TITLE: Find and correct mistakes and make your program work

LEARNING SCENARIO		
School:	***Duration (minutes):***	**90**
Teacher:	***Students age:***	**12**

Essential Idea:	**Most errors in writing a program come down to synthax.**

Topics:

- Pupils plan, anticipate, monitor, create and adjust programs.

Aims:

- Pupils design and create a working animation or program for a specific purpose.

Outcomes:

- Pupils plan, anticipate, monitor, create and adjust programs.

Work forms:

- individual work
- work in pairs
- group work

Methods:

- presentation
- discussion
- interactive exercise

ARTICULATION
Course of action (duration, minutes)
INTRODUCTION Teacher explains and starts discussion with pupils: Writing our first programs and having them do what we want to do is a lot of fun, but it's also frustrating to have errors that disable our program from working. This is especially frustrating if it's an error we can't quickly find so we waste our time which we could be spending by doing something else. Most errors in writing a program come down to synthax – we usually forget a bracket or an apostrophe, or maybe a block is not well atached.
MAIN PART **<u>Topics for discussion</u>** There are ways of avoiding errors in writing your programs, and some of them are: - Make sure your code is readable. It's unimportant what programming language you're using – whether it's blocks in something like Scratch or code in something like Python – it is very important that your code can be easily read by other people. And yes, that also goes for programs which won't be seen by anybody but yourself. Easily readable code means that the blocks are neatly and logically sorted, or the the text is written with clear spacings between lines when that is a logical thing to do. If your program has multiple parts it always needs to be clear which part does what. - Use comments whenever possible. Imagine you're a person who's seeing your code for the first time. If there's a part of the code that won't be instantly understandable to that person (assuming they know the language you're using) – you should use the option of writing comments which will explain what that part of the code is doing. - Read other peoples' programs. If you want the become a better photographer you should look at photographs the professionals are making, and if you want to become a better basketball player you should watch NBA games. It's always a good rule to learn from the professionals to become better at something, and programming is no exception.

Go online and see how others are writing their code in the same programming language you're using.

If you went through all of the steps above and you're still getting an error message, you could try one of the following:

- Look at what your program is telling you the mistake is.
Whatever you're using to write your program – it will almost always tell you where your mistake is. This can meand that it will graphically mark that part of the program, or it will tell you in what line of code the mistake has happened, or it will tell which command or variable was problematic.
Before you change anything else – concentrate on changing the thing your program has told you you should change

- Use Google.
If you can't find the error on your own, it's very possible that it's a very typical error and that you're not the first person in the world who has encountered it. An easy Google search for the type of error you're getting can often mean that you'll find a text where someone is explaining how they have struggled with the same error for hours, and more importantly how they have solved it.

- Ask somebody else to take a look at your program.
Programmers often get so „into“ their programs that they can't see the forest from all of the trees. Sometimes we come upon an error we would usually spot very easily, but that is hard to do after you've spent hours programming. In moments like these it's useful to ask somebody else to take a look at our program.

- Take a break.
If nothing else is working – quit programming, but just for a while.
Move away from the computer, eat something, watch an episode of your favourite show, go outside for a walk or go and take a nap.
The error will still be there when you come back, but your brain will be working in the background looking for a solution and it's very possible it will come up with an idea you have not yet tried.

Exercise 1
Teacher explains and give instructions how to solve tasks.
Pupils create a simple animation in Scratch.
They apply previously acquired knowledge and skills.
Pupils in their work deliberately create an error as a trap.

Pupils share their works and discover mistakes in other pupils 'programs.
Pupils solve tasks and present their solutions.

CONCLUSION

Pupils and teacher discuss and evaluate the presented solutions.

Methods		***Work forms***
presentation	***interview***	***individual work***
discussion	***demonstration***	***work in pairs***
work on the text	***role playing***	***group work***
graphic work		***frontal work***
interactive exercise /simulation on the computer		

Material:

-

Literature

PERSONAL OBSERVATIONS, COMMENTS AND NOTES