Options for naming nodes

To me it seems that the easiest naming and accessing system for nodes is using some simple functions, to be defined on the beginning of the code. This way the code is easy readable once you understand what these functions do. The naming of the functions explain most of their job. If you use this system, both of the naming systems described below will become equivalent, since you only need to use the functions. For both naming systems I have written the code to define these functions.

The functions to be defined

Finding the label of a node:
labelNode(conenr, levelnr, nodenr) *Returns the label of a node.*

Finding property of node (the input 'node' in these functions is the node label):
getConenr(node) *Returns the cone number of a given node.*
getLevelnr(node) *Returns the level number of a given node.*
getNodenr(node) *Returns the node number of a given node. Every cone and every level starts numbering their nodes from 0.*

Finding certain nodes:
getNodes(Graph, *conditions) *Returns a list with the labels of all the nodes. It has optional arguments conenr, levelnr and nodenr. If any of these 3 arguments are added, only nodes on that cone/level and/or nodes with that nodenr will be returned.*

Ways to name (so ways to define the above functions)

As a string

This is how it is right now. The label of node 3 on level 2 in cone 1 has label "1_2_3".

+ This uses the labelling system of the networkx package. The other idea uses the attribute system, the 'label' of a node loses its meaning.
+ 'asking' python for properties of a certain node is very intuitive. For example: asking for the list of neighbours of node nr. 1 on level 2 in cone 3 is done like this: list(G.neighbors("1_2_3")).
− It is non trivial to understand how to get the conenumber/levelnumber/nodenumber of a certain node, or how to get a list with all the nodes from a certain cone/level. However, this problem is solved by using the functions I defined above.

As a node attribute

In that case the labels of the nodes just go from 0 to nNodes, they don't have any meaning. Every node gets 3 attributes: conenr, levelnr and a nodenr.

+ It is very easy to understand how to get the conenumber/levelnumber/nodenumber of a certain node, as can be seen by the defined functions.
− This doesn't use the labelling system of the networkx package, the 'label' of a node loses its meaning. This has as a consequence that every time the conenumber/levelnumber/nodenumber is used, an extra translation has to be made from those meaningful variables to the meaningless label.
− The creation of the network gets more complicated. The adding of the attributes is easy, but the nodes still have to be relabelled when adding the Watts-Strogatz ring to the network. This means that more actions have to be taken to create the network. It is not very difficult

to code, but the more lines the code has, the more work it costs to fully understand the code.
- It is still not trivial to understand how to get a list with all the nodes from a certain cone/level. However, this problem is also solved by using the functions I defined above.

Conclusion
I am in favor of using the first method of naming, the method already implemented. When the functions I wrote are used, it is a very easy system to use. The second way costs more code to implement and doesn't make stuff easier, although it look easier on first sight.