



---

# Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2024/2025

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.

NIM	<71231046>
Nama Lengkap	<FREIRE HANAN PUTRA>
Minggu ke / Materi	10 / LIST

PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS KRISTEN DUTA WACANA  
YOGYAKARTA  
2025

## BAGIAN 1: MATERI MINGGU INI (40%)

### MATERI 1 Sifat-Sifat List

List dalam Python merupakan struktur data yang menyimpan sekumpulan nilai dengan satu nama. Berbeda dengan string yang terbatas pada urutan karakter, list menawarkan fleksibilitas yang lebih besar karena dapat berisi campuran tipe data seperti karakter, angka (integer dan float), dan bahkan list di dalamnya (list bersarang). Sintaks penulisan list menggunakan tanda kurung siku [] untuk mengapit elemen-elemennya.

```
1 nilai_ujian = [80,75,70,90,81,84,92,71,65,80,70]
2 nama_pahlawan = ['Sukarno', 'Diponegoro', 'Jend. Sudirman', 'Cut Nya Dhien']
3 nilai_campuran = ['Javascript', 20, 34.4, True]
4 list_dalam_list = [23, [22, 20], 45]
5
```

Selain perbedaan tipe data yang dapat disimpan, list dan string juga berbeda dalam hal kemudahan perubahan nilai. List di Python bersifat *mutable*, memungkinkan kita untuk mengganti, menambah, atau menghapus elemennya secara langsung pada objek list yang sudah ada. Sementara itu, string bersifat *immutable*. Ini berarti, setiap operasi yang tampak seperti mengubah string sebenarnya akan menghasilkan string baru, dan string aslinya tetap tidak berubah.

```
1 # definisikan list berisi 4 nilai
2 data = [10,20,30,40]
3 # ubah nilai index ke-0 menjadi 50
4 data[0] = 50
5 # isinya sekarang: 50, 20, 30, 40
6 print(data)
7
8
9 # definisikan sebuah string
10 nama = 'Antonius Rachmat'
11 # ubah karakter index ke-0 menjadi Z
12 nama[0] = 'Z'
13 # tidak akan mencapai baris ini karena muncul error
14 # TypeError: 'str' object does not support item assignment
15 print(nama)
16
```

Dalam hal penggunaan memori, string dan list berperilaku berbeda. Ketika dua variabel string memiliki nilai yang sama, Python secara efisien akan membuat kedua variabel tersebut mengarah ke satu objek string yang sama di memori. Sebaliknya, meskipun dua list memiliki elemen yang identik, Python

akan tetap mengalokasikan ruang memori terpisah untuk masing-masing list, sehingga keduanya menjadi objek yang berbeda.

```
1 a = 'banana'
2 b = 'banana'
3 a is b #True
4
5 a = [1, 2, 3]
6 b = [1, 2, 3]
7 a is b #False
```

## MATERI 2 Mengakses dan Mengubah Isi List

Di bawah ini merupakan cara untuk mengakses indeks pada list. List dapat diubah isi dari list dengan menggunakan indeksnya juga. Berikut ini adalah contoh kode untuk menerapkannya:

```
1 thislist = ["apple", "banana", "cherry"]
2 print(thislist[1])
3 print(thislist[-1])
4 print(thislist[2:5])
5 print(thislist[:4])
6
7 thislist[1] = "jeruk"
8 print(thislist)
9
10 thislist2 = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
11 thislist2[1:3] = ["blackcurrant", "watermelon"]
12 print(thislist2)
13
```

## MATERI 3 Fungsi-Fungsi Untuk List

Penambahan list juga bisa lakukan dengan menggunakan operator +, pengulangan menggunakan \*.

Contoh:

```
1 thislist = [1,2,3,4,5]
2 thislist2 = [6,7,8]
3 listbaru = thislist + thislist2
4 listbaru2 = thislist * 2
5
6 print(thislist)
7 print(thislist2)
8 print(listbaru)
9 print(listbaru2)
```

### **1. append():**

Metode `append()` digunakan untuk menambahkan satu elemen baru ke akhir sebuah list. Elemen yang ditambahkan ini diperlakukan sebagai satu kesatuan objek, meskipun elemen tersebut berupa list lain. Contohnya, jika kita memiliki list `nama` dan menggunakan `nama.append(['tejo', 'ujo'])`, maka `['tejo', 'ujo']` akan ditambahkan sebagai satu elemen terakhir dalam list `nama`.

