National Centre for Computing Education

Raspberry Pi

Pedagogy Quick Reads

Using worked examples supports novices to develop their programming practice

Worked examples demonstrate an 'expert' solution to a problem and are used in many subjects to support novices, who use the examples as blueprints for solving new but related problems. Learners who encounter worked examples in conjunction with practice problems are more likely to develop and assimilate strategies for solving similar problems.[1]

Worked example (partial)

```
#Use a turtle object and a finite loop to
#draw a square with two sides 100 pixels long.

#Add required functions
from turtle import Turtle

# Initialise a turtle object
t = Turtle()

# Move/turn once per side
for side in range (__): # <- What value here?
    t.forward(__)
    t.right(90)
```

- Integrated instructions
- Sub-goals labels highlighting stages in the solution
- Comments used as questions and reflection prompts
- Incomplete elements for learners to resolve

Write a program that uses a loop to draw another regular polygon.

Make a program that prints the message "Hip Hip Hooray!" three times.

- Possible problems focussed on finite iteration

Product-oriented worked example written in Python

Reducing cognitive load

Worked examples help reduce the extraneous cognitive load placed on a learner's working memory by providing a model solution for a problem, which the learner can read, understand, and adapt to solve similar problems.

When a learner is given a (partial or complete) solution, they do not need to recall as much from their long-term memory.[2] If a concept is included in the solution, the learner can quickly retrieve and apply their existing understanding relating to that concept. Partially complete problems help focus learners on particular concepts, because learners only need to focus on the missing aspects.

In focussing learners' attention on structural elements of problems, worked examples support students to organise their new knowledge into schemas.[3]

Summary

Well-designed worked examples:

- Help reduce extraneous cognitive load on learners
- Aid learners in assimilating new knowledge into their existing understanding
- Are especially useful for novices during the early stages of learning

Good worked examples:

- Include sub-goal labelling to highlight structure and common programming 'patterns'
- Present relevant information in an integrated manner
- Combine multiple modes of delivery, such as visual and aural explanations
- May only be partial and require learners to complete them as part of exploration

In a learning sequence:

- Combine worked examples with similar practice problems
- Alternate worked examples and practice problems to keep the example in mind
- Use a least two examples for each concept or 'pattern' explored
- Fade the use of worked examples over time
- Focus on examples that emphasise program structure over surface details

Illustrating process:

- Educators should explicitly model their approach to solving a problem
- Process-oriented worked examples emphasise **how** a solution was reached
- Product-oriented worked examples provide a possible solution

Product versus process

There are typically two types of worked examples[3] found in literature. Both support the learner by modelling solutions to problems.

- **Process-oriented examples** model the steps taken to reach a particular solution. They may be **written down**, demonstrated by an expert, or **captured on video**.
- **Product-oriented examples** model one possible solution and allow learners to examine and apply the solution to a new context.

There is evidence that complete novices benefit from process-oriented worked examples, as they provide rationale for each aspect of the solution. Learners with some experience will then benefit more from product-oriented examples, from which they can infer the rationale.[4]

Designing worked examples

When designing worked examples, educators should consider the following features that may affect learners' cognitive load and ability to follow an example.

The **split attention effect** occurs when information about a problem or example is presented separately. In order to follow the example or solve the problem, learners must first combine the separate sources of information in working memory. Where possible, educators should integrate all of the information into one clear representation.

Similarly, the **redundancy effect** occurs when information is duplicated within a problem unnecessarily, or other redundant information is included in a problem. The learner may still process the redundant information when they try to understand the problem, which results in an unnecessary cognitive burden.

Take advantage of the **multimodal effect** by presenting key information both visually and aurally, as the brain will process these separately. Studies have shown that presenting the same information — and more specifically, worked examples — in another mode, either simultaneously or sequentially, can support learners in their comprehension, and therefore, their ability to solve future problems.

It is broadly accepted that novices (in many fields) tend to focus on the context of a problem and the surface details, rather than the underlying structure and common elements to solutions. Educators can use **sub-goal labelling** to identify the important components or steps in a solution and highlight them to the learner. To do this, educators could use explanatory comments or annotations, visual labels or highlights, or white space to group related instructions into 'chunks'. [1]

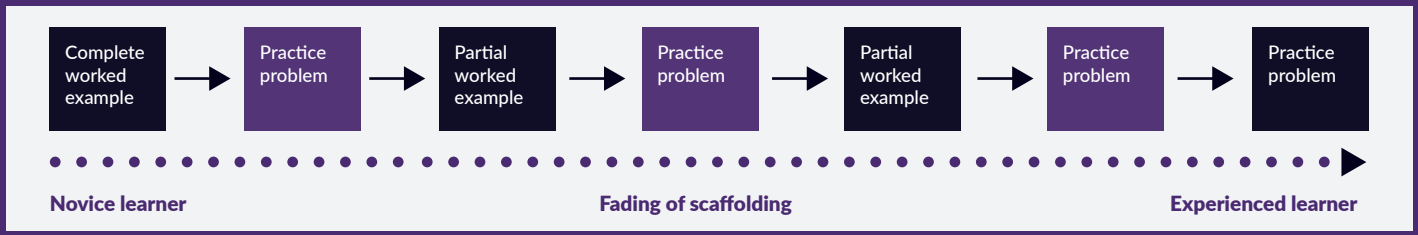
Integrating worked examples within a learning sequence

Educators should also consider how to combine worked examples with practice problems and other worked examples.

It is important to consider **variety**: presenting the same concept or programming pattern across multiple examples and problems within varied contexts. This variety helps learners to focus on structural connections between the solutions, and therefore, focus on the general concept, rather than the surface details of the problem.

Research suggests that the more worked examples learners experience, the more they benefit, and that learners should be exposed to **at least two worked examples** for each concept or pattern. Also, some studies suggest that learners should be presented with **example and practice pairs**,[5] which require them to understand an example, then apply it in practice. Alternatively, learners could review one example problem, then complete several practice problems, which would require them to hold the example in working memory for longer.

Worked examples are highly beneficial for **novices**, because they support them to build patterns for programs and procedures. However, as learners develop their expertise, they benefit more from solving new problems than from working from examples. Educators should use worked examples while a concept is new, then **fade** this support over time.[3]



References

1 Atkinson, R. K., Derry, S. J., Renkl, A. & Wortham, D. (2000) Learning from Examples: Instructional Principles from the Worked Examples Research. *Review of Educational Research*. 70 (2), 181–214.

2 Sweller, J., Ayres, P. & Kalyuga, S. (2011) *Cognitive Load Theory*. New York, Springer.

3 Sweller, J., van Merriënboer, J. J. G. & Paas, F. (2019) Cognitive Architecture and Instructional Design: 20 Years Later. *Educational Psychology Review*. 31 (2), 261–292. Available from: doi.org/10.1007/s10648-019-09465-5.

4 van Gog, T., Paas, F. & van Merriënboer, J.J. (2008) Effects of Studying Sequences of Process-Oriented and Product-Oriented Worked Examples on Troubleshooting Transfer Efficiency. *Learning and Instruction*. 18(3), 211–222.

5 Abdul-Rahman, S. S. and du Boulay, B. (2010) Learning Programming via Worked-examples. In: *Proceedings of PPIG-WIP 2010, 7–8 January 2010, Dundee*.

Raspberry Pi