1.Module Import and Management Scenario

You are developing a complex Python project with multiple modules. To manage the project effectively, you need to import and use various modules. Additionally, you want to organize the workload using namespaces and avoiding naming conflicts. Design a Python program that demonstrates the following. 1.Import multiple modules within your project. 2.Use the import statement to access functions and classes and variables from imported modules. 3.create your custom module that you use in your main program. 4. Handle naming conflicts and ensure proper namespacing. 5. Implement error handling for missing modules or incorrect module usage.

In [2]:
```
# Main program (main.py)

# Import multiple modules
import import_ipynb
import math
import random
import custom_module # Custom module created in the same directory

# Access functions, classes, and variables from imported modules
print("Lets check whether the math module is working or not:")
print(f"Square root of 25: {math.sqrt(25)}")
print(f"Value of pi: {math.pi}")
print('Yes,its working')

print("\nlets check random module is working or not:")
print(f"Random integer between 1 and 10000: {random.randint(1, 10000)}")

# Use the custom module
print("\nlets check the function from the custom modulle")
custom_module.custom_function()
print(f"Custom variable from the module: {custom_module.custom_variable}")

# Handle naming conflicts and ensure proper namespacing
import custom_module as cm # Alias the custom_module to avoid conflicts
print("\nUsing custom_module with alias:")
cm.custom_function()
print(f"Custom variable from the module with alias: {cm.custom_variable}")

# Implement error handling for missing modules or incorrect module usage
try:
    import no_module # This module does not exist
except ImportError as e:
    print(f"Error importing no_module: {e}")

try:
    # Attempt to access an undefined variable from math module
    print(math.undefined_variable)
except AttributeError as e:
    print(f"Error accessing undefined_variable from math module: {e}")
```

```
importing Jupyter notebook from custom_module.ipynb
Lets check whether the math module is working or not:
Square root of 25: 5.0
Value of pi: 3.141592653589793
Yes,its working

lets check random module is working or not:
Random integer between 1 and 10000: 7061

lets check the function from the custom modulle
Asalam Walaikum
Custom variable from the module: Wa rahmatullahi wabarkatuhu

Using custom_module with alias:
Asalam Walaikum
Custom variable from the module with alias: Wa rahmatullahi wabarkatuhu
Error importing no_module: No module named 'no_module'
Error accessing undefined_variable from math module: module 'math' has no att
ribute 'undefined_variable'
```

Virtual Environment Management Scenario: You are working on multiple Python projects with different dependencies and versions. To avoid conflicts and ensure project isolation, you decide to use virtual environments.

```
Create a Python program that demonstrates the following:

    1. Create a virtual environment for a specific project.

    2. Activate and deactivate virtual environments.

    3. Install, upgrade, and uninstall packages within a virtual envir
onment.

    4. List the installed packages in a virtual environment.

    5. Implement error handling for virtual environment operations.
```

In [1]:
```
# venv_project.py

import os
import subprocess

# Define the name of the virtual environment for your project
venv_name = "my_project_venv"

# 1. Create a virtual environment for a specific project
def create_venv():
    try:
        subprocess.run(["python", "-m", "venv", venv_name], check=True)
        print(f"Virtual environment '{venv_name}' created successfully.")
    except subprocess.CalledProcessError as e:
        print(f"Error creating virtual environment: {e}")

# 2. Activate and deactivate virtual environment
def activate_venv():
    venv_path = os.path.join(venv_name, "Scripts" if os.name == "nt" else "bin'
    activate_script = os.path.join(venv_path, "activate")

    try:
        subprocess.run([activate_script], shell=True, check=True)
        print(f"Activated virtual environment '{venv_name}'.")
    except subprocess.CalledProcessError as e:
        print(f"Error activating virtual environment: {e}")

def deactivate_venv():
    try:
        subprocess.run(["deactivate"], shell=True, check=True)
        print(f"Deactivated virtual environment '{venv_name}'.")
    except subprocess.CalledProcessError as e:
        print(f"Error deactivating virtual environment: {e}")

# 3. Install, upgrade, and uninstall packages within a virtual environment
def install_package(package_name):
    try:
        subprocess.run([f"pip install {package_name}"], shell=True, check=True
        print(f"Package '{package_name}' installed in the virtual environment.'
    except subprocess.CalledProcessError as e:
        print(f"Error installing package: {e}")

def upgrade_package(package_name):
    try:
        subprocess.run([f"pip install --upgrade {package_name}"], shell=True, c
        print(f"Package '{package_name}' upgraded in the virtual environment.")
    except subprocess.CalledProcessError as e:
        print(f"Error upgrading package: {e}")

def uninstall_package(package_name):
    try:
        subprocess.run([f"pip uninstall -y {package_name}"], shell=True, check=
        print(f"Package '{package_name}' uninstalled from the virtual environme
    except subprocess.CalledProcessError as e:
        print(f"Error uninstalling package: {e}")

# 4. List installed packages in a virtual environment
```

```
def list_installed_packages():
    try:
        subprocess.run(["pip list"], shell=True, check=True)
    except subprocess.CalledProcessError as e:
        print(f"Error listing installed packages: {e}")

if __name__ == "__main__":
    create_venv()
    activate_venv()

    # Example package operations
    install_package("requests")
    upgrade_package("requests")
    uninstall_package("requests")

    list_installed_packages()

    deactivate_venv()
```

```
Virtual environment 'my_project_venv' created successfully.
Activated virtual environment 'my_project_venv'.
Error installing package: Command '['pip install requests']' returned non-zer
o exit status 1.
Error upgrading package: Command '['pip install --upgrade requests']' returne
d non-zero exit status 1.
Error uninstalling package: Command '['pip uninstall -y requests']' returned
non-zero exit status 1.
Error listing installed packages: Command '['pip list']' returned non-zero ex
it status 1.
Deactivated virtual environment 'my_project_venv'.
```

