|  |  |
| --- | --- |
| A picture containing text, clipart  Description automatically generated | **Laporan Praktikum Algoritma dan Pemrograman**  Semester Genap 2023/2024 |

|  |  |
| --- | --- |
| **NIM** | **<71231038>** |
| **Nama Lengkap** | **<AMELIA AGUTSIN>** |
| **Minggu ke / Materi** | **05 / Struktur Kontrol Perulangan** |

**SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.**

**SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS KRISTEN DUTA WACANA  
YOGYAKARTA  
2024**

# BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

## MATERI 1

5.3.1 Definisi Perulangan

Sebuah kumpulan yang dimana menyatakan perintah yang harus dijalankan berkali-kali. Untuk perulangan ada dua macam untuk membedakan antara lain;

1. Perulangan FOR

Perulangan yang digunakan untuk mengulangan dengan jumlah perulangan yang sudah ditentukan/dibatasi dalam kondisinya. Perulangan FOR berupa range / rentang nilai merujuk pada sebuah struktur pengendalian yang digunakan untuk mengulangi sebuah blok kode untuk sejumlah iterasi yang telah ditentukan. Pada perulangan for yang menggunakan range atau rentang nilai, Anda dapat menentukan nilai awal, nilai akhir, dan langkah (step) untuk setiap iterasi.

for i in range(1, 10, 2):  # Akan menampilkan angka ganjil dari 1 hingga 9

    print(i)

1. Perulangan WHILE

Perulangan yang digunakan untuk mengulang dengan jumlah perulangan yang tanpa diketahui/ tergantung dengan suatu kondisi. Jika kondisi tersebut bernilai TRUE maka perulangan akan berlanjut. Struktur perulangan while akan terus mengeksekusi blok kode di dalamnya selama kondisi yang diberikan bernilai benar (True).

bilangan = 0

genap = False

while genap == False:

    bilangan = int(input('Masukkan bilangan genap: '))

    if bilangan % 2 == 0:

        genap = True

print(bilangan, 'yang anda masukkan adalah bilangan genap')

1. Rekursif adalah teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri dengan argumen yang berbeda hingga kondisi tertentu terpenuhi. Rekursif biasanya digunakan untuk menyelesaikan masalah yang dapat dipecah menjadi kasus-kasus yang lebih sederhana dari masalah yang sama. Dalam rekursif, ada kasus dasar (base case) yang menghentikan rekursi dan kasus rekursif (recursive case) yang memanggil fungsi itu sendiri dengan argumen yang berbeda.

def fibonacci(n):

# Kasus dasar

if n <= 0:

return 0

elif n == 1:

return 1

# Kasus rekursif

else:

return fibonacci(n - 1) + fibonacci(n - 2)

## MATERI 2

5.3.2 Bentuk Perulangan for.

Perulngan for biasanya digunakan dalam kondisi:

1. Mengiterasi elemen dalam kumpulan data terurut

Perulangan for sangat cocok digunakan ketika Anda ingin mengiterasi elemen-elemen dalam kumpulan data terurut seperti list, tuple, atau string. Ini memungkinkan Anda untuk mengakses setiap elemen secara berurutan dan melakukan operasi tertentu pada setiap elemen tersebut.

# Mengiterasi elemen dalam list

buah = ['apel', 'jeruk', 'mangga', 'pisang']

for buah\_item in buah:

    print(buah\_item)

# Mengiterasi karakter dalam string

nama = "Python"

for huruf in nama:

    print(huruf)

1. Mengiterasi dalam rentang angka tertentu

Dengan menggunakan fungsi range() dalam perulangan for, Anda dapat dengan mudah mengiterasi dalam rentang angka tertentu. Ini berguna ketika Anda perlu melakukan operasi berulang sebanyak jumlah iterasi yang diketahui sebelumnya.

