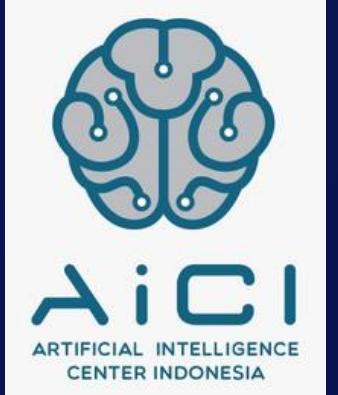




Kampus  
Merdeka  
INDONESIA JAYA



# File Handling



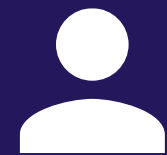
Penyusun Modul: Fitria Yunita Dewi  
Editor: M. Hamed Bagus Pratama



# File

# Operations

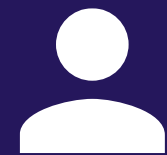




## File Operations



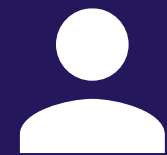
- File adalah kumpulan *byte* yang menyimpan data dalam format yang spesifik
- Secara umum terdapat 2 tipe file:
  - Tipe teks: txt, xml, html, csv, py, java, dsb.
  - Tipe biner: pdf, doc, jpg, png, mp4, zip, dsb.
- Kedua tipe file tersebut sama-sama berupa kumpulan bit, namun tiap bit pada tipe teks melambangkan karakter sedangkan pada biner melambangkan struktur data tertentu (*custom data*)
- *Encoding* menentukan bagaimana data disimpan di dalam file, beberapa *encoding* yang umum:
  - ASCII (128 karakter)
  - UNICODE (1.114.112 karakter)



