

- Sabotage: Teaching Debugging

When children first make the transition from using Scratch (a visual programming language) to Python (a text-based programming language), they can often become frustrated with the high numbers of syntax errors which prevent their scripts from working. This then creates negative associations with text-based programming, which can lead to pupils becoming disinterested with this part of the curriculum.


In order to address this frustration, and encourage children to identify exactly what was causing the error rather than waiting for the teacher to do so for them, we have designed Sabotage - a game which makes debugging much more enjoyable!

This activity should develop:

- **Analytical skills:** Pupils become more adept at analysing code, and assessing what makes it work
- **Practical problem solving:** Learners work together to make the changes that they suggest.
- **Teamwork:** Through working in pairs/teams to identify and resolve problems, pupils are encouraged to consider using their peers for assistance more often.

- Steps for the Activity

- 1) Everybody starts with a script that definitely works, e.g. an example from a book or their own coded solution to a problem. The 'Guess the Number Game' is ideal for this and can be found at www.exa.is/guessthenumber



```
print('hello! what is your name?')
myName = input()

number = random.randint(1, 20)
print('well, ' + myName + ', I am thinking of a number between 1 and 20.')

while guessesTaken < 6:
    print('Take a guess.') # There are four spaces in front of print.
    guess = input()
    guess = int(guess)
    guessesTaken = guessesTaken + 1

    if guess < number:
        print('Your guess is too low.') # There are eight spaces in front of print.
    if guess > number:
        print('Your guess is too high.')
    if guess == number:
        break

if guess == number:
    guessesTaken = str(guessesTaken)
    print('Good job, ' + myName + '! You guessed my number in ' + guessesTaken + ' guesses!')
```

- 2) We then test the game, just to make sure that we know it definitely works.
- 3) The teacher asks everyone in the class to think of examples of syntax errors that might prevent the game from working properly. They then share as many of these as they can with their partner in just one minute. Then, forming a group of 4, they see if they can think of any more examples.



- 4) The teacher then compiles a list of generic examples by asking each group to suggest one typical error.
- 5) The teacher then explains to the class that they are going to deliberately

This resource is also available at www.exa.foundation/resources

sabotage the working code by hiding 5 errors in there. They don't have to create 5 different types, but there must be 5 errors in total, no more, no less. It can be helpful to suggest that one should be pretty obvious and one should be very sneaky, e.g. using the character zero instead of the letter 'o'.

- **6)** The teacher asks the pupils to swap with their partner and take it in turns to debug the errors. The debugger should be coached by the saboteur, as the aim is for the saboteur to tie hints and clues that eventually enable the debugger to spot all 5 errors. They then swap over.
- **7)** Finally, ask for feedback. Who scored 5 out of 5? Who put really sneaky errors in? What were they? Who gets the prize for being the most helpful? Who gets the prize for being the most devious?