Input/Output and OS Interaction

Basic Python for Cybersecurity

Input/Output

Secara umum, program menyediakan berbagai versi input dan output sesuai dengan *use case* dan pihak yang berinteraksi (manusia/sistem lain).

Input

Metode	User	Deskripsi	Implementasi
Input interaktif	Manusia	User memberikan input saat program berjalan	`input("Enter your name: ")`
Command-line args	Manusia/sistem	Parameter yang diteruskan ketika program akan dijalankan	`sys.argv[1]` using `import sys`
File	Manusia/sistem	Membaca file eksternal sebagai sumber input	`open("log.txt").read()`
Variabel env	Sistem	Variabel untuk config dan key/secret	`os.getenv("API_KEY")`

Output

Metode	User	Deskripsi	Implementasi
Console	Manusia/sistem	Program menuliskan informasi di console ketika sedang berjalan. Paling mudah diimplementasikan. *Note*: output console akan hilang jika program mati/restart atau user keluar dari terminal/shell.	`print("[INFO] Database connected!")`
Output File	Manusia/sistem	Program menuliskan data ke dalam file yang sifatnya persisten.	`open("20250430101948_monitoring.log", "w").write(data)`
Return value	Sistem	Program mengembalikan value kepada pihak yang menjalankannya.	`return result`

Interaksi dengan OS (Operating System)

OS pada dasarnya adalah "lingkungan" dari aplikasi. Analoginya: jika program/aplikasi adalah toko, maka OS adalah mall. Toko bersinergi dengan mall untuk memberikan pengalaman terbaik bagi pengguna. Untuk itu, ada standar interaksi yang dapat dilakukan.

Dalam konteks program-OS, program umumnya berurusan dengan hal terkait:

- navigasi path
- manajemen resource: CPU, memory, dll
- manajemen akses
- manajemen process/subprocess

Contoh instruksi terkait OS di Python:

Perintah	Syntax Python
List files	`os.listdir(".")`
Jalankan perintah (seolah-olah user ketik di shell)	`os.system("ls -l")` or `subprocess.run()`
Cek file	`os.path.exists("config.txt")`
Cek folder saat ini	`os.getcwd()`

Perintah	Syntax Python
Pindah folder	`os.chdir("/home/user")`

Tentunya ada lebih banyak instruksi yang tersedia. Coba eksplorasi dan jalankan di komputer kamu.

Interaksi Antarprogram

Program Python dapat berinteraksi dengan program Python lain melalui beberapa mekanisme.

Method	Use Case	Example
Import as module	Code digunakan ulang	`import utils`
Subprocess	Satu program menjalankan program lain, peran sebagai eksekutor mewakili user	`subprocess.run(["python3", "tool.py"])`
Socket/network	Dua program memiliki interface (HTTP, websocket, dll) dan bertukar data	`socket.socket()` (advanced)
Inter-process file	Berbagi via files atau logs	Read/write file yang sama

Manfaat interaksi antarprogram

- Setiap developer dapat berkontribusi di bagian tertentu
- Mendorong kolaborasi
- Mendorong spesialisasi program, lebih mudah dikelola
- Mendorong desain modular dengan banyak pilihan, misal kita bisa memilih library untuk fungsi HTTP request

Praktik: Log Analysis Pipeline (Extraction + Reporting)

Kali ini kamu akan membuat pipeline analisis log. Prosesnya dijalankan oleh beberapa program yang berkesinambungan.

Untuk kasus ini, pipeline terdiri dari dua program:

1. `Extractor` --> mengambil data sensitif dari log, persis dengan Chapter 07, perbedaannya adalah: file log terpisah, return value, dan tambahan instruksi untuk melanjutkan ke proses berikutnya
2. `Reporter` --> menerima input dari `Extractor`, lalu mengirimkan datanya ke service lain via HTTP request

