Opening and Closing Files:

- **The *open* Function:**

  Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object which would be utilized to call other support methods associated with it.

- **Syntax:**

```
file object = open(file_name [, access_mode][, buffering])
```

```
with open('USD_index.csv', 'rb') as inp, open('USD_index2.csv', 'wb') as out:
    writer = csv.writer(out)
```

A list of the different modes of opening a file:

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the **default** mode.
r+	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.

A list of the different modes of opening a file:

wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

The *file* object atrributes:

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.

```
>>> fo = open("text.txt", "w")
>>> print(fo.name)
text.txt
>>> print(fo.closed)
False
>>> print(fo.mode)
w
>>>
```

1. Opening Files

Prepares the file for reading:

A. Links the file variable with the physical file (references to the file variable are references to the physical file).
B. Positions the file pointer at the start of the file.

Format:[1]

```
<file variable> = open(<file name>, "r")
```

Example:

(Constant file name)

```
inputFile = open("data.txt", "r")
```

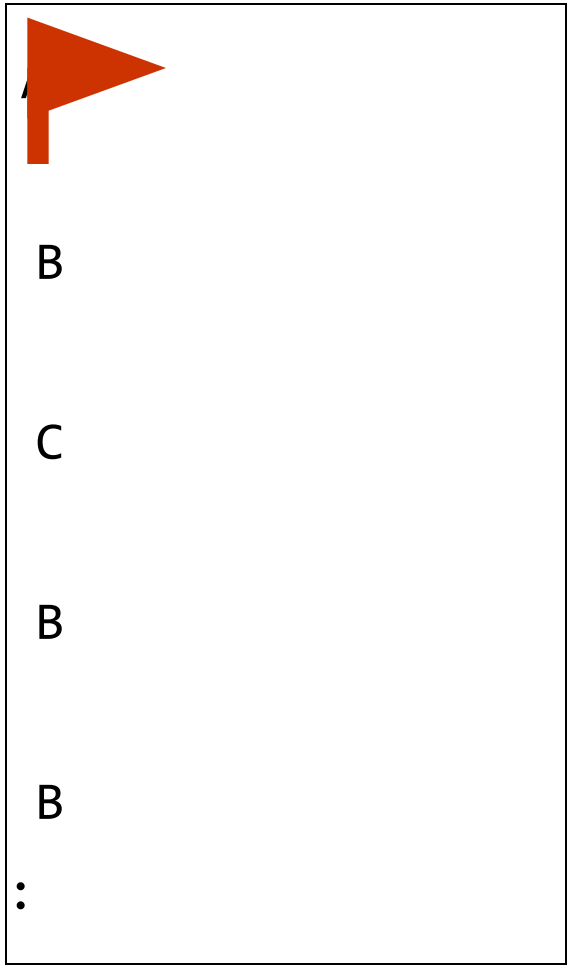
OR

(Variable file name: entered by user at runtime)

```
filename = input("Enter name of input file: ")
inputFile = open(filename, "r")
```

B. Positioning The File Pointer

letters.txt



2. Reading Information From Files

- Typically reading is done within the body of a loop
- Each execution of the loop will read a line from the file into a string

Format:

```
for <variable to store a string> in <name of file variable>:
    <Do something with the string read from file>
```

Example:

```
for line in inputFile:
    print(line)  # Echo file contents back onscreen
```

Closing The File

- Although a file is automatically closed when your program ends it is still a good style to explicitly close your file as soon as the program is done with it.
  - What if the program encounters a runtime error and crashes before it reaches the end? The input file may remain 'locked' an inaccessible state because it's still open.
- **Format:**

  `<name of file variable>.close()`

- **Example:**

```
inputFile.close()
```

1. Opening The File

Format[1]:

```
<name of file variable> = open(<file name>, "w")
```

Example:

(Constant file name)

```
outputFile = open("gpa.txt", "w")
```

(Variable file name: entered by user at runtime)

```
outputFileName = input("Enter the name of the output file
                        to record the GPA's to: ")
outputFile = open(outputFileName, "w")
```


1 Typically the file is created in the same directory/folder as the Python program.

3. Writing To A File

- You can use the '`write()`' function in conjunction with a file variable.
- Note however that this function will ONLY take a string parameter (everything else must be converted to this type first).

Format:

```
outputFile.write(temp)
```

Example:

```
# Assume that temp contains a string of characters.
outputFile.write (temp)
```

Reading From Files: Commonly Used Algorithm

- Pseudo-code:

```
Read a line from a file as a string
While (string is not empty)
    process the line
    Read another line from the file
```

Example Program: data_processing.py

```
inputFile = open ("employees.txt", "r")

print ("Reading from file input.txt")
for line in inputFile:
    name,job,income = line.split(',')
    last,first = name.split()
    income = int(income)
    income = income + (income * BONUS)
    print("Name: %s, %s\t\t\tJob: %s\t\tIncome $%.2f"
            %(first,last,job,income))

print ("Completed reading of file input.txt")
inputFile.close()
```

 # EMPLOYEES.TXT
Adama Lee,CAG,30000
Morris Heather,Heroine,0
Lee Bruce,JKD master,100000

Example Program: data_processing.py

```
inputFile = open ("employees.txt", "r")

print ("Reading from file input.txt")
for line in inputFile:
    name,job,income = line.split(',')
    last,first = name.split()
    income = int(income)
    income = income + (income * BONUS)
    print("Name: %s, %s\t\t\tJob: %s\t\tIncome $%.2f"
            %(first,last,job,income))

print ("Completed reading of file input.txt")
inputFile.close()
```

 # EMPLOYEES.TXT
Adama Lee,CAG,30000
Morris Heather,Heroine,0
Lee Bruce,JKD master,100000

```
Reading from file input.txt
Name: Lee, Adama                    Job: CAG            Income $63000.00
Name: Heather, Morris               Job: Heroine        Income $0.00
Name: Bruce, Lee                    Job: JKD master     Income $210000.00
Completed reading of file input.txt
```