



# Laboratório 2

Grupo A:

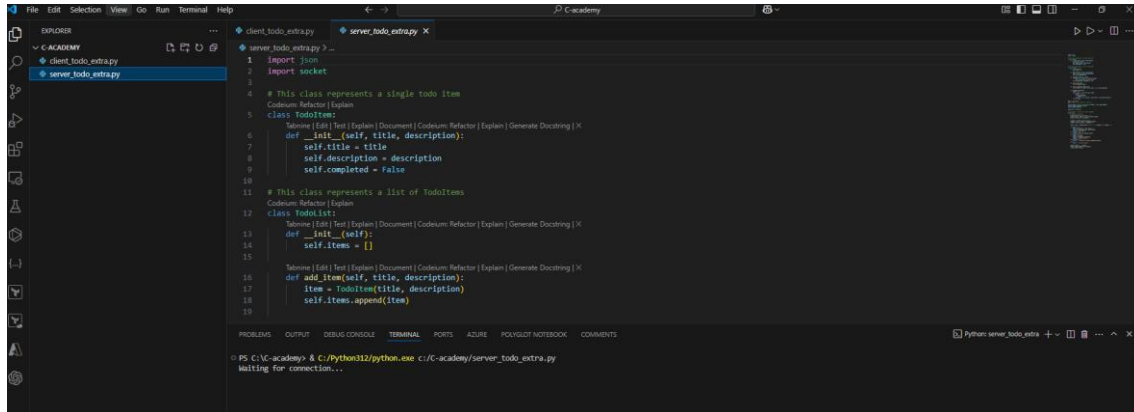
Teresa Correia  
Leonardo Abreu  
Miguel Rodrigues  
Sérgio Marcelino

## Laboratório 2 – Grupo A

### Exercício 1

1) link para o github : <https://github.com/mafar1968/repoLAB2>

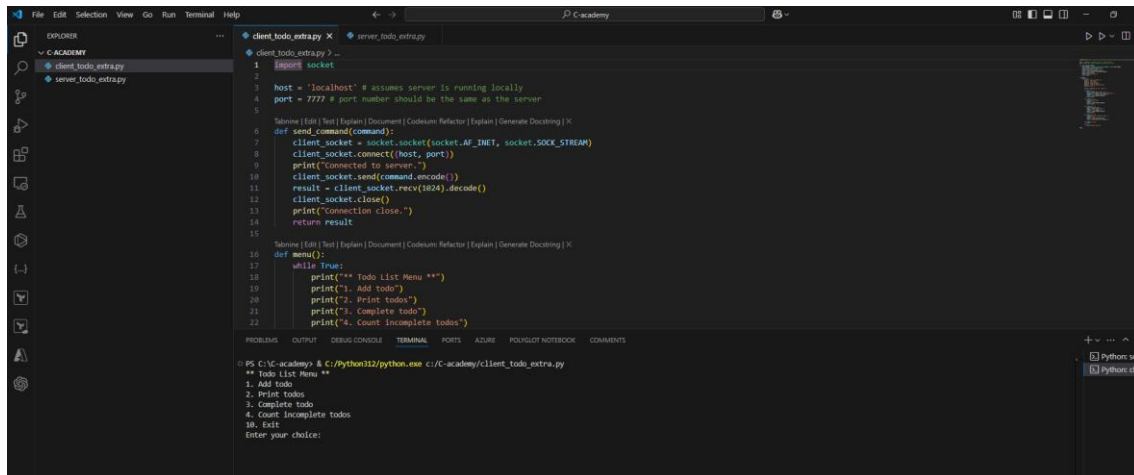
Servidor a funcionar, à espera de ligação por parte do cliente



```
1 import json
2 import socket
3
4 # This class represents a single todo item
5 class TodoItem:
6     def __init__(self, title, description):
7         self.title = title
8         self.description = description
9         self.completed = False
10
11 # This class represents a list of TodoItems
12 class TodoList:
13     def __init__(self):
14         self.items = []
15
16     def add_item(self, title, description):
17         item = TodoItem(title, description)
18         self.items.append(item)
```

PS C:\C-academy> C:\Python312\python.exe c:\C-academy\server\_todo\_extra.py  
Waiting for connection...

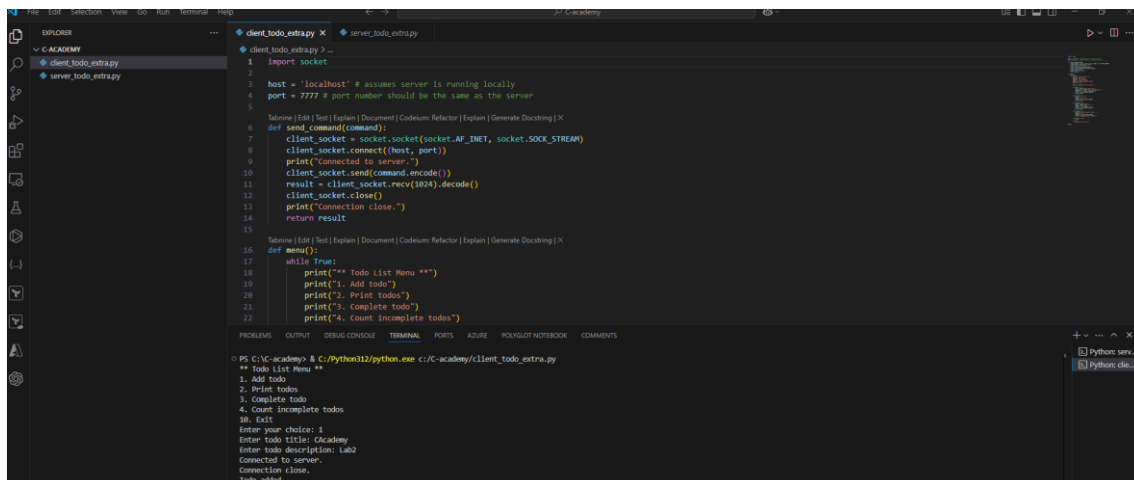
Cliente a funcionar.



```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17    while True:
18        print("*** Todo List Menu ***")
19        print("1. Add todo")
20        print("2. Print todos")
21        print("3. Complete todo")
22        print("4. Count incomplete todos")
23        choice = input("Enter your choice: ")
```

PS C:\C-academy> C:\Python312\python.exe c:\C-academy\client\_todo\_extra.py  
\*\* Todo List Menu \*\*  
1. Add todo  
2. Print todos  
3. Complete todo  
4. Count incomplete todos  
10. Exit  
Enter your choice:

Tarefa adicionada à ToDo List do lado do cliente

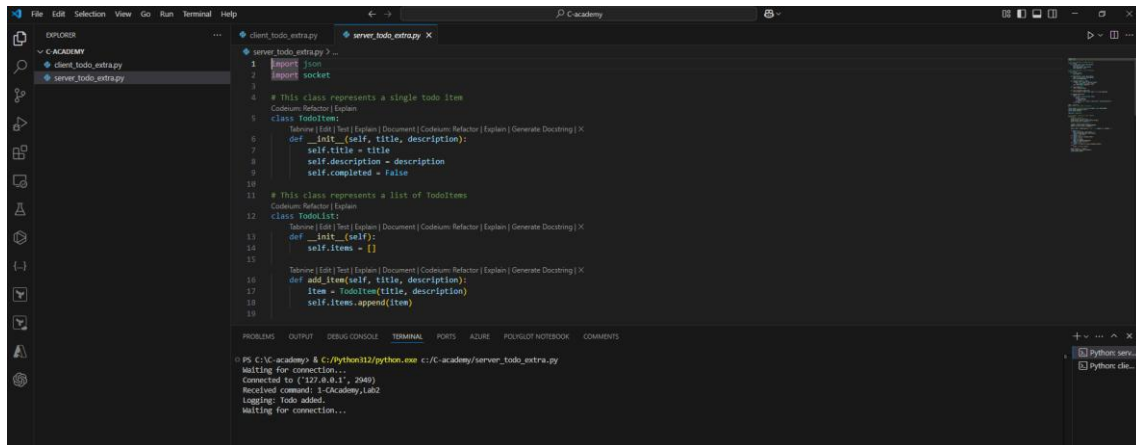


```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17    while True:
18        print("*** Todo List Menu ***")
19        print("1. Add todo")
20        print("2. Print todos")
21        print("3. Complete todo")
22        print("4. Count incomplete todos")
23        choice = input("Enter your choice: ")
24        if choice == "1":
25            title = input("Enter todo title: ")
26            description = input("Enter todo description: ")
27            command = json.dumps({"title": title, "description": description})
28            send_command(command)
```

PS C:\C-academy> C:\Python312\python.exe c:\C-academy\client\_todo\_extra.py  
\*\* Todo List Menu \*\*  
1. Add todo  
2. Print todos  
3. Complete todo  
4. Count incomplete todos  
10. Exit  
Enter your choice: 1  
Enter todo title: Ckadeemy  
Enter todo description: Lab2  
Connected to server.  
Connection close.  
Todo added.

