

Retrieving values

Since values are stored as Node objects, object retrieval will look something like this:

```
PYNDatabase.Node.Node.val
```

`val` is a variable which contains the value of the Node, and is linked to the original dictionary object.

To dynamically retrieve a `Node`, you can use the `get` method. The `get` method is also the only way to retrieve values which contain characters not in the alphabet (+the underscore character).

This way, you can avoid writing code like this:

```
eval(f'PYNDatabase.Node.{name_of_node}.Node.val')
```

Command Usage:

```
PYNDatabase.get(*names)
```

```
PYNDatabase.Node.get(*names)
```

If there is only 1 name given, the function will return a `Node` object. Otherwise, it will return a tuple of `Node`s.

Examples:

Static value retrieval

```
from pyndb import PYNDatabase
db = PYNDatabse('filename.pyndb')
db.set('hello', 'world')
print(db.hello.val) # <--
```

Dynamically retrieving a value

```
from pyndb import PYNDatabase
db = PYNDatabse('filename.pyndb')
db.set('123', 456) # Note that the value type doesn't matter

# These will work...
```

```
print(db.get('123').val)
# OR
node_name = 'helloworld'
print(db.get(node_name).val)
# OR
node_names = ['helloworld', 'test']
print(db.get(*node_names)[0].val) # Displays the value of the first Node returned

# But these will not.
print(db.123.val)
# OR
node_names = ['helloworld', 'test']
print(db.get(*node_names).val) # Why doesn't this work?
# Because the object returned is a tuple, not a Node!
```

Storing a Node inside a variable is also a great option!

```
from pyndb import PYNDatabase
db = PYNDatabse('filename.pyndb')
db.set('hello', 'world')
hello = db.hello # <--
print(hello.val)
```

Revision #7

Created 26 July 2021 22:50:22 by jvadair

Updated 19 February 2022 23:44:17 by jvadair