

Laboratorio 3b - Analizar diferentes tipos de datos con Python

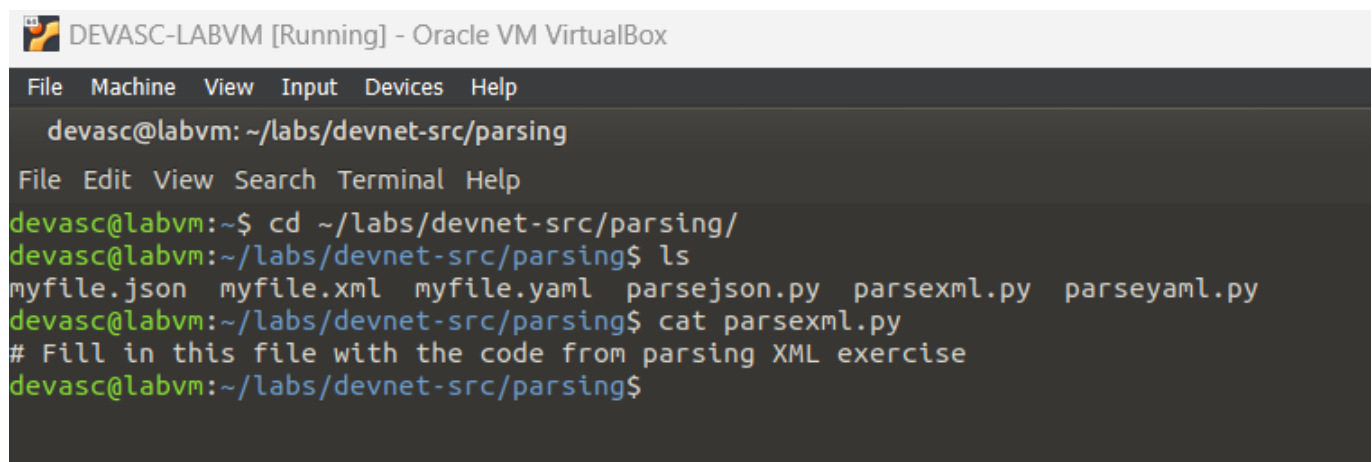
Instrucciones

Parte 1: Iniciar la máquina virtual (Virtual Machine) de DEVASC

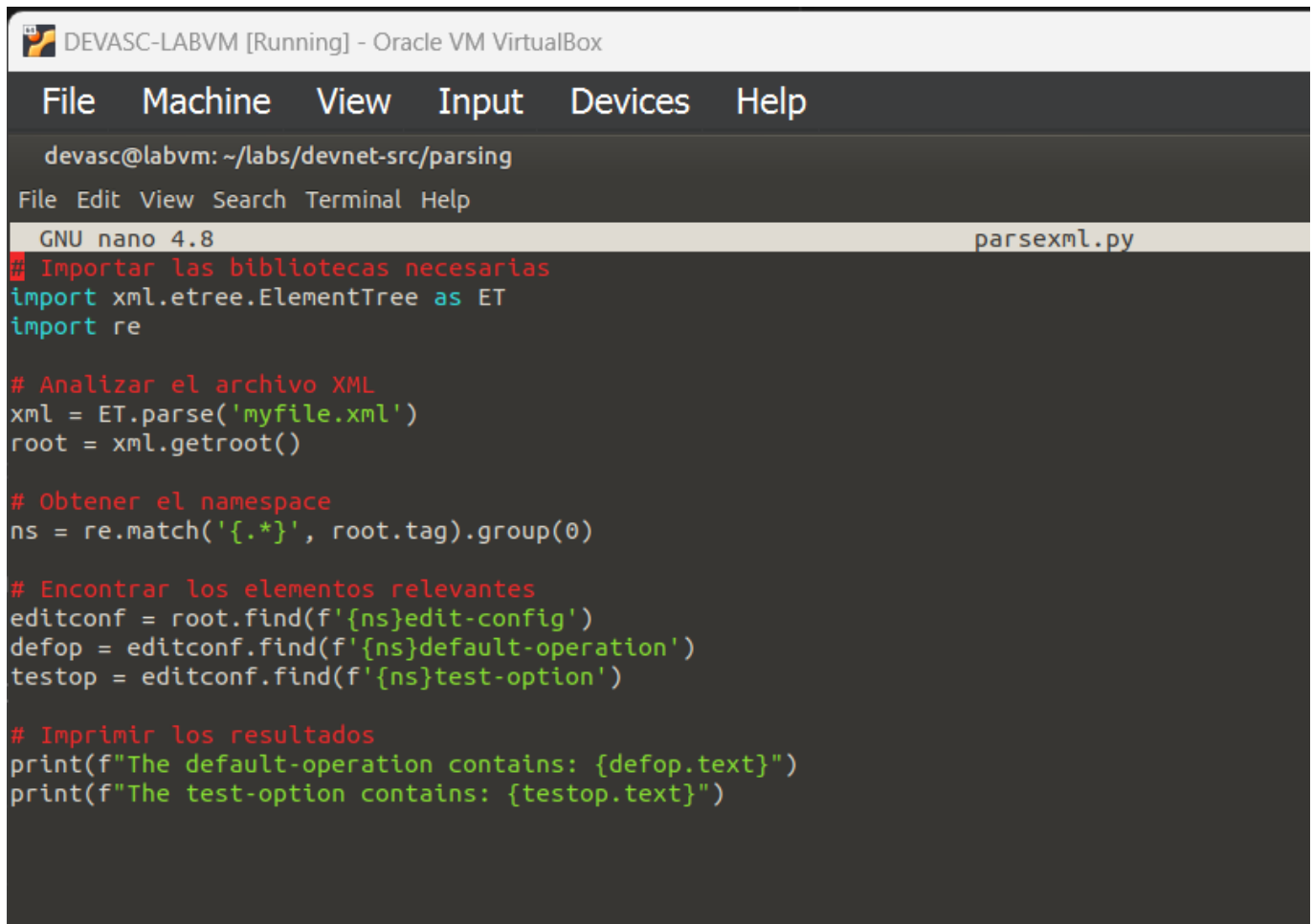
Hecho!

Parte 2: Analizar XML en Python

Paso 1: Crear un script para analizar los datos XML'



```
DEVASC-LABVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
devasc@labvm: ~/labs/devnet-src/parsing
File Edit View Search Terminal Help
devasc@labvm:~$ cd ~/labs/devnet-src/parsing/
devasc@labvm:~/labs/devnet-src/parsing$ ls
myfile.json  myfile.xml  myfile.yaml  parsejson.py  parsexml.py  parseyaml.py
devasc@labvm:~/labs/devnet-src/parsing$ cat parsexml.py
# Fill in this file with the code from parsing XML exercise
devasc@labvm:~/labs/devnet-src/parsing$
```



```

DEVASC-LABVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
devasc@labvm: ~/labs/devnet-src/parsing
File Edit View Search Terminal Help
GNU nano 4.8 parsexml.py
# Importar las bibliotecas necesarias
import xml.etree.ElementTree as ET
import re

# Analizar el archivo XML
xml = ET.parse('myfile.xml')
root = xml.getroot()

# Obtener el namespace
ns = re.match('{.*}', root.tag).group(0)

# Encontrar los elementos relevantes
editconf = root.find(f'{ns}edit-config')
defop = editconf.find(f'{ns}default-operation')
testop = editconf.find(f'{ns}test-option')

# Imprimir los resultados
print(f"The default-operation contains: {defop.text}")
print(f"The test-option contains: {testop.text}")

```

Paso 2: Ejecutar el script

```

devasc@labvm:~/labs/devnet-src/parsing$ python3 parsexml.py
The default-operation contains: merge
The test-option contains: set

```

Parte 3: Analizar JSON en Python

Analizar datos JSON utilizando la biblioteca json de Python y a convertirlos a YAML.