# Mencetak angka dari 1 sampai 5

for i in range(1, 6):

    print(i)

# Mencetak kelipatan 3 dari 0 sampai 20

for i in range(0, 21, 3):

    print(i)

1. Mengiterasi pasangan kunci-nilai dalam dictionary

Jika Anda memiliki sebuah dictionary dan ingin mengakses setiap pasangan kunci-nilai di dalamnya, Anda dapat menggunakan perulangan for dengan metode items() atau metode lainnya yang sesuai.

siswa = {

    'nama': 'John Doe',

    'kelas': '12A',

    'usia': 17

}

for kunci, nilai in siswa.items():

    print(f"{kunci}: {nilai}")

1. Mengiterasi baris dalam file

Ketika Anda membaca isi sebuah file, Anda dapat menggunakan perulangan for untuk mengiterasi setiap baris dalam file tersebut. Ini memudahkan pengolahan data atau informasi yang terdapat dalam file.

with open('data.txt', 'r') as file:

    for baris in file:

        print(baris.strip())

1. Mengiterasi elemen dalam objek yang dapat diiterasi

Python memungkinkan Anda untuk membuat objek yang dapat diiterasi dengan mendefinisikan metode \_\_iter\_\_() dan \_\_next\_\_(). Perulangan for dapat digunakan untuk mengiterasi elemen-elemen dalam objek seperti ini.

class PembacaAngka:

    def \_\_iter\_\_(self):

        self.angka = 1

        return self

    def \_\_next\_\_(self):

        if self.angka <= 5:

            hasil = self.angka

            self.angka += 1

            return hasil

        else:

            raise StopIteration()

pembaca = PembacaAngka()

for angka in pembaca:

    print(angka)

1. Melakukan operasi berulang pada serangkaian data

Perulangan for sering digunakan ketika Anda perlu melakukan operasi yang sama berulang kali pada serangkaian data atau objek yang dapat diiterasi. Ini dapat meningkatkan efisiensi dan mengurangi redundansi kode.

angka = [1, 2, 3, 4, 5]

for n in angka:

    print(n \* 2)

1. Mengiterasi baris dan kolom dalam matriks atau array dua dimensi

Dengan menggunakan perulangan for bersarang (nested for loops), Anda dapat dengan mudah mengiterasi baris dan kolom dalam matriks atau array dua dimensi.

matriks = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for baris in matriks:

    for elemen in baris:

        print(elemen, end=" ")

    print()

## MATERI 3

5.3.3 Bentuk Perulangan While

Dalam perulangan while, kondisi yang dievaluasi harus berupa ekspresi boolean. Jika kondisi bernilai True, maka blok kode di dalam perulangan akan dieksekusi. Setelah itu, kondisi akan dievaluasi kembali, dan jika masih bernilai True, blok kode akan dieksekusi lagi. Proses ini akan terus berlanjut hingga kondisi menjadi False. Terdapat dua kata kunci khusus yang dapat digunakan dalam perulangan while, yaitu:

1. Break

Kata kunci ini digunakan untuk keluar dari perulangan secara paksa, terlepas dari kondisi yang diberikan. Ketika kata kunci break ditemui dalam perulangan, maka perulangan akan segera berhenti dan eksekusi program akan melanjutkan ke baris kode setelah blok perulangan.

 for i in range(10):

    if i == 5:

        break

    print(i)

1. Continue

Kata kunci ini digunakan untuk melanjutkan ke iterasi berikutnya dalam perulangan. Ketika kata kunci continue ditemui, blok kode yang tersisa dalam iterasi tersebut akan dilewati, dan eksekusi akan kembali ke evaluasi kondisi perulangan untuk iterasi berikutnya.

for i in range(10):

    if i % 2 == 0:

        continue

    print(i)

## MATERI 3

5.3.4 Penggunaan Break dan Continue

Penggunaan break dan continue membantu dalam mengontrol alur eksekusi perulangan secara lebih efisien dan membuat kode menjadi lebih mudah dibaca dan dipahami. Namun, perlu diperhatikan bahwa penggunaan break dan continue yang berlebihan dapat membuat kode menjadi sulit dipahami dan dikelola. Oleh karena itu, sebaiknya gunakan break dan continue dengan bijak dan hanya saat diperlukan.

1. Break digunakan untuk mengakhiri perulangan secara paksa sebelum kondisi berhenti terpenuhi. Ketika break dieksekusi, program akan keluar dari blok perulangan terdekat dan melanjutkan eksekusi ke baris kode setelah blok perulangan tersebut.