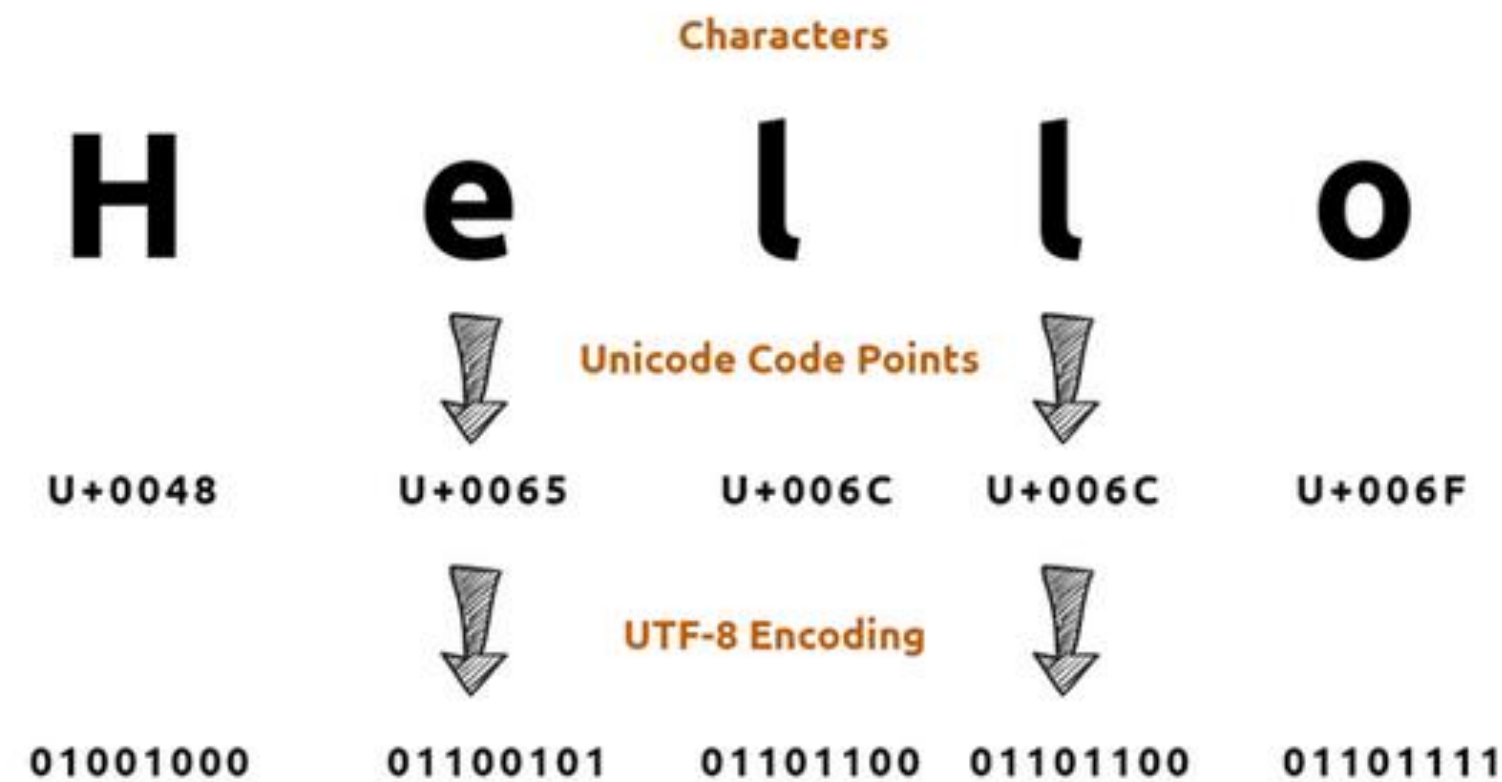
## Opening a file

- Sintaks untuk membuka file adalah  
`open (file, mode, encoding)`
- Parameter `file` adalah *path* dari file yang ingin dibuka. Jika berada dalam direktori yang sama maka path hanya berupa nama file saja
- Parameter `mode` adalah mode yang ingin dilakukan terhadap file. Secara default mode bernilai `'r'` (read mode)



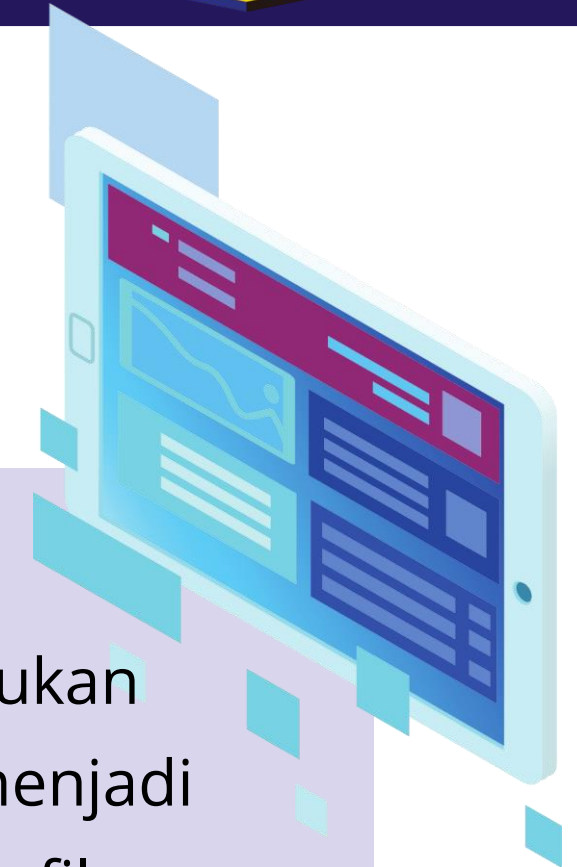


## Opening a File

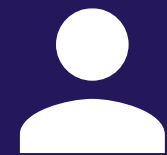


<https://nareshdevineniofficial.medium.com>

- Parameter `encoding` menentukan bagaimana data *byte* dibaca menjadi karakter, sehingga pembacaan file dapat berbeda ketika *encoding* yang digunakan berbeda
- *Encoding* yang paling populer dan sering digunakan adalah Unicode standard **'utf-8'**







## Opening a File

Contoh sebuah file txt

words.txt

```
1 Lorem
2 ipsum
3 dolor
4 sit
5 amet
6 €
```

```
Lorem
ipsum
dolor
sit
amet
â,~
```

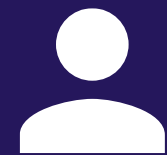
```
# without encoding
file = open('words.txt')
print(file.read())
```

```
# with encoding
file = open('words.txt', encoding='utf-8')
print(file.read())
```