## Laboratório 2 – Grupo A

Evidência, do lado do servidor, em como foi adicionada tarefa e escrita no log



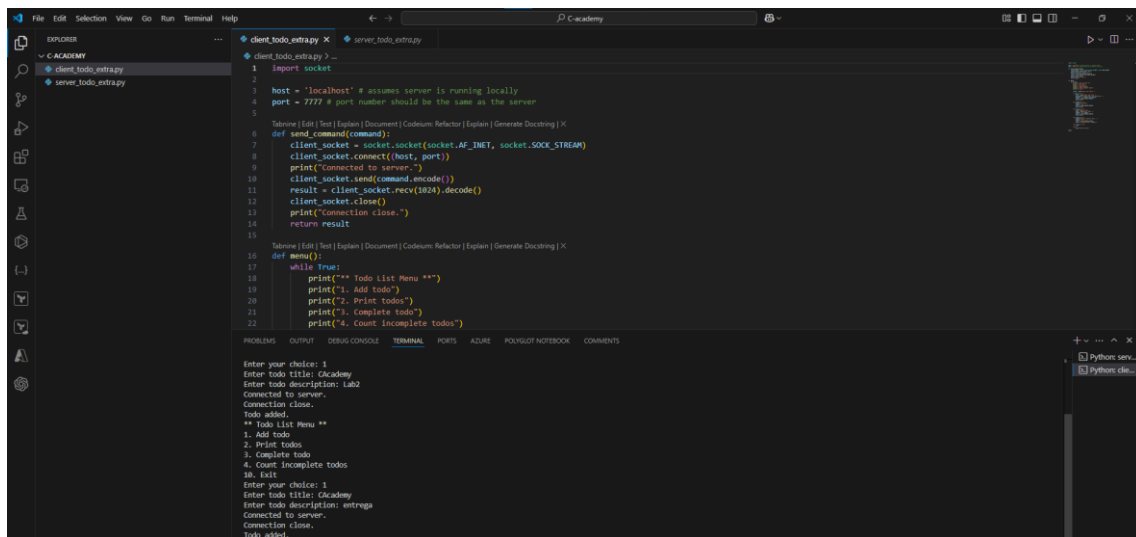
The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `server_todo_extra.py` file contains the following code:

```
1 import json
2 import socket
3
4 # This class represents a single todo item
5 class TodoItem:
6     def __init__(self, title, description):
7         self.title = title
8         self.description = description
9         self.completed = False
10
11 # This class represents a list of TodoItems
12 class TodoList:
13     def __init__(self):
14         self.items = []
15
16     def add_item(self, title, description):
17         item = TodoItem(title, description)
18         self.items.append(item)
```

The terminal output shows the server running and receiving a command:

```
PS C:\C-academy> & C:\Python12\python.exe c:\C-academy\server_todo_extra.py
Waiting for connection...
Connected to ('127.0.0.1', 2948)
Received command: 1-Cademy,lab2
Logging: todo added.
Waiting for connection...
```

Adicionada nova entrada na ToDo List



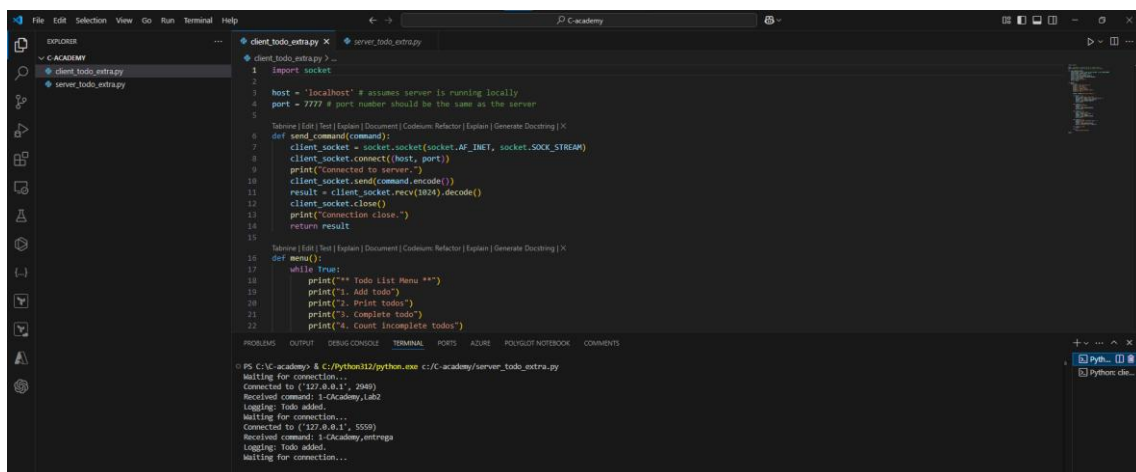
The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `client_todo_extra.py` file contains the following code:

```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17    while True:
18        print("*** Todo List Menu ***")
19        print("1. Add todo")
20        print("2. Print todos")
21        print("3. Complete todo")
22        print("4. Count incomplete todos")
23        print("0. Exit")
24        choice = input("Enter your choice: ")
25        if choice == "1":
26            title = input("Enter todo title: ")
27            description = input("Enter todo description: ")
28            command = f"1-{title},{description}"
29            send_command(command)
30            print("Connected to server.")
31            print("Connection close.")
32            print("Todo added.")
```

The terminal output shows the client running and sending a command:

```
Enter your choice: 1
Enter todo title: Cademy
Enter todo description: Lab2
Connected to server.
Connection close.
Todo added.
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
0. Exit
Enter your choice: 1
Enter todo title: Cademy
Enter todo description: entrega
Connected to server.
Connection close.
Todo added.
```

Entrada verificada do lado do servidor.



The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `server_todo_extra.py` file contains the following code:

```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17    while True:
18        print("*** Todo List Menu ***")
19        print("1. Add todo")
20        print("2. Print todos")
21        print("3. Complete todo")
22        print("4. Count incomplete todos")
23        print("0. Exit")
24        choice = input("Enter your choice: ")
25        if choice == "1":
26            title = input("Enter todo title: ")
27            description = input("Enter todo description: ")
28            command = f"1-{title},{description}"
29            send_command(command)
30            print("Connected to server.")
31            print("Connection close.")
32            print("Todo added.")
```

The terminal output shows the server running and receiving a command:

```
PS C:\C-academy> & C:\Python12\python.exe c:\C-academy\server_todo_extra.py
Waiting for connection...
Connected to ('127.0.0.1', 2948)
Received command: 1-Cademy,Lab2
Logging: todo added.
Waiting for connection...
Connected to ('127.0.0.1', 5558)
Received command: 1-Cademy,entrega
Logging: todo added.
Waiting for connection...
```

**Laboratório 2 – Grupo A**

## Print das tarefas no cliente

The image shows a VS Code editor window with a Python file named `server_todo_entry.py`. The script implements a simple chat server using the `socket` module. It listens for incoming connections on `localhost` at port `7777`. When a connection is established, it prints a message and enters a loop where it receives and decodes incoming messages, prints them, and then sends back a response based on the user's choice (1: Add todo, 2: Print todos, 3: Complete todo, 4: Count incomplete todos, 5: Exit).

```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17     while True:
18         print("*** Todo List Menu **")
19         print("1. Add todo")
20         print("2. Print todos")
21         print("3. Complete todo")
22         print("4. Count incomplete todos")
23
24 # Main execution
25 if __name__ == '__main__':
26     menu()
27     send_command("Enter todo description: ")
28     send_command("Connected to server.")
29     send_command("Connection close.")
30     send_command("Todo added.")
31     send_command("** Todo List Menu **")
32     send_command("1. Add todo")
33     send_command("2. Print todos")
34     send_command("3. Complete todo")
35     send_command("4. Count incomplete todos")
36     send_command("5. Exit")
37     send_command("Enter your choice: 2")
38     send_command("Print todos")
39     send_command("Connected to server.")
40     send_command("Connection close.")
41     send_command("0. [ ] CkAcademy: Lab2")
42     send_command("1. [ ] CkAcademy: entrega")
43     send_command("** Todo List Menu **")
44     send_command("1. Add todo")
45     send_command("2. Print todos")
46     send_command("3. Complete todo")
47     send_command("4. Count incomplete todos")
48     send_command("5. Exit")
49     send_command("Enter your choice: ")
```

The terminal output shows the execution of the script, including the menu prompts and the user's interactions with the application.

```
Enter todo title: CkAcademy
Enter todo description: entrega
Connected to server.
Connection close.
Todo added.
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 2
Print todos
Connected to server.
Connection close.
0. [ ] CkAcademy: Lab2
1. [ ] CkAcademy: entrega
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 
```

E do lado do servidor

The screenshot shows a VS Code editor with a Python file named `server_todo.py` open. The file contains a simple HTTP server implementation using the `socket` module. The server listens on `localhost` at port `7777`. It receives commands from a client and responds with a list of todos. The terminal output shows the server running and receiving commands from `1-Cademy: Lab2`.

```

1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17     while True:
18         print(""" Todo List Menu """)
19         print("1. add todo")
20         print("2. print todos")
21         print("3. complete todo")
22         print("4. Count incomplete todos")

```

The terminal output shows the server running and receiving commands from `1-Cademy: Lab2`:

