Issue Report: 802

Issue Link : https://github.com/RDFLib/rdflib/issues/802

About the issue:

The issue is about the serialization of turtle files with Blank Nodes. Initially, the turtle file is having 12 triples after parsing into graphs using Graph.parse() method. When we serialize this graph into another graph of turtle format, the new graph is having 14 triples (2 triples extra).

Code :-

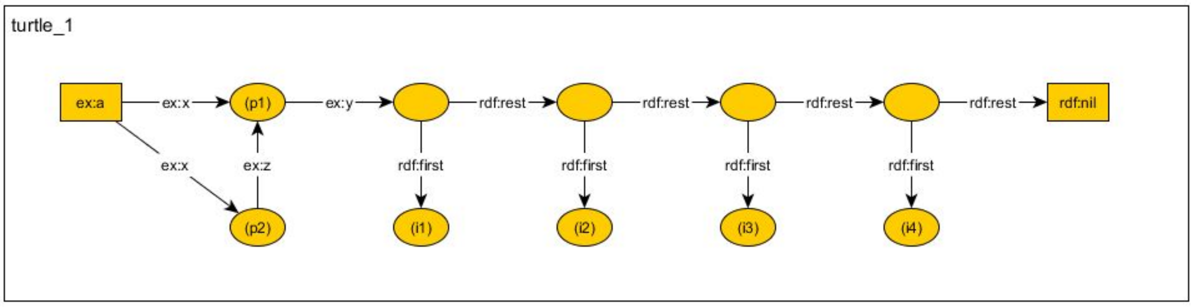
In turtle 1, we have a subject node as “ex:a” , predicate nodes as “ex:x” , “ex:y” and “ex:z” and all the other nodes are blank nodes . RDF graph for the code is given below :

```
# turtle_1

@prefix ex: <http://www.ex.org/> .

ex:a ex:x _:p1 , _:p2 .
_:p1 ex:y ( _:i1 _:i2 _:i3 _:i4 ) .
_:p2 ex:z _:p1.
```

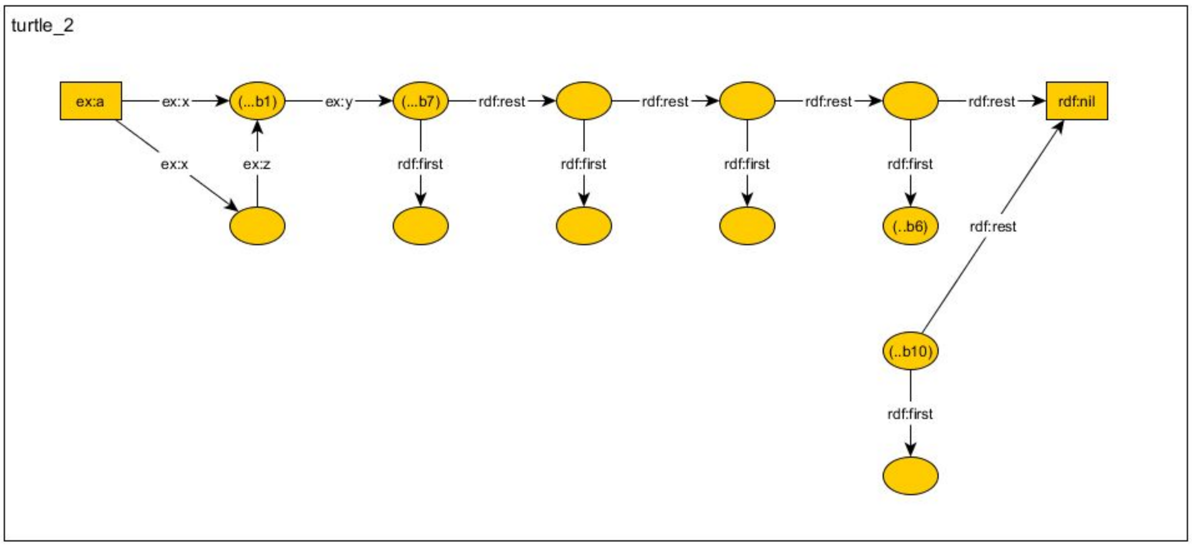
The above given turtle file is parsed into an RDF graph using Graph.parse() method.