### **2. extend():**

Metode `extend()` juga berfungsi untuk menambahkan elemen ke dalam list. Namun, berbeda dengan `append()`, `extend()` memperlakukan setiap elemen dalam iterable (seperti list lain) yang ditambahkan sebagai elemen individual dalam list awal. Sebagai contoh, jika kita menggunakan `nama.extend(['tejo', 'ujo'])` pada list `nama`, maka `'tejo'` dan `'ujo'` akan ditambahkan sebagai dua elemen terpisah di akhir list `nama`.

### **3. sort():**

Metode `sort()` digunakan untuk mengurutkan elemen-elemen yang ada di dalam sebuah list secara langsung (in-place). Setelah pemanggilan `nama.sort()`, urutan elemen dalam list `nama` akan diubah menjadi urutan tertentu (biasanya urutan ascending atau alfabetikal) tanpa perlu membuat list baru.

### **4. pop():**

Metode `pop()` berfungsi untuk menghapus elemen dari sebuah list. Jika kita memberikan indeks elemen yang ingin dihapus kepada metode `pop()`, elemen pada indeks tersebut akan dihapus. Selain menghapus, `pop()` juga mengembalikan nilai (elemen) yang telah dihapus tersebut. Ini berguna jika kita ingin menghapus elemen berdasarkan posisinya dan sekaligus mengetahui nilai elemen yang dihapus.

### **5. del().**

Metode untuk menghapus elemen list tapi tidak perlu menampilkan elemen yang dihapus. 

```
>>> nama = ['kuncoro', 'felix', 'ryan', 'yuan'] >>> del nama[2]
```

 6. `remove`. Metode untuk menghapus elemen list berdasarkan nilai tertentu. 

```
>>> nama = ['kuncoro', 'felix', 'ryan', 'yuan'] >>>
```

```
nama.remove('kuncoro') 7. reverse. Metode untuk membalik urutan list. >>> nama = ['kuncoro',  
'felix' 'ryan', 'yuan'] >>> nama.reverse()
```

Python menyediakan fitur ringkas yang disebut *list comprehension* untuk membuat list baru berdasarkan list yang sudah ada. Dengan sintaks yang lebih pendek, kita dapat melakukan iterasi melalui elemen-elemen sebuah list dan menerapkan suatu kondisi atau operasi pada setiap elemennya. Contohnya, kita dapat menggunakan *list comprehension* untuk memeriksa setiap string dalam sebuah list dan menghasilkan list baru yang berisi nilai boolean (True atau False) yang menunjukkan apakah string tersebut mengandung substring "or".

```
nama = ["anton", "kuncoro", "buntoro", "juworo", "margono"]  
newlist = [x for x in nama if "or" in x]
```

Contoh untuk mengubah semua elemen list menjadi huruf besar.

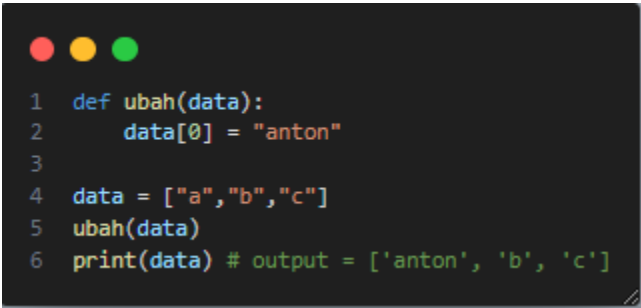
```
nama = ["anton", "kuncoro", "buntoro", "juworo", "margono"]  
newlist = [x.upper() for x in nama]
```

Contoh untuk memeriksa mengambil elemen list yang panjangnya lebih dari 6 huruf.

```
nama = ["anton", "kuncoro", "buntoro", "juworo", "margono"]  
newlist = [x for x in nama if len(x) > 6]
```