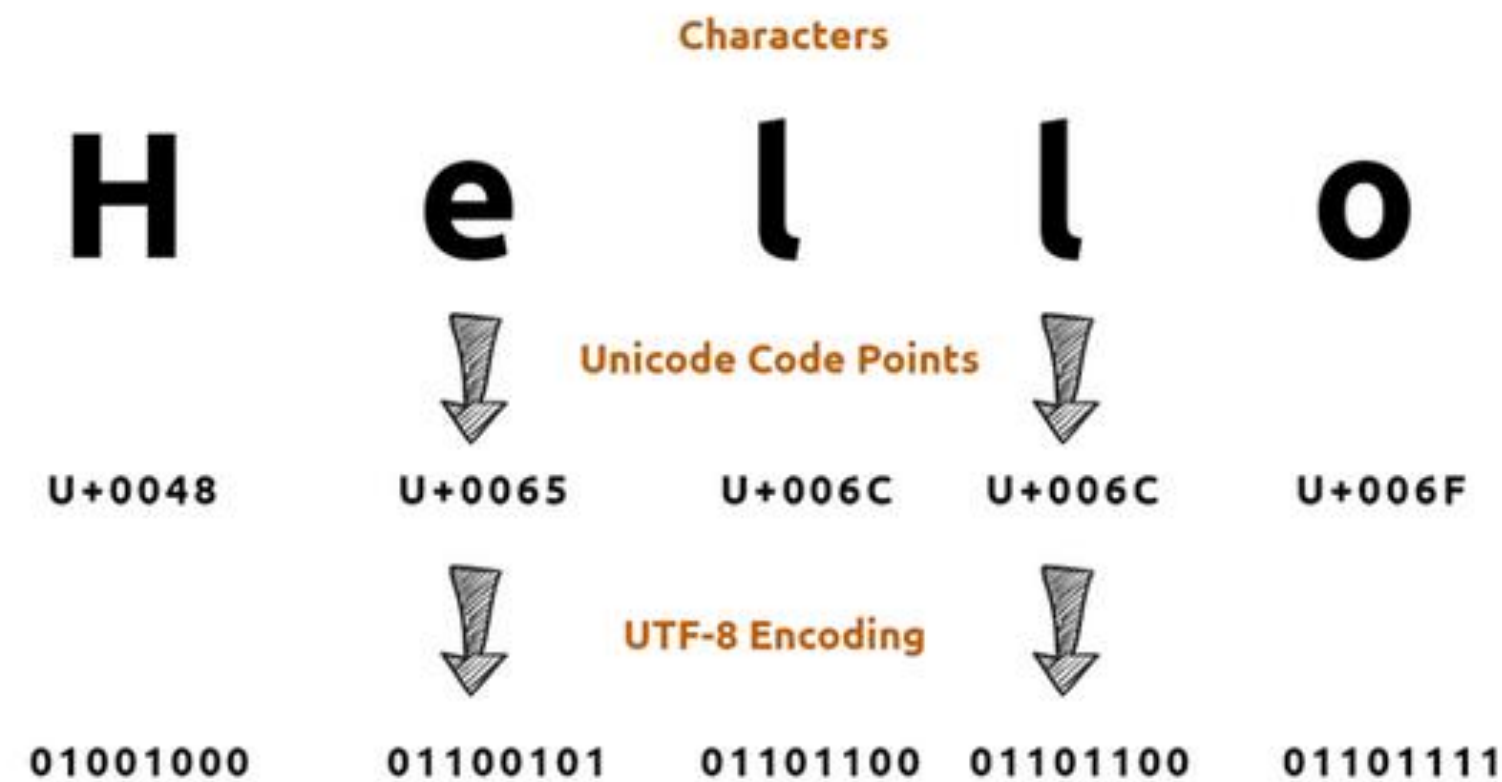
```
Lorem
ipsum
dolor
sit
amet
€
```

Untuk membuka konten keseluruhan dapat menggunakan metode `read()` pada objek file yang telah dibuka

Default value pada saat melakukan encoding bersifat *platform-dependent*, sehingga sangat dianjurkan untuk selalu menentukan tipe encoding pada saat membuka file untuk mencegah konsekuensi tak terduga

