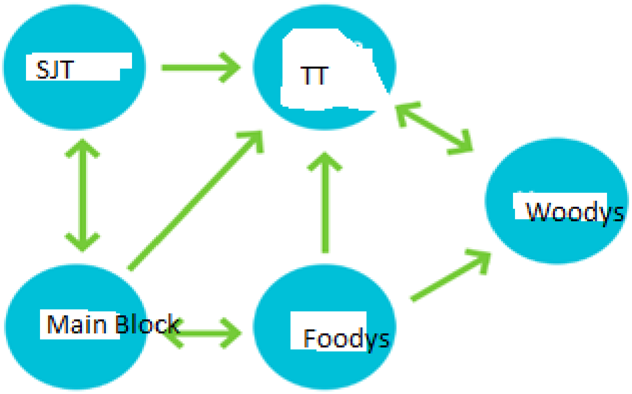
Lab DA-6, Winter 2020-21, L15+L16

WEB MINING

by Sharadindu Adhikari, 19BCE2105

PAGE RANKING ALGORITHM

Write a python program to find the ranks for the given graph. Use the damping factor as d = 0.85. Perform 7 iteration and print the final iteration value only.



- **Graph.py** (`gra.py`): **For constructing the graph as per the question.**

```
"""Graph class"""

class Graph(object):
    """
    Class that supports basic graph creation, manipulation, and analysis.

    Graphs support vertices of arbitrary (hashable) data types. Vertices are
stored as keys
    in a dictionary, whose value is a dictionary of vertices representing the
edges.
    Undirected graphs are represented as Graphs where every edge is bi-
directional.

    Parameters
    ----------
    graph_dict : dictionary
        Dictionary to initialize graph. If None (default) creates an empty
Graph.
    directed : boolean
        Boolean determining if graph is directed or undirected. Affects
verifying graph upon
        initialization, and adding of all edges will be done symmetrically.

    """

    def __init__(self, graph_dict=None, directed=True):
        if graph_dict:
            self.verify(graph_dict, directed)
        self.graph_dict = graph_dict or {}
        self.directed = directed
    def verify(self, graph_dict, directed):
        for vertex, edges in graph_dict.items():
            for edge in edges:
                if edge not in graph_dict.keys():
                    raise ValueError("{} is part of an edge but not added as 
vertex".format(edge))
                if not directed:
                    if vertex not in graph_dict[edge].keys():
                        raise ValueError("Edge ({}, {}) is unidirectional, this 
is not a valid undirected graph".format(vertex, edge))

    def add_vertex(self, v):
        self.graph_dict[v] = {}

    def add_edge(self, source, dest):
        # If vertices don't exist, add to graph
        if source not in self.graph_dict:
            self.graph_dict[source] = {}
        if dest not in self.graph_dict:
            self.graph_dict[dest] = {}
        # In future, can store edge attributes in {}
        self.graph_dict[source][dest] = {}

        if not self.directed:
            self.graph_dict[dest][source] = {}

    def remove_vertex(self, v):
        if v in self.graph_dict:
            del self.graph_dict[v]
        else:
            raise KeyError("Vertex {} is not in the graph".format(v))
        for e in self.graph_dict.values():
            if v in e:
                del e[v]

    def remove_edge(self, source, dest):
        if source in self.graph_dict and dest in self.graph_dict:
            if dest in self.graph_dict[source]:
                del self.graph_dict[source][dest]
            else:
                raise KeyError("Edge ({}, {}) is not in the 
graph".format(source, dest))
            # If undirected graph, delete edges in both directions
            if not self.directed:
                del self.graph_dict[dest][source]
        else:
            raise KeyError("Vertices ({}, {}) don't both exist in the 
graph".format(source, dest))

    def number_of_vertices(self):
        return len(self.graph_dict)

    def incoming_vertices(self, vertex):
        result = []
        for v, e in self.graph_dict.items():
            if vertex in e:
                result.append(v)
        return result

    def in_degree(self, v):
        return len(self.incoming_vertices(v))

    def out_degree(self, v):
        return len(self.graph_dict[v])
```

- **Pagerank.py** (`page.py`): **For PageRank calculation.**

```
def pagerank(graph, iterations=7, d=0.85):
    """ Calculate PageRank of vertices in a graph

    Paramters
    ----------
    graph : Graph
        Graph object on which to perform PageRank analysis
    iterations : int
        Number of iterations in PageRank calculation
    d : float
        Dampening factor in PageRank algorithm

    Returns
    -------
    pagerank: dictionary
        Dictionary of vertices with PageRank values

    """

    num_v = graph.number_of_vertices()
    # Initialize ranks to 1/N
    ranks = dict.fromkeys(graph.graph_dict, 1.0/float(num_v))
    for _ in range(iterations):
        for vertex, edges in graph.graph_dict.items():
            incoming = graph.incoming_vertices(vertex)
            weighted_ranks = [ranks[v]/len(graph.graph_dict[v]) for v in
incoming]
            ranks[vertex] = (1-d) + d*sum(weighted_ranks)
    print (ranks)
```

