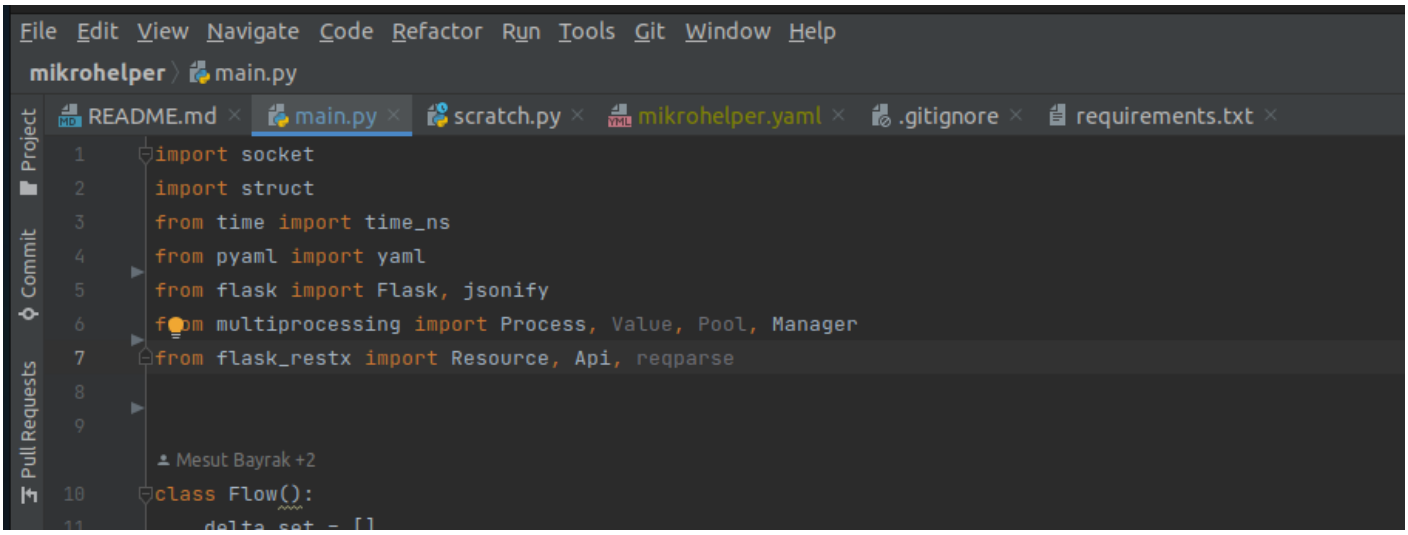
Sharing a dictionary between processes on python