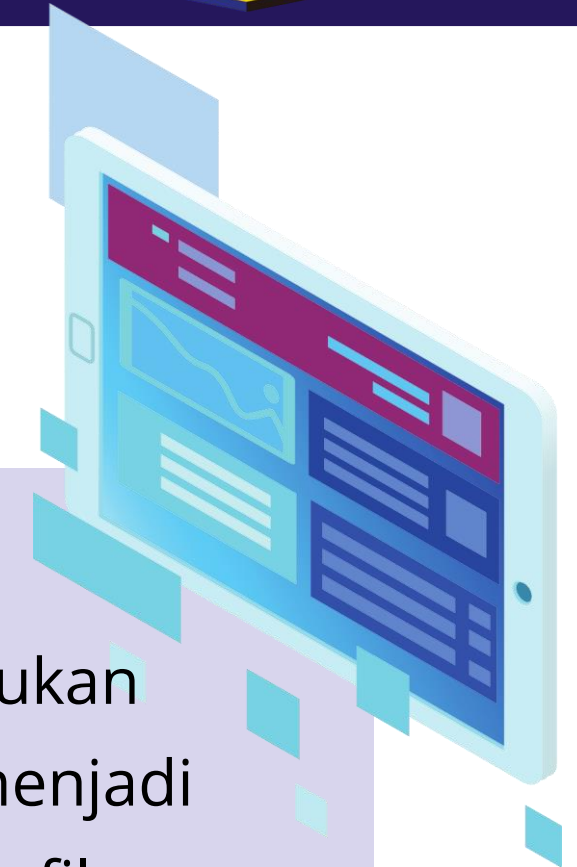


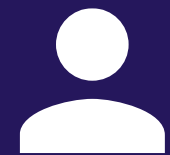
## Opening a File



<https://nareshdevineniofficial.medium.com>

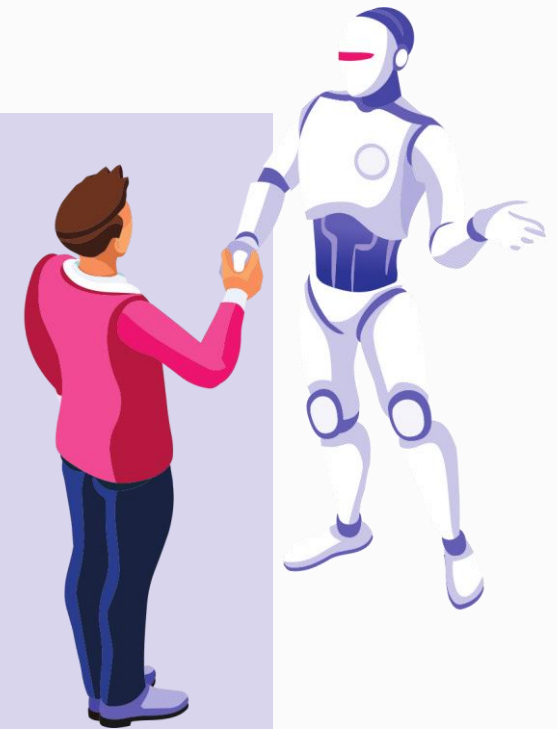
- Parameter `encoding` menentukan bagaimana data *byte* dibaca menjadi karakter, sehingga pembacaan file dapat berbeda ketika *encoding* yang digunakan berbeda
- *Encoding* yang paling populer dan sering digunakan adalah Unicode standard **'utf-8'**






## Opening a File

- Jangan lupa untuk selalu menutup file dengan `close()`
- Meskipun Python dapat menutup file otomatis, namun melakukan prosedur `close()` adalah praktik standar untuk mengurangi resiko modifikasi konten file tanpa sengaja
- Pada Python terdapat *context managers* yang memfasilitasi proses ini yaitu dengan menggunakan *keyword* `with`
- File hanya akan terbuka dalam scope `with`, sehingga kita tidak perlu secara manual memanggil metode `close()`

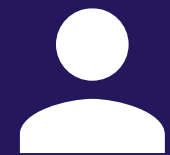


```
file = open('words.txt', encoding='utf-8')  
print(file.read())  
file.close()
```



```
with open('words.txt', encoding='utf-8') as file:  
    print(file.read())
```





# File Operations Modes

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

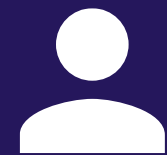
In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