- File log `20250601_001.log`

```
[2025-06-05 10:00:12] INFO  Starting service on port 8080
[2025-06-05 10:00:13] INFO  Connected to database at 127.0.0.1:5432
[2025-06-05 10:00:14] INFO  Environment: production
[2025-06-05 10:00:15] DEBUG Loaded configuration file: config.yml
[2025-06-05 10:00:16] INFO  Health check: OK
[2025-06-05 10:00:20] INFO  Received request: POST /api/v1/login
[2025-06-05 10:00:20] DEBUG Request payload: {"username": "admin", "password": "hunter2"}
[2025-06-05 10:00:21] INFO  Auth success for user admin
[2025-06-05 10:00:30] INFO  Received request: GET /api/v1/profile
[2025-06-05 10:00:30] DEBUG Auth token: Bearer abcdef1234567890abcdef
[2025-06-05 10:00:32] INFO  Request processed in 34ms
[2025-06-05 10:00:35] INFO  Scheduled job 'backup' started
[2025-06-05 10:00:35] INFO  Backup target: /var/backups/app
[2025-06-05 10:00:40] INFO  Received request: POST /api/v1/api-key
[2025-06-05 10:00:40] DEBUG Generated API key for user 'alice': sk_live_78tghasdhb7126asdad
[2025-06-05 10:00:41] INFO  Key creation successful
[2025-06-05 10:00:50] WARN  High memory usage detected
[2025-06-05 10:00:55] INFO  Received request: GET /api/v1/report
[2025-06-05 10:00:55] DEBUG Report parameters: {"date": "2025-06-01", "format": "csv"}
[2025-06-05 10:00:56] INFO  Report generated successfully
[2025-06-05 10:01:00] INFO  Received request: POST /api/v1/login
[2025-06-05 10:01:00] DEBUG Request payload: {"username": "bob", "password": "12345678"}
[2025-06-05 10:01:01] INFO  Auth failed for user bob
[2025-06-05 10:01:05] INFO  Received request: DELETE /api/v1/user
[2025-06-05 10:01:05] DEBUG Payload: {"user_id": "789", "confirmed": true}
[2025-06-05 10:01:06] INFO  User 789 deleted by admin
[2025-06-05 10:01:10] INFO  Received request: POST /api/v1/settings
[2025-06-05 10:01:10] DEBUG Payload: {"email": "test@example.com", "password": "qwerty123!"}
[2025-06-05 10:01:11] INFO  Settings updated
[2025-06-05 10:01:15] INFO  Gracefully shutting down
[2025-06-05 10:01:16] INFO  Service stopped
```

- File `extractor.py`

```python
# extractor.py
# For educational purposes only

import sys
import json
import subprocess

SENSITIVE_KEYS = ["username", "password", "API", "token"]

def extract_sensitive_data(filepath):
    findings = {}

    try:
        with open(filepath, "r") as file:
            lines = file.readlines()

        for keyword in SENSITIVE_KEYS:
            findings[keyword] = []

            for line in lines:
                if keyword in line:
                    findings[keyword].append(line.strip())
    except Exception as e:
        print(f"error: {str(e)}")
        sys.exit(1)

    return findings

def send_to_reporter(data):
    json_data = json.dumps(data)

    subprocess.run(["python3", "reporter.py", json_data])

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python3 extractor.py <logfile>")
        sys.exit(1)

    log_path = sys.argv[1]
    print(f"processing {log_path}...")

    try:
        sensitive_findings = extract_sensitive_data(log_path)
        print("log analysis success. moving to pipeline...")
        send_to_reporter(sensitive_findings)
    except Exception as e:
        print(f"error: {str(e)}")
```

- File `reporter.py`

```python
# reporter.py
# For educational purposes only

import sys
import json
import requests

REPORT_ENDPOINT = "https://httpbin.org/post" # sebagai mock endpoint, bisa diganti ke enpoint lain

def send_report(data, endpoint):
    try:
        print(f"sending data to {endpoint}...")
        response = requests.post(endpoint, json=data)
        print(f"server response: {response.status_code}")
        if (response.status_code >= 200 and response.status_code < 300):
            print(response.json())
        else:
            print("failed to send report")
    except Exception as e:
        print(f"error: {str(e)}")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python3 reporter.py <json_data>")
        sys.exit(1)

    try:
        findings = json.loads(sys.argv[1])
        send_report(findings, REPORT_ENDPOINT)
    except json.JSONDecodeError:
        print("invalid JSON input")
    except Exception as e:
        print(f"error: {str(e)}")
```

Extra Challenge

- Asumsikan ada file log yang di-generate setiap jam, buat agar pipeline tersebut selalu jalan, mengecek apakah ada file log terbaru (bisa auto deteksi file lama di-skip), lalu memprosesnya.
- Tantangan mandiri: buat program Python yang bisa mengkloning dirinya sendiri, alias menjalankan subprocess untuk memanggil dirinya sendiri :)

Referensi

- Modul `os` Python: https://www.w3schools.com/python/module_os.asp
- Modul `subprocess` Python: https://www.geeksforgeeks.org/python/python-subprocess-module/
- Fork bomb: https://id.wikipedia.org/wiki/Fork_bomb