**FINAL PROJECT DASAR PEMOGRAMAN**

**“PROGRAM PYTHON PENGADAAN BARANG”**

Dosen Pengajar : Walim, M.Kom



Makalah ini

Disusun oleh :

1. Abdul Aziz (19241791)
2. Ahmad Maulana Zuhdi (19240878)
3. Defa Raihan Agis (19240506)
4. Muhammad Nadhif Fadhilah Toziri (19240559)
5. Nero Aziz Saputra (19240522)

**JURUSAN SISTEM INFORMASI**

**FAKULTAS TEKNIK DAN INFORMATIKA**

**UNIVERSITAS BINA SARANA DAN INFORMATIKA**

**2024/2025**

**KATA PENGATAR**

Puji dan Syukur kami panjatkan kepada kehadirat Allah SWT / Tuhan Yang Maha Esa, karena beliaulah yang selalu senantiasa melimpahkan Rahmatnya dan Karunia-Nya sehingga kami dapat menyusun laporan makalah ini secara baik dan benar. Kami juga tidak lupa mengucapkan banyak terima kasih kepada dosen pengajar kami,Bapak. Walim, M.Kom yang telah membimbing kami dalam mengerjakan laporan makalah ini.

Makalah ini berisikan tentang Final Project Dasar Pemrograman “Program Sederhana Membedakan Penyakit Covid-19 dan Gejala Tifus“. Adapun maksud dan tujuan kami di buatnya makalah ini adalah untuk memenuhi salah satu tugas mata kuliah yakni Dasar Pemrograman.

Kami menyadari bahwa masih banyak kekurangan yang mendasar atas proses pembuatan makalah ini baik dari segi apapun. Oleh karena itu kami mohon maaf sebesar-besarnya dan kami mengharapkan saran serta kritik yang dapat membantu kami untuk penyempurnaan makalah ini selanjutnya.

Cikarang, 20 November 2024

Kelompok 6

**BAB I**

**PENDAHULUAN**

**1.1Latar Belakang**

Seiring dengan perkembangan teknologi, solusi digital mulai diadopsi untuk menyelesaikan permasalahan dalam proses pengadaan. Salah satu pendekatan yang menarik adalah pengembangan aplikasi pengadaan barang berbasis teknologi digital. Aplikasi ini dirancang untuk mengotomatiskan proses pengadaan, meminimalkan kesalahan, meningkatkan transparansi, serta mempercepat waktu pemrosesan.

Pengembangan aplikasi pengadaan barang dapat dilakukan menggunakan berbagai teknologi, termasuk Python sebagai struktur aplikasi dan MySQL untuk pengelolaan basis data, Python menjadi pilihan yang kuat untuk membangun aplikasi yang handal dan fleksibel karena kemudahan dalam memahami sintaks programnya.

Karena itu, dengan kemajuan di bidang teknologi saat ini, dibuatnya system untuk pengadaan barang agar memudahkan dalam update serta menganalisis stok barang/produk.

**1.2 Identifikasi Masalah**

Pengadaan barang secara manual memerlukan waktu lama, rawan kesalahan, serta kurangnya transparansi

**1.3 Rumusan Masalah**

* Bagaimana merancang program (Python) pengadaan barang yang dapat mengotomatisasi proses dan meningkatkan efisiensi?
* Bagaimana memastikan transparansi dalam seluruh tahapan pengadaan barang melalui program (Python) ?

**1.4 Tujuan**

* Meningkatkan efisiensi dalam proses pengadaan dengan mengotomatiskan tugas-tugas manual.
* Menciptakan sistem yang transparan dan akuntabel dalam setiap tahapan pengadaan barang.

**1.5 Ruang Lingkup**

Dalam penelitian ini kelompok kami membatasi permasalahan terkait pengadaan barang dengan menggunakan Bahasa Pemrograman Python dan basis data MySQL untuk menampilkan beberapa fitur yang dapat memperbarui data dengan cepat.

**BAB II**

**ISI**

**2.1 Pengertian**

* Pengertian Python

Menurut pengertian dari Python Software Foundation (2016), Python adalah bahasa pemrograman interpretatif, berorinetasi objek dan semantik yang dinamis. Python memiliki high-level struktur data, dynamic typing dan dynamic binding. Python memiliki sintaks sederhana dan mudah dipelajari untuk penekanan pada kemudahan membaca dan mengurangi biaya perbaikan program. Python mendukung modul dan paket untuk mendorong kemodularan program dan code reuse. Interpreter Python dan standard library-nya tersedia secara gratis untuk semua platform dan dapat secara bebas disebarkan. Bahasa pemrograman ini dibuat oleh Guido van.

* Pengertian MySQL

MySQL adalah sistem manajemen basis data relasional (*Relational Database Management System* atau RDBMS) yang menggunakan SQL (*Structured Query Language*) sebagai bahasa utama untuk mengelola data. MySQL dikembangkan oleh MySQL AB dan sekarang dimiliki oleh Oracle Corporation.

* Pengertian Pengadaan Barang

Pengadaan barang adalah proses penambahan, pemilihan, pembelian, dan pengelolaan barang yang diperlukan untuk memenuhi kebutuhan suatu organisasi, perusahaan, atau institusi. Proses ini bertujuan untuk memastikan bahwa barang yang dibutuhkan tersedia dalam jumlah yang cukup, kualitas yang sesuai, dan dengan biaya yang efisien.

**2.2 Program Pengadaan Barang**

Program Pengadaan Barang menggunakan Python dan MySQL adalah sebuah program yang dirancang untuk mengelola proses pengadaan barang secara efisien dengan menggunakan Python sebagai bahasa pemrograman dan MySQL sebagai basis data untuk penyimpanan informasi.

**BAB III**

**PROGRAM PYTHON**

**3.1 Listing Program**

# Aplikasi pengadaan barang dengan operasi SQL CRUD (Create, Read, Update, Delete)

# Install Xampp

# Nyalakan Apache dan Mysql

# search di browser localhost

# pilih phpMyAdmin

# masuk ke menu sql

# Paste sintaks dibawah ini (digunakan hanya untuk petama kali saja untuk mengkonfigurasi database):

# CREATE DATABASE db\_minimarket;

# USE db\_minimarket;

# CREATE TABLE tb\_product (

#     id INT(11) AUTO\_INCREMENT PRIMARY KEY,

#     name VARCHAR(255) NOT NULL,

#     category VARCHAR(255) NOT NULL,

#     weight INT(11) NOT NULL,

#     quantity INT(11) NOT NULL,

#     supplier VARCHAR(255) NOT NULL

# );

# CREATE TABLE users (

#     id INT(11) AUTO\_INCREMENT PRIMARY KEY,

#     username VARCHAR(16) NOT NULL,

#     password VARCHAR(16) NOT NULL,

#     position VARCHAR(25) NOT NULL

# );

# INSERT INTO `users` (`id`, `username`, `password`, `position`) VALUES (NULL, 'admin', 'admin', 'admin');

# INSERT INTO tb\_product (id)

# VALUES (

#     CONCAT(

#         CHAR(FLOOR(65 + (RAND() \* 26))),  -- Huruf acak (A-Z)

#         LPAD(FLOOR(RAND() \* 9999) + 1, 4, '0')  -- Nomor acak (1-9999)

#     )

# );

# Install library MySQL:

# pip install mysql.connector

# import library

import mysql.connector # library untuk menghubungkan kedalam database

import time # Library untuk waktu

import getpass # Library untuk password

import os # Library untuk berinteraksi dengan OS

# Function untuk clear terminal

def clear():

    if os.name == 'nt':  # Identitas OS windows

        os.system('cls')  # Sintaks windows untuk membersihkan layar terminal

    else:

        os.system('clear') # Sintaks untuk OS Unix membersihkan layar terminal

# Function untuk membuat koneksi kedalam database

def db\_connection():

    try: # Mencoba membuat koneksi kedalam database

        # Membuat koneksi dengan format:

        return mysql.connector.connect(

            host = 'localhost',

            user = 'root',

            password = '',

            database = 'db\_minimarket'