Beberapa kasus di mana break sering digunakan:

* 1. Validasi input pengguna Misalnya, dalam sebuah perulangan untuk meminta input dari pengguna, break dapat digunakan untuk keluar dari perulangan jika input yang diberikan valid.
  2. Pencarian elemen dalam kumpulan data Jika Anda mencari sebuah elemen dalam sebuah list atau struktur data lainnya, break dapat digunakan untuk menghentikan perulangan setelah elemen ditemukan.
  3. Kondisi khusus Break dapat digunakan untuk keluar dari perulangan jika suatu kondisi khusus terpenuhi, seperti mencapai batasan tertentu atau terjadi kesalahan.

1. Continue digunakan untuk melewati iterasi saat ini dalam perulangan dan melanjutkan ke iterasi berikutnya. Ketika continue dieksekusi, sisa kode dalam blok perulangan untuk iterasi saat ini akan dilewati, dan kontrol program akan kembali ke awal perulangan untuk iterasi berikutnya.

Beberapa kasus di mana continue sering digunakan:

* 1. Menyaring data Misalnya, dalam sebuah perulangan untuk mengolah data, continue dapat digunakan untuk melewati elemen data yang tidak memenuhi kriteria tertentu.
  2. Menghindari kondisi khusus Continue dapat digunakan untuk melewati iterasi saat ini jika suatu kondisi khusus terpenuhi, seperti menghindari pembagian dengan nol atau menangani kasus-kasus khusus lainnya.
  3. Pemrosesan kondisional Dalam beberapa situasi, Anda mungkin hanya ingin melakukan operasi tertentu pada iterasi-iterasi yang memenuhi kondisi tertentu, dan menggunakan continue untuk melewati iterasi lainnya.

## MATERI 3

5.3.5 Konversi dari Bentuk for Menjadi Bentuk while

1. Dari for ke while
2. Temukan variabel iterasi dan nilai awalnya.
3. Tulis kondisi perulangan while berdasarkan nilai akhir atau kondisi berhenti.
4. Dalam blok while, lakukan operasi yang sama seperti dalam for.
5. Increment/update variabel iterasi sesuai dengan langkahnya.

# Bentuk for

for i in range(5):

print(i)

# Bentuk while

i = 0

while i < 5:

print(i)

i += 1

# Bentuk for

buah = ['apel', 'jeruk', 'mangga']

for item in buah:

    print(item)

    # Bentuk while

buah = ['apel', 'jeruk', 'mangga']

indeks = 0

while indeks < len(buah):

    item = buah[indeks]

    print(item)

    indeks += 1

1. Dari while ke for
2. Tentukan variabel iterasi dan gunakan fungsi range() dengan nilai awal, akhir, dan langkah yang sesuai.
3. Lakukan operasi yang sama seperti dalam while, tapi dengan menggunakan variabel iterasi dari for.

# Bentuk while

i = 0

total = 0

while i < 5:

    total += i

    i += 1

print(total)

# Bentuk for

total = 0

for i in range(5):

    total += i

print(total)

1. Catatan
   1. Tidak semua perulangan dapat dikonversi dengan mudah, tergantung pada kompleksitas kondisi.
   2. Pilih bentuk yang paling sesuai dengan kasus penggunaan dan gaya pemrograman.
   3. Perulangan for biasanya digunakan untuk mengiterasi kumpulan data terurut, sedangkan while digunakan ketika jumlah iterasi tidak diketahui sebelumnya dan bergantung pada kondisi tertentu.

## MATERI 3

5.4.1 Deret Bilangan

Deret bilangan Fibonacci adalah deret bilangan yang dimulai dengan 0 dan 1, dan setiap suku selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Rumus untuk menghitung deret bilangan Fibonacci adalah:

Fn = Fn-1 + Fn-2

dimana:

* Fn adalah suku ke-n
* Fn-1 adalah suku ke-(n-1)
* Fn-2 adalah suku ke-(n-2)

def fibonacci(n):

  if n <= 1:

    return n

  else:

    return fibonacci(n-1) + fibonacci(n-2)

# Mencetak 10 suku pertama deret Fibonacci

for i in range(10):

  print(fibonacci(i))

# BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

## SOAL 1

def perkalian(a, b):

  """

  Fungsi untuk menghitung perkalian dua bilangan dengan menggunakan penjumlahan.