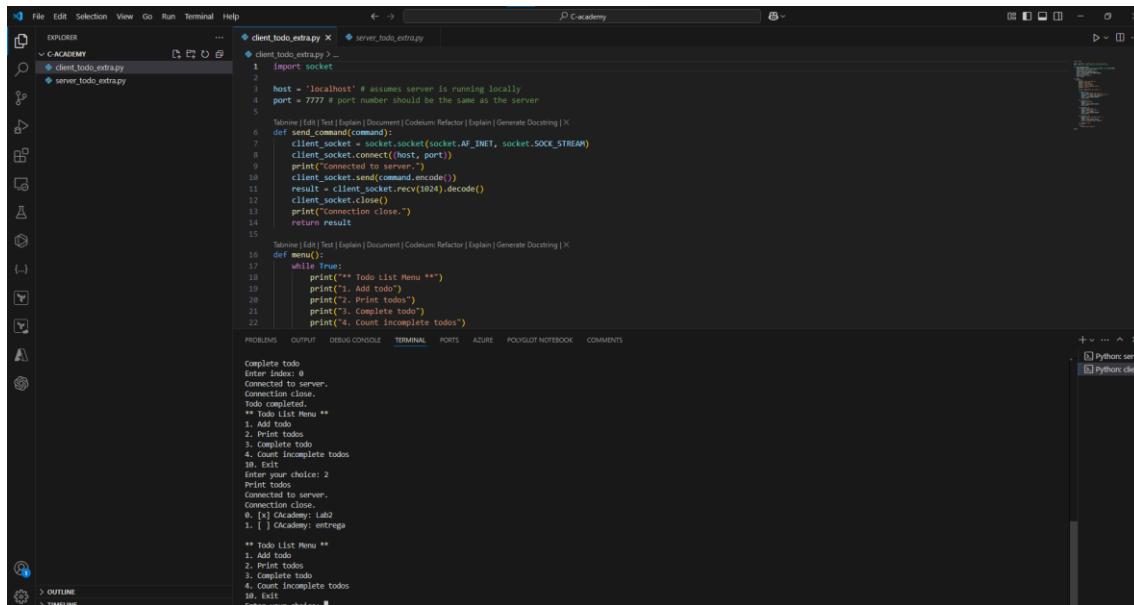
```

PS C:\C:\cademy> C:\python112\python.exe c:\C-cademy\server_todo_extra.py
Waiting for connection...
Connected to ('127.0.0.1', 7940)
Received command: 1-Cademy: Lab2
Logging: Todo added.
Waiting for connection...
Connected to ('127.0.0.1', 5550)
Received command: 1-Cademy: entrega
Logging: Todo added.
Waiting for connection...
Connected to ('127.0.0.1', 6987)
Received command: 2:
Logging: 0. [ 1-Cademy: Lab2
1. [ 1-Cademy: entrega
Waiting for connection...

```

## Laboratório 2 – Grupo A

Conclusão da tarefa 1 (index 0) e posterior listagem total das tarefas (lado do cliente)



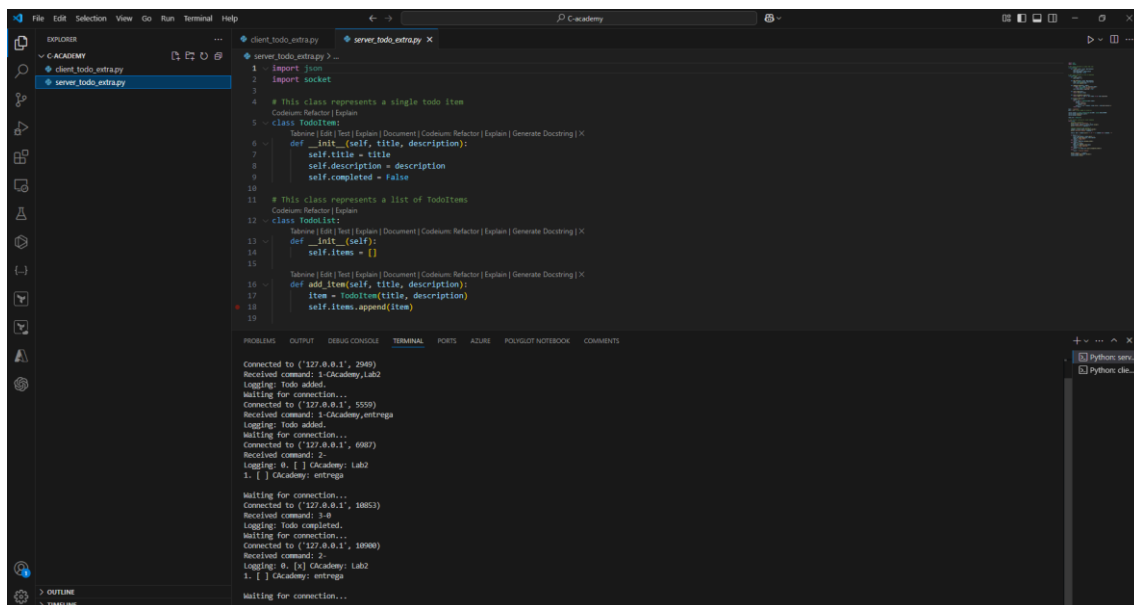
The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `client_todo_extra.py` file contains the following code:

```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17     while True:
18         print("\n** Todo List Menu **")
19         print("1. Add todo")
20         print("2. Print todos")
21         print("3. Complete todo")
22         print("4. Count incomplete todos")
23         print("5. Exit")
24         user_input = input("Enter your choice: ")
25         if user_input == "1":
26             title = input("Enter title: ")
27             description = input("Enter description: ")
28             command = f"1-Cademy,{title},{description}"
29             send_command(command)
30         elif user_input == "2":
31             command = "2-"
32             send_command(command)
33         elif user_input == "3":
34             command = "3-Cademy,entrega"
35             send_command(command)
36         elif user_input == "4":
37             command = "4-"
38             send_command(command)
39         elif user_input == "5":
40             command = "5-"
41             send_command(command)
42         else:
43             print("Invalid choice. Please try again.")
44
45 if __name__ == '__main__':
46     menu()
```

The terminal output shows the execution of the client code:

```
Complete todo
Enter index: 0
Connected to server.
Connection close.
Todo completed.
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 2
Print todos
Connected to server.
Connection close.
0. [x] Cademy: Lab2
1. [ ] Cademy: entrega
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 1
```

E do lado do servidor



The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `server_todo_extra.py` file contains the following code:

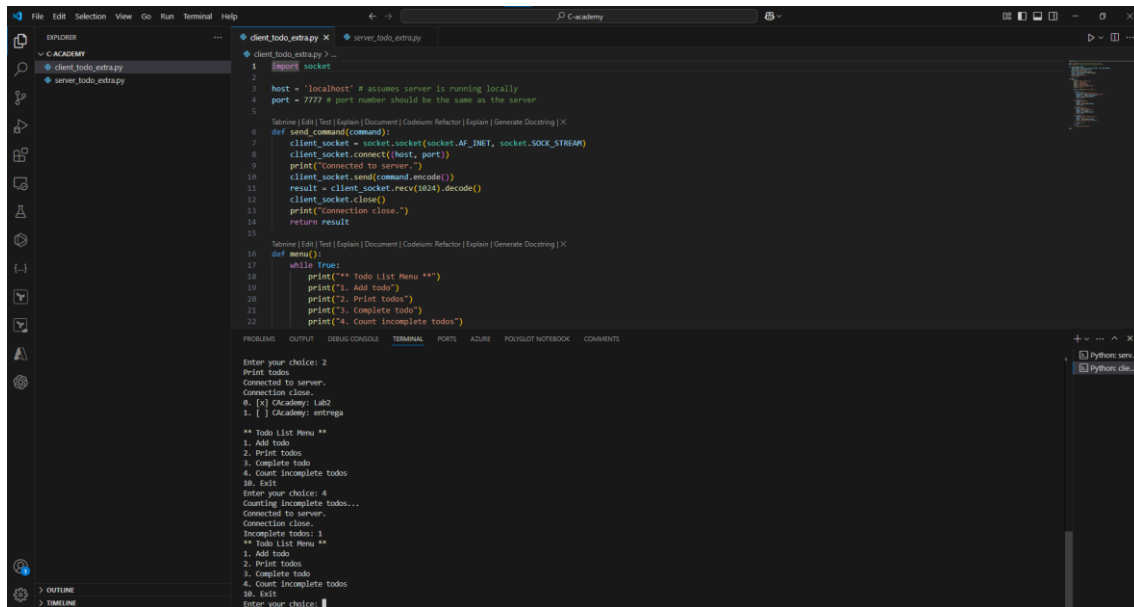
```
1 import sys
2 import socket
3
4 # This class represents a single todo item
5 class TodoItem:
6     def __init__(self, title, description):
7         self.title = title
8         self.description = description
9         self.completed = False
10
11 # This class represents a list of TodoItems
12 class TodoList:
13     def __init__(self):
14         self.items = []
15
16     def add_item(self, title, description):
17         item = TodoItem(title, description)
18         self.items.append(item)
19
20 if __name__ == '__main__':
21     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22     server_socket.bind(('localhost', 7777))
23     server_socket.listen(5)
24     print("Server is running on localhost:7777")
25
26     while True:
27         client_socket, address = server_socket.accept()
28         print(f"Connected to {address}")
29
30         command = client_socket.recv(1024).decode()
31         print(f"Received command: {command}")
32
33         if command.startswith("1-"):
34             title, description = command.split(',')
35             add_item(title, description)
36             print(f"Added item: {title} - {description}")
37
38         elif command.startswith("2-"):
39             print("Printing todos")
40             for item in items:
41                 print(f"{item.title} - {item.description} - {item.completed}")
42
43         elif command.startswith("3-"):
44             title, description = command.split(',')
45             complete_item(title, description)
46             print(f"Completed item: {title} - {description}")
47
48         elif command.startswith("4-"):
49             print("Counting incomplete todos")
50             count_incomplete()
51
52         elif command.startswith("5-"):
53             print("Exiting")
54             sys.exit(0)
55
56         client_socket.close()
57         print("Connection closed")
```