        )

    # Sintaks yang berjalan jika terjadi error pada kode try

    except mysql.connector.Error as e: # Menampilkan bagian yang error dari try koneksi ke MySQL

        print(f"Error: {e}")

        time.sleep(5) # Membuat delay selama 5 detik sebelum mengeskekusi sintaks selanjutnya

        clear() # Function clear

# Function untuk user login

def login(username, password):

    try: # Mencoba untuk fetch data sesuai dengan yang diinput oleh user (jika data True)

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor kedalam SQL

        cursor.execute("SELECT \* FROM users WHERE username = %s AND password = %s", (username, password)) # Cursor mengeksekusi query untuk memilih column yang sesuai dengan yang diinput user berdasarkan parameter function

        get\_user = cursor.fetchone() # Fetch 1 data dari kolom table

        if get\_user: # Jika data berhasil di fetch

            clear() # Function clear

            print("\n\033[32m\033[1m!--Login Berhasil--!\033[0m\033[30m\n") # Alert keterangan untuk user

            time.sleep(1) # Membuat delay waktu selama 1 detik kepada terminal sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

            main()  # Function main

        else: # Jika data yang di fetch tidak cocok

            clear() # Function clear

            print("\n\033[31m\033[1m!--Login Gagal--!\nusername atau password salah\033[0m\033[30m\n") # Alert keterangan untuk user

            time.sleep(1) # Membuat delay waktu selama 1 detik kepada terminal sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

    except ValueError as e: # Menampilkan value yang error dari kolom try login

        print(f"Error: {e}")

    finally: # setelah selesai, tutup koneksi dan kursor untuk menghemat RAM

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi untuk menghemat memori

# Function untuk menambah product baru

def add\_new(name, category, weight, quantity, supplier):

    try: # Mencoba mengeksekusi

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor kedalam SQL

        cursor.execute("SELECT \* FROM tb\_product WHERE name = %s AND weight = %s AND supplier = %s", (name, weight, supplier)) # Cursor mengeksekusi query untuk select semua column pada table

        cursor.fetchall() # Fetch semua data

        if cursor.rowcount == 0: # Jika produk yang ditambahkan belum ada pada column

            cursor.execute("INSERT INTO tb\_product (name, category, weight, quantity, supplier) VALUES(%s, %s, %s, %s, %s)", (name, category, weight, quantity, supplier)) # Cursor mengeksekusi query untuk menambahkan produk kedalam database

            db.commit() # Menyimpan perubahan kedalam database

            print("\n\033[32m\033[1m!--Product Berhasil Ditambahkan--!\033[0m\033[30m\n") # Alert keterangan produk berhasil ditambahkan

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks berikutnya

            clear() # Function clear

        else: # Jika Produk sudah ada dalam database

            print(f"\033[31m\033[1m!--Gagal Menambah Product Baru--!\nProduct Sudah Ada\033[0m\033[30m") # Alert Keterangan produk gagal ditambahkan

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks berikutnya

            clear() # Function clear

            main() # Function main

    except ValueError as e: # Jika terdapat sintaks error pada try

        print(f"Error: {e}")

        time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

        clear() # Function clear

    finally: # Jika sudah selesai

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi kedatabase untuk menghemat memori

# Function +qty product

def update\_qty(id, name, quantity, weight, supplier):

    try: # Mencoba mengeksekusi

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor kedalam SQL

        cursor.execute("SELECT id FROM tb\_product WHERE name = %s AND weight = %s AND supplier = %s", (name, weight, supplier)) # Cursor mengeksekusi query untuk memilih column id berdasarkan name, weight, dan supplier

        select = cursor.fetchone() # Fetch 1 data yang match

        if select: # Jika select true

            cursor.execute("UPDATE tb\_product SET quantity = quantity + %s WHERE id = %s AND name = %s AND weight = %s AND supplier = %s", (quantity, id, name, weight, supplier)) # Cursor mengeksekusi query untuk + quantity

            db.commit() # Menyimpan perubahan kedalam database

            print("\n\033[32m\033[1m!--Berhasil Menambahkan Jumlah Product--!\033[0m\033[30m\n") # Alert keterangan berhasil menambahkan jumlah produk

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

        else: # Jika select false

            print(f"\n\033[31m\033[1m!--Gagal Menambah Jumlah Product--!\033[0m\033[30m\n") # Alert keterangan gagal menambah jumlah produk

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

    except ValueError as e: # Jika terdapat error pada try

        print(f"Error: {e}")

    finally: # Jika selesai

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi untuk menghemat memori

# Function untuk - quantity

def reduce\_qty(id, name, quantity, weight, supplier):

    try: # Mencoba mengeksekusi

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor pada SQL

        cursor.execute("SELECT id FROM tb\_product WHERE name = %s AND weight = %s AND supplier = %s", (name, weight, supplier)) # Cursor mengeksekusi query untuk memilih column id berdasarkan name, weight, supplier

        select = cursor.fetchone() # Fetch 1 data yang macth

        if select: # Jika select true

            cursor.execute("UPDATE tb\_product SET quantity = quantity - %s WHERE id = %s AND name = %s AND weight = %s AND supplier = %s", (quantity, id, name, weight, supplier)) # Cursor mengeksekusi query untuk mengurangi quantity

            db.commit() # Menyimpan perubahan kedalam SQL

            print("\n\033[32m\033[1m!--Berhasil Mengurangi Jumlah Product--!\033[0m\033[30m\n") # Alert keterangan berhasil mengurangi jumlah produk

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

        else: # Jika select false

            print(f"\n\033[31m\033[1m!--Gagal Mengurangi Jumlah Product--!\033[0m\033[30m\n") # Alert keterangan gagal mengurangi jumlah produk

            time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

            clear() # Function clear

    except ValueError as e: # Jika terdapat error pada try

        print(f"Error: {e}")

    finally: # Jika selesai

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi untuk menghemat memori

# Function untuk menghapus produk

def remove\_product(id, name, weight, supplier):

    try: # Mencoba mengeksekusi

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor pada SQL

        cursor.execute("DELETE FROM tb\_product WHERE id = %s AND name = %s AND weight = %s AND supplier = %s", (id, name, weight, supplier)) # Cursor mengeksekusi query untuk menghapus produk sesuai dengan id, name, weight, dan supplier

        db.commit() # Menyimpan perubahan kedalam SQL

        print("\n\033[32m\033[1m!--Product berhasil dihapus--!\033[0m\033[30m\n") # Alert keterangan produk berhasil dihapus

        time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

        clear() # Function clear

    except ValueError as e: # Jika terdapat error pada try

        print(f"\033[31m\033[1m!--Gagal Menghapus Product--!\nError: {e}\033[0m\033[30m") # Alert keterangan gagal menghapus produk

        time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

        clear() # Function clear

    finally: # Jika selesai

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi untuk menghemat memori

# Function melihat semua produk

def show\_all():

    try: # Mencoba mengeksekusi

        db = db\_connection() # Mendeklarasikan variable untuk function db\_connection

        cursor = db.cursor() # Membuat cursor pada SQL

        cursor.execute("SELECT \* FROM tb\_product") # Cursor mengeksekusi query untuk select semua column

        show\_result = cursor.fetchall() # Fetch semua data

        # Format untuk output

        print("--- DAFTAR PRODUK ---".center(100))

        print("="\*100)

        print(f"{'ID':<5} {'Name':<25} {'Categpry':<20} {'Weight':<10} {'Quantity':<10} {'Supplier':<20}") # Membuat table dengan format lebar colomn

        print("="\*100)

        for cell in show\_result: # looping data sesuai dengan jumlah rows di database

            print(f"{cell[0]:<5} {cell[1]:<25} {cell[2]:<20} {cell[3]:<10} {cell[4]:<10} {cell[5]:<20}") # Membuat table dengan format cell sesuai dengan database

