FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Topic 5.1:
Text Files

CSGE601020 - Dasar-Dasar Pemrograman 1

Iis Afriyanti, M.Sc.

Acknowledgement

This slide is an adapted version of **Text Files** slides used in DDP1 Course (2020/2021) by Hafizh Rafizal Adnan, M.Kom, and **Files and Exceptions I** slides by Punch and Enbody (2013)

Some additional contents, illustrations and visual design elements are provided by **Lintang Matahari Hasani, M.Kom.**

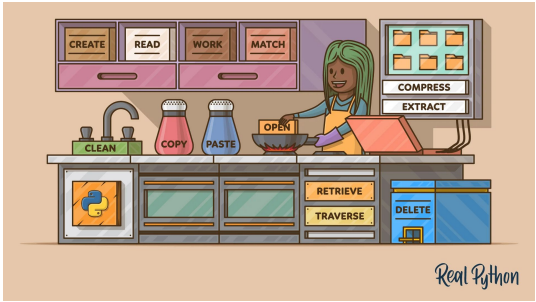
In this session, you will learn ...

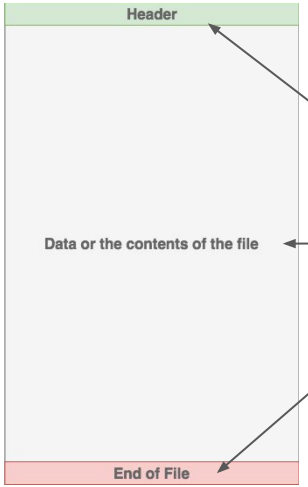
- What a file is
- How to create file object
- How to read file in Python
- How to write a file using Python

Files Overview

A file is **a collection of data** that is stored on **secondary storage** like a disk or a thumb drive (or USB flashdisk)

Accessing a file means establishing a **connection** between the file and the program and moving data between the two





File header describes the metadata of the file (name, size, type, etc.)

The contents of the file

End of file (EOF), a special character denoting the end of a file

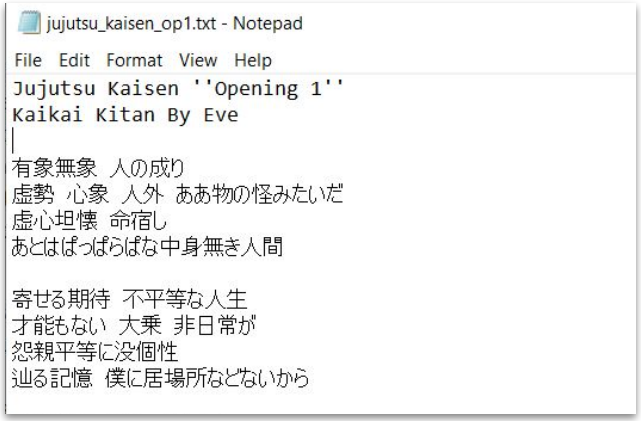
https://realpython.com/read-write-files-python/

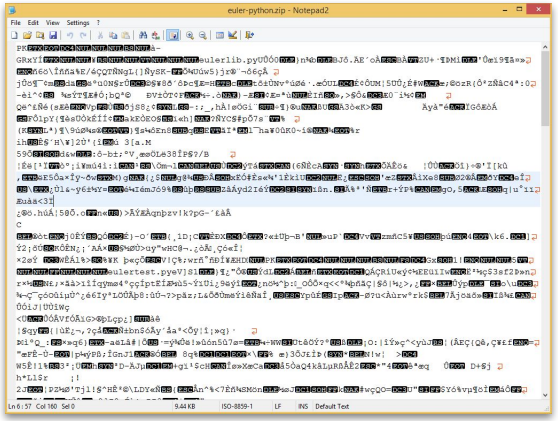
Types of Files

There are two general types of files that can be handled by Python:

1. **Text files**
   A text file is organized as Unicode/ASCII data and is generally human readable.

2. **Binary files**
   All the information is taken directly without translation as a sequence of bytes. Not human readable.