The terminal output shows the execution of the server code:

```
Connected to ('127.0.0.1', 2949)
Received command: 1-Cademy,Lab2
Logging: Todo added.
Waiting for connection...
Connected to ('127.0.0.1', 5509)
Received command: 1-Cademy,entrega
Logging: Todo added.
Waiting for connection...
Connected to ('127.0.0.1', 6987)
Received command: 2-
Logging: 0. [x] Cademy: Lab2
1. [ ] Cademy: entrega
Waiting for connection...
Connected to ('127.0.0.1', 18853)
Received command: 3-
Logging: Todo completed.
Waiting for connection...
Connected to ('127.0.0.1', 10900)
Received command: 4-
Logging: 0. [x] Cademy: Lab2
1. [ ] Cademy: entrega
Waiting for connection...
```

## Laboratório 2 – Grupo A

Contagem das tarefas por fazer (lado do cliente)



The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `client_todo_extra.py` file contains the following code:

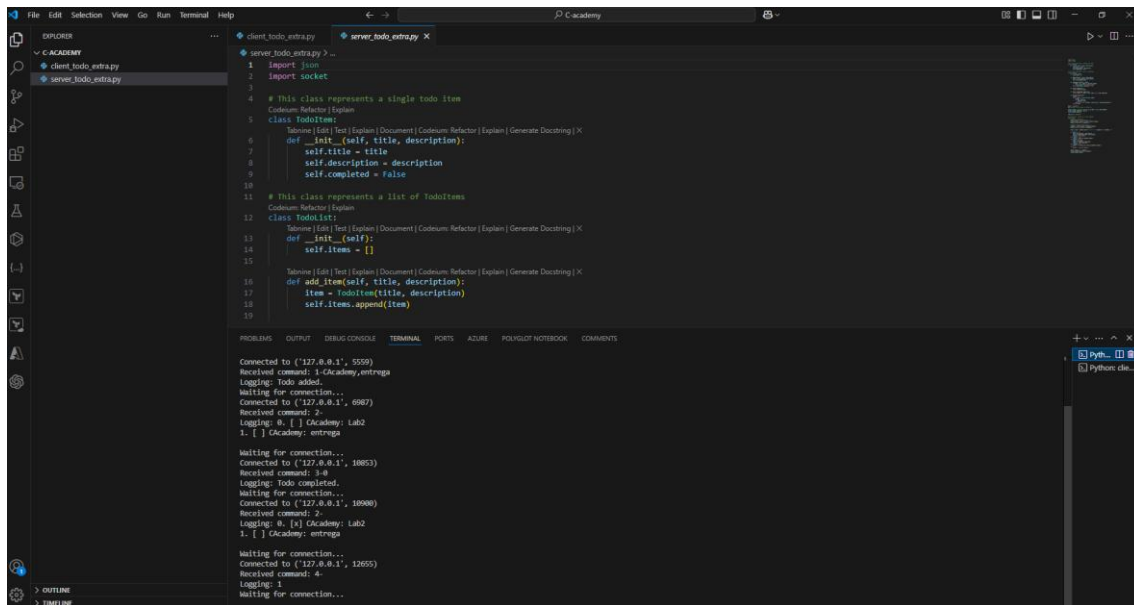
```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 7777 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17     while True:
18         print(""" Todo List Menu """)
19         print("1. Add todo")
20         print("2. Print todos")
21         print("3. Complete todo")
22         print("4. Count incomplete todos")
23         print("5. Exit")
24         choice = input("Enter your choice: ")
25         if choice == "1":
26             title = input("Enter title: ")
27             description = input("Enter description: ")
28             command = f"1:{title}:{description}"
29             send_command(command)
30         elif choice == "2":
31             command = "2:"
32             send_command(command)
33             result = send_command(command)
34             print(result)
35         elif choice == "3":
36             index = input("Enter index: ")
37             command = f"3:{index}"
38             send_command(command)
39         elif choice == "4":
40             command = "4:"
41             send_command(command)
42             result = send_command(command)
43             print(result)
44         elif choice == "5":
45             break
```

The terminal output shows the execution of the client code:

```
Enter your choice: 2
Print todos
Connected to server.
Connection close.
0. [x] CAcademy: Lab2
1. [ ] CAcademy: entrega

** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 4
Counting incomplete todos...
Connected to server.
Connection close.
Incomplete todos: 1
** Todo List Menu **
1. Add todo
2. Print todos
3. Complete todo
4. Count incomplete todos
5. Exit
Enter your choice: 1
```

Visto do lado do servidor.



The screenshot shows the VS Code editor with two files open: `client_todo_extra.py` and `server_todo_extra.py`. The `server_todo_extra.py` file contains the following code:

```
1 import json
2 import socket
3
4 # This class represents a single todo item
5 class TodoItem:
6     def __init__(self, title, description):
7         self.title = title
8         self.description = description
9         self.completed = False
10
11 # This class represents a list of TodoItems
12 class TodoList:
13     def __init__(self):
14         self.items = []
15
16     def add_item(self, title, description):
17         item = TodoItem(title, description)
18         self.items.append(item)
```

The terminal output shows the execution of the server code:

```
Connected to ('127.0.0.1', 5559)
Received command: 1-CAcademy,entrega
Logging: Todo added.
Waiting for connection...
Connected to ('127.0.0.1', 6987)
Received command: 2:
Logging: 0. [x] CAcademy: Lab2
1. [ ] CAcademy: entrega
Waiting for connection...
Connected to ('127.0.0.1', 10853)
Received command: 3-0
Logging: Todo completed.
Waiting for connection...
Connected to ('127.0.0.1', 10900)
Received command: 2:
Logging: 0. [x] CAcademy: Lab2
1. [ ] CAcademy: entrega
Waiting for connection...
Connected to ('127.0.0.1', 12655)
Received command: 4:
Logging: 1
Waiting for connection...
```

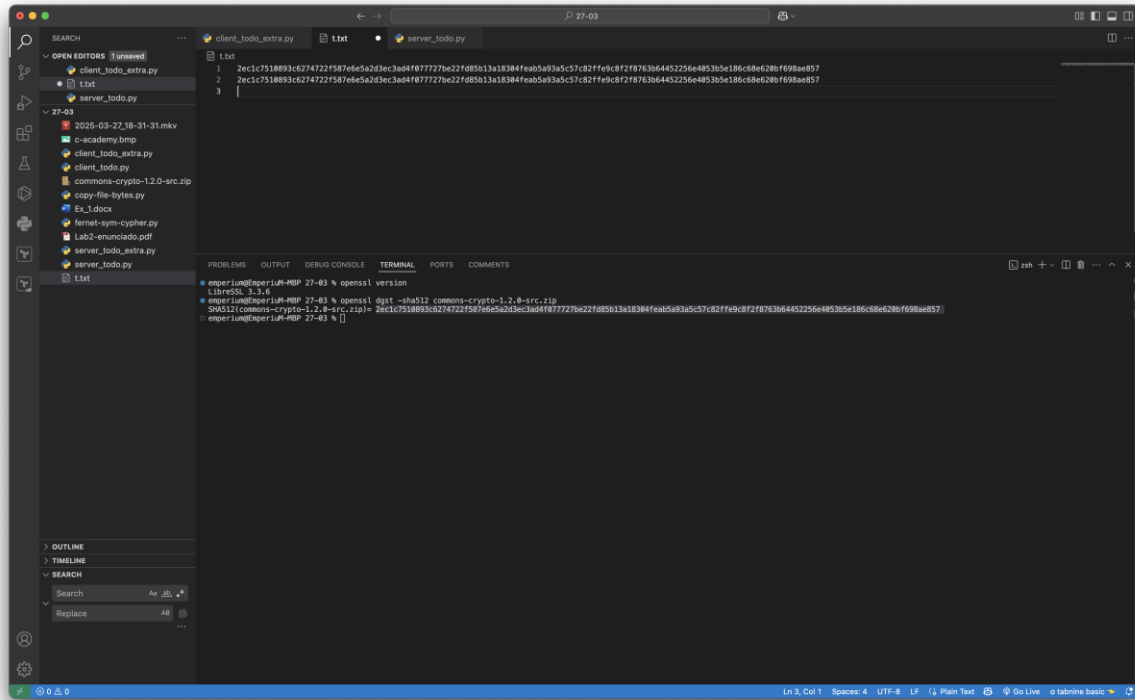
## Laboratório 2 – Grupo A

### Exercício 3

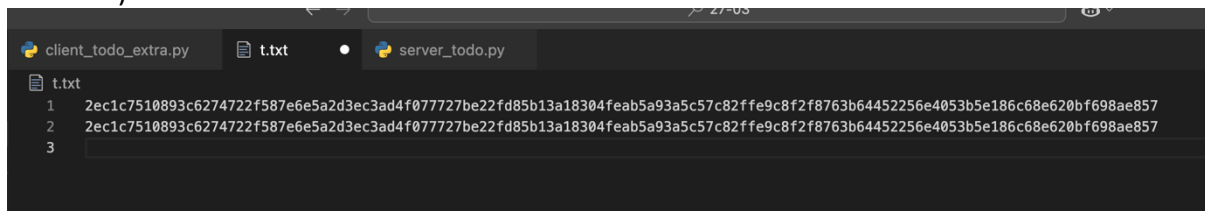
a)

i)

```
$openssl dgst -sha512 commons-crypto-1.2.0-src.zip
SHA512(common-crypto-1.2.0-
src.zip)=2ec1c7510893c6274722f587e6e5a2d3ec3ad4f077727be22fd85b13a18304feab
5a93a5c57c82ffe9c8f2f8763b64452256e4053b5e186c68e620bf698ae857
```

