



AUTOMATE YOUR TASKS WITH THESE 5 PYTHON SCRIPTS

File Transfer Script

System Monitoring Script

Web Scraping Script

Email Automation Script

Password Manager Script



FILE TRANSFER SCRIPT

```
import socket

# create socket
s = socket.socket()

# bind socket to a address and port
s.bind(('localhost', 12345))

# put the socket into listening mode
s.listen(5)

print('Server listening...')

# forever loop to keep server running
while True:
    # establish connection with client
    client, addr = s.accept()
    print(f'Got connection from {addr}')

    # receive the file name
    file_name = client.recv(1024).decode()

    try:
        # open the file for reading in binary
        with open(file_name, 'rb') as file:
            # read the file in chunks
            while True:
                chunk = file.read(1024)
                if not chunk:
                    break
                # send the chunk to the client
                client.sendall(chunk)

        print(f'File {file_name} sent successfully')
    except FileNotFoundError:
        # if file not found, send appropriate message
        client.sendall(b'File not found')
        print(f'File {file_name} not found')

# close the client connection
client.close()
```

SYSTEM MONITORING

SCRIPT

```
import psutil

# Get the current CPU usage
cpu_usage = psutil.cpu_percent()

# Get the current memory usage
memory_usage = psutil.virtual_memory().percent

# Get the current disk usage
disk_usage = psutil.disk_usage("/").percent

# Get the network activity
# Get the current input/output data rates for each network interface
io_counters = psutil.net_io_counters(pernic=True)
for interface, counters in io_counters.items():
    print(f"Interface {interface}:")
    print(f"  bytes sent: {counters.bytes_sent}")
    print(f"  bytes received: {counters.bytes_recv}")

# Get a list of active connections
connections = psutil.net_connections()
for connection in connections:
    print(f"{connection.laddr} <-> {connection.raddr} ({connection.status})")

# Print the collected data
print(f"CPU usage: {cpu_usage}%")
print(f"Memory usage: {memory_usage}%")
print(f"Disk usage: {disk_usage}%")
```

WEB SCRAPING SCRIPT **(THE MOST USED)**

```
import requests
from bs4 import BeautifulSoup

# Fetch a web page
page = requests.get("http://www.example.com")

# Parse the HTML content
soup = BeautifulSoup(page.content, "html.parser")

# Find all the links on the page
links = soup.find_all("a")

# Print the links
for link in links:
    print(link.get("href"))
```

EMAIL AUTOMATION

SCRIPT

```
import smtplib
from email.mime.text import MIMEText

# Set the SMTP server and login credentials
smtp_server = "smtp.gmail.com"
smtp_port = 587
username = "your@email.com"
password = "yourpassword"

# Set the email parameters
recipient = "recipient@email.com"
subject = "Test email from Python"
body = "This is a test email sent from Python."

# Create the email message
msg = MIMEText(body)
msg["Subject"] = subject
msg["To"] = recipient
msg["From"] = username

# Send the email
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()
server.login(username, password)
server.send_message(msg)
server.quit()
```

PASSWORD MANAGER

SCRIPT

```
import secrets
```

```
import string
```

```
# Generate a random password
```

```
def generate_password(length=16):
```

```
    characters = string.ascii_letters + string.digits + string.punctuation
```

```
    password = "".join(secrets.choice(characters) for i in range(length))
```

```
    return password
```

```
# Store a password in a secure way
```

```
def store_password(service, username, password):
```

```
    # Use a secure hashing function to store the password
```

```
    hashed_password = hash_function(password)
```

```
    # Store the hashed password in a database or file
```

```
    with open("password_database.txt", "a") as f:
```

```
        f.write(f"{service},{username},{hashed_password}\n")
```

```
# Retrieve a password
```

```
def get_password(service, username):
```

```
    # Look up the hashed password in the database or file
```

```
    with open("password_database.txt") as f:
```

```
        for line in f:
```

```
            service_, username_, hashed_password_ = line.strip().split(",")
```

```
            if service == service_ and username == username_:
```

```
                # Use a secure hashing function to compare the stored password with  
the provided password
```

```
                if hash_function(password) == hashed_password_:
```

```
                    return password
```

```
    return None
```