## MATERI List Sebagai Parameter Fungsi

Salah satu karakteristik penting dari list di Python adalah bahwa ketika ia digunakan sebagai parameter fungsi, ia bersifat *mutable*. Akibatnya, setiap perubahan yang Anda lakukan pada list di dalam fungsi akan memiliki efek samping, yaitu mengubah nilai dari list aslinya di luar cakupan fungsi. Ini perlu diperhatikan saat merancang fungsi yang menerima list sebagai input, karena operasi di dalamnya dapat memodifikasi data asli. Contoh:



```
1 def ubah(data):  
2     data[0] = "anton"  
3  
4     data = ["a", "b", "c"]  
5     ubah(data)  
6     print(data) # output = ['anton', 'b', 'c']
```

Ada beberapa keuntungan menggunakan list sebagai parameter:

- **Fleksibel:** Fungsi bisa menerima berbagai jenis data dalam list.
- **Bisa dipakai lagi:** Fungsi yang sama bisa digunakan untuk list yang berbeda.
- **Mudah dibagi:** List membantu memecah tugas besar jadi lebih kecil.
- **Data terstruktur:** List membantu mengorganisir data sebelum diproses fungsi.

Contoh lain untuk menghapus elemen pertama list:

```
1 def hapus(data):  
2     return data[1:]  
3  
4 data = ['anton', 'b', 'c']  
5 print(hapus(data))  
6 print(data)  
7 data = hapus(data)  
8 print(data)
```

Output sebagai berikut:

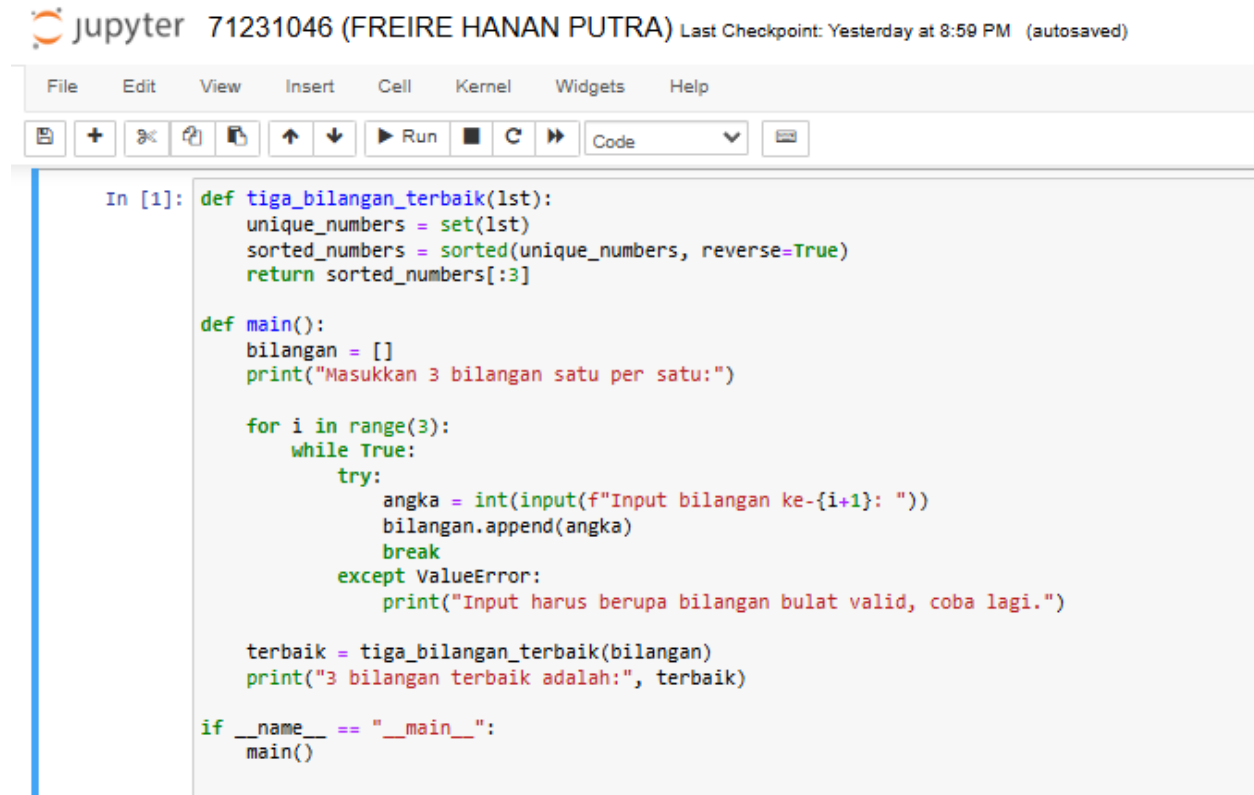
```
['b', 'c']  
['anton', 'b', 'c']  
['b', 'c']
```