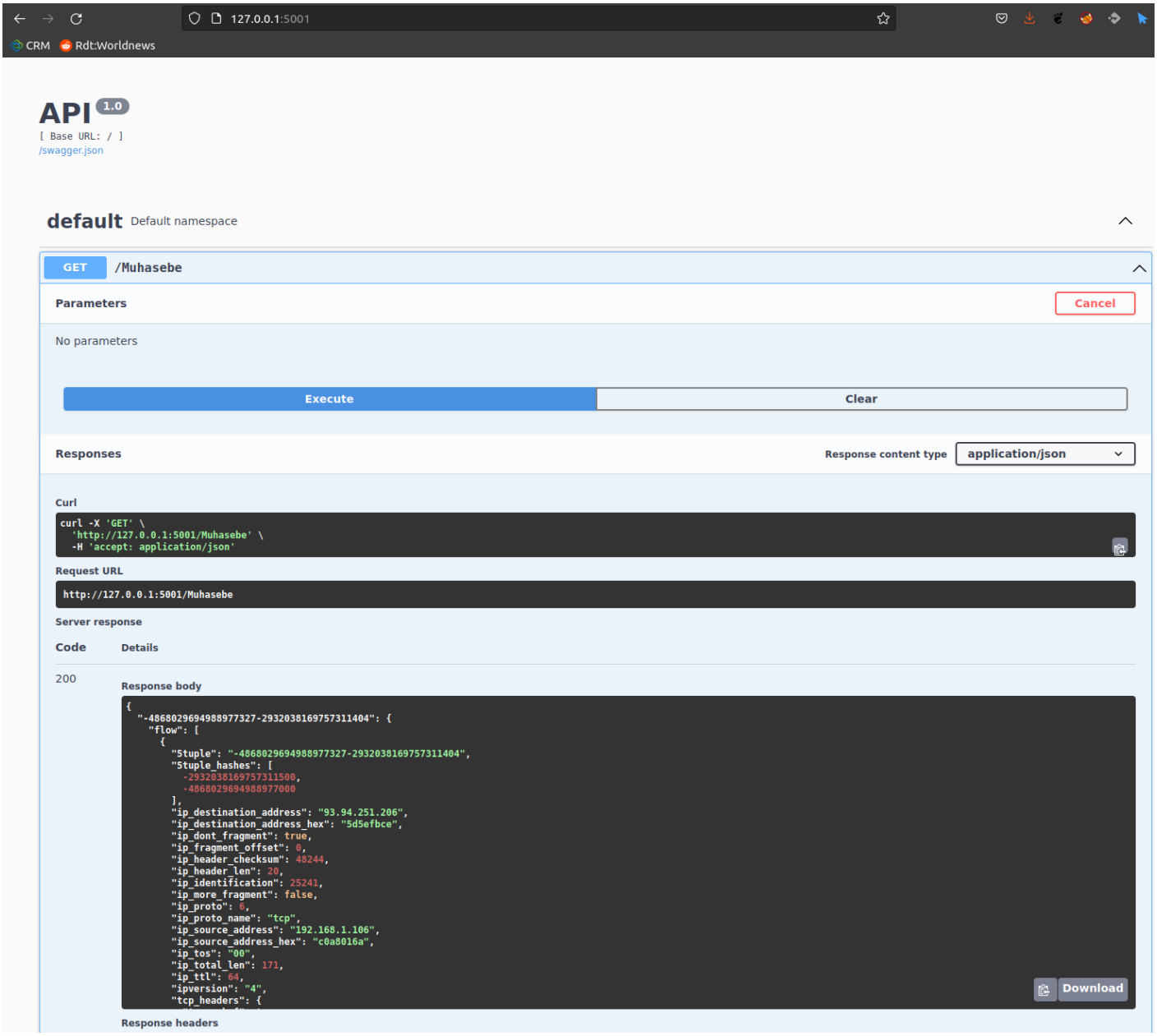
As you might know Python has a problem called as GIL which means global interpreter lock. This lock prevents sharing variables between processes. Basically process creates another **interpreted** processs which also means double the memory, and process power.

To overcome this problem, guys at python created a solution called Manager() not only you can share data between processes, you can share data between computers, nice isn't it ?

Quoted from python3 documentation;

Server process managers are more flexible than using shared memory objects because they can be made to support arbitrary object types. Also, a single manager can be shared by processes on different computers over a network. They are, however, slower than using shared memory.

