEARCH

REFERENCES

[1] Denny, P., Luxton-Reilly, A., & Simon, B. (2008). Evaluating a New Exam Question: Parsons Problems. *ICER '08: Proceedings of the Fourth International Workshop on Computing Education Research.* New York, ACM. 113-124. **helloworld.cc/parsons1**

[2] Izu, C. et al. (2019). Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories. *ITiCSE-WGR '19: Proceedings of the Working Group Reports on Innovation & Technology in Computer Science Education.* New York, ACM. 27-52. **helloworld.cc/parsons2**

[3] Harms, K.J., Chen, J., & Kelleher, C.L. (2016). Distractors in Parsons Problems Decrease Learning Efficiency for Young Novice Programmers. ICER '16: Proceedings of the 2016 ACM Conference on International Computing Education Research, New York, ACM. 241-250. **helloworld.cc/parsons3**

Rearrange the lines or blocks of code below to create a program that asks the user for their name, then for their favourite food, before telling them that their food choice is a good choice.

Python

```
print("Hi "+ name + ".   What is your favourite food?")
print(food  +  "  is a good choice "  + name)
food  =  input()
name  =  input()
print("Please enter your name:  ")
```

Scratch

ask [Please enter your name] and wait
when [flag] clicked
ask (join (Hi) (join (name) (. What is your favourite food?))) and wait
set [food] to (answer)

Figure 1 Examples of Parson's Problems in Python and in Scratch

Izu et al.[2] place Parson's Problems in the 'Blocks' row of the Block Model proposed by Schulte (see page 78). They state that "novice programmers should develop program comprehension skills as they learn to code so that they are able both to read and reason about code created by others, and to reflect on their code when writing, debugging or extending it". They also state that Parson's Problems support learners in developing their understanding of the 'notional machine' (see page 69).

Distractors

Some Parson's Problems include distractors. These are incorrect blocks or lines of code that are included in the set of provided code, meaning that learners need to be selective about which blocks they use; see the example below.

The inclusion of distractors can add an additional level of challenge[3] for more confident learners. However, care should be taken, as they may unnecessarily increase the cognitive load or the time spent on a task, or even result in a misconception or error being committed to long-term memory.

Advice for writing Parson's Problems

Explain clearly to your learners what the program should do when correctly sequenced; this reduces their cognitive load. Additionally, Denny et al.[1] recommend making sure that there is a unique answer for each question; that is, there should only be one ordering of the lines that achieves the goal.

Ensure that learners manipulate the actual lines of code, rather than using letters or numbers as a shorthand. Working with real lines of code helps to develop their familiarity with the syntax and the construction of the code.

> WORKING WITH REAL CODE HELPS DEVELOP FAMILIARITY WITH SYNTAX

In theory, it is possible for learners to guess the correct answer to a simple Parson's Problem without fully understanding the construct or logic being tested. Asking more than one question over time that tests the same logic or construct can reduce this concern.

Providing structure (such as braces, colons, or indentation) can make a question more accessible, as learners can use these visual clues to develop their solution. Providing this structure can also make it possible to tackle problems including more complex programming concepts. (HW)

EXAMPLE

Rearrange the lines of code to create a program that outputs the total cost to the customer. Be aware that there are two lines of code that will cause errors in your program if used.

```
price = 3.50
quantity = 5
total = price * Quantity
total = price * quantity
print(total)
print("total")
```

The Big Book of Computing Pedagogy 81