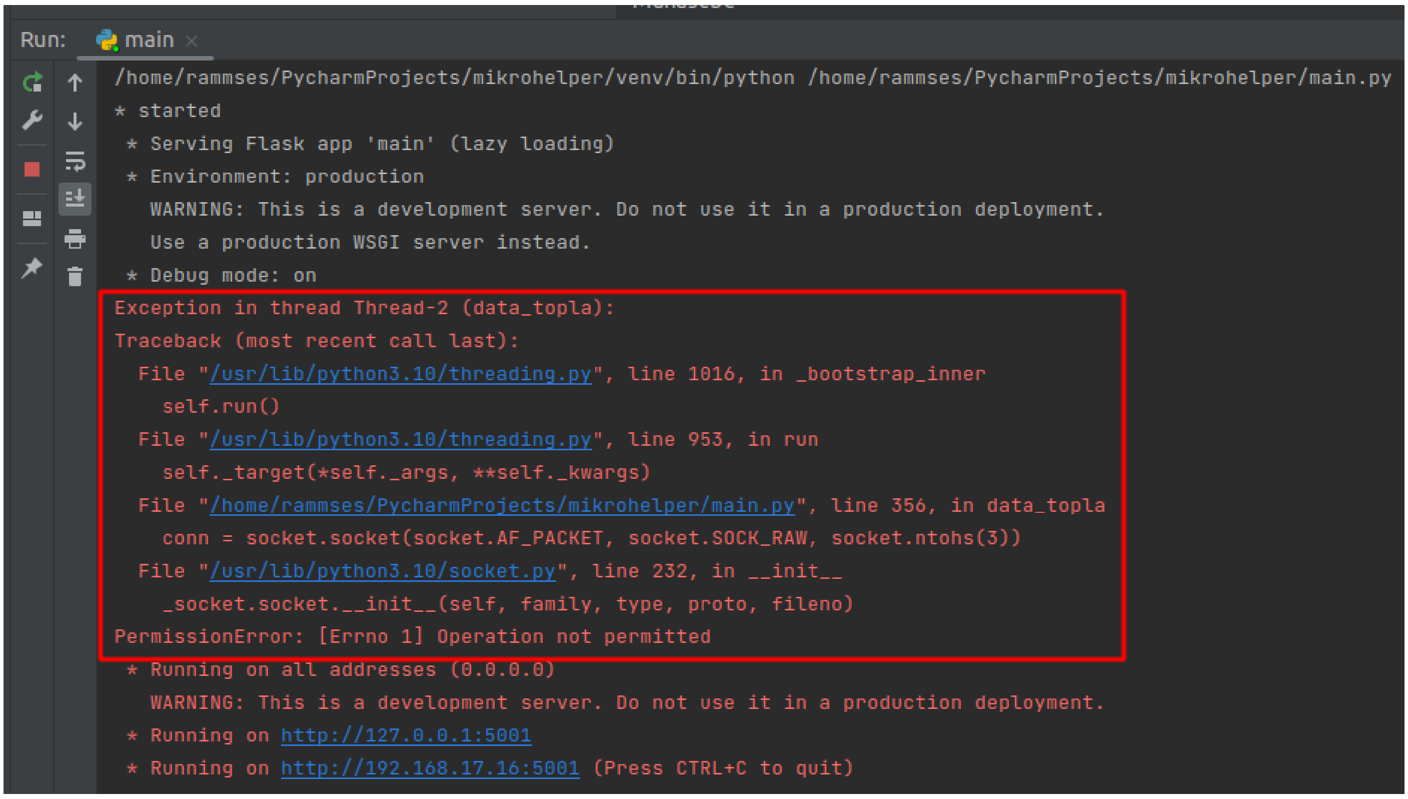
Mesut's Python problems and solutions

I have or had problems while coding python, you might find some fixes or some tricks

Permissions and working with raw sockets using pycharm



```
/home/rammses/PycharmProjects/mikrohelper/venv/bin/python /home/rammses/PycharmProjects/mikrohelper/main.py
* started
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
Exception in thread Thread-2 (data_topla):
Traceback (most recent call last):
  File "/usr/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.10/threading.py", line 953, in run
    self._target(*self._args, **self._kwargs)
  File "/home/rammses/PycharmProjects/mikrohelper/main.py", line 356, in data_topla
    conn = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))
  File "/usr/lib/python3.10/socket.py", line 232, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.17.16:5001 (Press CTRL+C to quit)
```

The problem is that raw_sockets require root privileges to initiate. This is not good and creates a lot of fuzz when you do runs on pycharm, you need to go terminal execute the code using sudo and stopping it creates a lot of additional fuzz if you are using threads or processes libraries. And this solution is only for developers that uses linux environments

To overcome this problem;

You can setcap the interpreter but you must be carefull because this is dangerous. After this point all the code you run with this feture will be commited with root privileges so if you are doing packet generation or some crazy stuff you might create network problems. Watch it !!!

