Using ODBC (Open Database Connectivity) for Database Operations

ODBC allows applications to interact with databases like Oracle, MySQL, SQL Server, etc., using a standardized API. Below, I'll show how to **connect to a database** and perform **basic operations** using **ODBC in Python**.

By using PYTHON

Download Python

Ø Go to the official Python website: https://www.python.org/downloads/

Ø Click the "Download Python" button (this will download the latest version)

Install Required Packages

First, install the `pyodbc` library (for Python):

To install above

1. `Win + X`, then click **Windows Terminal**
2. pip install pyodbc
3. python -c "import pyodbc; print(pyodbc.version)"

ODBC Connection Setup

To use ODBC, configure a **DSN (Data Source Name)** in your system:

Ø **Windows:** Go to Control panel → Windows tools (older versions of windows ADMINISTRATIVE TOOL)→ *ODBC Data Sources (64-bit)* → *Add* → Oracle in OraDb11g_home →

1. Data source name : orcl.
2. Description:
3. Tns service name:

4. User ID : sit2

→ click OK (it will ask password : sit)

Python Code for ODBC Database Operations

The following Python program connects to a database via ODBC and performs **CRUD (Create, Read, Update, Delete)** operations.

Ø Python Script Using ODBC

```
import pyodbc


# Oracle DSN connection details

dsn_name = "Orcl"  # Replace with your actual DSN

user = "sit2"

password = "sit"


try:

    # Connect to Oracle using DSN

    conn = pyodbc.connect(f"DSN={dsn_name};UID={user};PWD={password}")


    # Create a cursor

    cursor = conn.cursor()


    # 1. Create Table

    cursor.execute("Drop table acc")
    cursor.execute('''
      CREATE TABLE Acc (
      Account_No INT PRIMARY KEY,
      Holder_Name VARCHAR(100),
      Balance FLOAT
            )
    ''')
    print("Table Acc created successfully.")

    # 2 Insert Data
    cursor.execute("INSERT INTO Acc VALUES (101, 'Alice', 5000)")
    cursor.execute("INSERT INTO Acc VALUES (102, 'Bob', 3000)")
    conn.commit()

    # 3 Read Data
    cursor.execute("SELECT * FROM Acc")
    for row in cursor.fetchall():
      print(row)

    # 4 Update Data
    print("\nTable Acc before update.")
    cursor.execute("SELECT * FROM Acc")
    for row in cursor.fetchall():
      print(row)
    cursor.execute("UPDATE Acc SET Balance = Balance + 1000 WHERE Account_No =
101")
```

```
    conn.commit()

    print("\nTable Acc after update.")

    cursor.execute("SELECT * FROM Acc")

    for row in cursor.fetchall():

      print(row)

    # 5 Delete Data

    # cursor.execute("DELETE FROM Accounts WHERE Account_No = 102")

    conn.commit()




    # Close connection

    cursor.close()

    conn.close()


except Exception as e:

    print("Error:", e)
```

Explanation of Operations

1. **Connect to Database:**
   - Uses `pyodbc.connect()` to connect using ODBC.
   - Replace `your_server_name`, `your_database`, `your_username`, and `your_password` accordingly.
2. **Create a Table (`Accounts`)**
   - Defines columns: `Account_No`, `Holder_Name`, `Balance`.
3. **Insert Records**
   - Adds sample data (`Alice` and `Bob`).
4. **Retrieve Records**
   - Fetches all records and prints them.
5. **Update a Record**
   - Increases `Alice`'s balance by `1000`.
6. **Close Connection**
   - Closes the database connection to free resources.