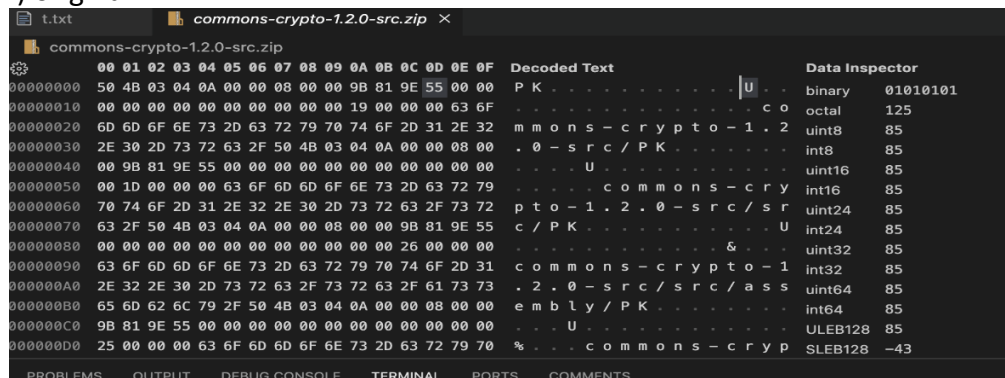


ii)

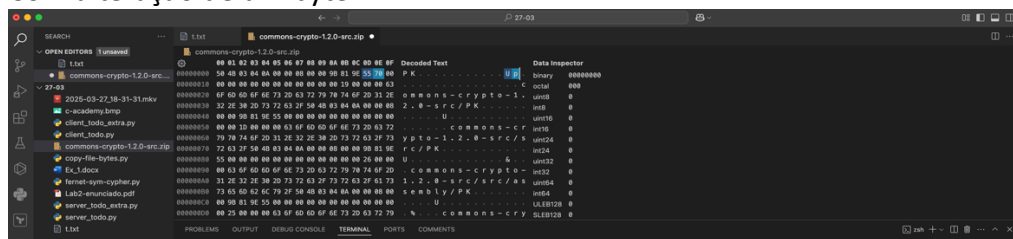


## Laboratório 2 – Grupo A

ii) original



Com alteração de um byte



Comparação de hashes após a alteração do byte:

**Original:**

```
2ec1c7510893c6274722f587e6e5a2d3ec3ad4f077727be22fd85b13a18304
feab5a93a5c57c82ffe9c8f2f8763b64452256e4053b5e186c68e620bf698a
e857
```

**Com alteração do byte:**

```
cd9437ed19dac76bbb3c010aa65413ef99e2a92d9305f434567c838c904389
7ef30c2e3cd849fb79e1c3a08c85b081ec1a6b55838dc3812b5e8d49f97bdc
4e03
```



## Laboratório 2 – Grupo A

iv)

The screenshot shows a VS Code editor with a terminal window. The terminal output shows a successful verification of the file commons-crypto-1.2.0-src.zip using SHA512. The hash matches the expected value.

```
esperim@esperim-MBP:~$ openssl dgst -sha512 commons-crypto-1.2.0-src.zip
SHA512(commons-crypto-1.2.0-src.zip)= 26c1c7518893c6274722f587ede5a2d3ec3ad4f07727b622fd85b13a18384fa05a93a5c57c82ffec8f2f8763864452256a4853b5e186c68e28b6f698ae857
esperim@esperim-MBP:~$
```

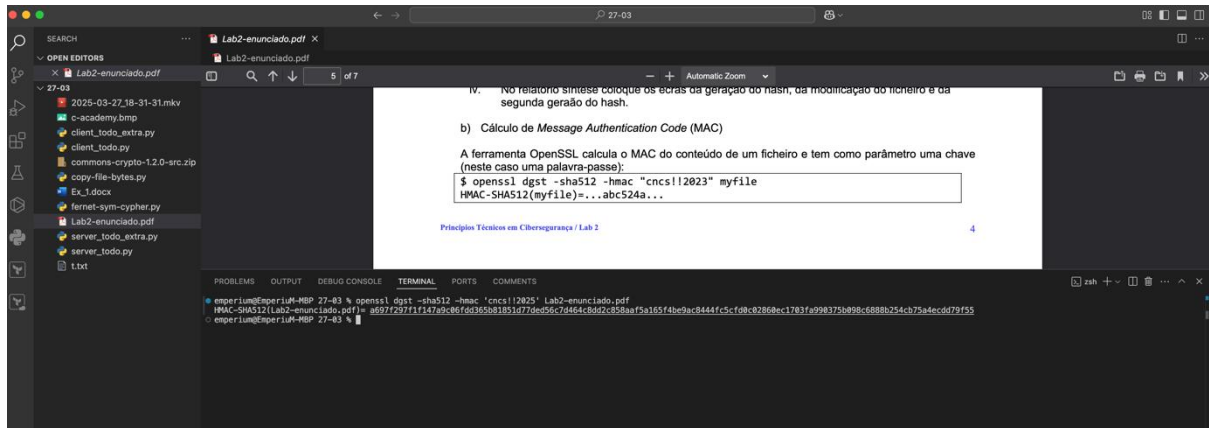
The screenshot shows a VS Code editor with a Data Inspector window. The Data Inspector displays the hex and decoded content of the file commons-crypto-1.2.0-src.zip. The decoded content shows the file name and its SHA512 hash.

Offset	Hex	Decoded Text	Data Inspector
00000000	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	P K	binary 00000000
00000001	50 48 03 04 0A 00 00 00 00 00 00 00 00 00 00 00	ommons - c r y p t o - 1 .	uint8 0
00000002	6F 6D 6D 6F 6E 73 2D 63 72 79 70 74 6F 2D 31 2E	2 . 0 - s r c / P K	uint8 0
00000003	32 2E 38 2D 73 72 63 2F 58 48 03 04 0A 00 00 00	U .	uint8 0
00000004	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ommons - c r	uint8 0
00000005	00 00 1D 00 00 00 00 00 00 00 00 00 00 00 00 00	y p t o - 1 . 2 . 0 - s r c / s	uint24 0
00000006	79 70 74 6F 2D 31 2E 32 2E 38 2D 73 72 63 2F 73	r c / P K	uint24 0
00000007	72 63 2F 58 48 03 04 0A 00 00 00 00 00 00 00 00	U .	uint32 0
00000008	55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ommons - c r y p t o -	uint32 0
00000009	00 63 6F 6D 6F 6E 73 2D 63 72 79 70 74 6F 2D	1 . 2 . 0 - s r c / s r c / a s	uint64 0
0000000A	31 2E 32 2E 38 2D 73 72 63 2F 73 72 63 2F 6A 73	s e a m l y / P K	uint64 0
0000000B	73 65 6D 6E 79 2F 58 48 03 04 0A 00 00 00 00 00	U .	uint64 0
0000000C	00 00 01 9E 55 00 00 00 00 00 00 00 00 00 00 00	S L E B 1 2 8	uint128 0
0000000D	00 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

```
esperim@esperim-MBP:~$ openssl dgst -sha512 commons-crypto-1.2.0-src.zip
SHA512(commons-crypto-1.2.0-src.zip)= cd9437ed19ac76bb3c818ae65413ef99e2a209385f434567c838c9043897ef30c2e3cd849fb79e1c3a88c83a081e1ca6b55838dc3b125e8d49f97bd4e03
esperim@esperim-MBP:~$
```

## Laboratório 2 – Grupo A

b) `$openssl dgst -sha512 -hmac 'cncs!!2023' Lab2-enunciado.pdf`



Resultado:

HMAC-SHA512(Lab2-enunciado.pdf)=  
a697f297f1f147a9c06fdd365b81851d77ded56c7d464c8dd2c858aaf5a165f4be9a  
c8444fc5cfd0c02860ec1703fa990375b098c6888b254cb75a4ecdd79f55

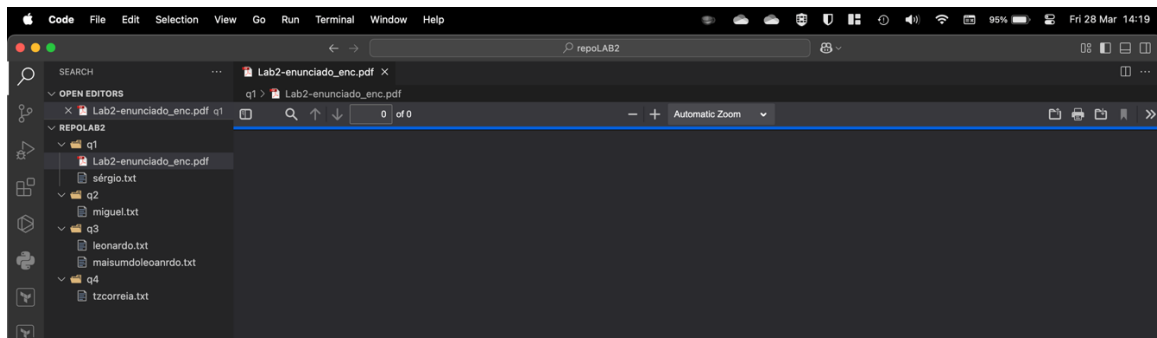
c)

i)

`$openssl enc -aes-256-cbc -in Lab2-enunciado.pdf -out Lab2-enunciado_enc.pdf`  
password usada: 123456