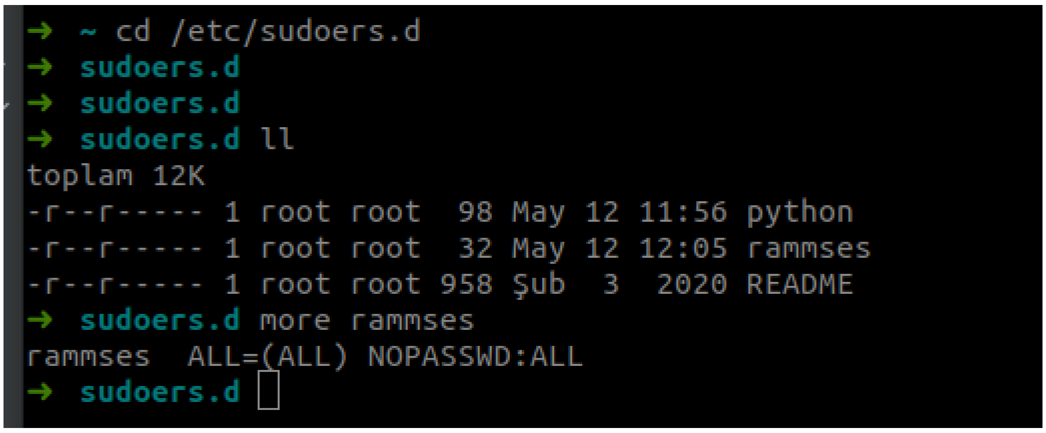
```
sudo setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /usr/bin/python3.10
```

Or run with sudo in pycharm, use the script below as your interpreter.

There is a catch, you should also require to be a nopassword user in your sudoers config.

1. Get your user name in to sudoers

   ```
   sudo echo "your_user_name ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/your_user_name
   ```