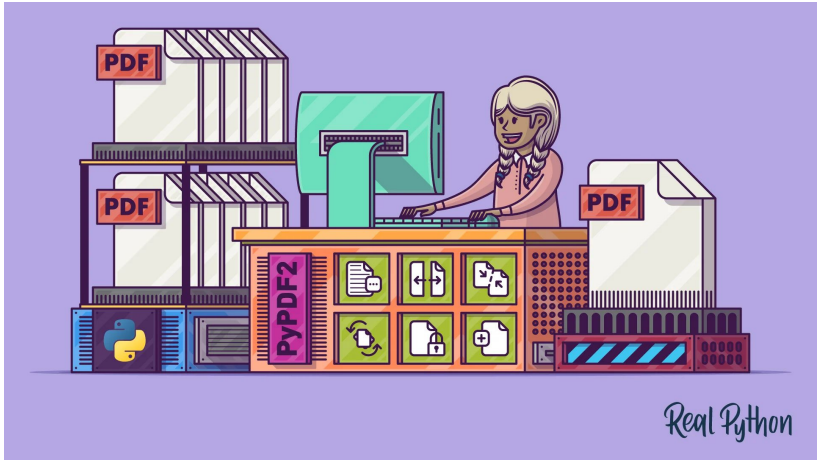




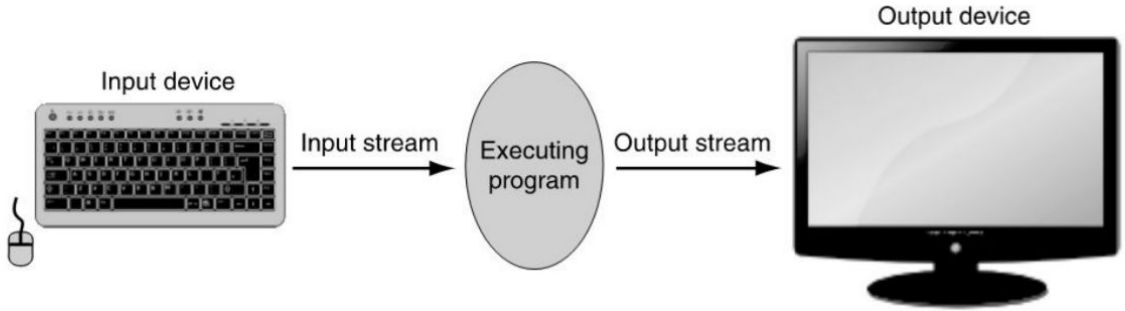
File Objects or Stream

When opening a file, you create a **file object** or **file stream** that is a connection between the file information on disk and the program.

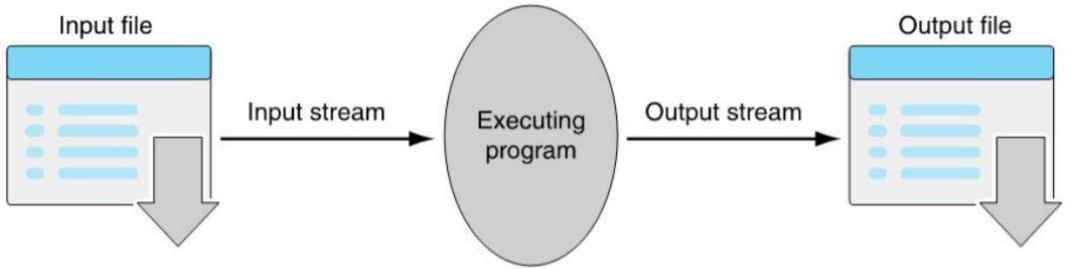
The stream contains a **buffer** of the information from the file, and provides the information to the program.



