código a revisar

```
""" A Python Class
A simple Python graph class, demonstrating the essential
facts and functionalities of graphs.
"""


class Graph(object):

    def _init_(self, graph_dict=None):
        """ initializes a graph object
            If no dictionary or None is given,
            an empty dictionary will be used
        """
        if graph_dict == None:
            graph_dict = {}
        self.__graph_dict = graph_dict

    def vertices(self):
        """ returns the vertices of a graph """
        return list(self.__graph_dict.keys())

    def edges(self):
        """ returns the edges of a graph """
        return self.__generate_edges()

    def add_vertex(self, vertex):
        """ If the vertex "vertex" is not in
            self.__graph_dict, a key "vertex" with an empty
            list as a value is added to the dictionary.
            Otherwise nothing has to be done.
        """
        if vertex not in self.__graph_dict:
            self.__graph_dict[vertex] = []

    def add_edge(self, edge):
        """ assumes that edge is of type set, tuple or list;
            between two vertices can be multiple edges!
        """
        edge = set(edge)
        (vertex1, vertex2) = tuple(edge)
        if vertex1 in self.__graph_dict:
            self.__graph_dict[vertex1].append(vertex2)
```

```
        else:
            self.__graph_dict[vertex1] = [vertex2]

    def __generate_edges(self):
        """ A static method generating the edges of the
            graph "graph". Edges are represented as sets
            with one (a loop back to the vertex) or two
            vertices
        """
        edges = []
        for vertex in self.__graph_dict:
            for neighbour in self.__graph_dict[vertex]:
                if {neighbour, vertex} not in edges:
                    edges.append({vertex, neighbour})
        return edges

    def _str_(self):
        res = "vertices: "
        for k in self.__graph_dict:
            res += str(k) + " "
        res += "\nedges: "
        for edge in self.__generate_edges():
            res += str(edge) + " "
        return res


if _name_ == "_main_":

    g = { "1" : ["2"],
          "2" : ["4","6"],
          "3" : ["2", "1"],
          "4" : ["5","3"],
          "5" : ["2"],
          "6" : ["5","7"]
          "7" : ["5","4"],
        }


    graph = Graph(g)

    print("Vertices of graph:")
    print(graph.vertices())

    print("Edges of graph:")
```

```
print(graph.edges())

print("Add vertex:")
graph.add_vertex("z")

print("Vertices of graph:")
print(graph.vertices())

print("Add an edge:")
graph.add_edge({"a","z"})

print("Vertices of graph:")
print(graph.vertices())

print("Edges of graph:")
print(graph.edges())

print('Adding an edge {"x","y"} with new vertices:')
graph.add_edge({"x","y"})
print("Vertices of graph:")
print(graph.vertices())
print("Edges of graph:")
print(graph.edges())
```