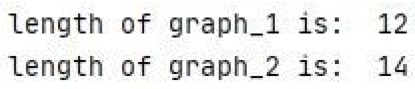


The parsed graph from turtle_1 is now serialized into another turtle file named turtle_2 using Graph.serialize() method. Again, the turtle_2 file is parsed into another graph as graph_2. But, the graph_2 is now having 2 triples extra with 14 triples in total.

```
graph_1 = Graph().parse(data=turtle_1, format='turtle')
turtle_2 = graph_1.serialize(format='turtle')
graph_2 = Graph().parse(data=turtle_2, format='turtle')
```

The visualization of graph_2 can be shown below:-





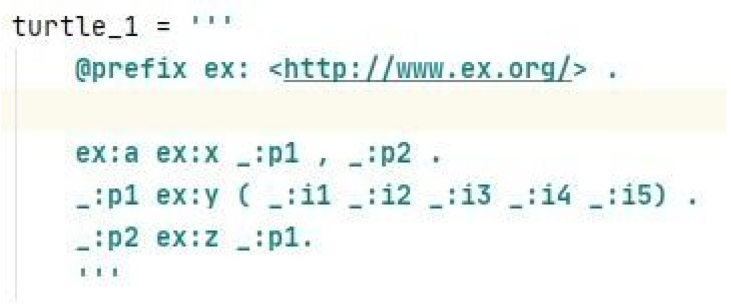
```
length of graph_1 is:  12
length of graph_2 is:  14
```

Solution for the issue:-

The above issue will not affect any of our expected functionality as all our original triples are present in it. The triples which are being added are an orphan list with one Blank Node.

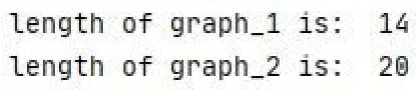
Observations:-

1) The problem is only arising with this particular format of turtle file with 4 and 5 blank nodes. The output with 5 blank nodes is shown below.



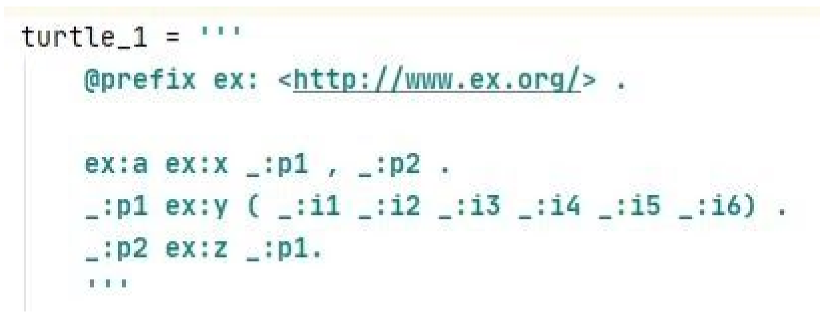
```
turtle_1 = '''
    @prefix ex: <http://www.ex.org/> .

    ex:a ex:x _:p1 , _:p2 .
    _:p1 ex:y ( _:i1 _:i2 _:i3 _:i4 _:i5) .
    _:p2 ex:z _:p1.
    '''
```



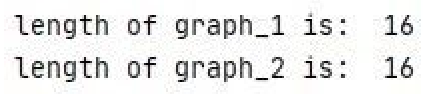
```
length of graph_1 is:  14
length of graph_2 is:  20
```

With 5 blank nodes also, some extra triples are being added into the turtle_2 file. But, this doesn't happen with 6 blank nodes or more.



```
turtle_1 = '''
    @prefix ex: <http://www.ex.org/> .

    ex:a ex:x _:p1 , _:p2 .
    _:p1 ex:y ( _:i1 _:i2 _:i3 _:i4 _:i5 _:i6) .
    _:p2 ex:z _:p1.
    '''
```

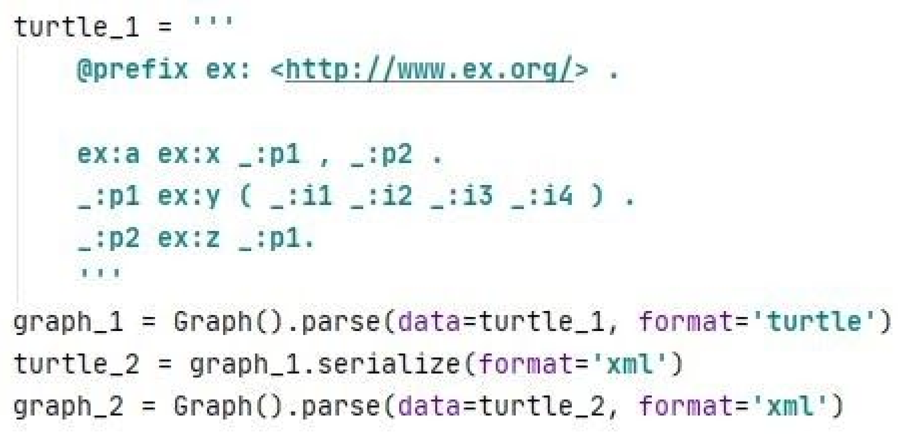


```
length of graph_1 is:  16
length of graph_2 is:  16
```

With 6 blank nodes, no extra triples will be added.

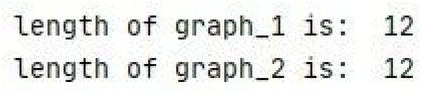
2) The Graph.serilaze() method has a parameter called ‘format’ which accepts all forms of representation like xml, n-triples, n-quads and many.

This issue doesn’t occur if we serialize the second graph using ‘xml format’.