  Args:

    a: Bilangan pertama.

    b: Bilangan kedua.

  Returns:

    Hasil perkalian a dan b.

  """

  if b < 0:

    return -perkalian(a, -b)

  elif b == 0:

    return 0

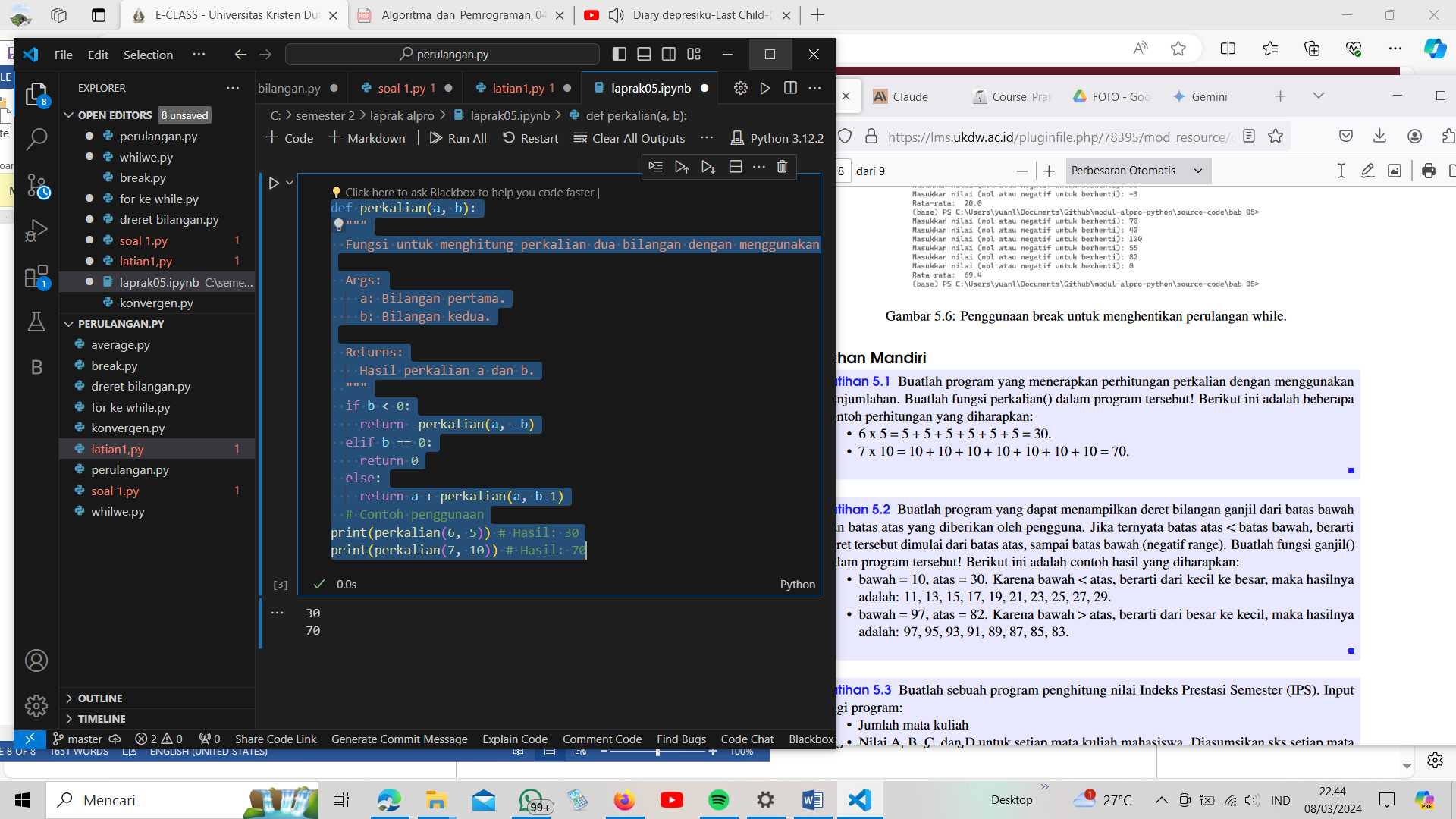
  else:

    return a + perkalian(a, b-1)