        print("="\*100)

        answer = input("Kembali ke manu? ")

        if answer:

            time.sleep(2)

            clear()

            main()

    except ValueError as e: # Jika terdapat error pada try

        print(f"Error: {e}")

    finally: # Jika selesai

        cursor.close() # Menutup cursor

        db.close() # Menutup koneksi untuk menghemat memori

# Function main(utama)

def main():

    while True: # Jika login berhasil / True

        print("\n\033[35m\033[1mSelamat Datang Di Minimarket\033[0m\033[30m\n")

        print("1. Tambah Product Baru")

        print("2. Update Jumlah Product")

        print("3. Hapus Product")

        print("4. Lihat Semua Product")

        print("5. Keluar")

        user\_answer = int(input("Masukkan pilihan kamu(1/2/3/4/5): ")) # Variable untuk menampung jawaban user

        if user\_answer == 1: # Jika user menginput 1

            clear() # Function clear

            print("Menambahkan  Product Baru\n")

            nama\_product = input("Nama Product: ")

            catg\_product = input("Kategoti Product: ")

            weig\_product = int(input(("Berat Product(gram): ")))

            qtyy\_product = int(input(("Jumlah Product: ")))

            splr\_product = input("Supplier Product: ")

            add\_new(nama\_product, catg\_product, weig\_product, qtyy\_product, splr\_product) # Menjalankan function add\_new dengan format

        elif user\_answer == 2: # Jika user menginput 2

            clear() # Function clear

            print("1. Tambahi Jumlah Product: ")

            print("2. Kurangi Jumlah Product: ")

            upd\_user\_answer = int(input("Masukkan pilihan kamu(1/2): ")) # Variable untuk menampung jawaban user

            if upd\_user\_answer == 1: # Jika user menginput 1

                clear() # Function clear

                print("Menambah Jumlah Product\n")

                frId\_product = int(input("ID Product: "))

                nama\_product = input("Nama Product: ")

                weig\_product = int(input("Berat Product(gram): "))

                splr\_product = input("Supplier Product: ")

                qtyy\_product = int(input("Tambah Jumlah Product: "))

                update\_qty(frId\_product, nama\_product, qtyy\_product, weig\_product, splr\_product) # Menjalankan function update\_qty dengan format

            elif upd\_user\_answer == 2: # Jika user menginput 2

                clear() # Function clear

                print("Mengurangi Jumlah Product\n")

                frId\_product = int(input("ID Product: "))

                nama\_product = input("Nama Product: ")

                weig\_product = int(input("Berat Product(gram): "))

                splr\_product = input("Supplier Product: ")

                qtyy\_product = int(input("Tambah Jumlah Product: "))

                reduce\_qty(frId\_product, nama\_product, qtyy\_product, weig\_product, splr\_product) # Menjalankan function reduce\_qty dengan format

            else: # Jika jawaban tidak valid

                clear() # Clear

                print("\033[31mJawaban tidak valid !\033[30m") # Alert keterangan jawaban tidak valid

                time.sleep(2) # Membuat delay selama 2 detik sebelum mengeksekusi sintaks selanjutnya

                clear() # Function clear

        elif user\_answer == 3: # Jika user menginput 3

            clear() # Function clear

            print("Menghapus Product\n")

            frId\_product = int(input("ID Product: "))

            nama\_product = input("Nama Product: ")

            weig\_product = int(input("Berat Product(gram): "))

            splr\_product = input("Supplier Product: ")

            remove\_product(frId\_product, nama\_product, weig\_product, splr\_product) # Menjalankan function remove\_product dengan format

        elif user\_answer == 4: # Jika user menginput 4

            clear()

            show\_all() # Menjalankan function show\_all

        elif user\_answer == 5: # Jika user menginput 5

            time.sleep(2)

            clear()

            break # menghentikan function main

        else: # Jika jawaban tidak valid

            clear() # Function clear

            print("\033[31mJawaban tidak valid !\033[30m") # Alert keterangan jawaban tidak valid

            clear() # Function clear

# if \_\_name\_\_ == "\_\_main\_\_":

clear() # Function clear

while True: # looping login jika gagal maka akan terus loop sampai meminta agar login berhasil

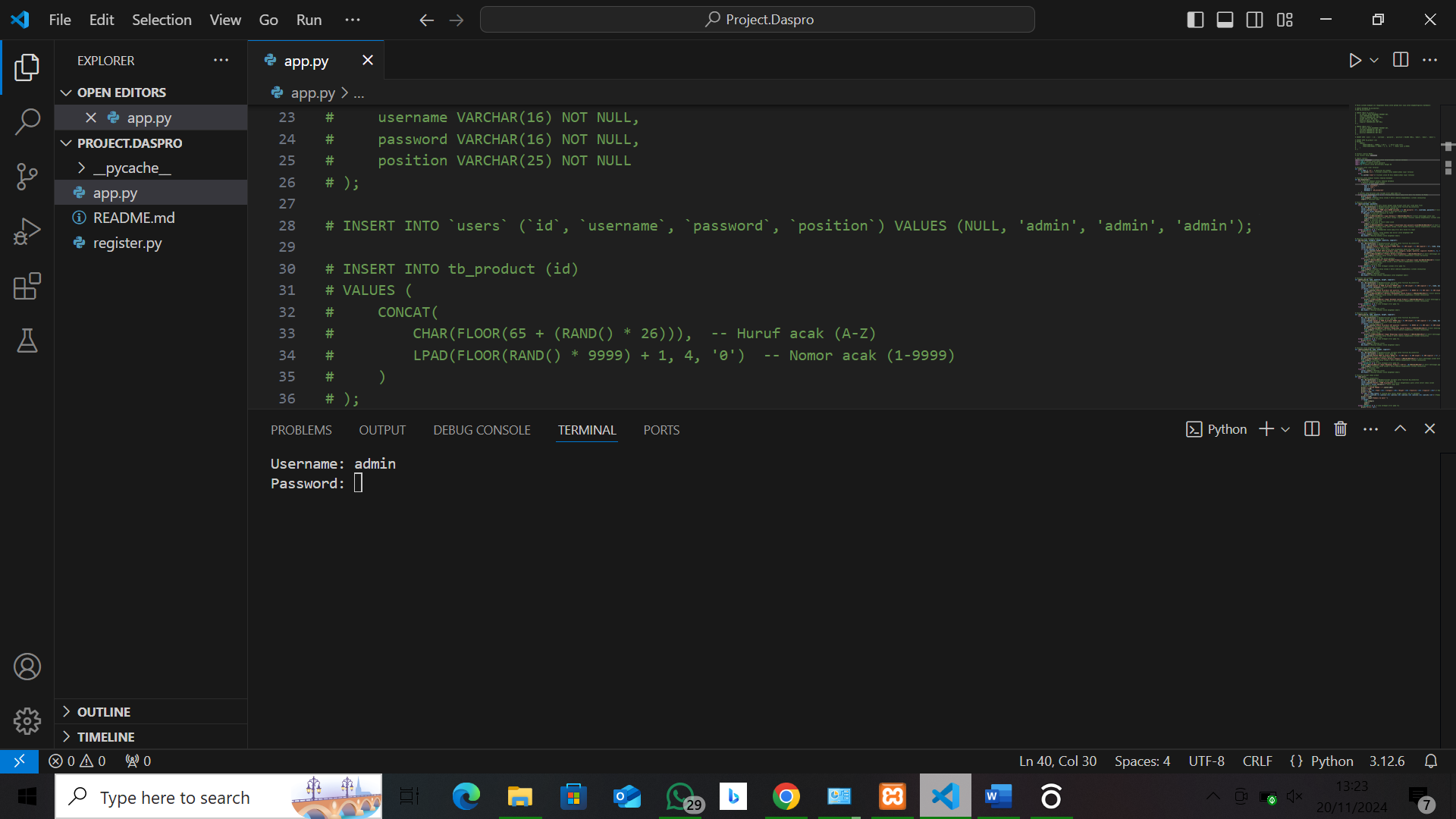
    username = input("Username: ")

    password = getpass.getpass("Password: ")