Module Dependency Resolution

Scenario: You are developing a Python application that relies on third-party packages. Managing dependencies and ensuring compatibility is crucial for your project's success.

Design a Python program that demonstrates the following:

```
1. Use a requirements.txt file to specify project dependencies.

2. Automatically install all project dependencies from the requirement
s.txt file.

3. Ensure that the versions of installed packages are compatible.

4. Implement error handling for dependency resolution and installatio
n.
```

Implement Inventory Management in Python with MySQL

```
1.Inventory management, a critical element of the supply chain, is the
tracking of inventory from manufacturers to warehouses, and from these
facilities to a point of sale. The goal of inventory management is to
have the right products in the right place at the right time.
2The required databases is Inventory, and the required tables are Purc
hases, Sales, and Inventory.
3.Note, apply these thoughts to demonstrate the database operations in
Python.
```

In [ ]:

```python
# 1.connecccting table to MySQl database
import mysql.connector
conn=mysql.connector.connect(user='root',
                             password='Tauheed333@333',
                             host='localhost',
                             database='inventory'

                             )
```

In [15]:

```python
print(conn)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001FFDE7B90D
0>
```

In [16]:

```python
curs1=conn.cursor()
```

In [18]:

```python
sql1="CREATE TABLE Purchases (purchase_id INT AUTO_INCREMENT PRIMARY KEY,produc

sql2="CREATE TABLE Sales (sale_id INT AUTO_INCREMENT PRIMARY KEY,product_name V

sql3="CREATE TABLE Inventory (product_id INT AUTO_INCREMENT PRIMARY KEY,product
```

In [20]:

```python
curs1.execute(sql1)
```

In [21]:

```python
curs1.execute(sql2)
```

In [22]:

```python
curs1.execute(sql3)
```

In [23]:
```
# Function to update inventory after a purchase
def update_inventory_purchase(product_name, quantity):
    sql = "INSERT INTO Purchases (product_name, purchase_date, quantity) VALUES
    curs1.execute(sql, (product_name, quantity))

    # Update the inventory table
    sql = "UPDATE Inventory SET quantity_in_stock = quantity_in_stock + %s WHER
    curs1.execute(sql, (quantity, product_name))

    conn.commit()

# Function to update inventory after a sale
def update_inventory_sale(product_name, quantity_sold):
    sql = "INSERT INTO Sales (product_name, sale_date, quantity_sold) VALUES (%
    curs1.execute(sql, (product_name, quantity_sold))

    # Update the inventory table
    sql = "UPDATE Inventory SET quantity_in_stock = quantity_in_stock - %s WHER
    curs1.execute(sql, (quantity_sold, product_name))

    conn.commit()

# Example usage
update_inventory_purchase("Product A", 50)
update_inventory_sale("Product A", 20)

# Close the cursor and database connection
curs1.close()
conn.close()
```

Customer Order Processing Scenario You are building a customer order processing system for an e-commerce company. The system needs to interact with a MySQL database to store customer orders, products, and other details.

```
1. Design a MySQL database schema for the order processing system, inc
luding tables for customer products and orders.
2. Write a Python program that connects the database and allows custom
ers to place new orders.
3. Implement a feature that calculates the total cost of an order and
updates product quantities in the database.
4. How would you handle cases where a product is no longer available w
hen a customer places an order?
5. Develop a function to generate order reports for the company's fina
nce department.
```

In [1]:
```
# 1.conneccting table to MySQl database
import mysql.connector
conn=mysql.connector.connect(user='root',
                            password='Tauheed333@333',
                            host='localhost',
                            database='orders'

                            )
```

In [2]:
```
print(conn)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001A6962F499
0>
```

In [3]:
```
curs2=conn.cursor()
```

In [4]:
```
sql4="CREATE TABLE Customers (customer_id INT AUTO_INCREMENT PRIMARY KEY,first_

sql5="CREATE TABLE Products (product_id INT AUTO_INCREMENT PRIMARY KEY,product_

sql6="CREATE TABLE Orders (order_id INT AUTO_INCREMENT PRIMARY KEY,customer_id 

sql7="CREATE TABLE OrderItems (order_item_id INT AUTO_INCREMENT PRIMARY KEY,or
```

In [ ]:
```
curs2.execute(sql4)
curs2.execute(sql5)
curs2.execute(sql6)
curs2.execute(sql7)
```

