**LAPORAN PRAKTIKUM**

**ALGORITMA & PEMPROGRAMAN**

*Sturktur kontrol percabangan*



*Dosen Pengampu : Ata Amrullah, M.Tr.Kom.*

Disusun Oleh :

Muhammad (24063004)

**PROGRAM STUDI INFORMATIKA**

**UNIVERSITAS ISLAM DARUL ‘ULUM LAMONGAN**

**2024/2025**

**BAB I**

**PENDAHULUAN**

**A. TUJUAN PEMBELAJARAN**

**Setelah menyelesaikan modul ini, peserta diharapkan mampu:**

**1. Menyusun boolean expression dengan benar.**

**2. Merangkai beberapa boolean expression menggunakan operator logical dengan benar.**

**3. Menyusun percabangan dengan if sesuai dengan permasalahan yang dihadapi.**

**4. Menangani jika terjadi kesalahan masukan pengguna menggunakan bentuk exception sederhana.**

Dalam era digital saat ini, pemrograman telah menjadi salah satu keterampilan yang paling dibutuhkan di berbagai bidang. Dari pengembangan perangkat lunak hingga analisis data, kemampuan untuk memahami dan menerapkan logika dasar sangatlah penting. Salah satu aspek krusial dalam pemrograman adalah penggunaan logika boolean dan pengendalian alur program. Modul ini dirancang untuk memberikan peserta pemahaman mendalam tentang kedua konsep tersebut, sehingga mereka dapat mengembangkan program yang lebih kompleks dan responsif.

Logika boolean adalah dasar dari keputusan dalam pemrograman. Dengan menggunakan ekspresi boolean, kita dapat membuat keputusan berdasarkan kondisi tertentu. Misalnya, dalam sebuah aplikasi pemesanan online, kita perlu memeriksa apakah produk yang diinginkan tersedia atau tidak sebelum melanjutkan ke proses pembayaran. Dengan memahami cara menyusun ekspresi boolean, peserta akan mampu menilai kondisi dengan tepat dan membuat program yang berfungsi sesuai harapan.

1. **Menyusun Boolean Expression dengan Benar**: Dalam bagian ini, peserta akan diajarkan cara menyusun ekspresi boolean yang sederhana hingga kompleks. Peserta akan belajar mengenai operator logika dasar, yaitu AND (&&), OR (||), dan NOT (!). Dengan contoh praktis, peserta akan melihat bagaimana kombinasi dari operator ini dapat menghasilkan keputusan yang berbeda. Misalnya, dalam pengolahan data pengguna, kita mungkin ingin memeriksa apakah pengguna berusia di atas 18 tahun dan memiliki alamat email yang valid. Pemahaman ini akan memberikan peserta keterampilan dasar dalam pengambilan keputusan.
2. **Merangkai Beberapa Boolean Expression Menggunakan Operator Logical dengan Benar**: Setelah memahami dasar-dasar ekspresi boolean, peserta akan belajar bagaimana merangkai beberapa ekspresi menjadi logika yang lebih kompleks. Dalam pengembangan aplikasi, seringkali kita harus menggabungkan berbagai kondisi untuk menentukan alur program. Misalnya, dalam sistem pendaftaran, kita mungkin ingin memeriksa beberapa kondisi: apakah nama pengguna tersedia, apakah kata sandi memenuhi syarat keamanan, dan apakah pengguna berusia di atas 18 tahun. Dengan merangkai ekspresi boolean, peserta dapat menciptakan logika yang lebih canggih dan responsif terhadap situasi yang berbeda.
3. **Menyusun Percabangan dengan If Sesuai dengan Permasalahan yang Dihadapi**: Struktur kontrol seperti if, else if, dan else merupakan alat yang sangat penting dalam pengembangan program. Peserta akan mempelajari cara menggunakan struktur percabangan ini untuk mengendalikan alur eksekusi program berdasarkan kondisi tertentu. Contoh nyata dalam penggunaan percabangan bisa dilihat dalam aplikasi yang mengharuskan pengguna memilih metode pembayaran. Berdasarkan pilihan pengguna, program akan menampilkan langkah-langkah yang berbeda. Dengan memahami cara menyusun percabangan dengan benar, peserta akan dapat membuat aplikasi yang lebih interaktif dan mudah digunakan.
4. **Menangani Jika Terjadi Kesalahan Masukan Pengguna Menggunakan Bentuk Exception Sederhana**: Kesalahan masukan pengguna adalah salah satu tantangan yang sering dihadapi dalam pengembangan aplikasi. Modul ini akan mengajarkan peserta bagaimana cara menangani situasi ini menggunakan teknik exception handling. Dengan pendekatan ini, peserta akan belajar untuk menggunakan blok try-catch untuk menangkap kesalahan yang mungkin terjadi, seperti input yang tidak valid atau kesalahan dalam format data. Memberikan umpan balik yang jelas kepada pengguna tentang kesalahan mereka merupakan bagian penting dari pengalaman pengguna yang baik. Peserta akan memahami bagaimana menangani kesalahan dengan cara yang ramah dan informatif, sehingga pengguna dapat dengan mudah memperbaiki kesalahan mereka.

