

Additional Features

- The `where()` method
- Converting back to a dictionary
- Node representation

The where() method

Syntax:

```
Node.where(**kwargs)
```

Let's say you have a lot of users, sorted by IDs. Now, you want to find a user with a specific, unique email. Sounds like a pain, right? Nope. Here's how to do it with pyntree:

```
db = Node("users.pyn")  
my_user = db.where(email="their@email.com") # -> [Node({"id": ...})]
```

Keep in mind that this will always return a list of Nodes (or an empty list).

Thus, finding multiple users with the same name is no problem:

```
db = Node("users.pyn")  
my_user = db.where(name="John") # -> [Node({"id": ...}), Node({"id": ...}), ...]
```

Converting back to a dictionary

Syntax:

```
dict(Node)
```

This is pretty straightforward, as shown above. The command will make a new dictionary. If you want to directly manipulate a Node's data instead, you can simply perform methods on the called function, as shown in the section [Math and object manipulation](#).

Node representation

Syntax:

```
str(Node)
# or
repr(Node)
```

-
- `str(Node)` will return a string representation of the data stored within the Node
 - `repr(Node)` will return the same as `str(Node)`, but wrapped within the text "`Node()`"