- 'rb' berarti read in binary mode
- 'rt' berarti read in text mode



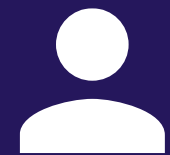


## Reading binary files

```
# read a binary file
# mode='rb'
with open('python_logo.png', mode='rb') as file:
    content = file.read()
    print(content)
```

```
b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02\x02\x00\x00\x02\x00\x08\x03\x00\x00\x00\xc7S\
\x00\x01sRGB\x00\xae\xce\x1c\xe9\x00\x00\x03\x00PLTEGpL7t\xa97x\xae\xfd\xcd<7l\x99\xff\xd3?
\xff\xcc;3r\xa6\xff\xbf@\xff\xdaK\xb2\xc9\\6n\x9d\xfd\xd1@6m\x9c7r\xa45m\x9b9u\xa7\xff\xcf>\xfc
```

- Ketika kita membaca file biner (pada contoh di atas adalah file png), output dapat berupa sekuens panjang berisi kumpulan *byte*. Karena metode `print()` menampilkan data dalam bentuk string, maka kita akan melihat representasi karakter dari kumpulan *byte* tersebut.
- Untuk dapat membaca data dalam bentuk gambar biasanya digunakan library tambahan misalnya yang biasanya dipakai adalah matplotlib, opencv, atau PIL, namun tidak akan pada modul ini



## Reading text files

Terdapat beberapa metode pembacaan *text file*

- `read(size)` : parameter `size` menentukan jumlah karakter yang dibaca, jika dikosongkan maka seluruh konten akan dibaca
- `readline()` : setiap kali dipanggil hanya membaca satu baris, pemanggilan berikutnya akan membaca baris selanjutnya. Program mengetahui adanya pergantian baris jika bertemu dengan karakter *newline* (`\n`)
- `readlines()` : membaca setiap baris yang belum terbaca pada file menjadi berbentuk list

Contoh:

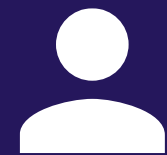
```
with open('words.txt', mode='rt', encoding='utf-8') as file:  
    print("read(size=3):",file.read(3))  
    print("readline() pertama:",file.readline())  
    print("readline() kedua:",file.readline())  
    print("readlines():",file.readlines())
```

`read(size=3):` Lor

`readline() pertama:` em

`readline() kedua:` ipsum

`readlines():` ['dolor\n', 'sit\n', 'amet\n', '€']



## Writing text files

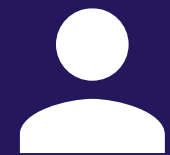
Seperti pada pembacaan, terdapat beberapa metode penulisan *text file*

- `write(string)` : menulis konten `string` ke dalam file
- `writelines()` : menulis kumpulan `string` ke dalam file

Dalam melakukan penulisan terdapat dua mode editing

- `w` (write): file dibuka dalam mode *write-only*, pointer dimulai pada permulaan file sehingga menimpa konten apapun yang sudah ada di dalam file (*DANGEROUS*)
- `a` (append): konten yang baru akan ditambahkan dari bagian terakhir pada file sehingga konten yang lama tidak tertimpa (*SAFE*)





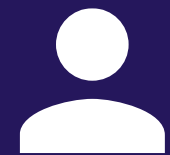
## Writing text files

```
# open with a mode
with open('words.txt', mode='a') as file:
    file.write('\nNew Line with append mode...')

# append multiple lines at once
with open('words.txt', mode='a') as file:
    list_to_append = ['\nMulti Line 1', '\nMulti Line 2', '\nMulti Line 3']
    file.writelines(list_to_append)

with open('words.txt', mode='rt', encoding='utf-8') as file:
    print(file.read())
```

Lorem  
ipsum  
dolor  
sit  
amet  
€  
New Line with append mode...  
Multi Line 1  
Multi Line 2  
Multi Line 3



## Rename, Delete and Create Files

Untuk kedua proses pertama perlu menggunakan modul `os`

### Rename

```
import os

os.rename('file_to_rename.txt',
          'file_with_new_name.txt')
```

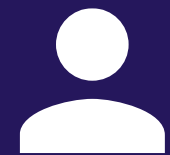
Jika file tidak ditemukan maka akan muncul error berikut