The Process



a) Standard input and output

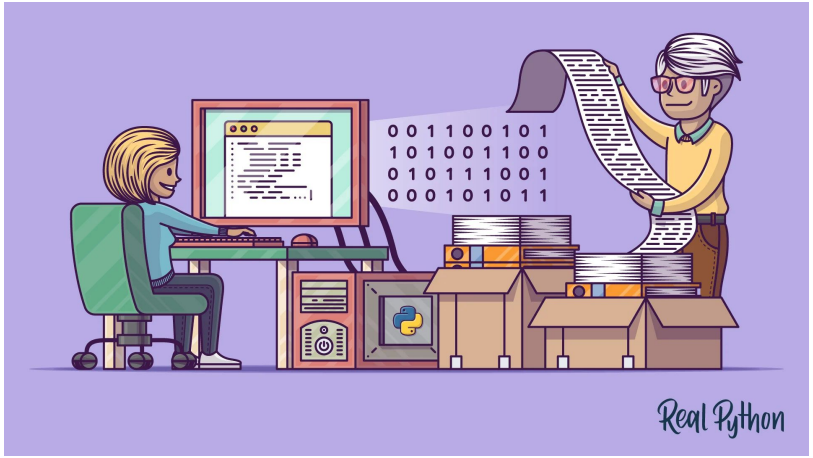


b) File input and output

FIGURE 5.1 Input-output streams.

Buffering

- ➔ **Reading from a disk is very slow**. Thus the computer will read a lot of data from a file in the hopes that, if you need the data in the future, it will be buffered in the file object.
- ➔ This means that the file object contains **a copy of information** from the file called a cache (pronounced "cash")



Making a File Object

```
my_file = open("my_file.txt", "r")
```

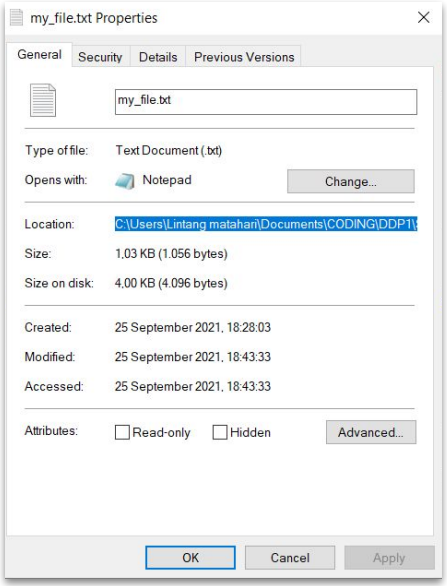
`my_file` is the file object. It contains the buffer of information. The open function creates the connection between the disk file and the file object.

The first quoted string is the file name on disk, the second is the mode to open it (here,`"r"` means to read)

Mind the File Path

When opened, the name of the file can come in one of two forms:

- ➔ `"my_file.txt"` assumes the file name is my_file.txt and it is located in the **current program directory**
- ➔ `"c:\\bill\\my_file.txt"` is the **fully qualified file name** and includes the directory information



A common error:

```
1 my_file = open('my_file.txt', 'r')
```

```
Shell
Python 3.7.9 (bundled)
>>> %cd 'C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 28-09-2021'
>>> %Run my_file.py
Traceback (most recent call last):
  File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 28-09-2021\my_file.py", line 1, in <module>
    my_file = open('my_file.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'my_file.txt'
>>>
```

➔ **FileNotFoundError**: Maybe we declared wrong file name or did not include the right directory information

Code Example: Read a File

Example 1

```
my_file = open("my_file.txt", "r")
print("Reading from my_file.txt:" )

# print each line from the file by preserving the new lines from the file
for line in my_file:
        print(line, end = '')

# flush the buffer ^^
my_file.close()
```

***Example 2** (The Fully Qualified Version)*

```
my_file_fully_qualified =  open("D:\Proyek\ddp\my_file.txt" , "r")
print("Reading from my_file.txt:" )

# print each line from the file by preserving the new lines from the file
for line in my_file_fully_qualified:
        print(line, end = '')

# flush the buffer ^^
my_file_fully_qualified. close()
```

File Modes

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of the file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open for updating (reading and writing)

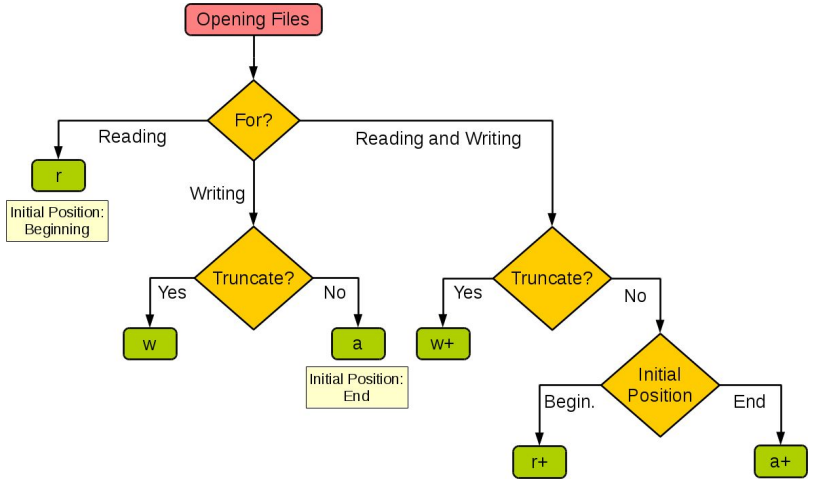
https://docs.python.org/3/library/functions.html#open