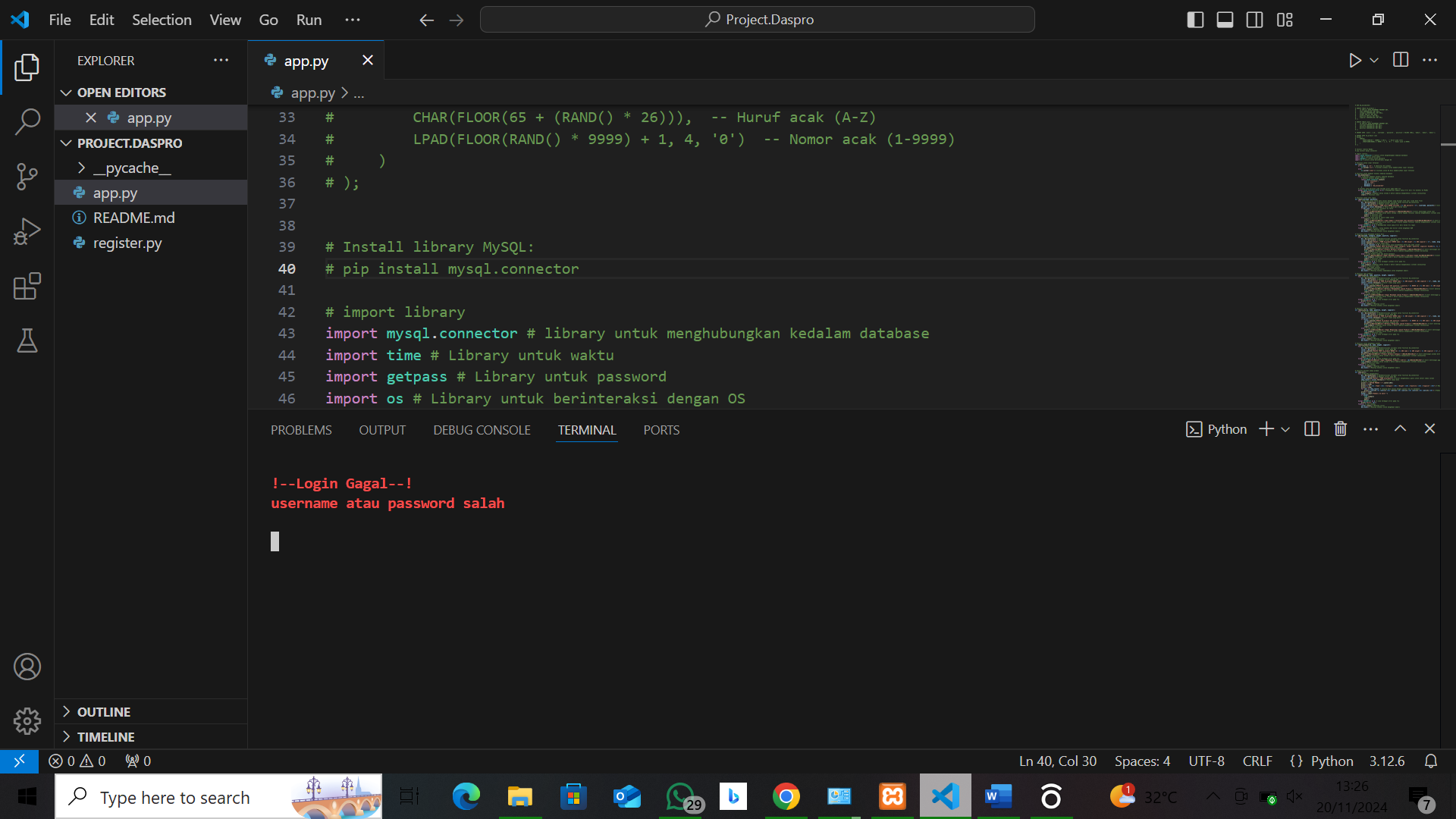
    login(username, password) # Menjalankan function login dengan format

