U5VGKQ2SY : <@U2BS4M1RV> Check this out:
```
def _query_filter(entry: str, include: list = None, exclude: list = None):
    if include:
        if entry in include and entry not in exclude:
            return True
    else:
        if entry not in exclude:
            return True
    return False

print("Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: "
      + str(_query_filter('test', ['this', 'is', 'a', 'test'], ['whatever'])))
print("Entry: 'test', include is [], exclude is ['whatever']: "
      + str(_query_filter('test', [], ['whatever'])))
```
output:
```
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: True
Entry: 'test', include is [], exclude is ['whatever']: True
```

U5VGKQ2SY : `if include` has no value b/c the very next line you are checking for the exact same thing that the `else` checks
U5VGKQ2SY : as it is written, `_query_filter()` returns True if `entry` is NOT in `exclude` and False for anything else
U2BS4M1RV : I think you misunderstand the meaning of include.
U5VGKQ2SY :
```
def _query_filter(entry: str, include: list = None, exclude: list = None):
    if include:
        if entry in include and entry not in exclude:
            return True
    else:
        if entry not in exclude:
            return True
    return False

print("Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: "
      + str(_query_filter('test', ['this', 'is', 'a', 'test'], ['whatever'])))
print("Entry: 'test', include is [], exclude is ['whatever']: "
      + str(_query_filter('test', [], ['whatever'])))
print("Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever', 'test']: "
      + str(_query_filter('test', ['this', 'is', 'a', 'test'], ['whatever', 'test'])))
print("Entry: 'test', include is [], exclude is ['whatever', 'test']: "
      + str(_query_filter('test', [], ['whatever', 'test'])))
```
output:
```
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: True
Entry: 'test', include is [], exclude is ['whatever']: True
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever', 'test']: False
Entry: 'test', include is [], exclude is ['whatever', 'test']: False
```

U2BS4M1RV : Here is one set of my tests for it. The first test shows what the raw data is.
```
    def test_counts_query_unfiltered(self):
        counts = pypihole.counts_query(self.test_log)
        self.assertEqual(counts['unifi'], 2)
        self.assertEqual(counts['openvpn'], 2)
        self.assertEqual(counts['<http://docs.google.com|docs.google.com>'], 1)

    def test_counts_query_include(self):
        counts = pypihole.counts_query(self.test_log, ['unifi'])
        self.assertEqual(counts['unifi'], 2)
        self.assertEqual(counts['openvpn'], 0)

    def test_counts_query_exclude(self):
        counts = pypihole.counts_query(self.test_log, exclude=['openvpn'])
        self.assertEqual(counts['unifi'], 2)
        self.assertEqual(counts['openvpn'], 0)
        self.assertEqual(counts['<http://docs.google.com|docs.google.com>'], 1)

    def test_counts_query_include_and_exclude(self):
        counts = pypihole.counts_query(self.test_log,
                                       include=['unifi', '<http://docs.google.com|docs.google.com>'],
                                       exclude=['openvpn'])
        self.assertEqual(counts['unifi'], 2)
        self.assertEqual(counts['openvpn'], 0)
        self.assertEqual(counts['<http://docs.google.com|docs.google.com>'], 1)
```

U2BS4M1RV : All of the tests pass as it is currently.
U2BS4M1RV : Include is a whitelist, but only used if it isn't an empty list. If include is an empty list then it is ignored.
U2BS4M1RV : Your example is the intended result.
U2BS4M1RV : It works similarly to the include used on some routers, or like grep in a way. If include isn't specified everything is returned True. If include is populated then it acts as a whiltelist. And exclude works as a blacklist.
U5VGKQ2SY : output:
```
'Entry: 'test' NOT IN exclude:
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: True
Entry: 'test', include is [], exclude is ['whatever']: True
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever']: True
Entry: 'test', include is [], exclude is ['whatever']: True
Entry: 'test' IN exclude:
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever', 'test']: False
Entry: 'test', include is [], exclude is ['whatever', 'test']: False
Entry: 'test', include is ['this', 'is', 'a', 'test'], exclude is ['whatever', 'test']: False
Entry: 'test', include is [], exclude is ['whatever', 'test']: False
```

U5VGKQ2SY : &gt; If include isn't specified everything is returned TrueThat's not what is happening.

U2BS4M1RV : Sorry, that should say if include isn't specified, and exclude doesn't trigger, then it is returned True.
U5VGKQ2SY : but I could be wrong. Probably best to wait for someone else to chime in
U2BS4M1RV : There is no problem with that code, it tests perfectly.
U5VGKQ2SY : ok
U2BS4M1RV : Sorry if I was unclear on that.
U2BS4M1RV : My question is about the namedtuple.
U2BS4M1RV : Rather than sending entry[index] I am wondering, short of a dict or a class, if there is a better way, that is closer to sending entry.query.
U2BS4M1RV : <@U5VGKQ2SY>, here you go: <https://github.com/TomFaulkner/pypihole>
U5VGKQ2SY : Posting this b/c it got pushed up a ways. In case anyone else wants to give it a shot.
U2BS4M1RV : Thanks.
U2BS4M1RV : Full code to the above: <https://github.com/TomFaulkner/pypihole>
U1UFZTD4J : <@U2BS4M1RV> the easy answer is that you should never change the indexes/order on a named tuple
U1UFZTD4J : Thats a breaking change. If your concerned about that, you should probably use a class or dictionary and not let people use indexes
U2BS4M1RV : Yeah, I think I probably have to agree there. As nice as namedtuples are, that does make it easy to break things should the data being analyzed changed.
U1PCHFXMH : <@U2BS4M1RV> Are you trying to get the index of the property, when you first defined it?`Query._fields.index('client')`

U1PCHFXMH : Not sure if I understood correctly
U2BS4M1RV : You did, that works well.

U2BS4M1RV : However, it also starts with an underscore. I can't imagine standard lib changing that, but...
U2BS4M1RV : Actually, in the case of a namedtuple, it might make sense to do the underscore to avoid someone naming one of their fields 'fields' and breaking things.
U5VGKQ2SY : I know we already went over this but I really think that:```
```
    if include:
        if entry in include and entry not in exclude:
            return True
    else:
        if entry not in exclude:
            return True
    return False
```
Should read be:
```
    if include and entry in include and entry not in exclude:
        return True
    else:
        return False
```

U2BS4M1RV : Except then if include is empty it returns false, which is not the desired effect.
U2BS4M1RV : It is an optional whitelist.
U5VGKQ2SY : ok
U1PCHFXMH : <@U2BS4M1RV> Just checked the docs, it says "In addition to the methods inherited from tuples, named tuples support three additional methods and two attributes. To prevent conflicts with field names, the method and attribute names start with an underscore."
U2BS4M1RV : Nice. :smile:
U2BS4M1RV : Thank you!
U2BS4M1RV : Ended up using getattr for this on the namedtuple. getattr(mytuple, 'fieldname') :slightly_smiling_face:
U1PCHFXMH : <@U2BS4M1RV> Wouldn't that be the same as mytuple.fieldname ?
U1BP42MRS : That's what I've been wondering the whole time, I feel like the point of a named tuple is to be able to use the `.` operator.