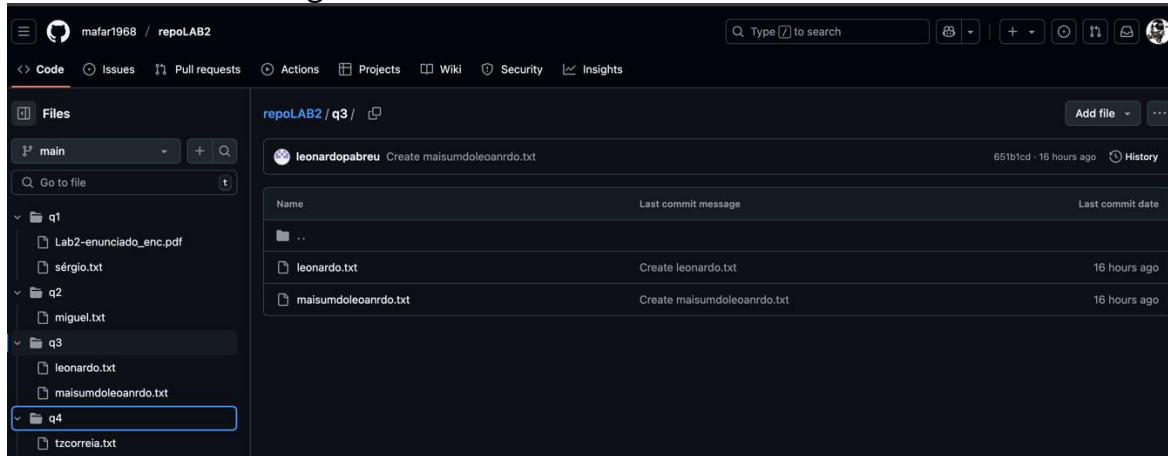


ii) adicionar o Lab2-enunciado\_enc.pdf ao github



## Laboratório 2 – Grupo A

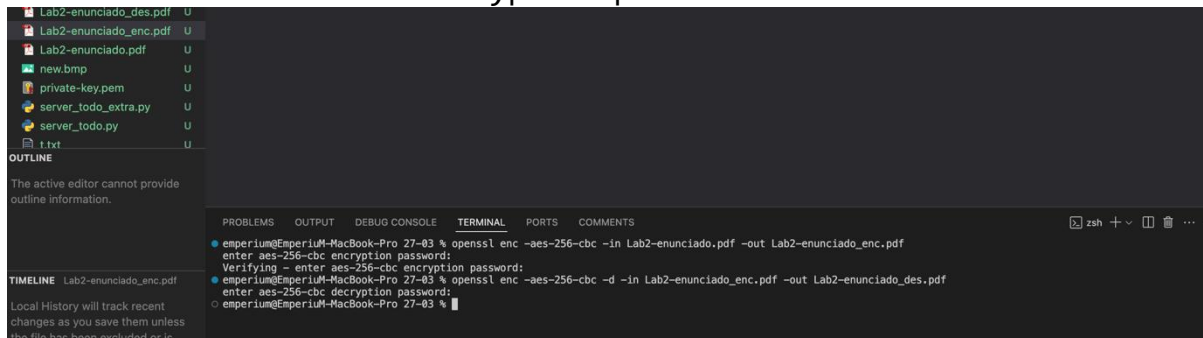
Vista no site do github.com



iii)

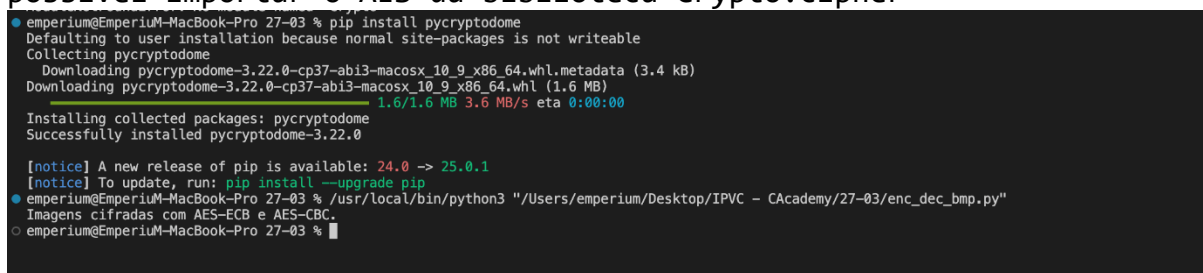
```
$ openssl enc -aes-256-cbc -d -in Lab2-enunciado_enc.pdf -out Lab2-enunciado_des.pdf
```

enter aes-256-cbc decryption password:

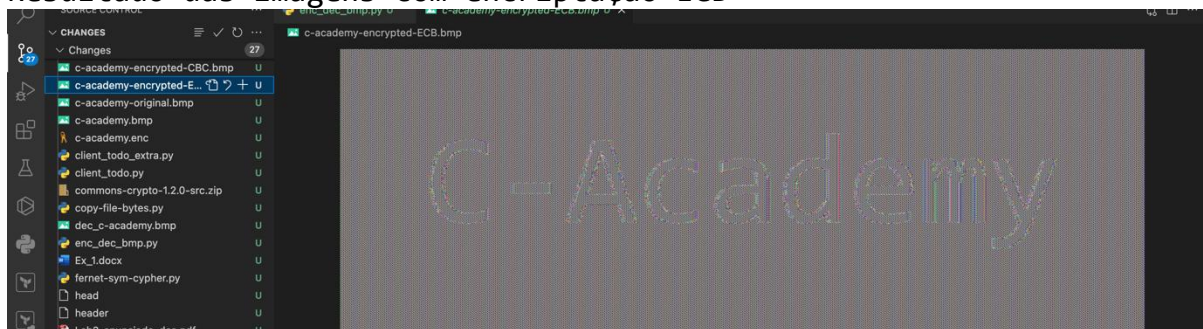


d)

Instalada o pycryptodome (pip install pycryptodome) para que seja possível importar o AES da biblioteca Crypto.cipher