In [ ]:
```
# Function to place an order
def place_order(customer_id, product_id, quantity):
    # Check product availability
    cursor.execute("SELECT quantity_available, price FROM products WHERE produc
    result = cursor.fetchone()
    if result:
        available_quantity, price = result
        if quantity <= available_quantity:
            subtotal = price * quantity
            cursor.execute("INSERT INTO order_details (order_id, product_id, qu
                           (order_id, product_id, quantity, subtotal))
            db.commit()
            return True
        else:
            return False
    else:
        return False

# Calculate total cost of an order
def calculate_total_cost(order_id):
    cursor.execute("SELECT SUM(subtotal) FROM order_details WHERE order_id = %s
    result = cursor.fetchone()
    return result[0] if result[0] else 0.0

# Handle product unavailability when placing an order
if not place_order(customer_id, product_id, quantity):
    print("Product is no longer available.")

# Generate order report for finance department
def generate_order_report(order_id):
    cursor.execute("SELECT customers.name, customers.address, orders.order_date
    result = cursor.fetchall()
    for row in result:
        print(f"Customer Name: {row[0]}")
        print(f"Customer Address: {row[1]}")
        print(f"Order Date: {row[2]}")
        print(f"Product: {row[4]}, Quantity: {row[3]}, Subtotal: {row[5]}")

# Close the database connection
db.close()
```

You are tasked with developing a Python program that connects to a MySQL database, retrieves data from a table, performs some operations on the data, and updates the database with the modified data. Please write Python code to accomplish this task.

Instructions:

```
1. Assume that you have a MySQL database server running with the follo
wing details:

i. Host: localhost

ii. Port: 3306

iii. Username: your username

iv. Password: your password

V. Database Name: your database

vi. Table Name: your table

vil. The table has the following columns: id (int), name (varchar). qu
antity (int).

2. Your Python program should:

    1. Connect to the MySQL database.

    2. Retrieve all records from the your table table.

iii. Calculate the total quantity of all records retrieved.

iv. Update the quantity column of each record by doubling its value.

v. Commit the changes to the database.

vi. Close the database connection.

3. Handle any potential errors that may occur during the database conn
ection and data manipulation, such as connection failures or SQL error
s:

4. Provide comments in your code to explain each step
```

In [ ]:

```python
# 1.conneccting table to MySQL database
import mysql.connector
conn=mysql.connector.connect(user='root',
                            password='Tauheed333@333',
                            host='localhost',
                            database='tauheer'

                            )
```

In [3]:
```
print(conn)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001FFDC4F97D
0>
```

In [4]:
```
curs=conn.cursor()
```

In [7]:
```
sql="create table your_table(id int,name varchar(20),quantity int)"
```

In [ ]:
```
curs.execute(sql)
```

In [27]:
```
cus_insert="insert into your_table(id,name,quantity) values(%s,%s,%s)"
cus_val=[(1,'shaik',100),
        (2,'tauheer',200),
        (3,'shaik',300)
        ]
curs.executemany(cus_insert,cus_val)
conn.commit()
print(curs.rowcount,'are inserted in a row')
```

```
3 are inserted in a row
```

In [28]:
```
curs.execute("select * from your_table")
for i in curs:
    print(i)
```

```
(1, 'shaik', 100)
(2, 'tauheer', 200)
(3, 'shaik', 300)
```

In [9]:
```
double="UPDATE your_table SET quantity=quantity*2"
curs.execute(double)
conn.commit()
```

In [10]:
```
curs.close()
conn.close()
```

You are developing an employee management system for a company. The database should store employee information, including name, salary, department, and hire date. Managers should be able to view and update employee details.

```
1. Design the database schema for the employee management system.
```

In [2]:
```
# 1.conneccting table to MySQl database
import mysql.connector
conn=mysql.connector.connect(user='root',
                             password='Tauheed333@333',
                             host='localhost',
                             database='employee_management'

                             )
print(conn)
curs3=conn.cursor()
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001E86EC1C59
0>
```

In [3]:
```
sql8="CREATE TABLE emp_info (salary INT ,name VARCHAR(255) NOT NULL,hire_date D
```

In [ ]:
```
curs3.execute(sql8)
```

In [59]:
```
cus_insert="insert into emp_info(salary,name,hire_date,department) values(%s,%s
cus_val=[(100,'shaik','2000-12-16','admin'),
        (200,'tauheer','2000-11-16','admin'),
        (300,'uzair','2000-10-16','management'),
        (400,'tausif','2000-9-16','management')
        ]
```

In [60]:
```
curs3.executemany(cus_insert,cus_val)
conn.commit()
print(curs3.rowcount,'are inserted in a row')
```

```
4 are inserted in a row
```

In [6]:
```
my="select name from emp_info where department ='admin'"
curs3.execute(my)
#result=curs3.fetchall()
for i in curs3:
    print(i)
```

```
('shaik',)
('tauheer',)
```

In [12]:
```
my1="update emp_info set department='admin' where salary='3500'"
curs3.execute(my1)
conn.commit()
print(curs3.rowcount,'are updated in a row')
```

```
0 are updated in a row
```