**3.2 Hasil Running**

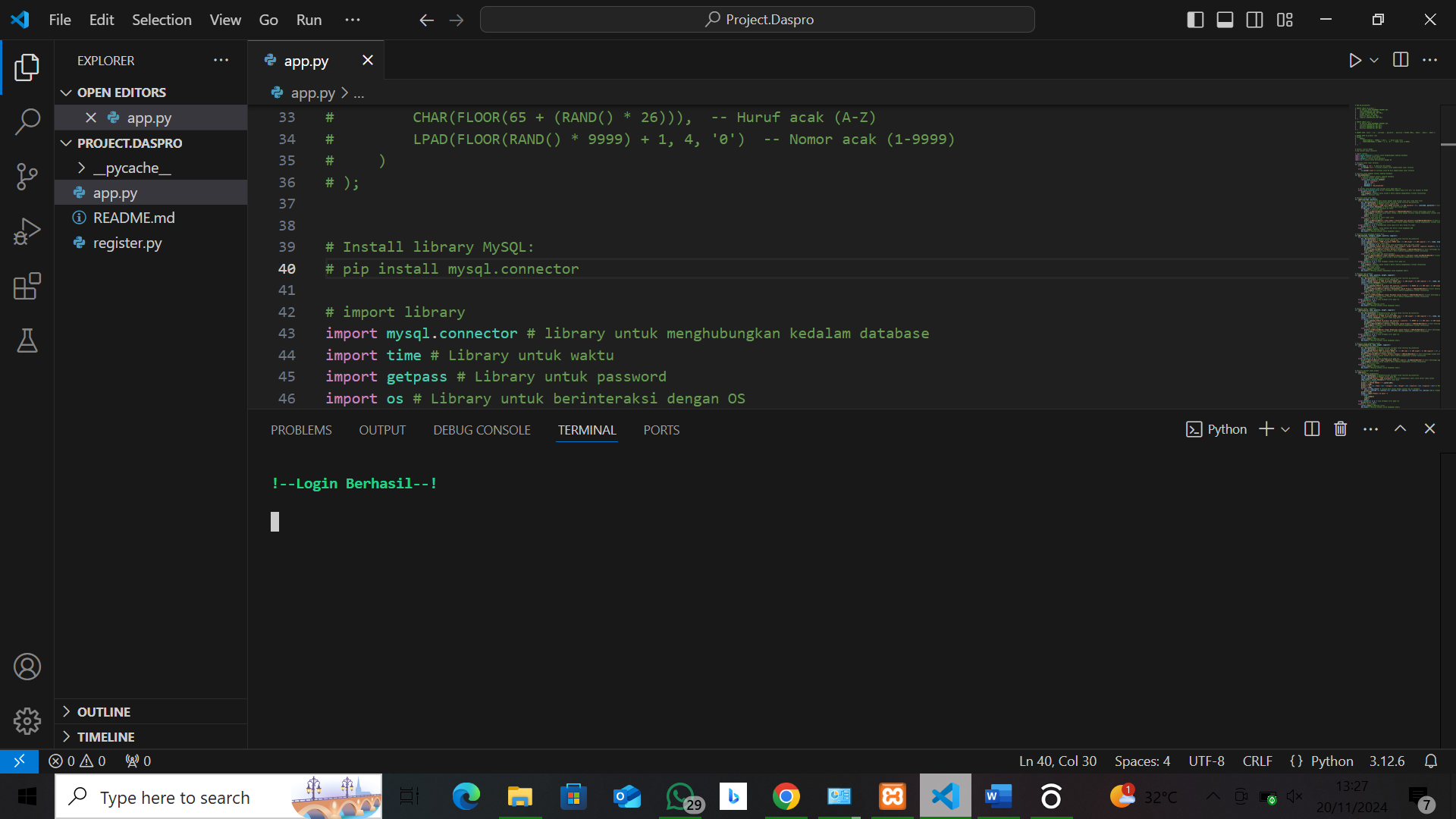
3.2.1 Login Output



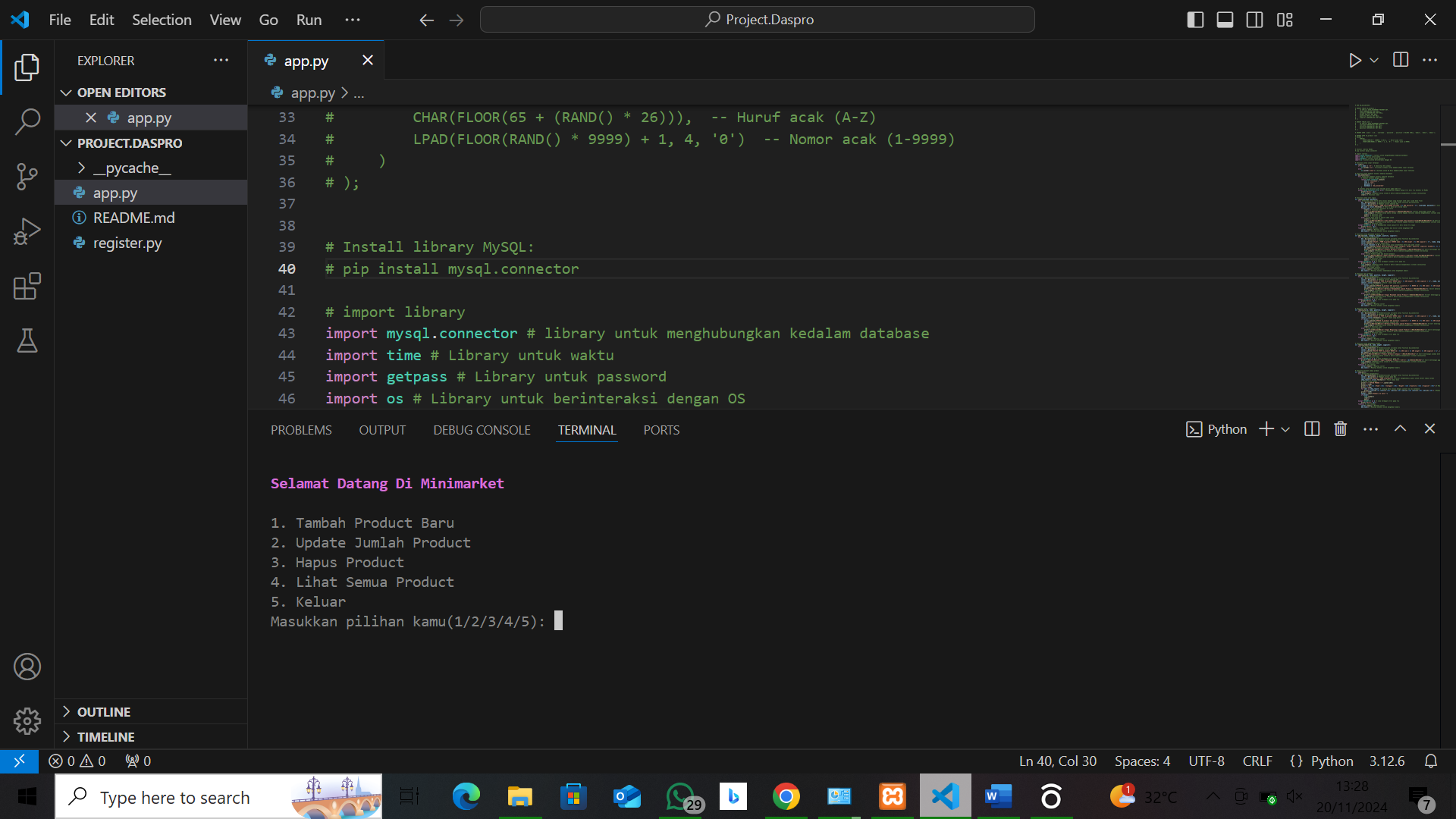
3.2.2 Jika login gagal



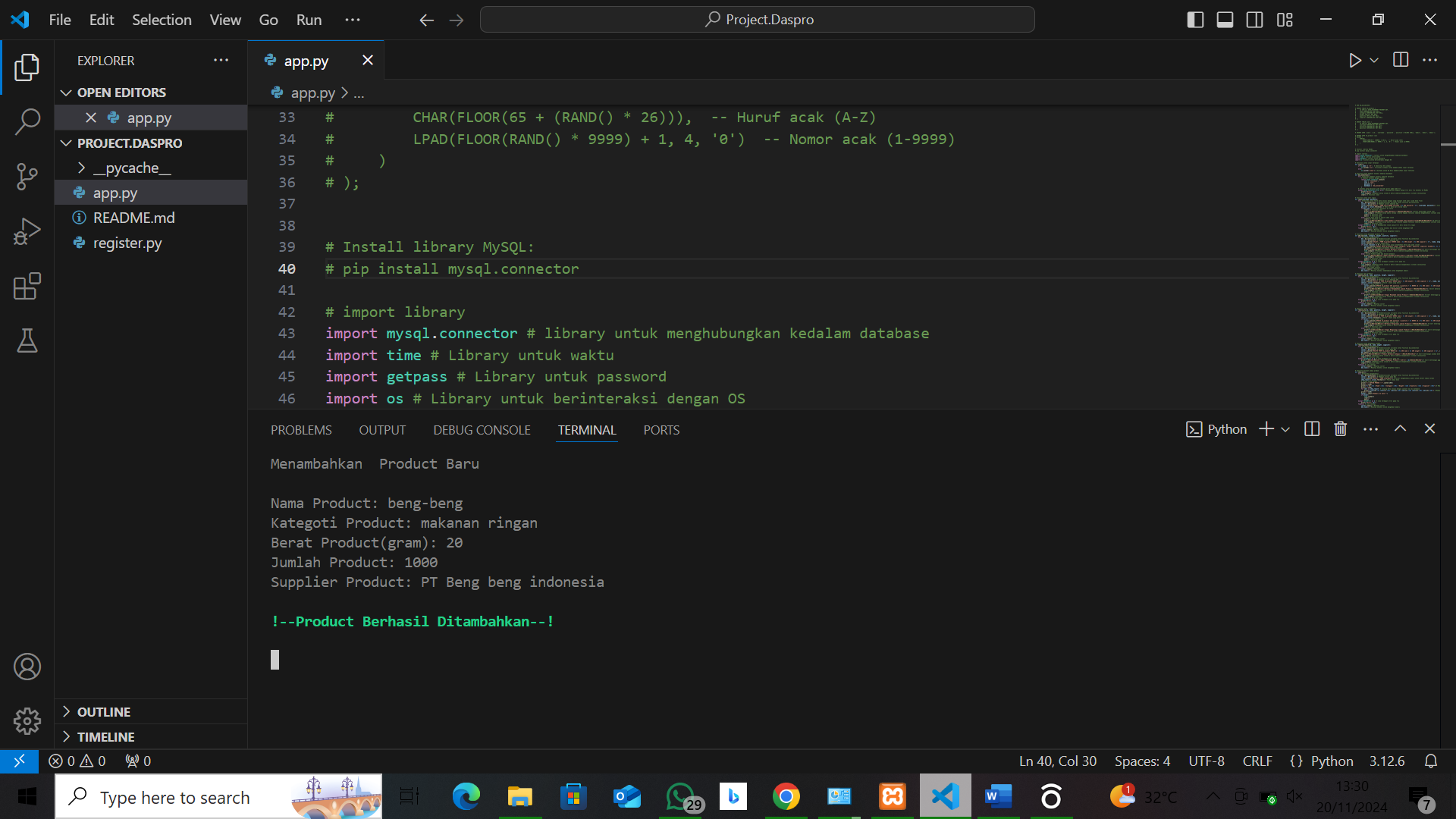