`FileNotFoundError`

Traceback (most recent call last)

Untuk mengatasi ini kita  
dapat menggunakan  
blok `try-except`

```
try:
    os.rename('file_to_rename.txt',
              'file_with_new_name.txt')
except:
    print('File not found with this name.')
```



## Rename, Delete and Create Files

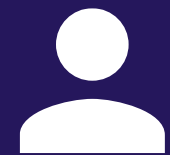
### Delete

```
import os  
  
os.remove('file_with_new_name.txt')
```

Seperti biasa untuk mencegah munculnya error  
karena file yang tidak ditemukan kita gunakan blok

try-except

```
try:  
    os.remove('file_with_new_name.txt')  
except:  
    print('File not found with this name.')
```



## Rename, Delete and Create Files

### Create

```
import os  
  
os.remove('file_with_new_name.txt')
```

Pembuatan file dapat menggunakan fungsi `open()` dengan pemilihan mode `'x'`. Jika file dengan nama tersebut sudah ada maka akan muncul error

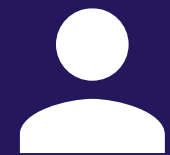
-----  
`FileExistsError`

Traceback (most recent call last)

Agar lebih aman maka kita gunakan blok `try-except` seperti berikut

```
try:  
    with open('file_to_create.txt', mode='x') as new_file:  
        new_file.write('This is a new file')  
except:  
    print('File already exist....')
```





## Getting folders and files list

### Cara 1

Menggunakan  
`os.listdir(path)`

```
path = os.getcwd()
content = os.listdir(path)
for c in content:
    print("content:", c)
```

### Cara 2

Menggunakan  
`os.scandir(path)`

```
path = os.getcwd()
folder = os.scandir(path)
for f in folder:
    print(f)
```

\*Untuk mengakses `path` direktori  
saat ini kita dapat menggunakan  
`os.getcwd()`

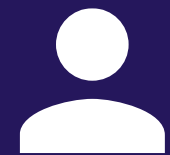
### Cara 3

Menggunakan  
modul `pathlib`

```
from pathlib import Path

path_content = Path(path)

for inner_content in path_content.iterdir():
    print(inner_content)
```



## Create folders

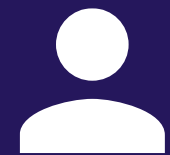
### Single directory

```
import os

os.mkdir('example_dir_1')
os.mkdir('example_dir_2')
# FileNotFoundError -> if you create again
```

### Multiple directories

```
# create a folder tree
os.makedirs('level 1/level 2/level 3')
os.makedirs('level 1/level 2/level 3', exist_ok=True) # to overcome FileNotFoundError
```

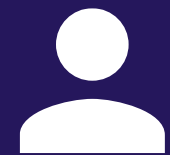


## Latihan

1. Buatlah program yang menghitung jumlah baris dalam file "story.txt" yang tidak diawali oleh huruf "T"

≡ story.txt ×

```
1  There is a playground.  
2  An aeroplane is in the sky.  
3  The sky is pink.  
4  Alphabets and numbers are allowed in the password.
```



## Latihan

1. Buatlah program yang menghitung jumlah baris dalam file "story.txt" yang tidak diawali oleh huruf "T"

≡ story.txt ×

```
1  There is a playground.  
2  An aeroplane is in the sky.  
3  The sky is pink.  
4  Alphabets and numbers are allowed in the password.
```

## Solusi:

```
count = 0  
with open("story.txt", "r", encoding='utf-8') as file:  
    for line in file:  
        if line[0] not in 'T':  
            count += 1  
print("No of lines not starting with 'T'=", count)
```

No of lines not starting with 'T' = 2





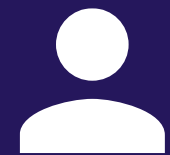
## Latihan



2. Buatlah program yang menghitung jumlah seluruh kata dalam file "story.txt"



## Latihan



2. Buatlah program yang menghitung jumlah seluruh kata dalam file "story.txt"

Solusi:

```
with open("story.txt", "r", encoding='utf-8') as file:  
    data = file.read()  
    words = data.split()  
    count = len(words)  
print("Total words are", count)
```

Total words are 22