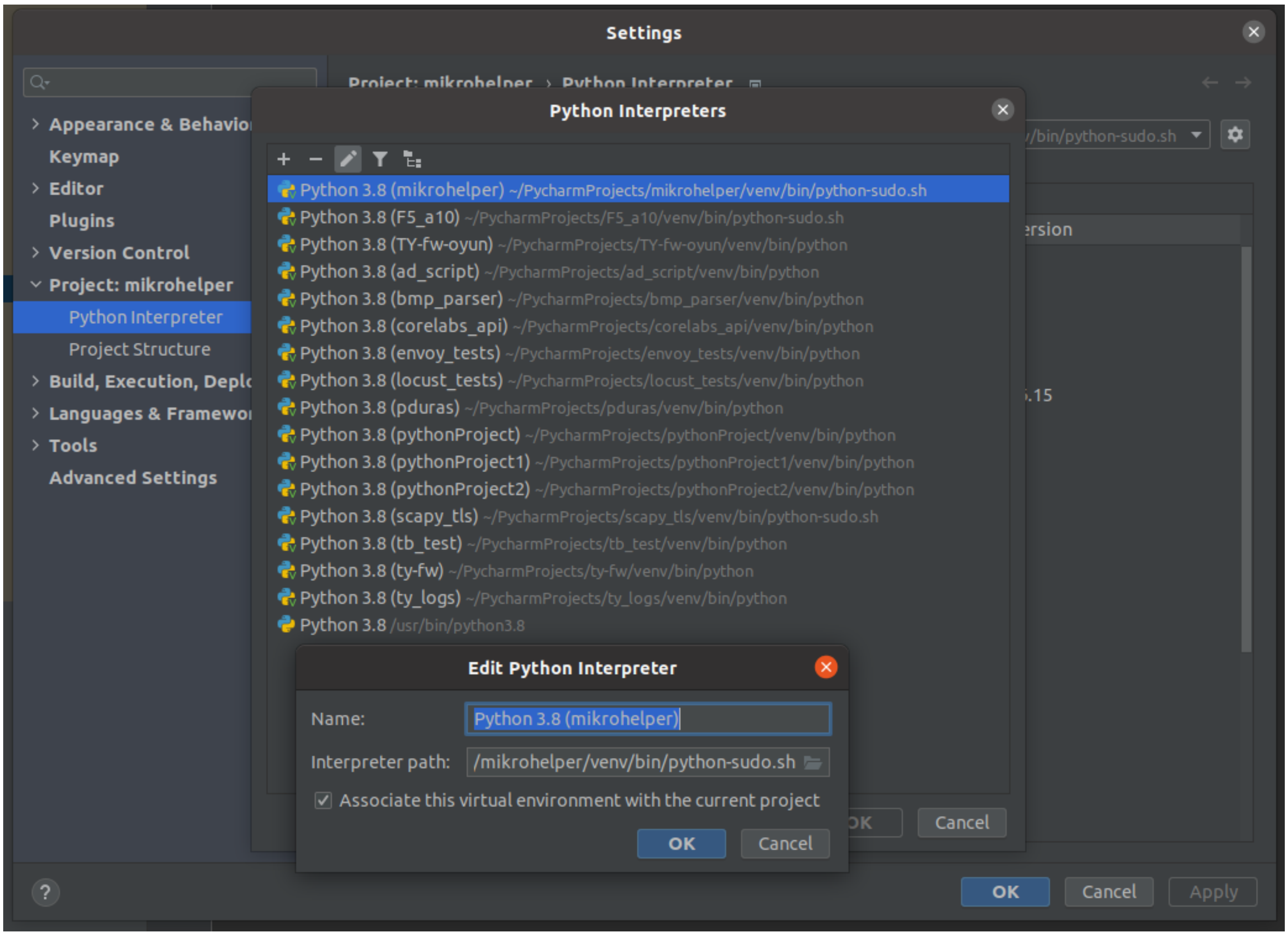
```
→ ~ cd /etc/sudoers.d
→ sudoers.d
→ sudoers.d
→ sudoers.d ll
toplam 12K
-r--r----- 1 root root  98 May 12 11:56 python
-r--r----- 1 root root  32 May 12 12:05 rammses
-r--r----- 1 root root 958 Şub  3  2020 README
→ sudoers.d more rammses
rammses  ALL=(ALL) NOPASSWD:ALL
→ sudoers.d
```

2. Change your pycharm config to use the python-sudo
   Copy this and don't forget to change the "home/rammses/PycharmProjects/mikrohelper" section before creating the python-sudo.sh file

   ```
   #!/bin/bash

   sudo /home/rammses/PycharmProjects/mikrohelper/venv/bin/python "$@"
   ```

   - put it into venv/bin folder
   - name it python-sudo.sh
   - set it executable chmod +x ./python-sudo.sh
   - set your project interpreter as shown below



Settings

Project: mikrohelper › Python Interpreter

Python Interpreters

```
Python 3.8 (mikrohelper) ~/PycharmProjects/mikrohelper/venv/bin/python-sudo.sh
Python 3.8 (F5_a10) ~/PycharmProjects/F5_a10/venv/bin/python-sudo.sh
Python 3.8 (TY-fw-oyun) ~/PycharmProjects/TY-fw-oyun/venv/bin/python
Python 3.8 (ad_script) ~/PycharmProjects/ad_script/venv/bin/python
Python 3.8 (bmp_parser) ~/PycharmProjects/bmp_parser/venv/bin/python
Python 3.8 (corelabs_api) ~/PycharmProjects/corelabs_api/venv/bin/python
Python 3.8 (envoy_tests) ~/PycharmProjects/envoy_tests/venv/bin/python
Python 3.8 (locust_tests) ~/PycharmProjects/locust_tests/venv/bin/python
Python 3.8 (pduras) ~/PycharmProjects/pduras/venv/bin/python
Python 3.8 (pythonProject) ~/PycharmProjects/pythonProject/venv/bin/python
Python 3.8 (pythonProject1) ~/PycharmProjects/pythonProject1/venv/bin/python
Python 3.8 (pythonProject2) ~/PycharmProjects/pythonProject2/venv/bin/python
Python 3.8 (scapy_tls) ~/PycharmProjects/scapy_tls/venv/bin/python-sudo.sh
Python 3.8 (tb_test) ~/PycharmProjects/tb_test/venv/bin/python
Python 3.8 (ty-fw) ~/PycharmProjects/ty-fw/venv/bin/python
Python 3.8 (ty_logs) ~/PycharmProjects/ty_logs/venv/bin/python
Python 3.8 /usr/bin/python3.8
```

