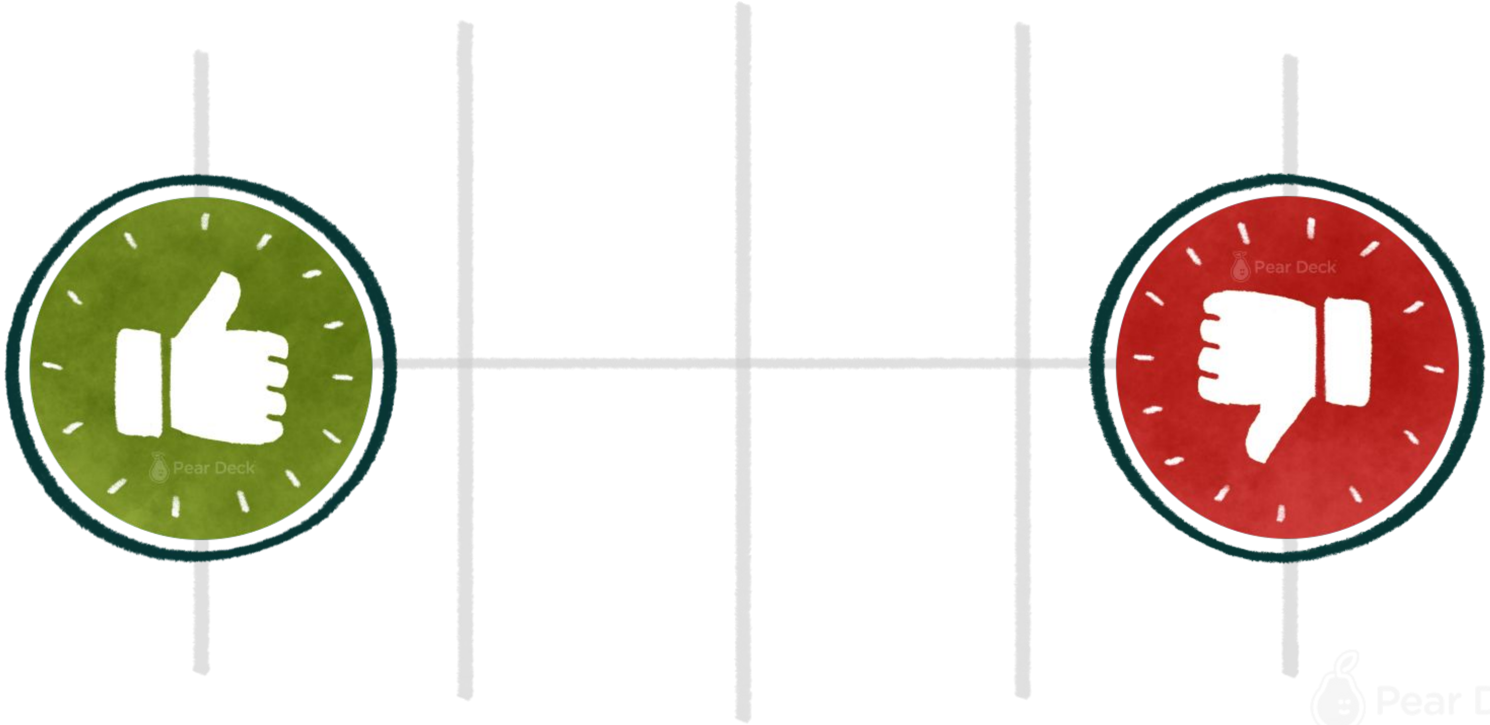
Writing Files

Table of Contents

1

Writing to File with `.write()` Method

How was the pre-class content of the “*writing to files*”?



Students, drag the icon!

Explain the difference between modes `"a"` and `"w"`

Students, write your response!

