What is it?

A tool to run the analysis of the metric data from the Excel table.

The goal of this part of the project is to provide a tool and a mathematical model for a multiple-criteria comparison of different programming languages.

How it works?

First things first: you need to install Python3 and the openpyxl package (to work with Excel tables).

Don’t forget to specify your Excel file name inside the script (will make the usage easier soon) before running the script.

To run the script:

```
$ python3 pl_compare.py
```

The result will be the list of scores for each metric for each language.

What’s inside?

`generate_pl()` function is used to create a new PL structure from the Excel table. It takes a new PL’s name and a letter which corresponds to the column in the Excel table.

Generated PLs should be used inside the metric functions (the script provides 2 examples: `calc_best_web()` and `calc_best_business()`) which take the relevant table data and run the metric calculations. The coefficients are separated to be loaded from another file (pre-optimized) in the future.

As a baseline metric we decided to pick relevant features of programming languages for developing web applications (Built-in exception handling support, Model of resolving conflict related to conflicting versions, Start Up (Launch) time for GUI becnhmarks etc) and for developing applications with complex business logic (Kind of typification , Support for immutable objects, Ability to change the routine signature while inheriting etc).

Here by web applications we mean a typical client-server program, for example a simple retail store, and by a complex business logic application we understand, for example, a modern CMS system supporting hundreds of entity types with different inheritance and architectural patterns.

Keep in mind that the provided examples are just our first assumptions and the user of the tool can specify any number of features and provide a relevant metric function to create a new metric for comparison.

Let's take a closer look at one of the example functions to understand how to create a new one.

```
def calc_best_web(pl):
    coef_6 = 1.0
    coef_21 = 1.0
```

```
    coef_22 = 1.0
    coef_28 = 1.0
    coef_54 = 1.0
    coef_67 = 1.0
    coef_68 = 1.0
    coef_85 = 1.0
    coef_86 = 1.0
    coef_88 = 1.0
    coef_89 = 1.0
    coef_90 = 1.0
    coef_91 = 1.0
    return (coef_6 * pl[("%s6" % pl["letter"])] + coef_21 * pl[("%s21" %
pl["letter"])]
          + coef_22 * pl[("%s22" % pl["letter"])] + coef_28 * pl[("%s28" %
pl["letter"])]
          + coef_54 * pl[("%s54" % pl["letter"])] + coef_67 * pl[("%s67" %
pl["letter"])]
          + coef_68 * pl[("%s68" % pl["letter"])] + coef_85 * pl[("%s85" %
pl["letter"])]
          + coef_86 * pl[("%s86" % pl["letter"])] + coef_88 * pl[("%s88" %
pl["letter"])]
          + coef_89 * pl[("%s89" % pl["letter"])] + coef_90 * pl[("%s90" %
pl["letter"])]
          + coef_91 * pl[("%s91" % pl["letter"])])
```

Here we use a generated PL structure inside a new metric function. The coeficients correspond to the relevant entries inside the Excel table. In our baseline assumption we just calculate the sum of scores for each feature.