**BAB II**

**PEMBAHASAN**

Latihan 3.1 Implementasikan penanganan kesalahan input pengguna dari program-program

pada Contoh Case I dan Case II. □

Latihan 3.2 Implementasikan percabangan pada Case II (Positif-Negatif) menggunakan

ternary operator. □

Latihan 3.3 Kamu sedang mengembangkan sebuah aplikasi kalender digital. Salah satu fitur

yang paling penting adalah kemampuan untuk menampilkan jumlah hari dalam setiap bulan.

Coba tuliskan kode programnya. Sebagai contoh, perhatikan input dan output berikut ini:



Lengkapi program tersebut dengan penanganan kesalahan jika pengguna memasukkan bulan

yang salah. Penanganan kesalahan dalam bentuk memunculkan pesan bahwa bulan yang

diinputkan oleh pengguna tersebut tidak valid.

Latihan 3.4 Kamu adalah seorang detektif yang sedang menyelidiki sebuah kasus. Di tempat

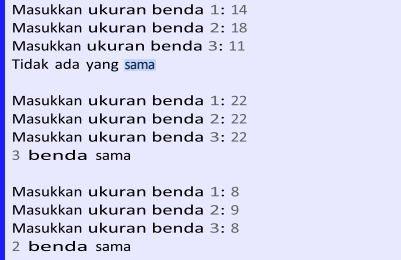
kejadian perkara, kamu menemukan tiga benda misterius dengan ukuran yang berbeda. Untuk

mengungkap kebenaran di balik kasus ini, masukkan ukuran ketiga benda tersebut. Jika ketiga

benda tersebut semua ukurannya sama, tampilkan pesan: "3 benda sama". Jika hanya ada dua

yang sama, tampilkan pesan "2 benda sama". Jika tidak ada yang sama maka tampilkan pesan:

"Tidak ada yang sama". Sebagai contoh, perhatikan input dan output berikut ini:



Lengkapi program tersebut dengan penanganan kesalahan jika pengguna memasukkan input

yang tidak valid.

**Latihan 3.1 case 1**

try:

    suhu = int(input("Masukkan suhu tubuh: "))

    if suhu >= 38:

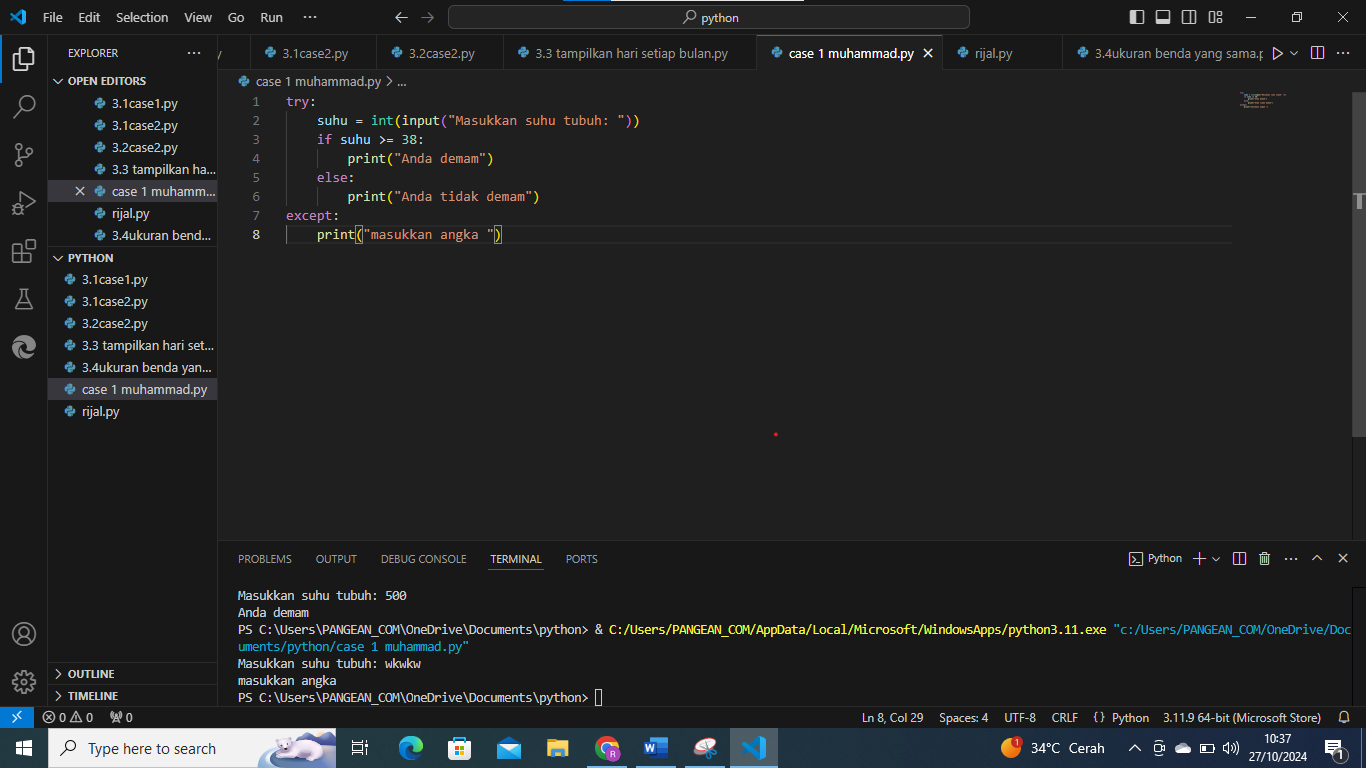
        print("Anda demam")

    else:

        print("Anda tidak demam")

except:

    print("masukkan angka ")



**Case 2**

try:

    bilangan = int(input("masukan sebuah bilangan"))

    if bilangan > 0:

        print("positif")

    elif bilangan < 0:

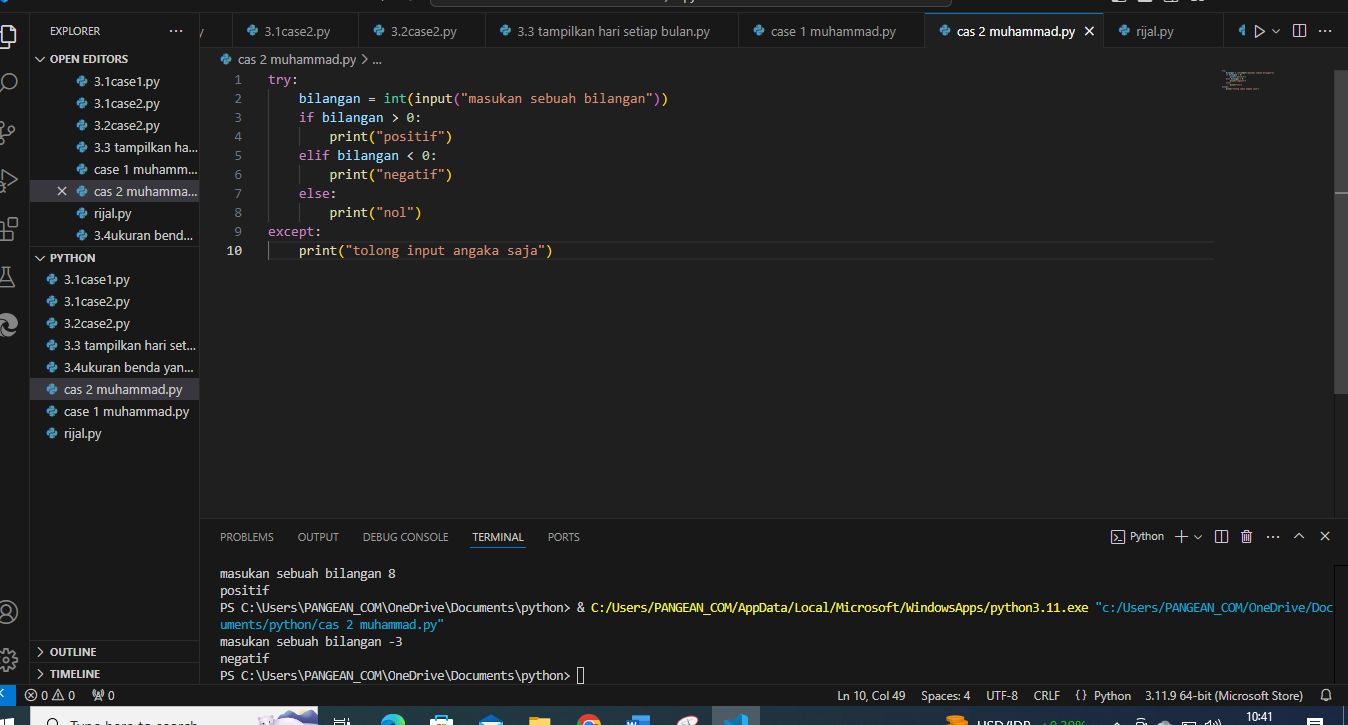
        print("negatif")

    else:

        print("nol")

except:

    print("tolong input angaka saja")



**Latihan 3.2**

try:

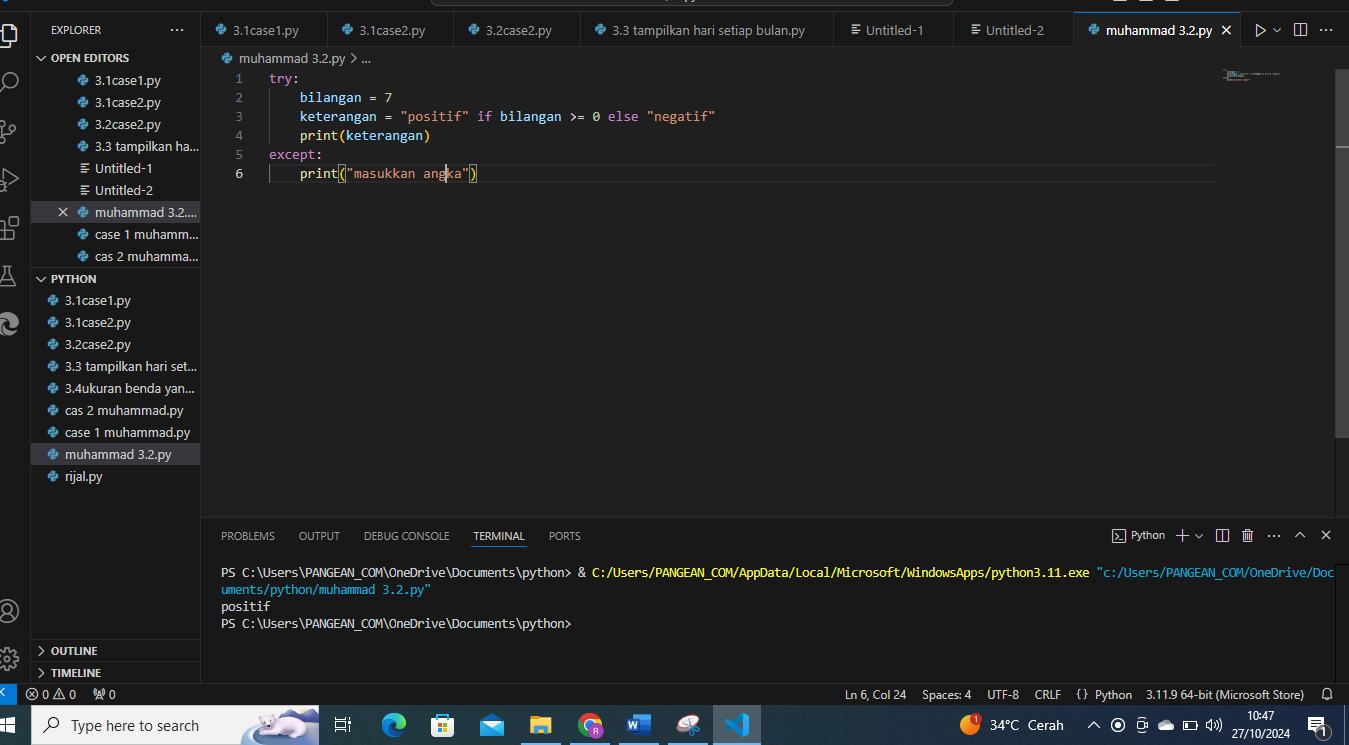
    bilangan = 7

    keterangan = "positif" if bilangan >= 0 else "negatif"

    print(keterangan)

except:

    print("masukkan angka")



**Latihan 3.3**

try :

    bulan = int(input("Masukkan nomor bulan (1-12): "))

    if bulan < 1 or bulan > 12:

        print("Bulan yang diinputkan tidak valid. Harus antara 1 dan 12.")