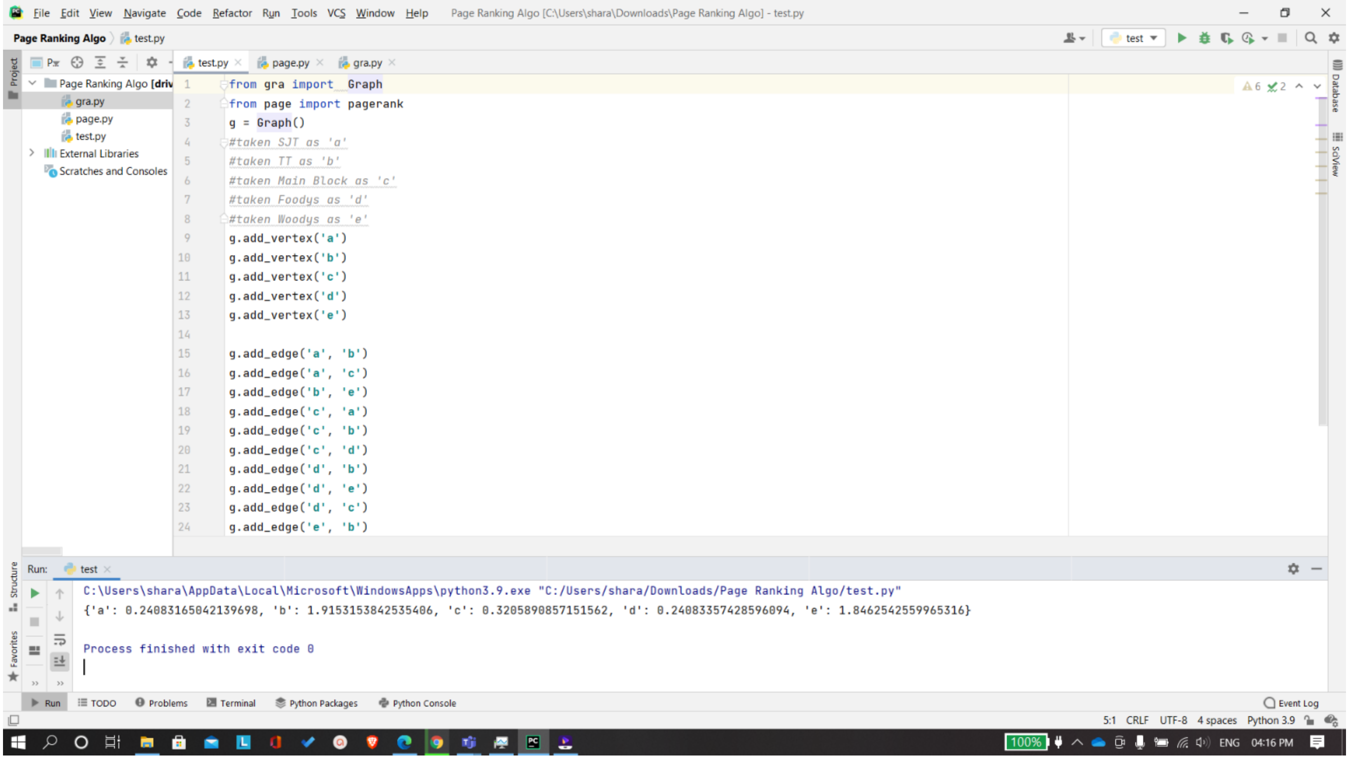
- **Test.py** (`test.py`): **Combining both** `gra.py` **and** `page.py` **to get the results.**

```
from gra import  Graph
from page import pagerank
g = Graph()
#taken SJT as 'a'
#taken TT as 'b'
#taken Main Block as 'c'
#taken Foodys as 'd'
#taken Woodys as 'e'
g.add_vertex('a')
g.add_vertex('b')
g.add_vertex('c')
g.add_vertex('d')
g.add_vertex('e')

g.add_edge('a', 'b')
g.add_edge('a', 'c')
g.add_edge('b', 'e')
g.add_edge('c', 'a')
g.add_edge('c', 'b')
g.add_edge('c', 'd')
g.add_edge('d', 'b')
g.add_edge('d', 'e')
g.add_edge('d', 'c')
g.add_edge('e', 'b')

ranks = pagerank(g)
```

OUTPUT:

```
{'a': 0.24083165042139698, 'b': 1.9153153842535406, 'c': 0.3205890857151562,
'd': 0.24083357428596094, 'e': 1.8462542559965316}
```

```
where,
#taken SJT as 'a'
#taken TT as 'b'
#taken Main Block as 'c'
#taken Foodys as 'd'
#taken Woodys as 'e'
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help
Page Ranking Algo [C:\Users\shara\Downloads\Page Ranking Algo] - test.py
Page Ranking Algo
test.py
page.py
gra.py
External Libraries
Scratches and Consoles
Run: test
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
5:1 CRLF UTF-8 4 spaces Python 3.9
ENG 04:16 PM

```python
from gra import  Graph
from page import pagerank
g = Graph()
#taken SJT as 'a'
#taken TT as 'b'
#taken Main Block as 'c'
#taken Foodys as 'd'
#taken Woodys as 'e'
g.add_vertex('a')
g.add_vertex('b')
g.add_vertex('c')
g.add_vertex('d')
g.add_vertex('e')

g.add_edge('a', 'b')
g.add_edge('a', 'c')
g.add_edge('b', 'e')
g.add_edge('c', 'a')
g.add_edge('c', 'b')
g.add_edge('c', 'd')
g.add_edge('d', 'b')
g.add_edge('d', 'e')
g.add_edge('d', 'c')
g.add_edge('e', 'b')
```

```
C:\Users\shara\AppData\Local\Microsoft\WindowsApps\python3.9.exe "C:/Users/shara/Downloads/Page Ranking Algo/test.py"
{'a': 0.24083165042139698, 'b': 1.9153153842535406, 'c': 0.3205890857151562, 'd': 0.24083357428596094, 'e': 1.8462542559965316}

Process finished with exit code 0
```