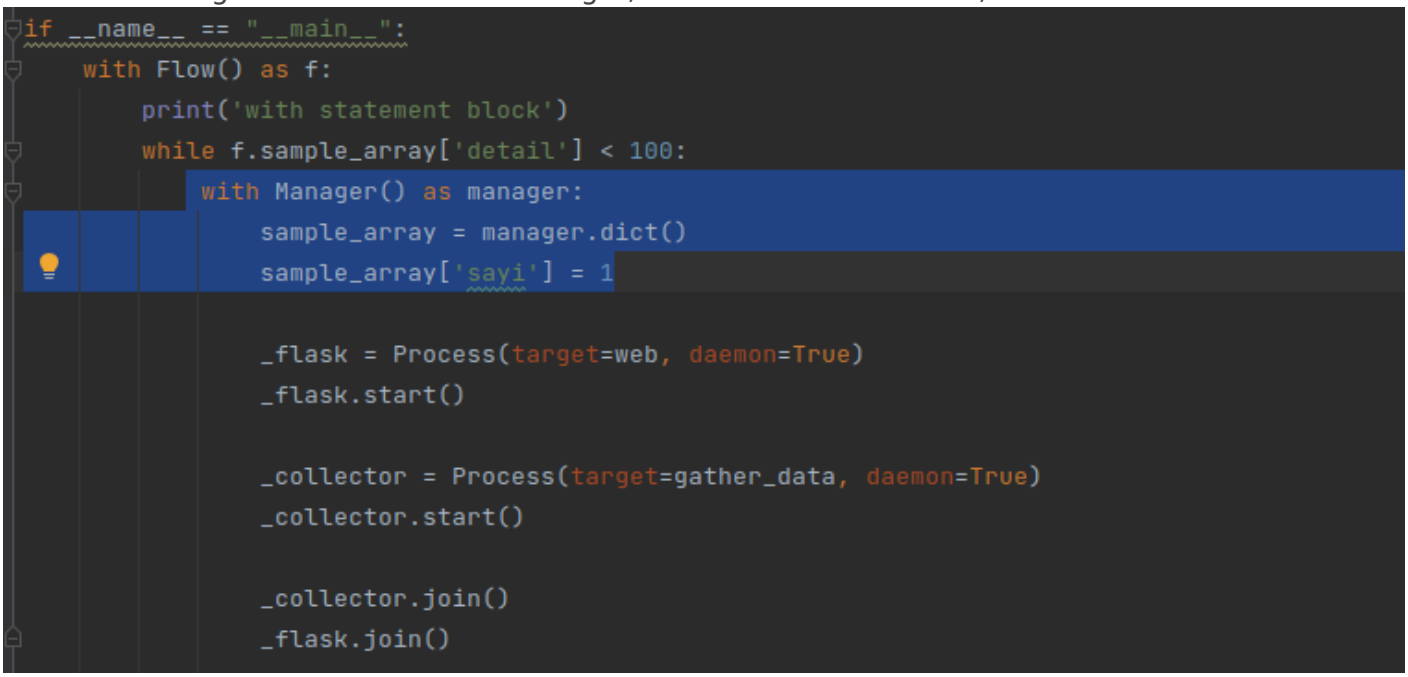
On my case, i needed to update a dictionary on one process and a flask api was going to serve requests based on this dictionary



However there was a problem, when i needed to update nested dictionaries on process i saw that dict was never updated, and look's like the Manager() class has a bug which doesn't update the values on dictionaries. Quoting from python "https://docs.python.org/3/library/multiprocessing.html#proxy-objects"

> If standard (non-proxy) `list` or `dict` objects are contained in a referent, modifications to those mutable values will not be propagated through the manager because the proxy has no way of knowing when the values contained within are modified. However, storing a value in a container proxy (which triggers a `__setitem__` on the proxy object) does propagate through the manager and so to effectively modify such an item, one could re-assign the modified value to the container proxy:

I did use manager within a context manager, it was initiated like this ;



```
if __name__ == "__main__":
    with Flow() as f:
        print('with statement block')
        while f.sample_array['detail'] < 100:
            with Manager() as manager:
                sample_array = manager.dict()
                sample_array['sayi'] = 1

                _flask = Process(target=web, daemon=True)
                _flask.start()

                _collector = Process(target=gather_data, daemon=True)
                _collector.start()

                _collector.join()
                _flask.join()
```

here is an example to update the manager() owned dictionary ;

```
class Flow():
    def __init__(self):
        print('init method called')
        self.sample_array = {}

    def __enter__(self):
        print('enter method called')
        self.sample_array['detail']=0
        return self
    def increase_detail(self,count):
        self.sample_array['detail'] += count

    def __exit__(self, exc_type, exc_val, exc_tb):
        print("exited")
        print(self.sample_array)
    # def __exit__(self, exc_type, exc_value, exc_traceback):
    #     print('exit method called')
```

the increase_details() function represents a stream to update the sample_array['detail'] by 1 on every interval, how ever this wasn't happening and i couldn't find any legitimate solution to this.

What i did was to create a copy of the array, do the nested updates on new array and copy the whole array back to manager. Like this;

```
    def increase_detail(self,count):
    # self.sample_array['detail'] += count
    _sa_array= self.sample_array
    _sa_array['detail'] += count
    self.sample_array=_sa_array
```

Lame but solved my problem, i'd like to know possible solutions to this problem, if you have one mail me mailto mesut

---

Revision #3
Created 2022-11-09 12:13:49 UTC by Mesut Bayrak
Updated 2022-11-09 12:46:20 UTC by Mesut Bayrak