## BAGIAN 2: LATIHAN MANDIRI (60%)

Link Github: [https://github.com/Freirehnn23/prak\\_alpro\\_week11](https://github.com/Freirehnn23/prak_alpro_week11)

### SOAL 1

Code:



```
In [1]: def tiga_bilangan_terbaik(lst):
        unique_numbers = set(lst)
        sorted_numbers = sorted(unique_numbers, reverse=True)
        return sorted_numbers[:3]

        def main():
            bilangan = []
            print("Masukkan 3 bilangan satu per satu:")

            for i in range(3):
                while True:
                    try:
                        angka = int(input(f"Input bilangan ke-{i+1}: "))
                        bilangan.append(angka)
                        break
                    except ValueError:
                        print("Input harus berupa bilangan bulat valid, coba lagi.")

            terbaik = tiga_bilangan_terbaik(bilangan)
            print("3 bilangan terbaik adalah:", terbaik)

        if __name__ == "__main__":
            main()
```

Penerapan:

```
Masukkan 3 bilangan satu per satu:
Input bilangan ke-1: 22
Input bilangan ke-2: 22
Input bilangan ke-3: 22
3 bilangan terbaik adalah: [22]
```

Penjelasan:

**def tiga\_bilangan\_terbaik(lst):**

- Mendefinisikan fungsi bernama tiga\_bilangan\_terbaik.
- Parameter lst adalah daftar (list) angka yang diberikan pengguna.

**unique\_numbers = set(lst)**

- Menghapus angka-angka yang sama (duplikat) dari daftar.
- Misalnya: [8, 9, 9] akan menjadi {8, 9}.

**sorted\_numbers = sorted(unique\_numbers, reverse=True)**

- Mengurutkan angka-angka tersebut dari yang paling besar ke paling kecil.
- Contoh: {8, 9} → [9, 8].