Mode	How Opened	File Exists	File Does Not Exist
'r'	read-only	Opens that file	Error
'w'	write-only	Clears the file contents	Creates and opens a new file
'a'	write-only	File contents left intact and new data appended at file's end	Creates and opens a new file
'r+'	read and write	Reads and overwrites from the file's beginning	Error
'w+'	read and write	Clears the file contents	Creates and opens a new file
'a+'	read and write	File contents left intact and read and write at file's end	Creates and opens a new file

TABLE 5.1 File Modes

Careful with write modes

- Be careful if you open a file with the `'w'` mode. **It sets an existing file's contents to be empty**, destroying any existing data.
- The `'a'` mode is nicer, allowing you to **write to the end** of an existing file **without changing** the existing contents



Triggering Question 1

Assume we want to open a file 'test.txt' located in the same directory with the open_file.py program as illustrated below.

```
/
|
├── documents/
|   ├── codes/  ← Your current working directory is here
|   |   └── test.txt  ← Accessing this file
|   |   └── open_file.py
|   └── nilai_ddp1.txt
└── jujutsu_kaisen_ep1.mp4
```

Which of the following is the correct way to open the file for reading as a text file in open_file.py? Select all that apply.

Write the answer in the **comment section**

a.
```
my_file = open("test.txt", "wt")
```

b.
```
my_file = open("test.txt", "r")
```

c.
```
my_file = open("/documents/codes/test.txt", "rb")
```

d.
```
my_file = open("test.txt", "w")
```

e.
```
my_file = open("test.txt", "rt")
```

Text files use strings

If you are interacting with text files, remember that **everything is a string**, everything read is a string. if you write to a file, you **can only write a string**

Getting File Contents

Once you have a file object:

➔ `fileObject.read()`

reads the entire contents of the file as a string and returns it. It can take an optional argument integer to limit the read to N bytes, that is `fileObject.read(N)`

```
my_file = open("my_file.txt", "r")

# Print entire lines
print(my_file.read())

# flush the buffer ^^
my_file.close()
```

➔ `fileObject.readline()`

delivers the next line as a string

```
my_file = open("my_file.txt", "r")

# Print line 1
print(my_file.readline(), end = '')
# Print line 2
print(my_file.readline(), end = '')
# Print line 3
print(my_file.readline(), end = '')

# flush the buffer ^^
my_file.close()
```

More File Reads

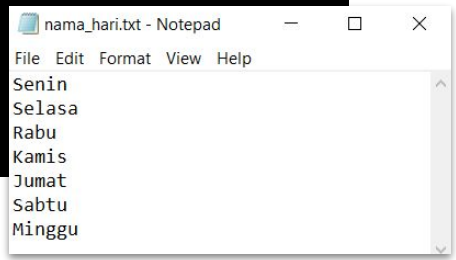
➔ `fileObject.readlines()`

returns **a single list of all the lines** from the file

```
my_file = open("nama_hari.txt", "r")

# Print entire lines
print(my_file.readlines())

# flush the buffer ^^
my_file.close()
```




```
>>> %Run my_file.py
['Senin\n', 'Selasa\n', 'Rabu\n', 'Kamis\n', 'Jumat\n', 'Sabtu\n', 'Minggu']
```

➔ `for line in fileObject:`
iterator to **go through the lines** of a file

```
my_file = open("nama_hari.txt", "r")

# Print entire lines
for line in my_file:
      print(line, end = '')

# flush the buffer ^^
my_file.close()
```

```
>>> %Run my_file.py
Senin
Selasa
Rabu
Kamis
Jumat
Sabtu
Minggu
```

