```
import os
import time

# Criar sandbox (uma pasta isolada)
sandbox = "sandbox_vm"
os.makedirs(sandbox, exist_ok=True)

# Criar honeyfiles (arquivos isca dentro da sandbox)
honeyfiles = ["senha.txt", "dados_banco.xlsx", "chave_api.key"]

for f in honeyfiles:
    path = os.path.join(sandbox, f)
    if not os.path.exists(path):
        with open(path, "w") as file:
            file.write("ARQUIVO ISCA - ACESSO SUSPEITO\n")

print("Sandbox e Honeyfiles criados com sucesso!")

# Função para monitorar os honeyfiles
def monitorar():
    print("\nMonitoramento iniciado (Ctrl+C para parar)...")
    timestamps = {f: os.path.getmtime(os.path.join(sandbox, f)) for f in honeyfiles}

    try:
        while True:
            time.sleep(2)
            for f in honeyfiles:
                path = os.path.join(sandbox, f)
                if os.path.getmtime(path) != timestamps[f]:
                    print(f"[ALERTA] O arquivo {f} foi modificado!")
                    timestamps[f] = os.path.getmtime(path)
    except KeyboardInterrupt:
        print("\nMonitoramento encerrado.")

monitorar()
```

1. Criar a sandbox (mini VM simulada)

```
sandbox = "sandbox_vm"
os.makedirs(sandbox, exist_ok=True)
```

- Aqui criamos uma **pasta chamada `sandbox_vm`**.

- Ela simula a ideia de uma **mini máquina virtual isolada** onde o antivírus deixaria o vírus rodar sem afetar o sistema real.

- O `exist_ok=True` serve para não dar erro caso a pasta já exista.

---

2. Criar os honeyfiles (arquivos isca)

```
honeyfiles = ["senha.txt", "dados_banco.xlsx", "chave_api.key"]

for f in honeyfiles:
    path = os.path.join(sandbox, f)
    if not os.path.exists(path):
        with open(path, "w") as file:
            file.write("ARQUIVO ISCA - ACESSO SUSPEITO\n")
```

- Aqui definimos uma lista de arquivos **falsos**, que parecem importantes (senhas, dados bancários, chave API).

- Esses arquivos são colocados **dentro da sandbox**.

- O conteúdo é só um texto: `"ARQUIVO ISCA - ACESSO SUSPEITO"`.

- O antivírus usa isso para enganar um malware → se ele tocar nesses arquivos, já é considerado suspeito.

---

3. Mensagem de confirmação

```
print("Sandbox e Honeyfiles criados com sucesso!")
```

- Apenas informa ao usuário que a parte inicial foi concluída.

---

4. Monitoramento dos arquivos

```
def monitorar():
    print("\nMonitoramento iniciado (Ctrl+C para parar)...")
    timestamps = {f: os.path.getmtime(os.path.join(sandbox, f)) for
f in honeyfiles}
```

- Criamos uma função `monitorar()`.
- Ela guarda a **data/hora da última modificação** de cada arquivo (`getmtime`).
- Assim podemos saber depois se alguém mexeu neles.

---

5. Loop de verificação

```
try:
    while True:
        time.sleep(2)
        for f in honeyfiles:
            path = os.path.join(sandbox, f)
            if os.path.getmtime(path) != timestamps[f]:
                print(f"[ALERTA] O arquivo {f} foi modificado!")
                timestamps[f] = os.path.getmtime(path)
```

- `while True` → loop infinito, fica sempre verificando.
- `time.sleep(2)` → espera 2 segundos entre cada checagem (pra não pesar).
- Para cada arquivo, ele vê se a data de modificação mudou.
- Se mudou → significa que alguém **alterou o honeyfile** → logo o programa avisa com um **[ALERTA]**.
- Depois atualiza o timestamp para continuar monitorando.

---

6. Encerrando o monitoramento

```
except KeyboardInterrupt:
    print("\nMonitoramento encerrado.")
```

- Se você apertar `Ctrl + C`, o programa sai do loop de forma controlada.

Resumindo a lógica:

- Sandbox = **espaço isolado**.
- Honeyfiles = **iscas** para atrair malware.
- Monitoramento = **detecção em tempo real** de acessos suspeitos.