Lab 1 Documentation

I used Python to implement the graph. I defined a class called Graph for the graph itself, a class UI for printing the menu and using the operations defined in graph and an error class for error handling. I also used 2 additional functions to save the graph to a file and one to load a graph from a file.

The class Graph has the following functions:

vertices_iterator(self):

Iterator for the vertices

neighbours_iterator(self, vertex)

Iterator for the neighbours of a vertex

```
edges_iterator(self)

Iterator for the edges
```

```
is_vertex(self, vertex)

Check if a vertex belongs to the graph
```

```
is_edge(self, vertex1, vertex2)

Check if the edge of vertex1 and vertex2 belongs to the graph
```

```
count_vertices(self)

return: Number of vertices in the graph
```

```
count_edges(self)

return: Number of edges in the graph
```

```
in_degree(self, vertex)

return: Number of edges with endpoint vertex
```

```
out_degree(self, vertex)

return: Number of edges with startpoint vertex
```

```
get_edge_cost(self, vertex1, vertex2)

return: The cost of an edge
```

```
set_edge_cost(self, vertex1, vertex2, new_cost)

Set the cost of an edge
```

```
add_vertex(self, vertex)

Add a vertex to the graph
```

```
add_edge(self, vertex1, vertex2, edge_cost = 0)

Add an edge to the graph
```

```
remove_edge(self, vertex1, vertex2)

Remove edge from the graph
```

```
remove_vertex(self, vertex)

Remove a vertex from the graph
```

```
copy(self)

return: Deepcopy of the graph
```

The graph is initialized with: self._vertices = the number of vertices of the graph (a set)

- Self._neighbours = number of neighbours of a vertex(a dict)
- Self._transpose = number of inbound neighbours of a vertex(a dict)

- Self._cost = cost of an edge (a dict)

Additional functions:

```
read_file(file_path)
```

Reads a graph from a file

```
write_file(file_path, g)
```

Saves a graph to a file

```
random_graph(vertices_no, edges_no)
```

Creates a random graph with a given number of vertices and edges