3.2.3 Jika login berhasil



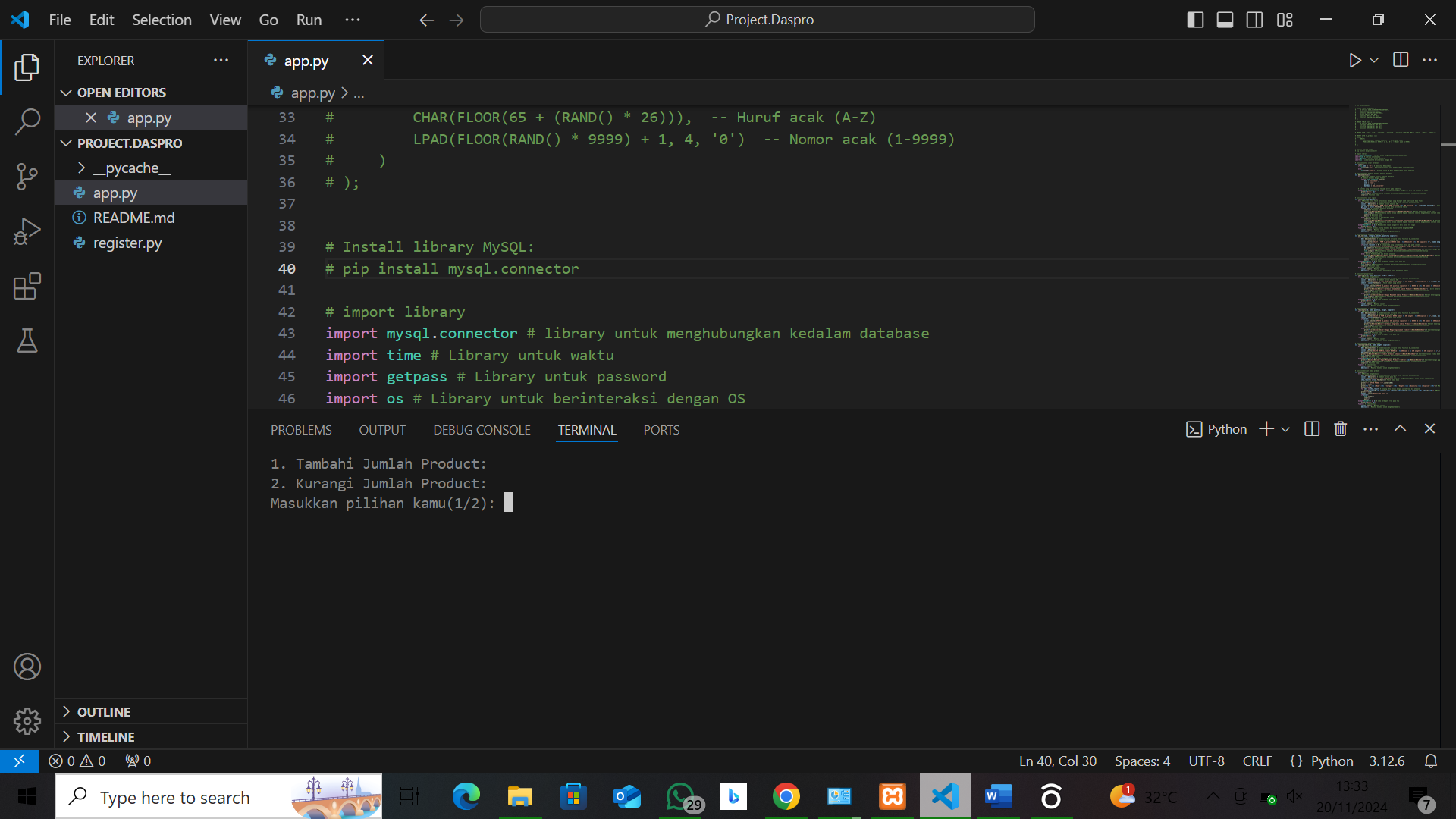
3.2.4 Menu Utama



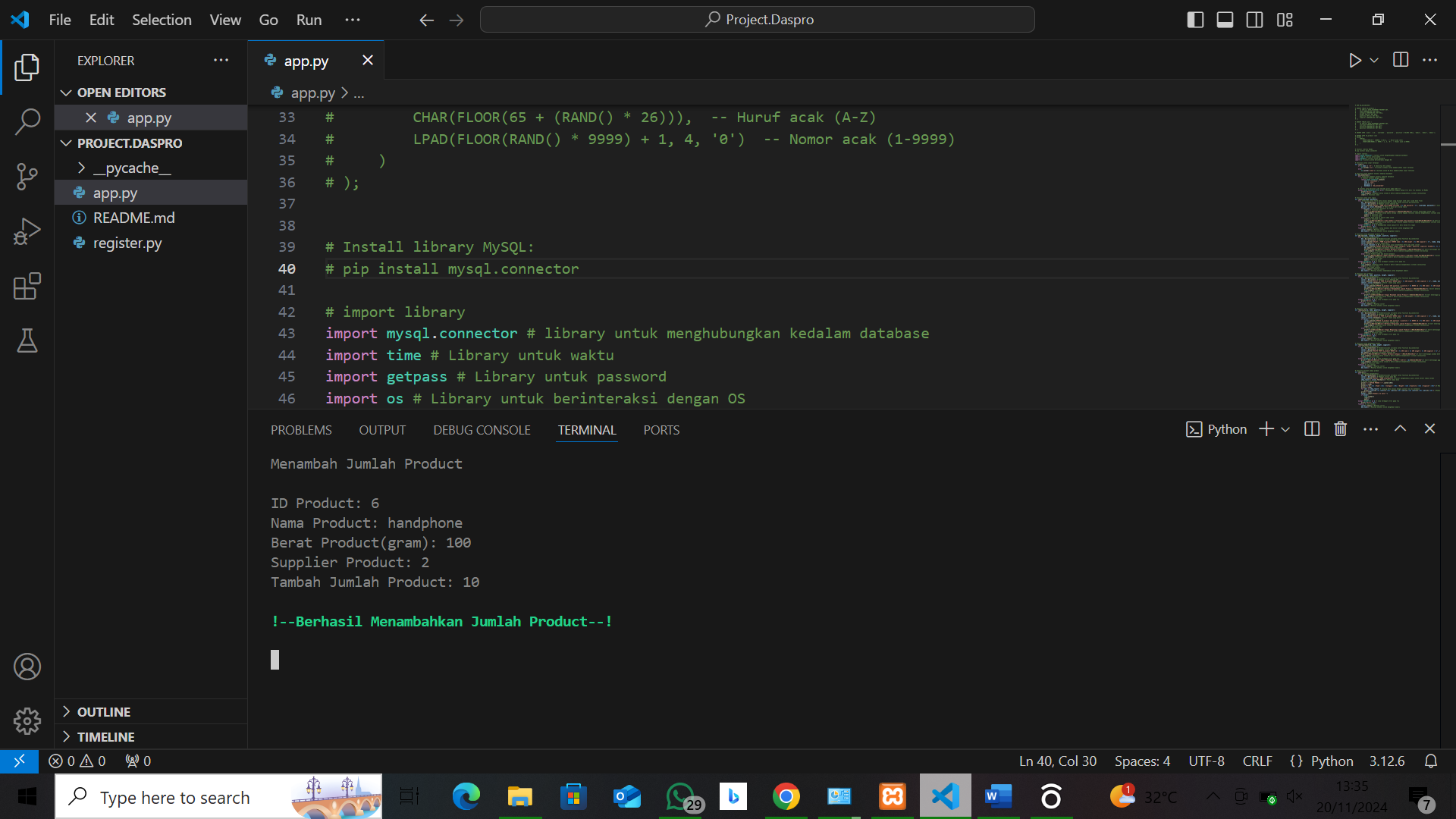
3.2.5 Tambah Produk Baru



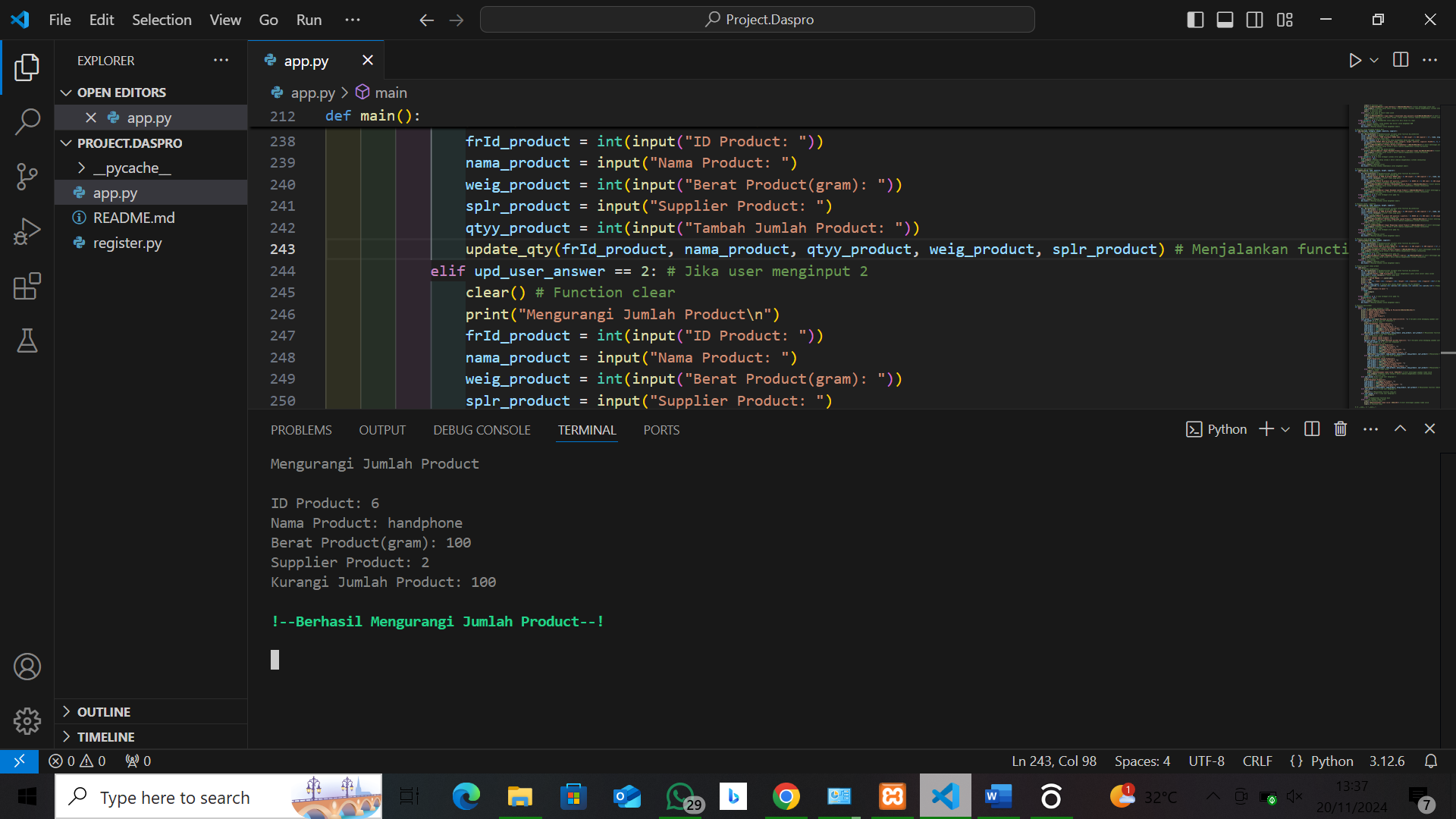