  # Contoh penggunaan

print(perkalian(6, 5)) # Hasil: 30

print(perkalian(7, 10)) # Hasil: 70



penjelasannya

* Fungsi perkalian menerima dua argumen, yaitu a dan b.
* Fungsi ini menggunakan rekursi untuk menghitung perkalian.
* Kasus dasar (base case) adalah ketika b sama dengan 0. Dalam kasus ini, hasil perkalian adalah 0.
* Kasus rekursif (recursive case) adalah ketika b lebih besar dari 0. Dalam kasus ini, hasil perkalian dihitung dengan menambahkan a ke hasil perkalian a dan b-1.
* Contoh penggunaan menunjukkan cara menggunakan fungsi perkalian untuk menghitung perkalian dua bilangan.

## SOAL 2

def ganjil(bawah, atas):

  """

  Fungsi untuk menampilkan deret bilangan ganjil dari batas bawah dan batas atas.

  Args:

    bawah: Batas bawah deret bilangan.

    atas: Batas atas deret bilangan.

  """

  if bawah > atas:

    # Deret bilangan dari besar ke kecil

    for i in range(atas, bawah+1, 2):

      print(i)

  else:

    # Deret bilangan dari kecil ke besar

    for i in range(bawah, atas+1, 2):

      print(i)

# Contoh penggunaan

bawah = 10

atas = 30

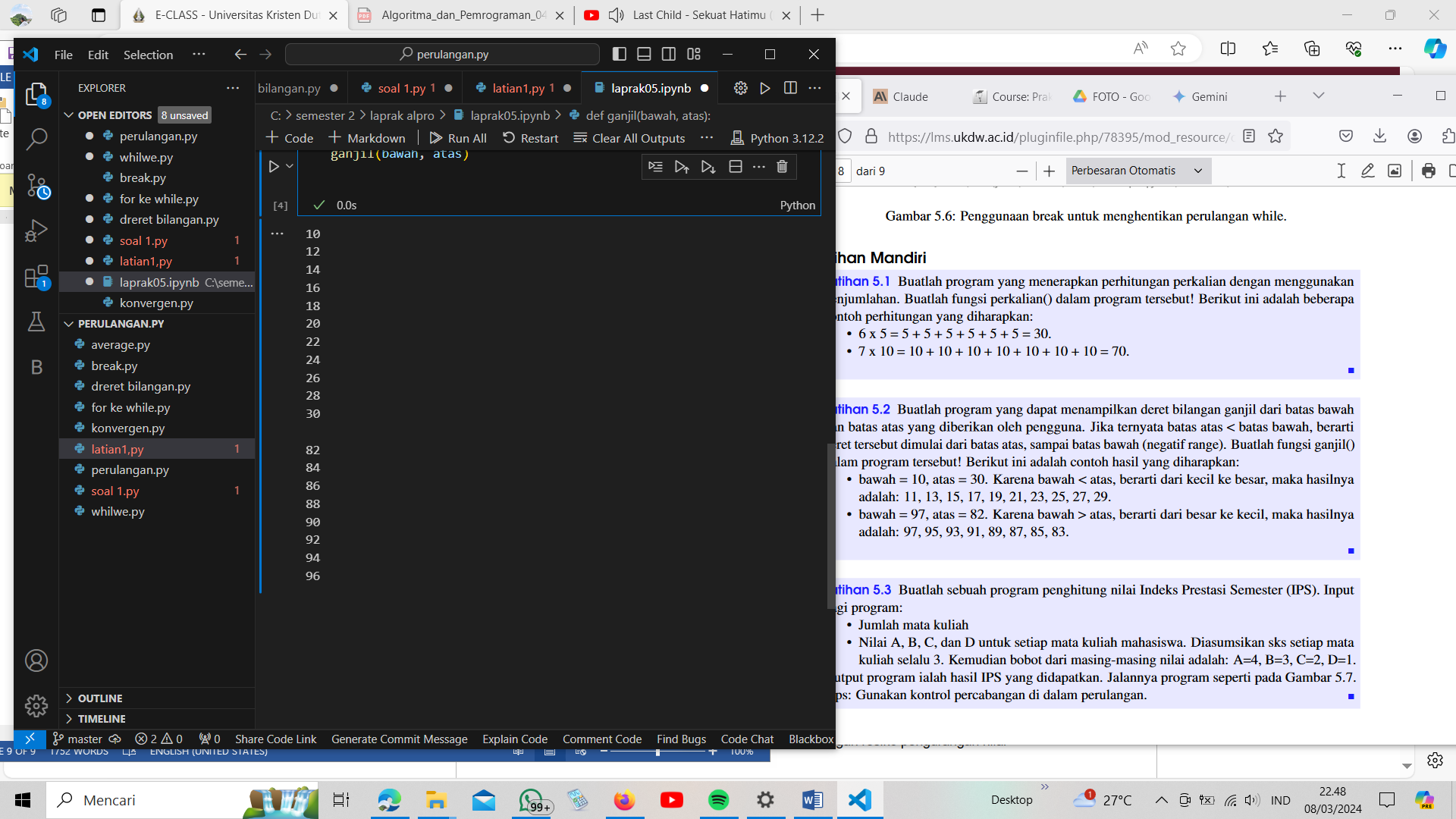
ganjil(bawah, atas)