Paso 1: Crear un script para analizar los datos JSON

Vemos contenido del json:

```

devasc@labvm:~/labs/devnet-src/parsing$ ls
myfile.json myfile.xml myfile.yaml parsejson.py parsexml.py parseyaml.py
devasc@labvm:~/labs/devnet-src/parsing$ cat myfile.json
{
  "access_token": "ZDI3MGEyYzQtNmFlNS00NDNhLWFlNzAtZGVjNjE0MGU1OGZmZWNmZDEwN2ItYTU3",
  "expires_in": 1209600,
  "refresh_token": "MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4OTExMzQ1Njc4",
  "refresh_tokenexpires_in": 7776000
}

```

Codificamos el script usando el método `json.load (archivo_json)` para analizar los datos JSON.

Paso 2: Ejecutar el script para imprimir los datos JSON y luego modificarlo para imprimir datos de interés

Después de ejecutar el script:

```
devasc@labvm:~/labs/devnet-src/parsing$ cat parsejson.py
import json
import yaml

with open('myfile.json', 'r') as json_file:
    ourjson = json.load(json_file)

print(ourjson)
devasc@labvm:~/labs/devnet-src/parsing$ python3 parsejson.py
{'access_token': 'ZDI3MGEyYzQtNmFlNS00NDNhLWFlNzAtZGVjNjE0MGU1OGZmZWNmZDEwN2ItYTU3', 'expires_in': 1209600, 'AxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4', 'refresh_tokenexpires_in': 7776000}
```

Paso 3: Mostrar los datos JSON analizados en un formato de datos YAML

```
devasc@labvm:~/labs/devnet-src/parsing$ cat parsejson.py
import json
import yaml

with open('myfile.json', 'r') as json_file:
    ourjson = json.load(json_file)

print('\n\n')
print (yaml.dump (ourjson))
devasc@labvm:~/labs/devnet-src/parsing$ python3 parsejson.py

access_token: ZDI3MGEyYzQtNmFlNS00NDNhLWFlNzAtZGVjNjE0MGU1OGZmZWNmZDEwN2ItYTU3
expires_in: 1209600
refresh_token: MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4
refresh_tokenexpires_in: 7776000
```

Parte 4: Analizar YAML en Python

En esta parte, aprenderá a analizar datos YAML y convertirlos a JSON.

Paso 1: Crear un script para analizar los datos de YAML

Usamos el método `yaml.safe_load (archivo_yaml)` para analizar los datos de YAML.

Paso 2: Ejecutar el script para imprimir los datos de YAML y luego modificarlo para imprimir datos de interés

```
devasc@labvm:~/labs/devnet-src/parsing$ vim parseyaml.py
devasc@labvm:~/labs/devnet-src/parsing$ cat parseyaml.py
import json
import yaml

with open('myfile.yaml', 'r') as yaml_file:
    ouryaml = yaml.safe_load(yaml_file)

print(ouryaml)
devasc@labvm:~/labs/devnet-src/parsing$ python3 parseyaml.py
{'access_token': 'ZDI3MGEyYzQtNmFlNS00NDNhLWFlNzAtZGVjNjE0MGU1OGZmZWNmZDEwN2ItYTU3', 'expires_in': 1209600, 'refresh_token': 'MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4', 'refresh_tokenexpires_in': 7776000}
```

Paso 3: Mostrar los datos YAML analizados en un formato de datos JSON

```
devasc@labvm:~/labs/devnet-src/parsing$ vim parseyaml.py
devasc@labvm:~/labs/devnet-src/parsing$ cat parseyaml.py
import json
import yaml

with open('myfile.yaml', 'r') as yaml_file:
    ouryaml = yaml.safe_load(yaml_file)

print('\n\n')
print(json.dumps(ouryaml,indent=4))
devasc@labvm:~/labs/devnet-src/parsing$ python3 parseyaml.py

{
  "access_token": "ZDI3MGEyYzQtNmFlNS00NDNhLWFlNzAtZGVjNjE0MGU1OGZmZWNmZDEwN2ItYTU3",
  "expires_in": 1209600,
  "refresh_token": "MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MDEyMzQ1Njc4",
  "refreshtokenexpires_in": 7776000
}
```

Observamos un formato más legible por la indentación incluida.

Reflexión

En este laboratorio se aprendió a analizar datos XML, JSON y YAML. El uso de bibliotecas de Python para analizar datos es una buena manera de aprender.