More File Reads: Non Latin Alphabet Characters

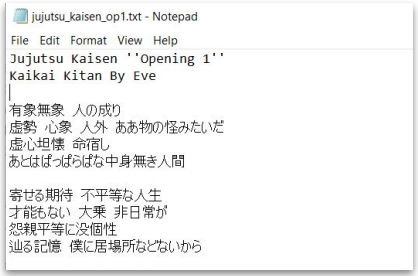
➔ Sometimes, it is necessary to use `encoding = UTF-8` in `open()` function

A common error:

```
my_file = open("jujutsu_kaisen_op1.txt", "r")

# Print entire lines
print(my_file.read())

# flush the buffer ^^
my_file.close()
```




```
Traceback (most recent call last):
  File "C:\Users\Lintang matahari\Documents\CODING\DDP1\Session 28-09-2021\tes.py", line 4, in <module>
    print(my_file.read())
  File "C:\Users\Lintang matahari\AppData\Local\Programs\Thonny\lib\encodings\cp1252.py", line 23, in decode
    return codecs.charmap_decode(input,self.errors,decoding_table)[0]
UnicodeDecodeError: 'charmap' codec can't decode byte 0x81 in position 70: character maps to <undefined>
```

Solution:

```
my_file = open("jujutsu_kaisen_op1.txt", "r",encoding = "UTF-8")
```

Triggering Question 2

What does this program do? Assume that no file named 'test.txt' existed in the program directory

If test.txt existed, what would have happened if the program was executed?

Write the answer in the **comment section**

```
my_file = open("test.txt", "rt")

number = 0
for line in my_file:
    number += 1
print(number)

my_file.close()
```

1

```
my_file = open("test.txt", "r")

number = 0
for line in my_file:
    number = number + len(line)
print(number)

my_file.close()
```

2

Writing to a File

➔ Once you have created a file object, opened for writing, you can use the `print()` command

you add `file = file_to_write` to the `print()` command

```
# open file for writing:
#    creates file if it does not exist
#    overwrites file if it exists
>>> temp_file = open("temp.txt","w")
>>> print("first line", file=temp_file)
>>> print("second line", file=temp_file)
>>> temp_file.close()
```

Code Example: Write a File

```
my_file = open("my_file_w_mode.txt", "w")

print("Mantappu Jiwaa!", file = my_file)
print("", file = my_file)
print(12345678, file = my_file)

my_file.close()
```

Close Method

When the program is finished with a file, we close the file

1. **flush the buffer contents** from the computer to the file
2. **tear down the connection** to the file

close is a method of a file obj `file_obj.close()`

All files should be closed!

Code Example: Using Close Method

```
input_file = open("my_file.txt","r")
output_file = open("my_file_inverse.txt","w")
line_counter = 1

for line_str in input_file:
     new_str = ''
     line_str = line_str.strip() # get rid of carriage return (newline)

     for char in line_str:
          new_str = char + new_str

     print(new_str,file = output_file)

     # observe progress
     print('Line {:d}: {:s} reversed is {:s}'.format(line_counter,line_str,new_str))

     line_counter += 1

input_file.close()
output_file.close()
```

Review Question (1)

```
my_file_example = open('Tes.txt', 'w+')
print('ABCD', file = my_file_example)
print('EFGH')
print(1234, file = my_file_example)
my_file_example.close()
```

1. Assuming `Tes.txt` is not yet existed, **what is written in `Tes.txt` after the program execution?**

Review Question (2)

```
my_file_example = open('Tes.txt', 'w+')
print('ABCD',file = my_file_example)
print('EFGH')
print(1234, file = my_file_example)
my_file_example.close()
```

2. What happen if this program is **executed twice**?

Review Question 3

My Diary:

Create a program to write anything you wanna write.

The written input needs to be saved in a file with additional information including today's date (*Hint: use python* `date.today()`), author name (program input), current location (program input), count of characters written, and count of words. The files append the new input with the old one.

my_diary.TXT

```
Ditulis oleh [Nama] di [Lokasi] pada [Tanggal Hari Ini]

[Pesan yang ditulis]

============================
```

Post your code in the Scele Forum

UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

Q&A Session