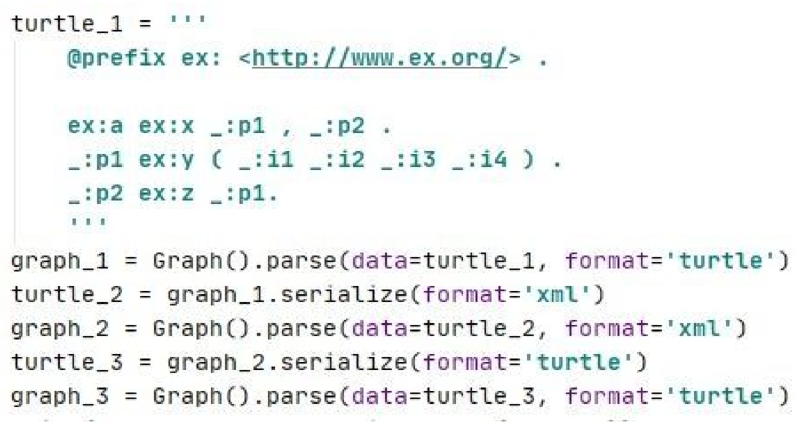
```
turtle_1 = '''
    @prefix ex: <http://www.ex.org/> .

    ex:a ex:x _:p1 , _:p2 .
    _:p1 ex:y ( _:i1 _:i2 _:i3 _:i4 ) .
    _:p2 ex:z _:p1.
    '''
graph_1 = Graph().parse(data=turtle_1, format='turtle')
turtle_2 = graph_1.serialize(format='xml')
graph_2 = Graph().parse(data=turtle_2, format='xml')
```



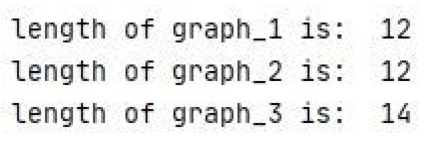
```
length of graph_1 is:  12
length of graph_2 is:  12
```

Even after converting graph_2 of xml format into turtle format, we are still getting an extra list with one blank node.



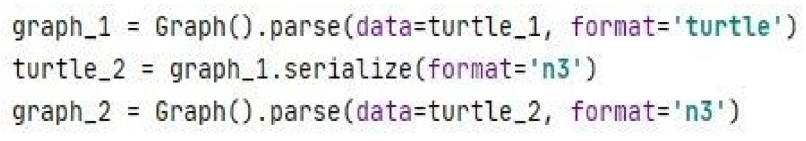
```
turtle_1 = '''
    @prefix ex: <http://www.ex.org/> .

    ex:a ex:x _:p1 , _:p2 .
    _:p1 ex:y ( _:i1 _:i2 _:i3 _:i4 ) .
    _:p2 ex:z _:p1.
    '''
graph_1 = Graph().parse(data=turtle_1, format='turtle')
turtle_2 = graph_1.serialize(format='xml')
graph_2 = Graph().parse(data=turtle_2, format='xml')
turtle_3 = graph_2.serialize(format='turtle')
graph_3 = Graph().parse(data=turtle_3, format='turtle')
```

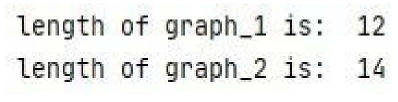


```
length of graph_1 is:  12
length of graph_2 is:  12
length of graph_3 is:  14
```

3) We have converted the graph to various other formats. In the below code, we have serialized the graph to ‘n3’ format. Even then an extra blank node list is added.



```
graph_1 = Graph().parse(data=turtle_1, format='turtle')
turtle_2 = graph_1.serialize(format='n3')
graph_2 = Graph().parse(data=turtle_2, format='n3')
```

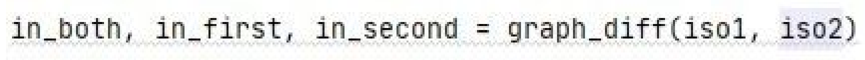


```
length of graph_1 is:  12
length of graph_2 is:  14
```

4) We also checked using ‘isomorphic’ and ‘to_isomorphic’ methods in rdflib.compare module.

The first thing for two graphs to be isomorphic is, both the graphs must have the same number of vertices and edges. But, as here the number of triples changes, the graphs are not isomorphic to each other.

5) We then tried using **graph_diff()** method from rdflib.compare module.



```
in_both, in_first, in_second = graph_diff(iso1, iso2)
```

in_both:- This in_both contains the triples which are different from two graphs if two graphs are not same. But, if the graphs are the same, this in_both will contain all the triples.

in_first:- This contains the triples which are extra in graph 1.

in_second:- This contains the triples which are extra in graph 2.

We tried removing the triples which are in_both form graph_2. Then we thought that we could remove extra triples . But, this cannot happen, because when we serialize a graph, the blank nodes will get different ids all the time which are randomly generated. Those random ids could not be compared and removed from any graph.

We have defined a function **myComparison(g1,g2)** which takes the two graphs (graph_1 and graph2) and performs all the operations that are mentioned in point 4 and 5.