



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

---

Experiment No. 6
Serialization in Python using Pickle
Date of Performance:27-02-2024
Date of Submission:05-02-2024



## ExperimentNo6

Title: Serialization in Python using  
Pickle

Aim: To study and implement serialization using Pickle  
in Python

Objective: To introduce serialization and deserialization using Pickle  
module in Python

Theor  
y:

Serialization and deserialization play crucial roles in data handling, especially in  
scenarios

where data needs to be stored or transmitted efficiently. Pickle, being a built-in  
module in

Python, simplifies this process by offering a convenient way to serialize and  
deserialize

Python  
objects.

One important aspect to note about Pickle is its ability to handle complex data  
structures

seamlessly. It can serialize and deserialize not only basic data types like strings  
and integers



but also more complex objects like lists, dictionaries, and even user-defined classes.

Additionally, Pickle provides support for protocol versions, allowing developers to choose

the appropriate protocol based on factors such as compatibility and efficiency. The protocol

version determines the format of the serialized data and can impact factors like file size and

serialization/deserialization speed.

It's worth mentioning that while Pickle is powerful and convenient, it's not without

limitations. One notable limitation is that the serialized data is not human-readable, making it

unsuitable for scenarios where human-readable data is required. Also, Pickle may not be the

most efficient solution for large datasets or scenarios where interoperability with non-Python

systems is a requirement.

Despite these limitations, Pickle remains a valuable tool in the Python ecosystem for many use cases, offering a quick and straightforward solution for serialization and deserialization

tasks. By understanding its capabilities and limitations, developers can leverage Pickle

effectively to manage data in their Python



applications.

CODE:

```
import pickle

data = {
    'username': 'Anchal',
    'email': 'anchalbiyani09@gmail.com',
    'password': '@Anchal09'
}

print('Data:', data)

# Writing data to a pickle file
with open('example.pickle', 'wb') as file:
    pickle.dump(data, file)
print("Pickle Success")

# Reading data from the pickle file
with open('example.pickle', 'rb') as file:
    value = pickle.load(file)
print('Unpickle Success')
print('Unpickled Data:', value)
```



## OUTPUT:

A screenshot of a Python terminal window. The window title is 'Run' with a sub-tab 'main'. The command prompt shows the execution of a Python script: `C:\Users\anand\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\anand\PycharmProjects\pythonProject1\main.py`. The output of the script is as follows:  
Data: {'username': 'Anchal', 'email': 'anchalbiyani09@gmail.com', 'password': '@Anchal09'}  
Pickle Success  
Unpickle Success  
Unpickled Data: {'username': 'Anchal', 'email': 'anchalbiyani09@gmail.com', 'password': '@Anchal09'}  
Process finished with exit code 0

## CONCLUSION:

The experiment successfully demonstrates the serialization and deserialization of Python

objects using the Pickle module. By serializing instances of the Student class into a file and

deserializing them back, we have effectively stored and retrieved object data. This process

showcases the practical utility of serialization for data persistence and transfer in Python

programming. Through Pickle, we can easily maintain the state of objects across sessions,

enhancing the versatility and efficiency of our code.