print("")

bawah = 97

atas = 82

ganjil(bawah, atas)



* Fungsi ganjil menerima dua argumen, yaitu bawah dan atas.
* Fungsi ini terlebih dahulu memeriksa apakah bawah lebih besar dari atas.
* Jika bawah lebih besar dari atas, maka deret bilangan ditampilkan dari besar ke kecil.
* Jika bawah tidak lebih besar dari atas, maka deret bilangan ditampilkan dari kecil ke besar.
* Dalam kedua kasus, loop for digunakan untuk iterasi dari bawah ke atas dengan selisih 2 (untuk mendapatkan bilangan ganjil).
* Contoh penggunaan menunjukkan cara menggunakan fungsi ganjil untuk menampilkan deret bilangan ganjil dari dua batas yang berbeda.

## SOAL 3

def hitung\_ips(jumlah\_matkul):

  """

  Fungsi untuk menghitung nilai Indeks Prestasi Semester (IPS).

  Args:

    jumlah\_matkul: Jumlah mata kuliah yang diambil.

  Returns:

    Nilai IPS.

  """

  total\_bobot = 0

  total\_sks = 0

  for i in range(jumlah\_matkul):

    # Input nilai dan SKS

    nilai = input("Masukkan nilai mata kuliah {} (A/B/C/D): ".format(i+1)).upper()

    sks = 3

    # Hitung bobot

    if nilai == "A":

      bobot = 4

    elif nilai == "B":

      bobot = 3

    elif nilai == "C":

      bobot = 2

    elif nilai == "D":

      bobot = 1

    else:

      print("Nilai tidak valid.")