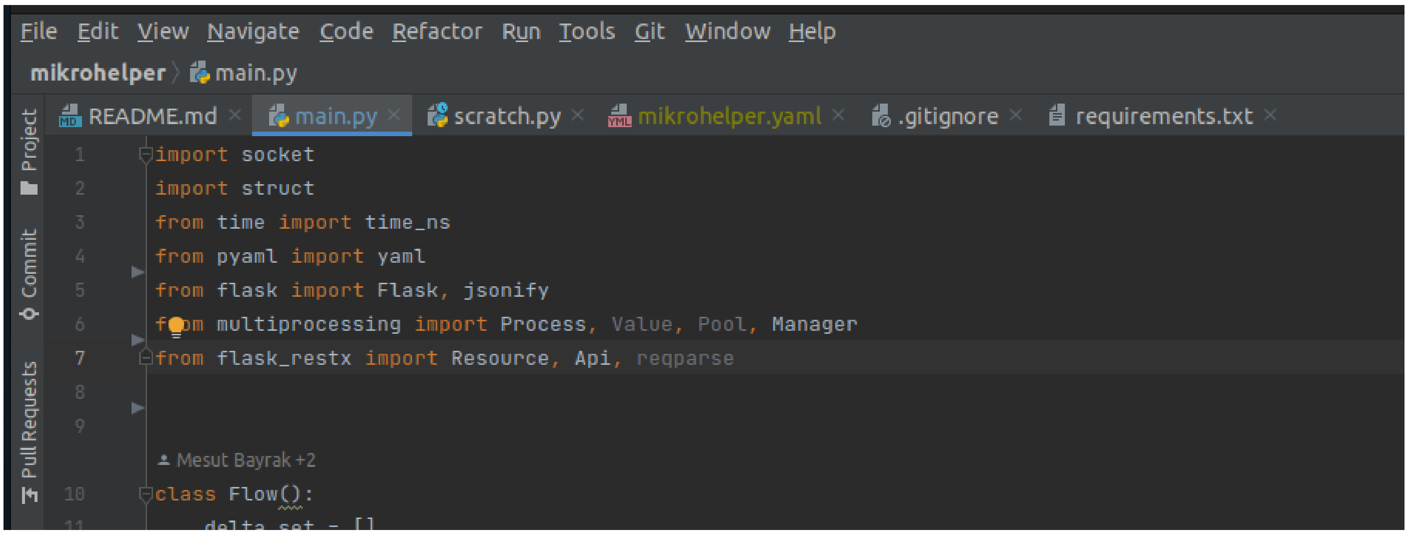
Edit Python Interpreter

Name: Python 3.8 (mikrohelper)

Interpreter path: /mikrohelper/venv/bin/python-sudo.sh

[✓] Associate this virtual environment with the current project

Sharing a dictionary between processes on python



```
import socket
import struct
from time import time_ns
from pyaml import yaml
from flask import Flask, jsonify
from multiprocessing import Process, Value, Pool, Manager
from flask_restx import Resource, Api, reqparse


class Flow():
    delta_set = []
```