3.2.6 Update Jumlah Produk



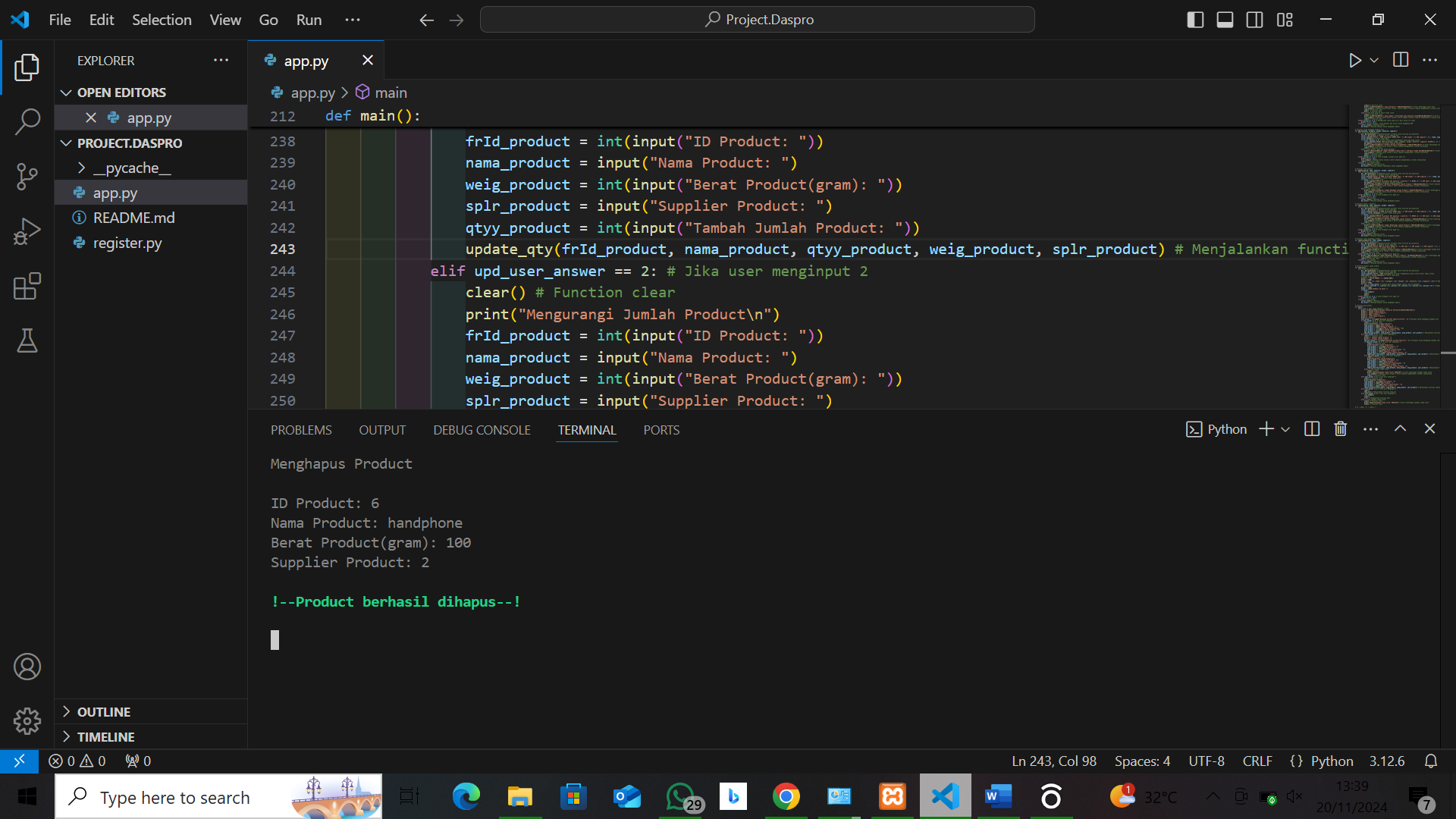
3.2.6.1 Tambah Jumlah Produk



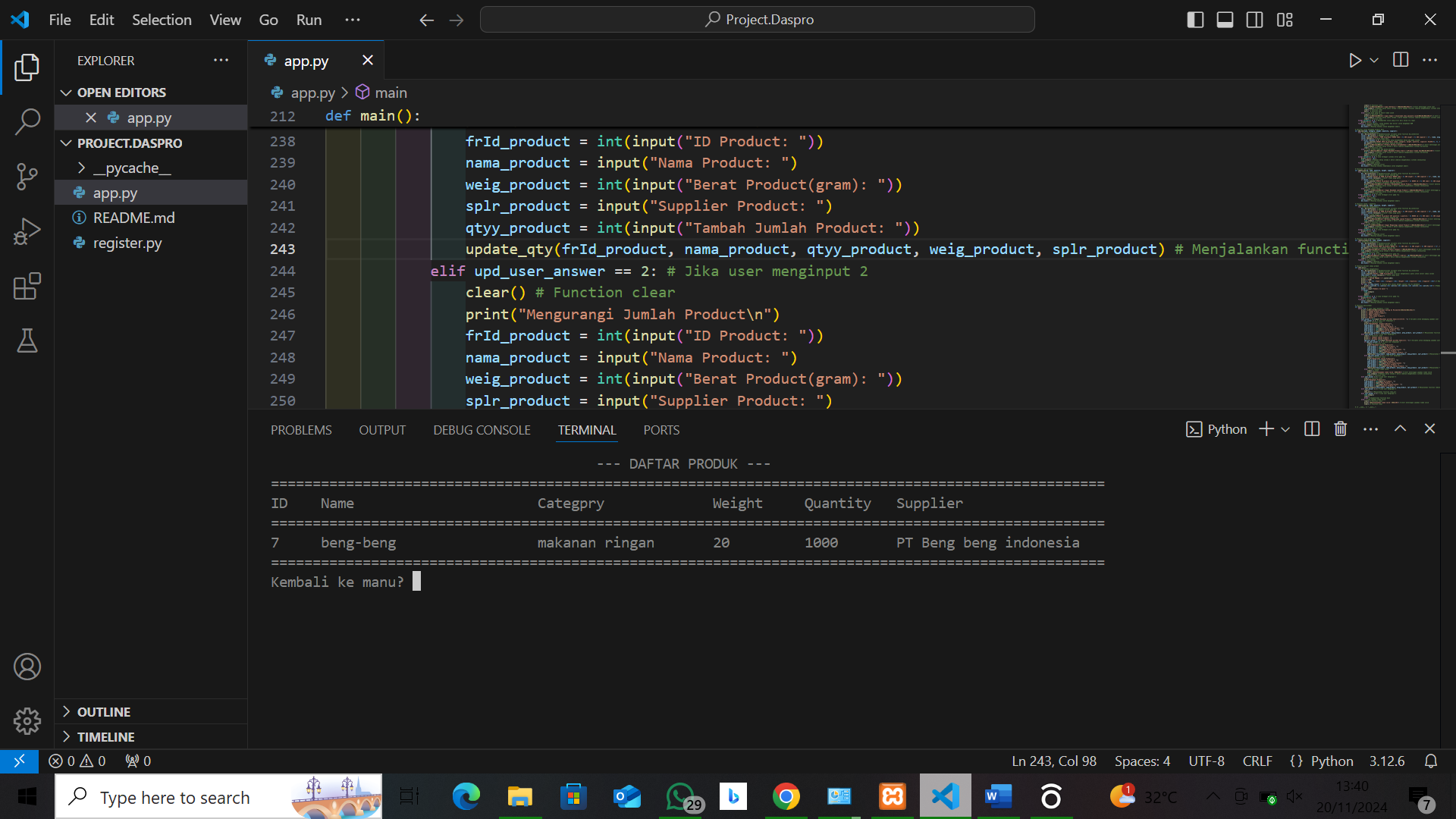
3.2.6.2 Kurangi Jumlah Produk



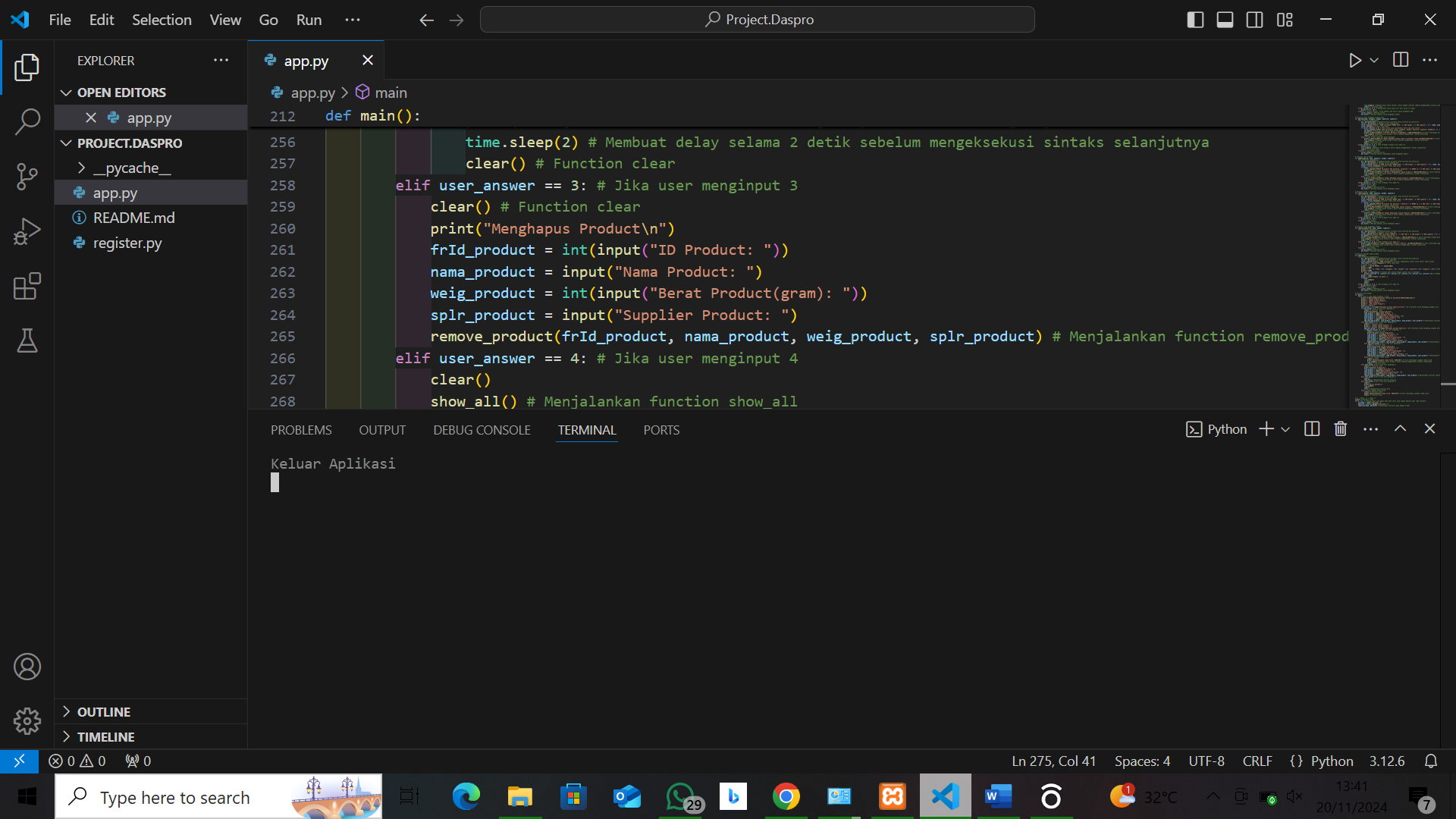
3.2.7 Hapus Produk



3.2.8 Lihat Semua Produk



3.2.9 Keluar Aplikasi



**BAB IV**

**PENUTUP**

**4.1 Kesimpulan**

Dengan di buatnya Bahasa pemrograman berbasis python ini adalah untuk Pengadaan barang