In [6]:

```
from sympy.matrices import Matrix
import sympy as sp
import numpy as np
from Exercise import Exercise, MarkdownBlock

try:
    from config import URL, TOKEN
except:
    None

# TODO: replace with supplied strings
Exercise.URL = URL
Exercise.TOKEN = TOKEN
```

Introduction

In this notebook, you are about to create some (linear algebra) exercises using the developed `Exercise` Python library aiming to facilitate authoring parameterized mathematics exercises at a high level of abstraction (i.e. access to a scripting language and the libraries available in there, including as SymPy, NumPy and Matplotlib). Created exercises can be 'played' inline, using the web-based player developed as part of this project. Roughly speaking this project is new combination of existing approaches: MEGUA-like parameterized text, SymPy's CAS functionality and exercise-setup as used by Grasple and SageMath for working with mathematical objects in notebooks.

The goal is to evaluate the usability of the developed library and the authoring setup (this notebook). Note that by no means you or your skills are being tested, it is by no means a problem if exercises are left uncompleted. Notes, comments and suggestions are very welcome, please write these either as code-comments or in the Markdown cells in the notebook. All feedback will be reported and reflected upon anonymously. Completing the notebook should take about 30 minutes, depending on setup time, prior knowledge about this project, familiarity with linear algebra and the supplied frameworks etc. Please download the notebook when done and send it by email. After completion, in a brief semi-structured interview, you can further elaborate upon your experiences.

To start creating exercises, please replace the `URL` and `TOKEN` in the block above with the strings supplied by email:

```
Exercise.URL = "<supplied_url_here>"
Exercise.TOKEN = "<supplied_token_here>"
```

Assumptions:

- Familiarity with Python, Markdown, LaTeX
- Familiarity with Jupyter-Notebook
- Familiarity with the very basics of linear algebra

Recommendations:

- Use Binder (www.mybinder.org) to edit this notebook, if you prefer local setup instead, see README.md.
- Use Firefox, the iFrame exercise player embeddings do not work in Chrome or Safari due to global cross-origin policies set by these browsers.
- Other browsers (Chrome, Safari) can still be used, however, playing exercises is only possible outside of the notebook by clicking the generated exercise links, which is rather inconvenient.

Notes:

- Documentation can for the Python library can be found in the `html` directory.
- Within Jupyter-Notebook, function documentation can be viewed by writing a `?` after the function, like so: `Exercise("What is $1 + 1$?").add_answer?`
- Within exercises, only inline math notation is supported.
- Preview-exercises are purged from the server from time to time, don't expect long-term, persistent availability of any played exercises.
- Please skip an exercise in case completing it requires more than a few minutes.

Happy coding ;)

Exercise Basics

The most basic exercise contains a Markdown string with the exercise content and a single answer rule specifying the correct answer. Mathematics notation can be written inline in LaTeX between dollar signs.

In [2]:

```
# Create an exercise instance
e = Exercise("What is $1 + 1$?")
# Add 2 as a correct answer
e.add_answer(2, True, "Correct!")
```

```
# Verify that the exercise is working correctly
e.play()
# Note: as of now, all basic arithmatic is simplified by sp.simplify(...), there is not yet a way to con
# therefore writing 1 + 1 in the answer box is accepted correct
# Details on what is simplified: https://docs.sympy.org/latest/tutorial/simplification.html
```

Check

Toggle

```
Published succesfully, preview at: https://www.mscthesis.nl/preview?id=adf3222c-f2f0-423f-9a86-
f89d19bdc3c8
```

Let's imagine the typical student mistake for this exercise is computing $1 - 1 = 0$ instead. We add an answer rule to catch that error and provide the student with answer-specific feedback.

In [3]:

```
e.add_answer(0, False, "▯▯ That's not right, did you compute $1 - 1 = 0$ instead?")
# Verify that the specific feedback is shown
e.play()
```

Check

Toggle

```
Published succesfully, preview at: https://www.mscthesis.nl/preview?id=9928eaed-6b16-4002-a5d5-
58bce474db14
```

Task 1

Create an exercise asking learners to compute $3/3$. Provide answer-specific feedback in case learners compute $3*3$ instead. Add default feedback (using `e.add_default_feedback(...)`) with a link pointing to a source of preference explaining (integer) devision (hint: `[link](www.example.com)`). Feel free to embed your favorite meme or xkcd at a correct/incorrect answer (hint `![img](www.example.com/img)`).