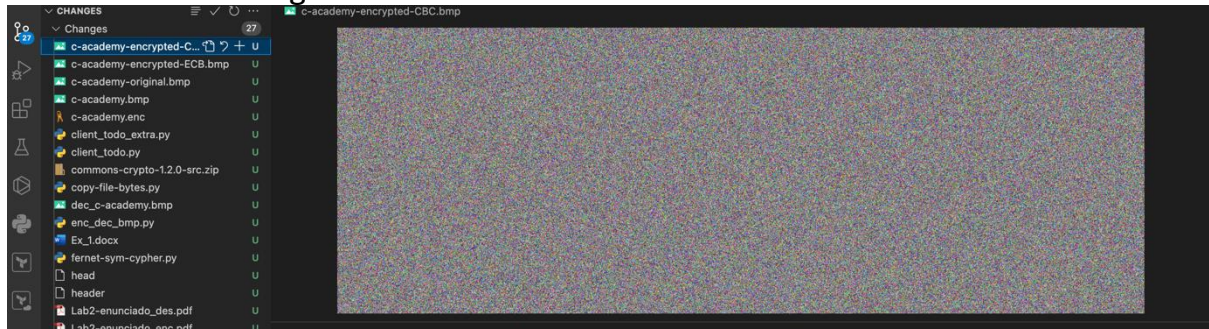


Resultado das imagens com encriptação ECB

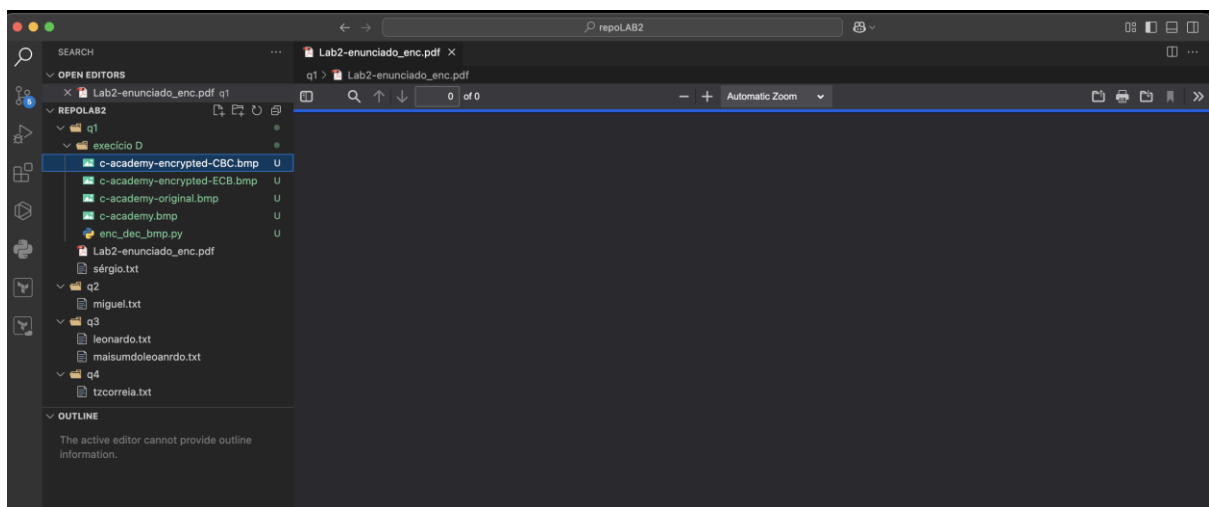


## Laboratório 2 – Grupo A

### Resultado da imagem em CBC

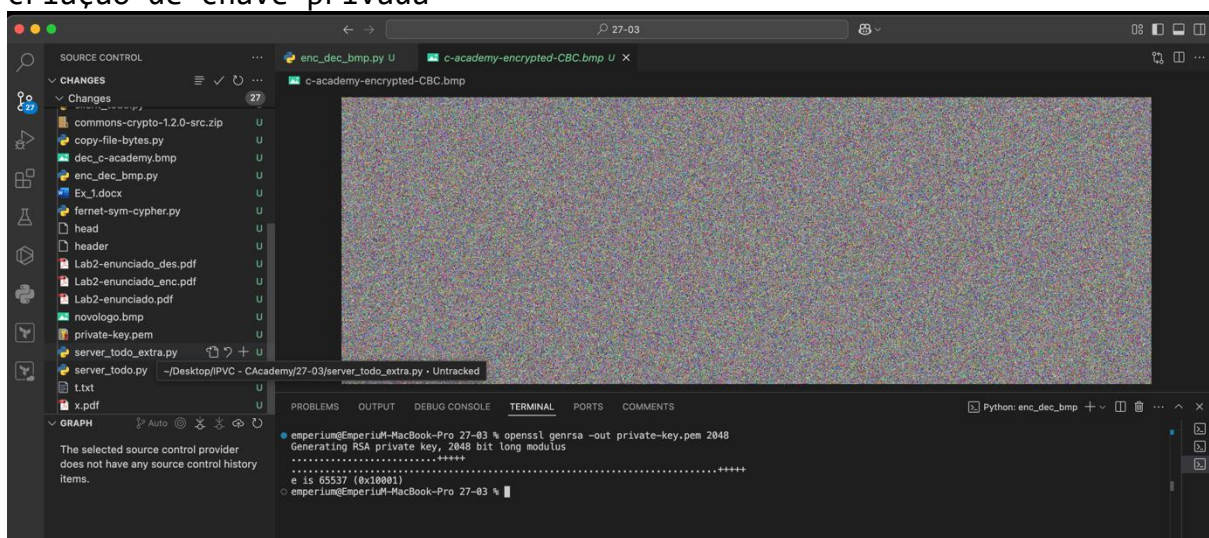


### Preparação do files para o github (*commit* e *push*)



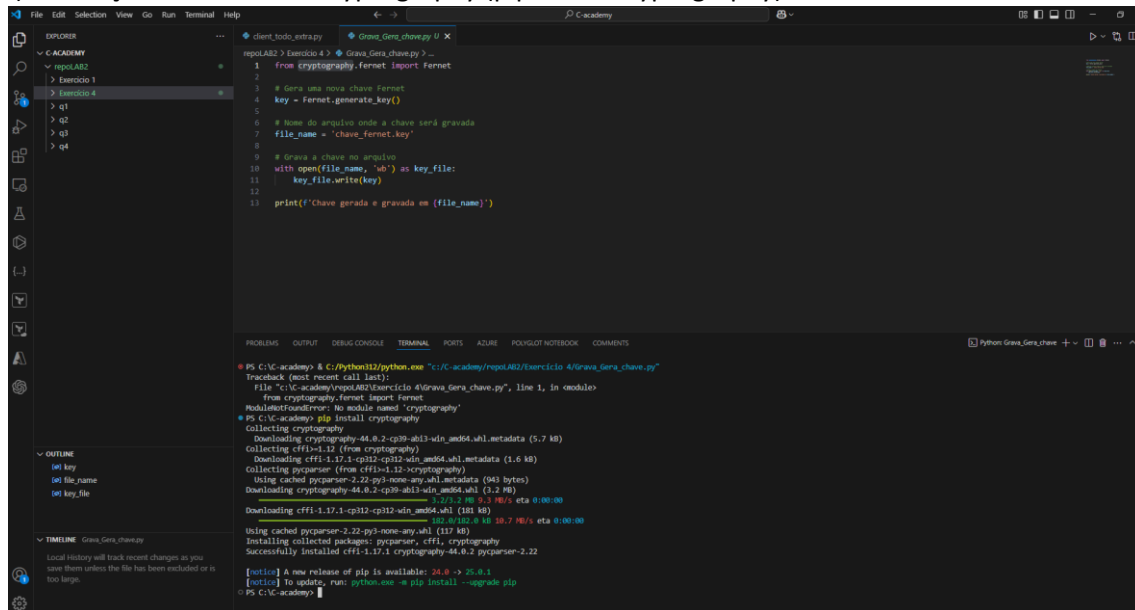
Como é possível verificar, a encriptação ECB não é tão eficaz como a CBC, pois é visível vislumbrar o teor da mensagem original.

### e) criação de chave privada



### Exercício 4

#### i) Instalação da biblioteca cryptography (pip install cryptography)



```

1 from cryptography.fernet import Fernet
2
3 # Gera uma nova chave Fernet
4 key = Fernet.generate_key()
5
6 # Nome do arquivo onde a chave será gravada
7 file_name = 'Chave_fernet.key'
8
9 # Grava a chave no arquivo
10 with open(file_name, 'wb') as key_file:
11     key_file.write(key)
12
13 print(f'Chave gerada e gravada em {file_name}')

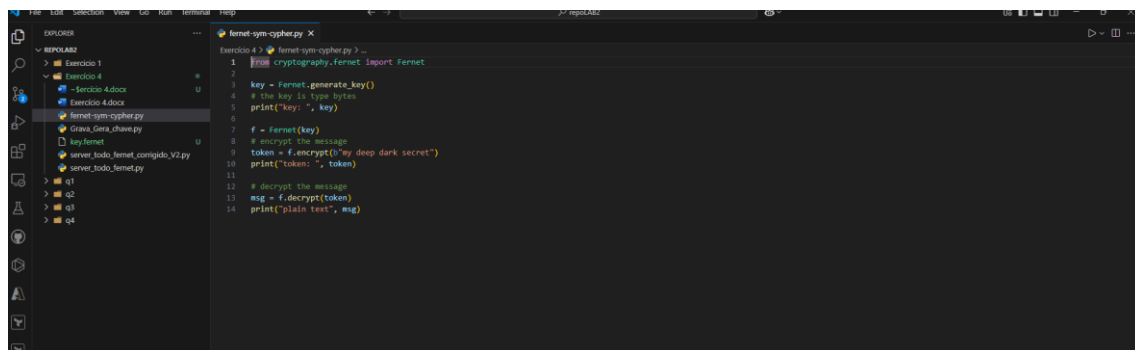
```

```

PS C:\C-academy> pip install cryptography
Collecting cryptography
  Downloading cryptography-44.0.2-cp39-abi3-win_amd64.whl.metadata (5.7 kB)
Collecting cffi>=1.12 (from cryptography)
  Downloading cffi-1.17.1-cp312-cp312-win_amd64.whl.metadata (1.6 kB)
Collecting pycparser (from cffi>=1.12->cryptography)
  Using cached pycparser-2.22-py3-none-any.whl.metadata (943 bytes)
Using cached cryptography-44.0.2-cp39-abi3-win_amd64.whl (3.2 MB)
Installing collected packages: pycparser, cffi, cryptography
Successfully installed cffi-1.17.1 cryptography-44.0.2 pycparser-2.22
[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update: run: python.exe -m pip install --upgrade pip
PS C:\C-academy>

```

#### ii) chave gerada (key.fernet)



```

1 from cryptography.fernet import Fernet
2
3 key = Fernet.generate_key()
4 # the key is type bytes
5 print("key:", key)
6
7 f = Fernet(key)
8 # encrypt the message
9 token = f.encrypt(b"my deep dark secret")
10 print("token:", token)
11
12 # decrypt the message
13 msg = f.decrypt(token)
14 print("plain text:", msg)

```

```

[Running] python -u "c:\C-academy\repoLAB2\Exercício 4\fernet-sym-cypher.py"
key:  b'2Kp8c6geZU39p8K7a9umhDWNrRY3qp k1-ily819vo-'
token:  b'gAAAAABn7XZF7u6AEjUxy7iTDnJA9zWu-u19NpdXCJ2L1xtj96Cyl4A4CvUHu4k0BS1Z_g-Ire8yB4ve4H9P5HMV-zxnouN-1Gz64Mw6D7g1ZTFHCNkpIw8-'
plain text b'my deep dark secret'
[Done] exited with code=0 in 0.076 seconds