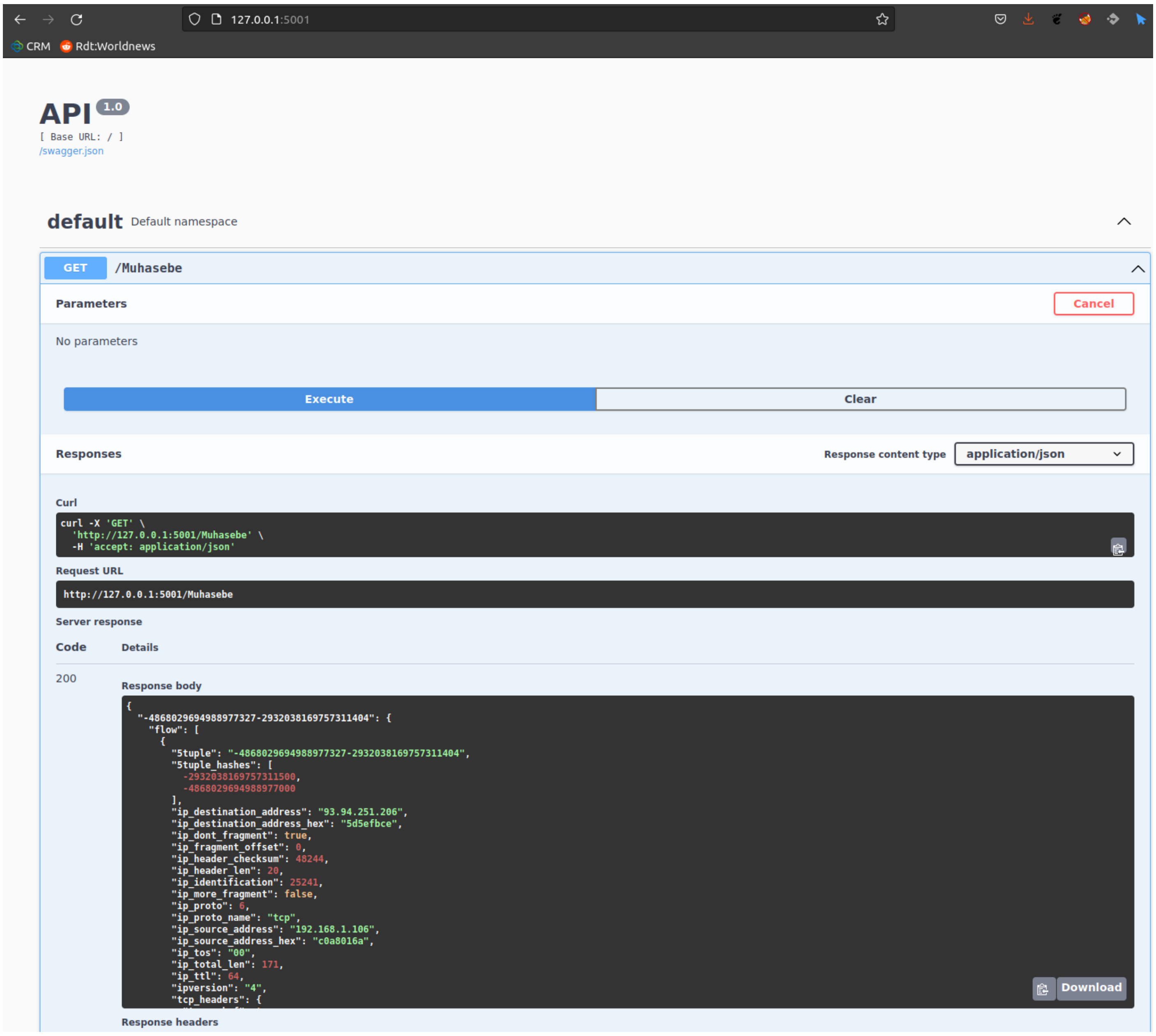
As you might know Python has a problem called as GIL which means global interpreter lock. This lock prevents sharing variables between processes. Basically process creates another **interpreted** processs which also means double the memory, and process power.

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



```
curl -X 'GET' \
  'http://127.0.0.1:5001/Muhasebe' \
  -H 'accept: application/json'
```