      return None

    # Hitung total bobot dan SKS

    total\_bobot += bobot \* sks

    total\_sks += sks

  # Hitung IPS

  ips = total\_bobot / total\_sks

  return ips

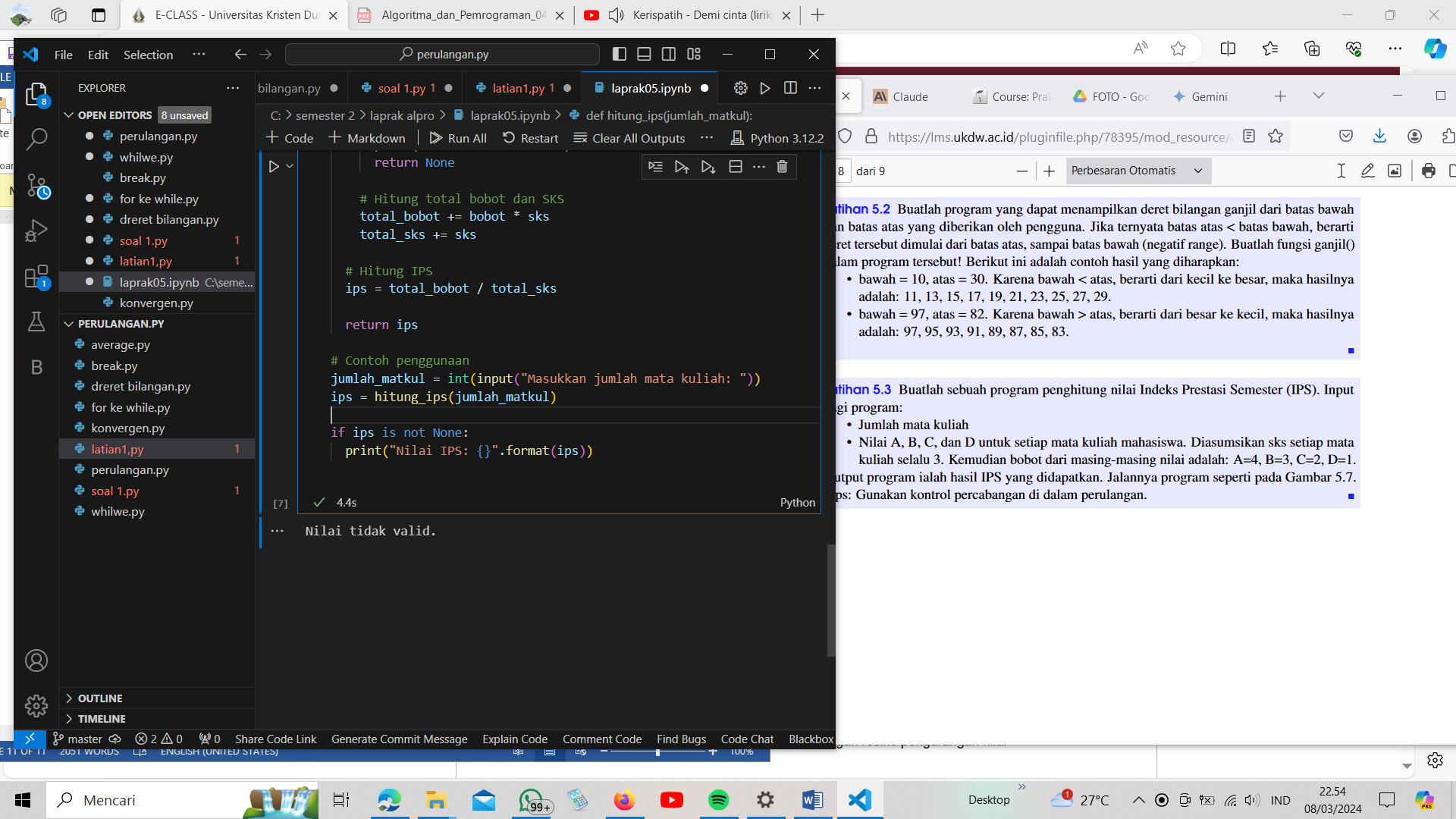
# Contoh penggunaan

jumlah\_matkul = int(input("Masukkan jumlah mata kuliah: "))

ips = hitung\_ips(jumlah\_matkul)

if ips is not None:

  print("Nilai IPS: {}".format(ips))



* Fungsi hitung\_ips menerima satu argumen, yaitu jumlah\_matkul.
* Fungsi ini menggunakan perulangan for untuk iterasi sebanyak jumlah\_matkul.
* Di dalam perulangan, pengguna diminta untuk memasukkan nilai dan SKS untuk setiap mata kuliah.
* Nilai diubah menjadi huruf besar dan kemudian divalidasi.
* Bobot dihitung berdasarkan nilai.
* Total bobot dan SKS dihitung.
* IPS dihitung dengan membagi total bobot dengan total SKS.
* Fungsi ini mengembalikan nilai IPS.