    # Jumlah hari dalam setiap bulan

    if bulan == 1:   # Januari

        jumlah\_hari = 31

    elif bulan == 2: # Februari

        jumlah\_hari = 28  # Untuk kesederhanaan, tidak menghitung tahun kabisat

    elif bulan == 3: # Maret

        jumlah\_hari = 31

    elif bulan == 4: # April

        jumlah\_hari = 30

    elif bulan == 5: # Mei

        jumlah\_hari = 31

    elif bulan == 6: # Juni

        jumlah\_hari = 30

    elif bulan == 7: # Juli

        jumlah\_hari = 31

    elif bulan == 8: # Agustus

        jumlah\_hari = 31

    elif bulan == 9: # September

        jumlah\_hari = 30

    elif bulan == 10: # Oktober

        jumlah\_hari = 31

    elif bulan == 11: # November

        jumlah\_hari = 30

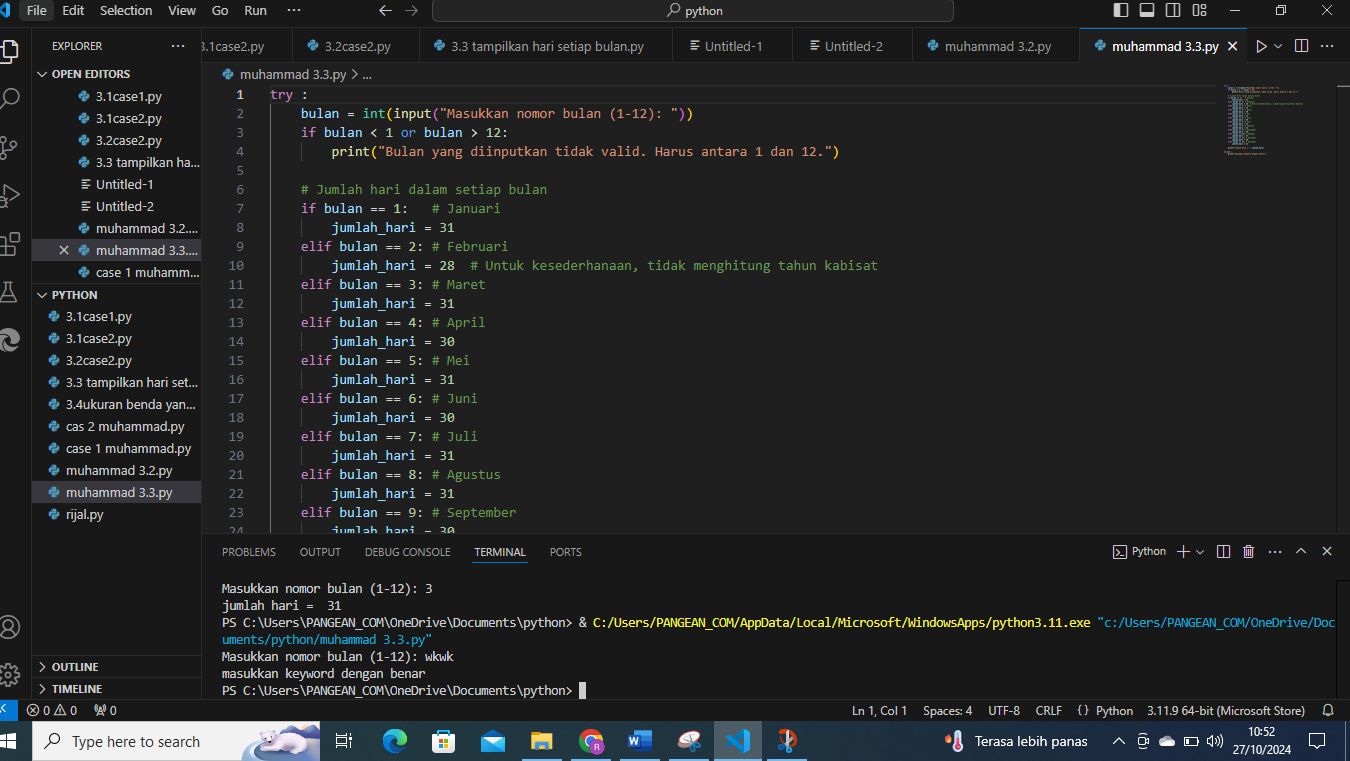
    elif bulan == 12: # Desember

        jumlah\_hari = 31

    print("jumlah hari = ", jumlah\_hari)

except :

    print("masukkan keyword dengan benar")



**Latihan3.4**

try:

    # Meminta input ukuran dari tiga benda

    ukuran1 = float(input("Masukkan ukuran benda 1: "))

    ukuran2 = float(input("Masukkan ukuran benda 2: "))

    ukuran3 = float(input("Masukkan ukuran benda 3: "))

    # Menentukan apakah ada benda yang sama

    if ukuran1 == ukuran2 == ukuran3:

        print("3 benda sama")

    elif ukuran1 == ukuran2 or ukuran1 == ukuran3 or ukuran2 == ukuran3:

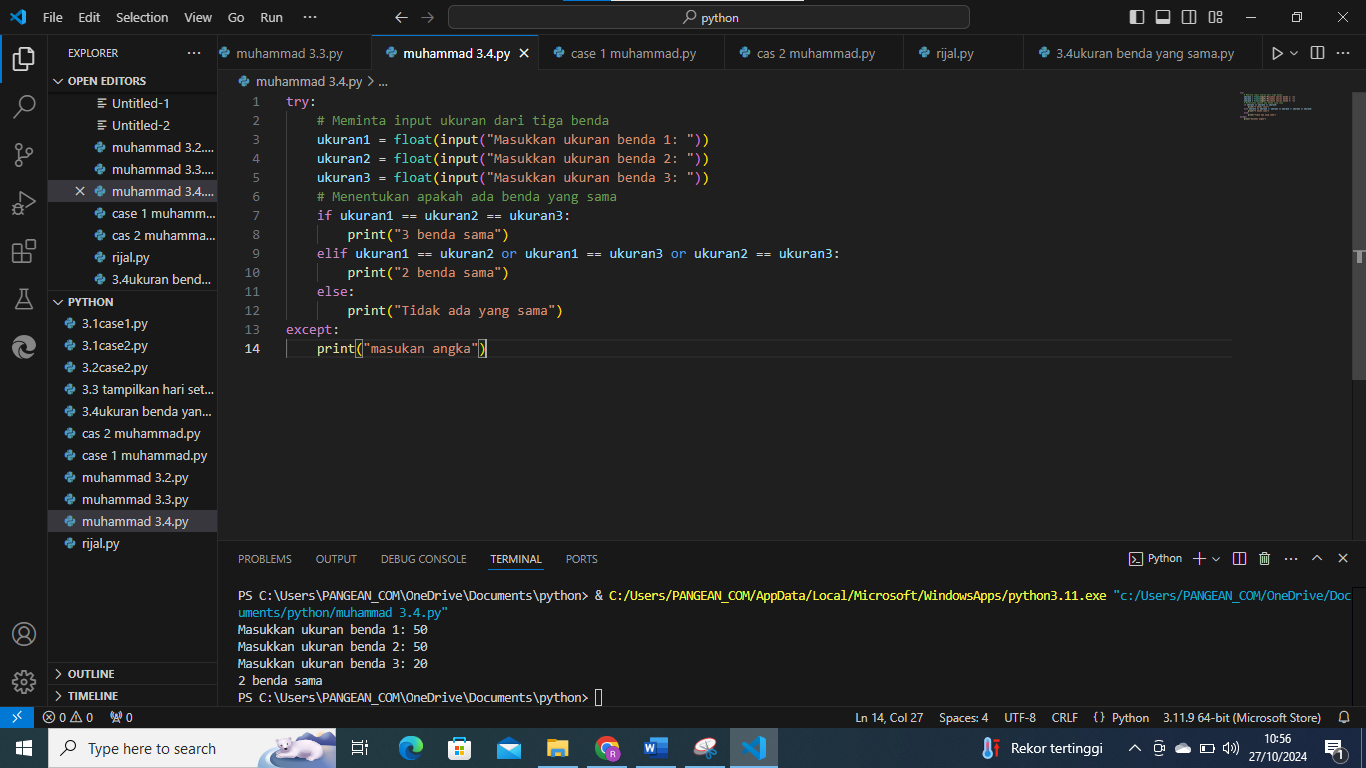
        print("2 benda sama")

    else:

        print("Tidak ada yang sama")

except:

    print("masukan angka")

****

**PENUTUP**

**KESIMPULAN**

Struktur kontrol percabangan merupakan elemen penting dalam pemrograman yang memungkinkan program untuk membuat keputusan berdasarkan kondisi tertentu. Dengan menggunakan pernyataan seperti if, else if, else, dan switch, programmer dapat mengendalikan alur eksekusi program dengan lebih fleksibel. Hal ini meningkatkan kejelasan kode, mempermudah pembacaan, dan memungkinkan program untuk beradaptasi dengan berbagai situasi. Secara keseluruhan, struktur kontrol percabangan sangat penting untuk menciptakan program yang efektif dan responsif.