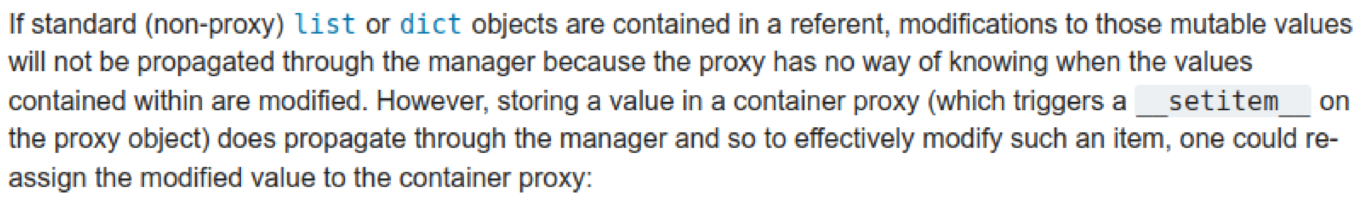
```
http://127.0.0.1:5001/Muhasebe
```

200

Response body

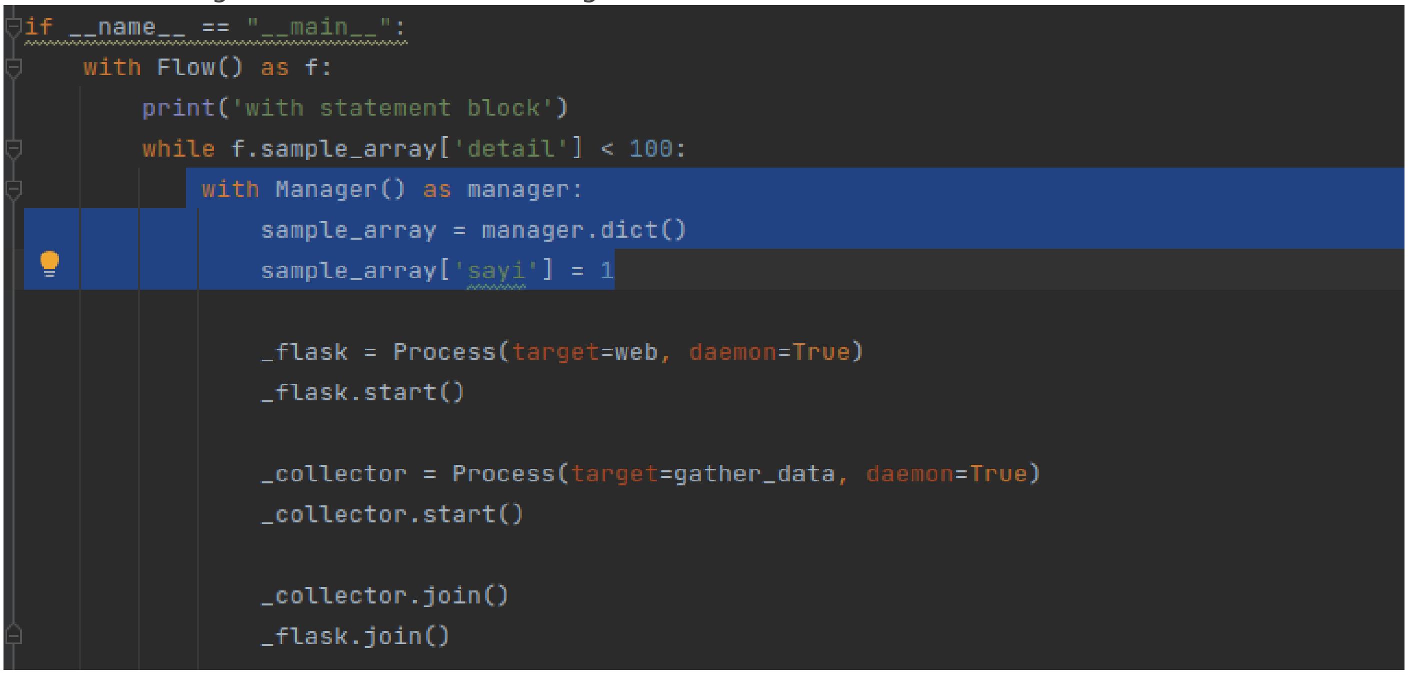
```
{
  "-4868029694988977327-2932038169757311404": {
    "flow": [
      {
        "5tuple": "-4868029694988977327-2932038169757311404",
        "5tuple_hashes": [
          -2932038169757311500,
          -4868029694988977000
        ],
        "ip_destination_address": "93.94.251.206",
        "ip_destination_address_hex": "5d5efbce",
        "ip_dont_fragment": true,
        "ip_fragment_offset": 0,
        "ip_header_checksum": 48244,
        "ip_header_len": 20,
        "ip_identification": 25241,
        "ip_more_fragment": false,
        "ip_proto": 6,
        "ip_proto_name": "tcp",
        "ip_source_address": "192.168.1.106",
        "ip_source_address_hex": "c0a8016a",
        "ip_tos": "00",
        "ip_total_len": 171,
        "ip_ttl": 64,
        "ipversion": "4",
        "tcp_headers": {
```

