REVISÃO – Parte 1

Criando Servidor Web

```python
# Importa o módulo http.server
import http.server
import socketserver

# Define a porta a ser utilizada
porta = 8000

# Configura o manipulador (handler) para o servidor
handler = http.server.SimpleHTTPRequestHandler

# Cria um servidor na porta especificada
with socketserver.TCPServer(("", porta), handler) as httpd:
    print(f"Servidor iniciado na porta {porta}")
    # Mantém o servidor em execução
    httpd.serve_forever()
```

REVISÃO – Parte 1

Definindo página de inicialização do Servidor Web
- Manipulando método **list_directory** da Classe **MyHandler**

```python
import os
from http.server import SimpleHTTPRequestHandler
import socketserver

class MyHandler(SimpleHTTPRequestHandler):
    def list_directory(self, path):
        try:
            # Tenta abrir o arquivo index.html
            f = open(os.path.join(path, 'index.html'), 'r')
            # Se existir, envia o conteúdo do arquivo
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(f.read().encode('utf-8'))
            f.close()
            return None
        except FileNotFoundError:
            pass

        return super().list_directory(path)
```

REVISÃO – Parte 1

Abrindo arquivos **html** para exibí-los **como resposta do servidor**

```
class MyHandler(SimpleHTTPRequestHandler):
    def list_directory(self, path):
        try:
            # Tenta abrir o arquivo index.html
            f = open(os.path.join(path, 'index.html'), 'r')
            # Se existir, envia o conteúdo do arquivo
```

```
def do_GET(self):
    if self.path == '/login':
        # Tenta abrir o arquivo login.html
        try:
            with open(os.path.join(os.getcwd(), 'login.html'), 'r') as login_file:
                content = login_file.read()
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(content.encode('utf-8'))
        except FileNotFoundError:
            self.send_error(404, "File not found")
```

REVISÃO – Parte 1

Exibindo **resposta do servidor com página html**

```
class MyHandler(SimpleHTTPRequestHandler):
    def list_directory(self, path):
        try:
            # Tenta abrir o arquivo idx.html
            f = open(os.path.join(path, 'idx.html'), 'r')
            # Se existir, envia o conteúdo do arquivo
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(f.read().encode('utf-8'))
            f.close()
```

Exibindo **resposta do servidor com mensagem apenas**

```
if self.usuario_existente(login, senha):
    # Responde ao cliente indicando que o usuário logou com sucesso
    self.send_response(200)
    self.send_header("Content-type", "text/html; charset=utf-8")
    self.end_headers()
    mensagem = f"Usuário {login} logado com sucesso!!!"
    self.wfile.write(mensagem.encode('utf-8'))
```

REVISÃO – Parte 1

Exibindo **resposta do servidor – redirecionando para outra rota**

```
# Verifica se o login já existe no arquivo
if any(line.startswith(f"{login};") for line in open('dados_login.txt', 'r', encoding='utf-8')):
    # Redireciona o cliente para a rota "/login_failed"
    self.send_response(302)
    self.send_header('Location', '/login_failed')
    self.end_headers()
    return  # Adicionando um return para evitar a execução do restante do código
```

REVISÃO – Parte 1

Exibindo **resposta do servidor – redirecionando para outra rota e enviando parâmetros na URL**

```
# Redireciona o cliente para a rota "/cadastro" com os dados de login e senha
self.send_response(302)
self.send_header('Location', f'/cadastro?login={login}&senha={senha}')
self.end_headers()

return  # Adicionando um return para evitar a execução do restante do código
```

REVISÃO – Parte 1

Definindo Rotas – End Points

- Novas rotas manipulando método **do_GET** da Classe **MyHandler**

```
def do_GET(self):
    if self.path == '/login':


    elif self.path == '/login_failed':


    elif self.path.startswith('/cadastro'):
```

REVISÃO – Parte 1

Definindo Rotas – End Points

- Novas rotas manipulando método **do_POST** da Classe **MyHandler**

```
def do_POST(self):
    # Verifica se a rota é "/enviar_login"
    if self.path == '/enviar_login':

    elif self.path.startswith('/confirmar_cadastro'):
```

REVISÃO – Parte 1

Extraindo dados recebidos no método **do_GET**

```python
from urllib.parse import urlparse, parse_qs



elif self.path.startswith('/cadastro'):
    # Extraindo os parâmetros da URL
    query_params = parse_qs(urlparse(self.path).query)
    login = query_params.get('login', [''])[0] #Se o parametro login estiver presente retornará o primeiro paramtro devido ao [0]
                                               #Caso o parametro login não esteja presente ele retorna '' devido ao ['']
    senha = query_params.get('senha', [''])[0]
```

REVISÃO – Parte 1

Extraindo dados recebidos no método **do_POST**

```python
from urllib.parse import urlparse, parse_qs


def do_POST(self):
    # Verifica se a rota é "/enviar_login"
    if self.path == '/enviar_login':
        # Obtém o comprimento do corpo da requisição
        content_length = int(self.headers['Content-Length'])
        # Lê o corpo da requisição
        body = self.rfile.read(content_length).decode('utf-8')
        # Parseia os dados do formulário
        form_data = parse_qs(body, keep_blank_values=True)

        login = form_data.get('email', [''])[0]
        senha = form_data.get('senha', [''])[0]
```

REVISÃO – Parte 1

Gravando dados em um **arquivo TXT**

```python
# Armazena os dados em um arquivo TXT
with open('dados_login.txt', 'a') as file:
    login = form_data.get('email', [''])[0]
    senha = form_data.get('senha', [''])[0]
    file.write(f"{login};{senha}\n")
```

Lendo dados em um **arquivo TXT**

```python
def usuario_existente(self, login, senha):
    # Verifica se o login já existe no arquivo

    with open('dados_login.txt', 'r', encoding='utf-8') as file:
        for line in file:
            if line.strip():
                stored_login, stored_senha_hash, stored_nome = line.strip().split(';')
                if login == stored_login:
```

REVISÃO – Parte 1

DESAFIO:

1 – Criar uma nova rota em **do_GET** chamada **/turmas**

2 – Crie um formulário HTML com o título "**Cadastro de Turmas**" com os campos "**codigo**" e "**descrição**" para responder na nova rota.

3 – Crie uma nova rota em **do_POST** chamada **/cad_turma** para receber os dados do formulário.

4 – Armazene os dados recebidos na rota acima em um arquivo txt com o nome **dados_turma.txt** separando os campo com ";" (ponto e vírgula)

5 – Gere respostas em um arquivo **.html** quando os dados forem armazenas com sucesso.

6 – Refaça o procedimento dos itens 1 ao 5 para cadastrar **Atividades** com os seguintes critérios:
- rota para exibir a página "**Cadastro de Atividades**": **/atividades**
- rota para receber os dados de "Atividades": **/cad_atividade**
- nome do arquivo **dados_atividade.txt**
- campos: "**codigo**" e "**descricao**" separados por ";"