```

#### Visualização da chave criada

```

FileNotFoundError: [Errno 2] No such file or directory: 'key.fernet'

[Done] exited with code=1 in 0.086 seconds

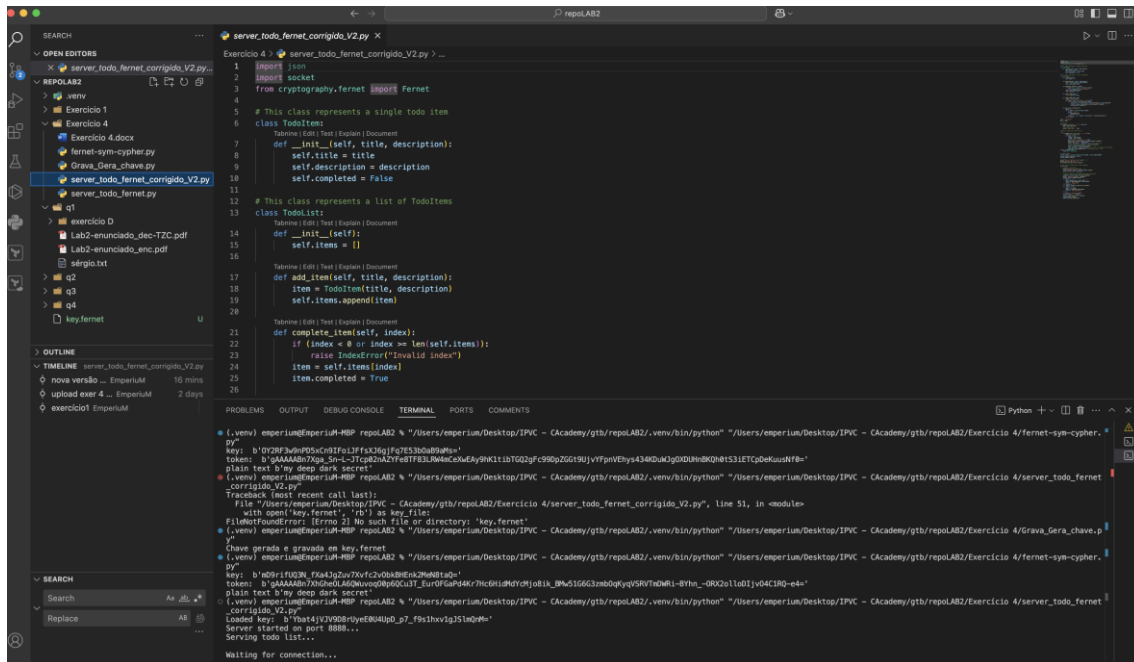
[Running] python -u "c:\C-academy\repoLAB2\Exercício 4\fernet-sym-cypher.py"
key:  b'2Kp8c6geZU39p8K7a9umhDWNrRY3qp k1-ily819vo-'
token:  b'gAAAAABn7XZF7u6AEjUxy7iTDnJA9zWu-u19NpdXCJ2L1xtj96Cyl4A4CvUHu4k0BS1Z_g-Ire8yB4ve4H9P5HMV-zxnouN-1Gz64Mw6D7g1ZTFHCNkpIw8-'
plain text b'my deep dark secret'
[Done] exited with code=0 in 0.076 seconds

```

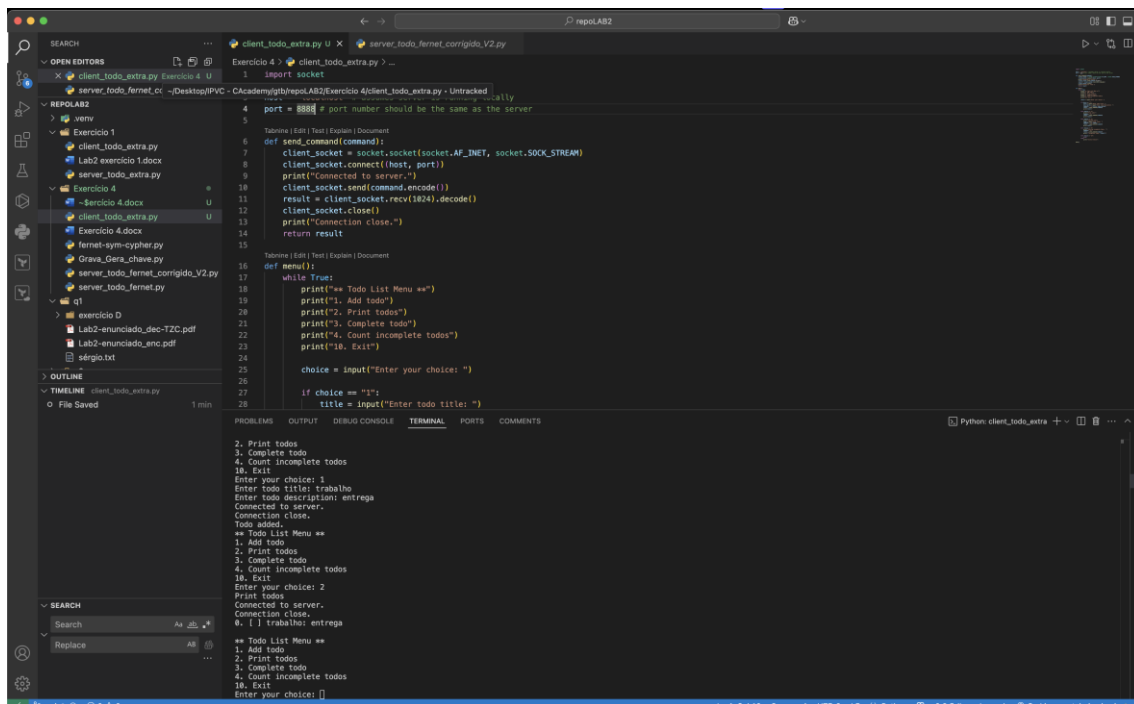


## Laboratório 2 – Grupo A

iii) *Server\_ToDo* a correr e a aguardar ligação do cliente

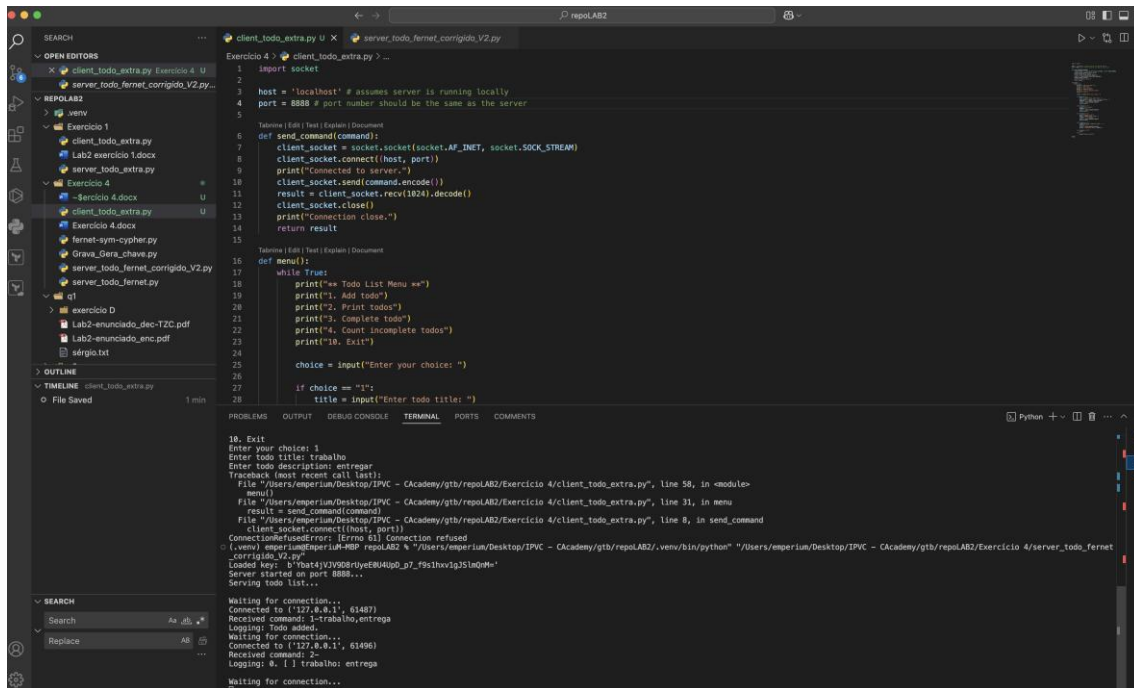


iv) Cliente a correr com tarefa adicionada e listada



## Laboratório 2 – Grupo A

Visualização, do lado do servidor, com a entrada



The screenshot shows a VS Code editor with two Python files: `client_todo_extra.py` and `server_todo_fernet_corrigido_V2.py`. The `client_todo_extra.py` file contains the following code:

```
1 import socket
2
3 host = 'localhost' # assumes server is running locally
4 port = 8888 # port number should be the same as the server
5
6 def send_command(command):
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9     print("Connected to server.")
10    client_socket.send(command.encode())
11    result = client_socket.recv(1024).decode()
12    client_socket.close()
13    print("Connection close.")
14    return result
15
16 def menu():
17     while True:
18         print("\nTodo List Menu\n")
19         print("1. Add todo")
20         print("2. Print todos")
21         print("3. Complete todo")
22         print("4. Count incomplete todos")
23         print("10. Exit")
24         choice = input("Enter your choice: ")
25         if choice == "1":
26             title = input("Enter todo title: ")
```

The `server_todo_fernet_corrigido_V2.py` file contains the following code:

```
1 import socket
2 import fernet
3
4 host = 'localhost'
5 port = 8888
6
7 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 server_socket.bind((host, port))
9 server_socket.listen(5)
10 print("Server started on port 8888...")
11
12 while True:
13     client_socket, addr = server_socket.accept()
14     print("Client connected from", addr)
15     data = client_socket.recv(1024)
16     if not data:
17         break
18     command = data.decode()
19     print("Received command:", command)
20     result = send_command(command)
21     client_socket.send(result.encode())
22     client_socket.close()
23     print("Connection closed")
```

The terminal window shows the output of the server, including the following messages:

```
Server started on port 8888...
Client connected from ('127.0.0.1', 61497)
Received command: 1-trabalho,entrega
Logging: Todo added.
Waiting for connection...
Client connected from ('127.0.0.1', 61496)
Received command: 2-
Logging: 0. [1] trabalho: entrega
Waiting for connection...
Client connected from ('127.0.0.1', 61497)
Received command: 3-trabalho,entrega
Logging: Todo added.
Waiting for connection...
Client connected from ('127.0.0.1', 61496)
Received command: 2-
Logging: 0. [1] trabalho: entrega
Waiting for connection...
```