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

Connecting to MS SQL Server with python

The problem

Sql server connection cant find the sql engine

The last line looks for a sql server engine named **SQL SERVER** however we don't use it anymore.

```
Collecting pypyodbc
Using cached
https://files.pythonhosted.org/packages/62/94/a5bb72a83366c3249d60c7c465b25cb4252b6be4cc2b13eef048e8
e73085/pypyodbc-1.3.6.tar.gz
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from pypyodbc)
Building wheels for collected packages: pypyodbc
Running setup.py bdist_wheel for pypyodbc ... done
Stored in directory: /root/.cache/pip/wheels/bd/68/0f/b23f408ec9ad90e883abc69ae16bf87ac822259de735c9f77c
Successfully built pypyodbc
Installing collected packages: pypyodbc
Successfully installed pypyodbc-1.3.6
root@miharbines:/home/miharbines/Test# python3 dbtest1.py
Traceback (most recent call last):
```

```
File "dbtest1.py", line 3, in <module>
connection = pypyodbc.connect('Driver={SQL
Server};Server=x.y.z.d;DATABASE=trade;UID=sa;PWD=x.x.x.x.x.x')
File "/usr/local/lib/python3.6/dist-packages/pypyodbc.py", line 2454, in _init_
self.connect(connectString, autocommit, ansi, timeout, unicode_results, readonly)
File "/usr/local/lib/python3.6/dist-packages/pypyodbc.py", line 2507, in connect
check_success(self, ret)
File "/usr/local/lib/python3.6/dist-packages/pypyodbc.py", line 1009, in check_success
ctrl_err(SQL_HANDLE_DBC, ODBC_obj.dbc_h, ret, ODBC_obj.ansi)
File "/usr/local/lib/python3.6/dist-packages/pypyodbc.py", line 983, in ctrl_err
raise OperationalError(state,err_text)
pypyodbc.OperationalError: ('01000', "[01000] [unixODBC][Driver Manager]Can't open lib 'SQL Server' : file not
found")
```

Solution

İnstall MS SQL version 17 db engine from ms repos

```
sudo -i
curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
curl https://packages.microsoft.com/config/ubuntu/$(lsb_release -rs)/prod.list > /etc/apt/sources.list.d/mssql-
release.list
sudo apt update
sudo ACCEPT_EULA=Y apt-get install -y msodbcsql17
sudo apt-get install -y unixodbc-dev
```