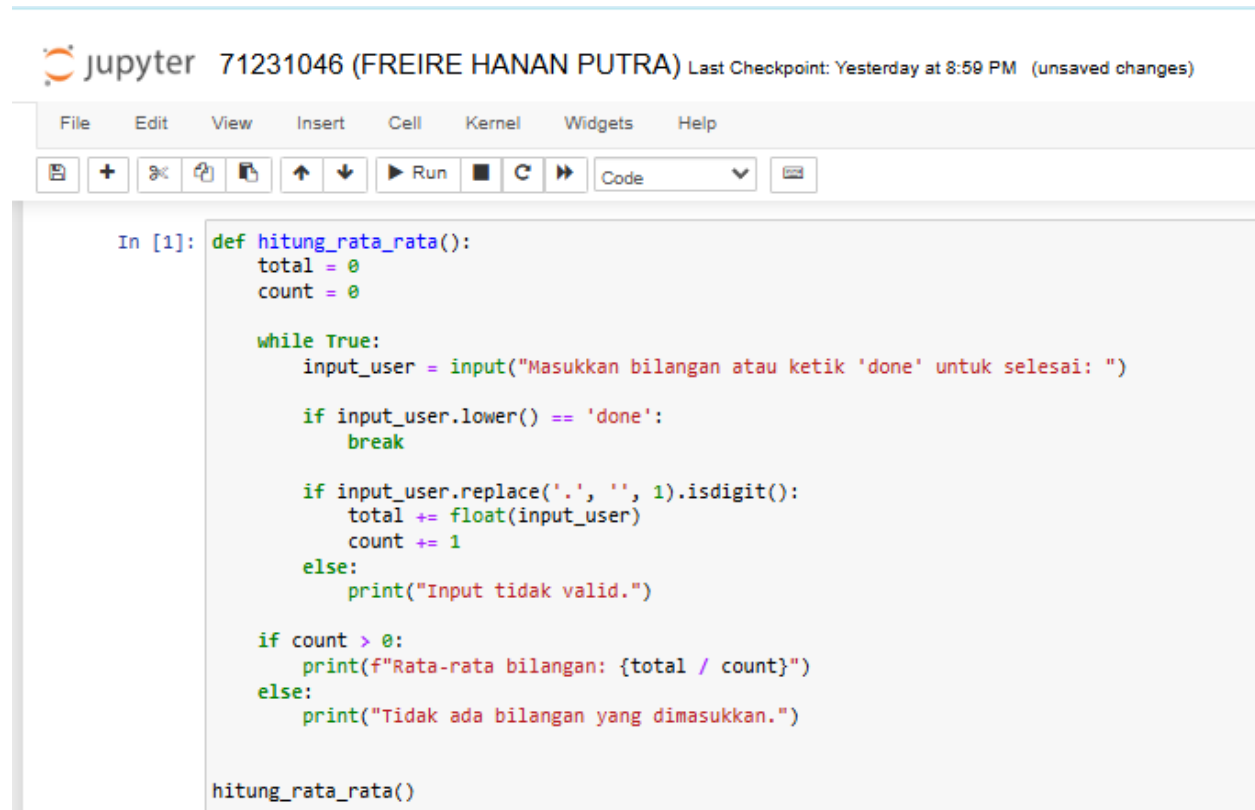
**return sorted\_numbers[:3]**

- Mengembalikan tiga bilangan teratas (jika ada).
- Bila jumlah elemen < 3, akan mengembalikan semua yang tersedia.



## SOAL 2

Code:



The screenshot shows a Jupyter Notebook window with the title "71231046 (FREIRE HANAN PUTRA)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell contains the following Python code:

```
In [1]: def hitung_rata_rata():
        total = 0
        count = 0

        while True:
            input_user = input("Masukkan bilangan atau ketik 'done' untuk selesai: ")

            if input_user.lower() == 'done':
                break

            if input_user.replace('.', '', 1).isdigit():
                total += float(input_user)
                count += 1
            else:
                print("Input tidak valid.")

        if count > 0:
            print(f"Rata-rata bilangan: {total / count}")
        else:
            print("Tidak ada bilangan yang dimasukkan.")

        hitung_rata_rata()
```

Penerapan:

```
Masukkan bilangan atau ketik 'done' untuk selesai: 21
Masukkan bilangan atau ketik 'done' untuk selesai: 21
Masukkan bilangan atau ketik 'done' untuk selesai: 21
Masukkan bilangan atau ketik 'done' untuk selesai: done
Rata-rata bilangan: 21.0
```

Penjelasan:

**total = 0 dan count = 0**

- Menginisialisasi variabel total untuk menjumlahkan semua bilangan.
- count menghitung berapa banyak bilangan yang dimasukkan.

**while True:**

- Perulangan tak terbatas sampai pengguna mengetik 'done'.

**input\_user = input(...)**

- Menerima input dari pengguna.
- Bisa berupa bilangan atau 'done'.

**if input\_user.lower() == 'done':**

- Mengecek apakah input adalah 'done' (dalam huruf kecil).
- Jika ya, keluar dari perulangan.

**if input\_user.replace('.', '', 1).isdigit():**

- Mengecek apakah input merupakan angka (termasuk desimal).
- `replace('.', '', 1)` menghapus satu titik desimal untuk memungkinkan pengecekan bilangan float seperti "12.5".
- Jika valid, input diubah ke float dan ditambahkan ke total.
- count ditambah 1.

**else:**

- Jika input bukan angka dan bukan 'done', maka dianggap tidak valid.
- Menampilkan pesan kesalahan.

**if count > 0:**

- Setelah input selesai, jika ada angka yang valid:
- Menghitung dan mencetak rata-rata: `total / count`.

**else:**

- Jika tidak ada angka valid yang dimasukkan, menampilkan pesan bahwa tidak ada input yang dihitung.