In [4]:

```
# Task 1 user code:
```

Templating Exercises

Exercises can be parameterized/templated (still looking for the correct terminology on this one), this allows for two things:

1. Randomization. By making part of the content random, multiple instances can be generated, allowing for repeated practice.
2. Abstraction. By utilizing the functionality of SymPy objects to be translated to LaTeX, authoring exercises remains efficient and effective.

The integer-exercise can be randomized as follows:

In [4]:

```
string = """
### Integer addition

Please compute $@a + @b$
"""

params = {}
# avoid 0 + 0 instance, since 0 + 0 == 0 - 0, answer same in case our typical mistake is made
params["a"] = np.random.randint(0, 10)
params["b"] = np.random.randint(1, 10)
params["ans_correct"] = params["a"] + params["b"]
params["ans_incorrect"] = params["a"] - params["b"]

e = Exercise(MarkdownBlock(string, params))
e.add_answer(params["ans_correct"], True, "Correct!")
e.add_answer(params["ans_incorrect"], False, MarkdownBlock("Did you compute $@a - @b = @ans_incorrect$ i

e.play()
```

Check

Toggle

```
Published succesfully, preview at: https://www.mscthesis.nl/preview?id=8b88dd0c-1aa2-4db0-aabc-
b147c767c419
```

Currently, only a single instance is generated played at a time. Support for multi-instance generation is planned.

Working with SymPy objects to represent mathematical objects

We can work with SymPy objects to represent mathematical objects, like vectors and matrices. An vector addition exercise can be created as follows:

In [7]:

```
string = "What is $@v_1 + @v_2$?"

params["v_1"] = sp.Matrix([1, 2, 3])
params["v_2"] = sp.Matrix([4, 5, 6])
params["ans"] = params["v_1"] + params["v_2"]

e = Exercise(MarkdownBlock(string, params))
e.add_answer(params["ans"], True, "That's right!")

e.play()
```

Check

Toggle

```
Published succesfully, preview at: https://www.mscthesis.nl/preview?id=ede21274-5cc5-4113-87a0-
d509c3158854
```

Task 2 Parameterized vector addition

Create an exercise asking learners to compute the sum of two vectors of random length (within reasonable limits), with random integer values. Note: if you prefer NumPy for working with matrices, you are in luck! NumPy objects can be passed to the SymPy matrix constructor, e.g. `sp.Matrix(np.arange(4))` .

In [9]:

```
# Task 2 user code:
```

Task 3 - Matrix indexing

Create an exercise asking learners to identify a value at randomized indices (but within bounds) in a 5 by 5 matrix. Please make sure all values are unique so there is only one correct answer.

In [10]:

```
# Task 3 user code:
```

Task 4 - Matrix multiplication

Create an exercise asking users to multiply two matrices. Provide a default answer explaining the procedure in case a wrong answer is supplied. You can use the `symbolic_matrix` and `explain_multiply` functions supplied in `helpers.py` as follows:

In [16]:

```
from helpers import symbolic_matrix, explain_multiply
a = symbolic_matrix("a", 2, 2)
b = symbolic_matrix("b", 2, 2)
display(explain_multiply(a, b))

a = sp.Matrix([1,2,3])
b = sp.Matrix(np.matrix([5,6,7]).reshape(-1))
display(explain_multiply(a, b))
```

```
$\displaystyle \left[\begin{matrix}{a}_{1, 1} \cdot {b}_{1, 1} + {a}_{1, 2} \cdot {b}_{2, 1} & {a}_{1, 1} \cdot {b}_{1, 2} + {a}_{1, 2} \cdot {b}_{2, 2}\\{a}_{2, 1} \cdot {b}_{1, 1} + {a}_{2, 2} \cdot {b}_{2, 1} & {a}_{2, 1} \cdot {b}_{1, 2} + {a}_{2, 2} \cdot {b}_{2, 2}\end{matrix}\right]$

$\displaystyle \left[\begin{matrix}1 \cdot 5 & 1 \cdot 6 & 1 \cdot 7\\2 \cdot 5 & 2 \cdot 6 & 2 \cdot 7\\3 \cdot 5 & 3 \cdot 6 & 3 \cdot 7\end{matrix}\right]$
```

In [5]:

```
# Task 4 user code:
```

Hooray! If you made it this far, you completed the notebook! Please add any additonal comments below. Thank you for participating!

Write any additional comments here...

In [ ]:

Loading [MathJax]/extensions/Safe.js