Switch the driver

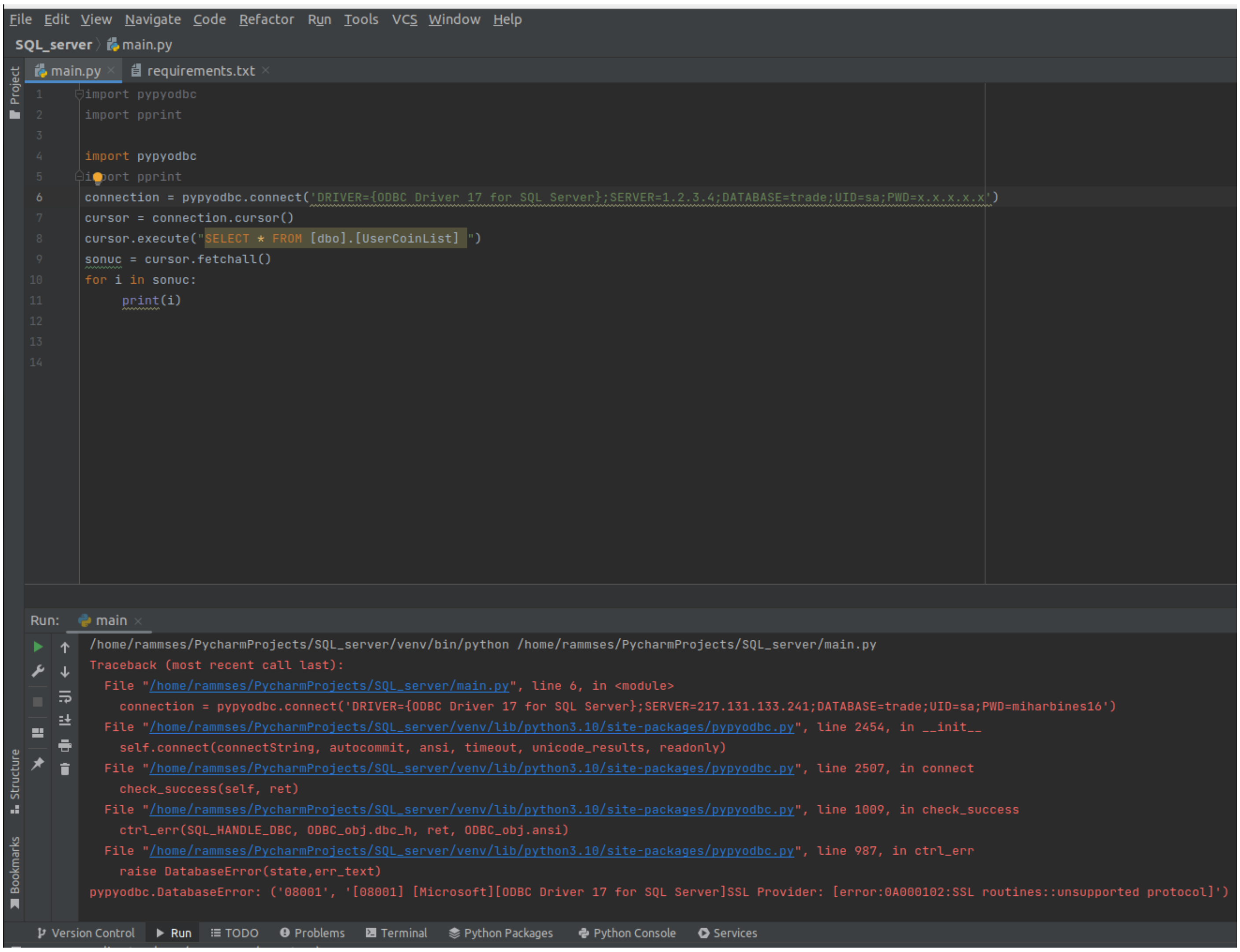
From

```
conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=server_name;DATABASE=database_name;UID=user;PWD=password')
```

to

```
conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=server_name;DATABASE=database_name;UID=user;PWD=password')
```

Output



```python
import pypyodbc
import pprint

import pypyodbc
import pprint
connection = pypyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=1.2.3.4;DATABASE=trade;UID=sa;PWD=x.x.x.x.x')
cursor = connection.cursor()
cursor.execute("SELECT * FROM [dbo].[UserCoinList] ")
sonuc = cursor.fetchall()
for i in sonuc:
    print(i)
```

```
/home/rammses/PycharmProjects/SQL_server/venv/bin/python /home/rammses/PycharmProjects/SQL_server/main.py
Traceback (most recent call last):
  File "/home/rammses/PycharmProjects/SQL_server/main.py", line 6, in <module>
    connection = pypyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=217.131.133.241;DATABASE=trade;UID=sa;PWD=miharbines16')
  File "/home/rammses/PycharmProjects/SQL_server/venv/lib/python3.10/site-packages/pypyodbc.py", line 2454, in __init__
    self.connect(connectString, autocommit, ansi, timeout, unicode_results, readonly)
  File "/home/rammses/PycharmProjects/SQL_server/venv/lib/python3.10/site-packages/pypyodbc.py", line 2507, in connect
    check_success(self, ret)
  File "/home/rammses/PycharmProjects/SQL_server/venv/lib/python3.10/site-packages/pypyodbc.py", line 1009, in check_success
    ctrl_err(SQL_HANDLE_DBC, ODBC_obj.dbc_h, ret, ODBC_obj.ansi)
  File "/home/rammses/PycharmProjects/SQL_server/venv/lib/python3.10/site-packages/pypyodbc.py", line 987, in ctrl_err
    raise DatabaseError(state,err_text)
pypyodbc.DatabaseError: ('08001', '[08001] [Microsoft][ODBC Driver 17 for SQL Server]SSL Provider: [error:0A000102:SSL routines::unsupported protocol]')
```

2nd Problem

My friend was using MS SQL 2014 default install and TLS 1.0 was the only option from produced by MS SQL 2014, we had to install the latest service packs so it can support newer TLS version such as tls 1.2 or 1.3

here is the link for turkish one

After patching the server 2014, the connection got established and code worked as expected.