PHASMG501 - Coursework 1
Submission Deadline: 17 February 2017.

The goal of this coursework is to write a solver for diffusion problems of the form

$$\begin{aligned} -\nabla \cdot [\sigma(x)\nabla u(x)] &= f(x), \text{ in } \Omega \\ u(x) &= 0, \text{ on } \partial\Omega \end{aligned}$$

Here, Ω is a given domain in two dimensions, f is a given function in Ω, σ is a scalar field that is defined over Ω and u is the wanted solution.
The solver should provide the following capabilities:

- Read in a given triangular grid in the legacy VTK 2.0 format.
- Define σ and f as Python functions in the form

```
def f(x):
...
    return result

def sigma(x):
....
    return result
```

- Provide a global routine solve of the form

```
def solve(vtk_file_name, f, sigma, integration_order=4):
...
```

  which performs the computation and stores the result in a vtk file `result.vtk`. Here, `vtk_file_name` is a string containing the name of a vtk file with the grid. `f` and `sigma` are Python functions as defined above, and `integration_order` is an integer with the order of the underlying quadrature rule. The implementation should assume low-order three noded triangles and simple nodal P1 basis functions.

  In order to perform this exercise you should first put some thought into the organization of your code. You will need routines for import/export, grid data structures, a description of the mapping from the reference element to the actual element, local element assembler, and a global assembly of the matrix. You should create corresponding classes, and split up the code into different modules where suitable.

  The linear system solve itself can be performed using the sparse direct solver routines provided in Scipy. These are sufficient for most 2d problems.

  You should submit your code as a zip or tar.gz archive containing a directory called 'source' and a file solution.ipynb. The directory 'source' should contain the code 'modules'. The file 'solution.ipynb' should contain detailed descriptions and a worked out example including graphical visualization.

The marking scheme for this exercise is as follows:

- The pure functionality makes up 60% of the coursework mark.
- Code quality is 20%. This includes PEP8 compliance, checked with the tool `pylint`. and structuring of the code into functions, classes and modules.
- Performance is 20%. All solutions will be benchmarked and the fastest solution will automatically receive full marks here. An anonymized benchmark list will be published. To speed up the code you will be allowed any Python module, including accelerators such as Cython, Numba and others. Explicitly writing C/C++ or Fortran code is not allowed.