## SOAL 3

Code:

jupyter 71231046 (FREIRE HANAN PUTRA) Last Checkpoint: Yesterday at 8:59 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run

```
In [2]: def cari_kata_unik(file_path):
        try:
            with open(file_path, 'r') as file:
                artikel = file.read()
                artikel = artikel.lower()
                daftar_kata = artikel.split()
                kata_unik = set(daftar_kata)
                print("Kata-kata unik yang ditemukan:")
                for kata in kata_unik:
                    print(kata)

        except FileNotFoundError:
            print(f"File {file_path} tidak ditemukan.")
        except Exception as e:
            print(f"Terjadi kesalahan: {e}")

        file_path = input("Masukkan path file teks: ")
        cari_kata_unik(file_path)
```

Penerapan:

```
Masukkan path file teks: text.txt
Kata-kata unik yang ditemukan:
bersepeda,
orang
jakarta.
cuaca
lainnya
mereka
membuat
ini
beberapa
rumah
banyak
di
matahari
cerah
taman.
jalan,
lebih
sepanjang
sangat
juga
keluar
menikmati
makanan
segar
untuk
merasa
berjalan
ringan.
memilih
sementara
kaki
yang
dan
bahagia
bersinar.
energik.
berolahraga.
hari
udara
```

Penjelasan:

**1. def cari\_kata\_unik(file\_path):**

Baris ini mendefinisikan fungsi dengan nama `cari_kata_unik`, yang memiliki satu parameter `file_path`. Parameter ini mewakili lokasi file teks yang akan dibaca oleh program.

**2. try:**

Menandakan bahwa baris-baris selanjutnya akan diperiksa untuk kemungkinan terjadi error. Jika ada error saat membuka atau membaca file, program akan berpindah ke bagian `except`.

**3. with open(file\_path, 'r') as file:**

Membuka file teks dari lokasi yang diberikan oleh `file_path`. Mode `'r'` berarti file dibuka untuk dibaca (read-only).

Kata kunci `with` digunakan agar file otomatis ditutup setelah selesai dibaca, tanpa perlu menuliskan `file.close()` secara manual.

**4. artikel = file.read()**

Membaca seluruh isi file dan menyimpannya ke dalam variabel `artikel`. Tipe data hasil dari `read()` adalah string.

**5. artikel = artikel.lower()**

Mengubah seluruh huruf dalam string `artikel` menjadi huruf kecil (lowercase). Hal ini bertujuan agar kata yang sama tetapi menggunakan huruf besar dan kecil (misalnya Python dan python) dianggap sama.

**6. daftar\_kata = artikel.split()**

Memisahkan isi string `artikel` menjadi kata-kata individual. Fungsi `split()` memisahkan kata berdasarkan spasi atau whitespace. Hasilnya adalah list (`daftar`) dari kata-kata dalam teks.

**7. kata\_unik = set(daftar\_kata)**

Mengonversi `daftar_kata` menjadi set, yaitu struktur data yang hanya menyimpan nilai-nilai unik (tanpa duplikat).

**8. print("Kata-kata unik yang ditemukan:")**

Menampilkan judul ke layar sebelum daftar kata unik ditampilkan satu per satu.

**9. for kata in kata\_unik:**

Melakukan perulangan (loop) untuk setiap elemen (kata) di dalam `kata_unik`.

**10. print(kata)**

Dalam setiap iterasi, mencetak satu kata dari `kata_unik` ke layar. Karena set tidak terurut, urutan kata yang dicetak bisa berbeda setiap kali program dijalankan.

**11. except FileNotFoundError:**

Jika file yang diberikan tidak ditemukan pada path yang dimasukkan, maka baris ini akan dijalankan.

**12. `print(f"File {file_path} tidak ditemukan.")`**

Menampilkan pesan error yang menjelaskan bahwa file tidak ditemukan.

**13. `except Exception as e:`**

Menangkap error lain yang tidak spesifik, misalnya error membaca file yang rusak atau encoding error.

**14. `print(f"Terjadi kesalahan: {e}")`**

Menampilkan pesan error yang terjadi, agar pengguna tahu apa masalahnya.

**15. `file_path = input("Masukkan path file teks: ")`**

Meminta pengguna untuk memasukkan lokasi atau path file teks yang ingin diproses.

**16. `cari_kata_unik(file_path)`**

Memanggil fungsi `cari_kata_unik` dengan path yang telah dimasukkan pengguna agar seluruh proses di atas berjalan.