Sudoku Solver

# Introduction

I have never been a fan of puzzles such as Sudoku. Those where it’s obvious that they could be solved by an algorithm and computer. With some available time at the beginning of 2023, I decided to investigate how to make a Sudoku Solver in Python.

After several furtive attempts, I searched online and found a great resource on the [Tech with Tim](https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking/) website which lays out the “backtracking” algorithm with simple efficient code.

This code is great for getting the solution. I pass the strongest Kudos to Mr. Tim the author and thank him for sharing it. However, here I saw an opportunity to extend this code, with extra functionality, commented code, and development features such as unit tests to create a showcase project for my developer skills.

## ObjectOriented

The first change is to enable scalability and flexibility by converting the function from a functional oriented paradigm to a project oriented paradigm.

## General Principles

To make code compliant with Pep-8 and the Zen of Python, I follow these general principles:

### Descriptive Variable names

Code is read more than it is written. The code below is easy to read, and understand its function, even if one is not familiar with python.

# Set flag for multiple workbooks read.  
sheets\_in\_workbook = len(re\_excel\_reader.sheet\_names)  
multiple\_sheets\_exist = sheets\_in\_workbook > 1  
read\_multiple = multiple\_sheets\_exist and load\_multiple\_sheets