Writing to File with `.write()` Method (review)

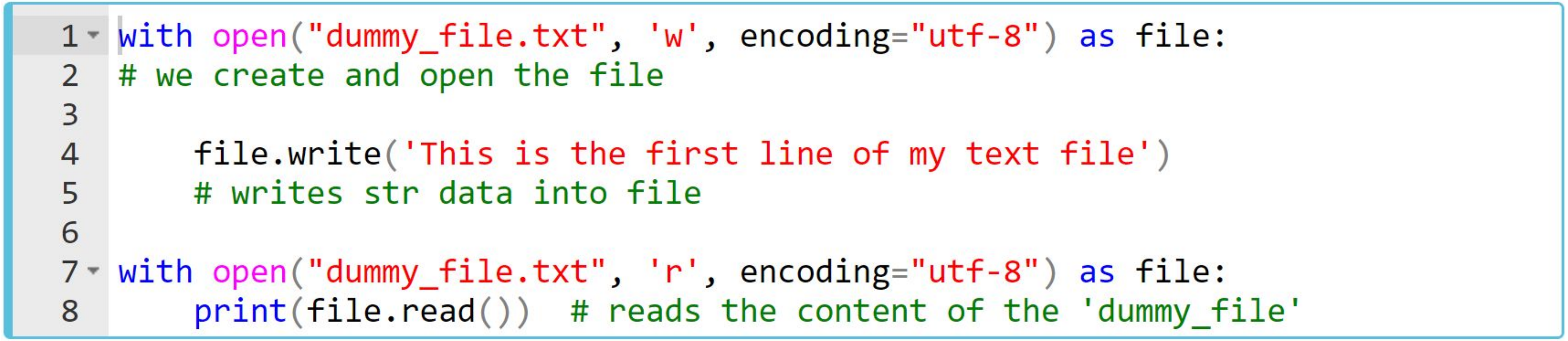
- As we mentioned earlier, we can **overwrite** a text to a file using `'w'` mode, which means that every time we use `'w'`, the content of the file is deleted and new content is written. If there isn't any file then it will be created automatically.

💡**Tips:**

- Note that data in the other type that we intend to write to the file must be converted to **string type** before the writing process.

Writing to File with `.write()` Method (review)

- Let's create and write string data to a file. We're going to use `.write()` method for writing :



```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
# we create and open the file

    file.write('This is the first line of my text file')
    # writes str data into file

with open("dummy_file.txt", 'r', encoding="utf-8") as file:
    print(file.read())  # reads the content of the 'dummy_file'
```

What is the output? Try to figure out in your mind...

Writing to File with `.write()` Method (review)

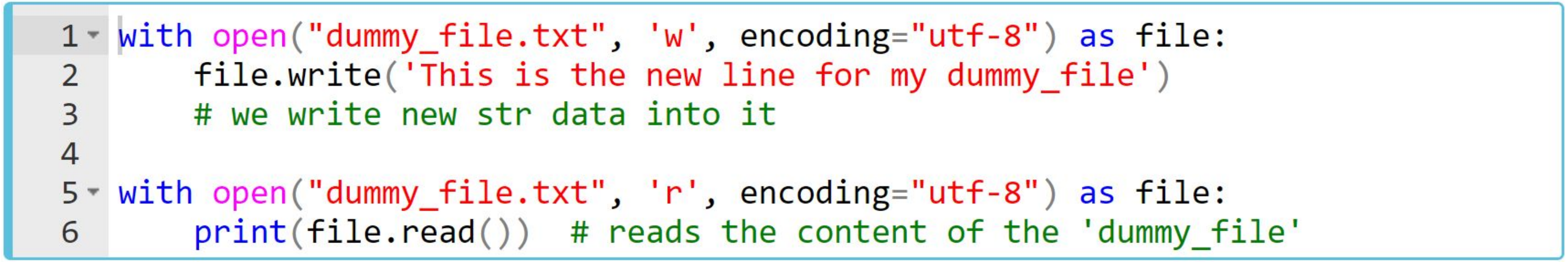
- It gives an output what we entered using `.write()` method.

```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
# we create and open the file

    file.write('This is the first line of my text file')
    # writes str data into file

with open("dummy_file.txt", 'r', encoding="utf-8") as file:
    print(file.read())  # reads the content of the 'dummy_file'
```

```
This is the first line of my text file

```

Writing to File with `.write()` Method (review)

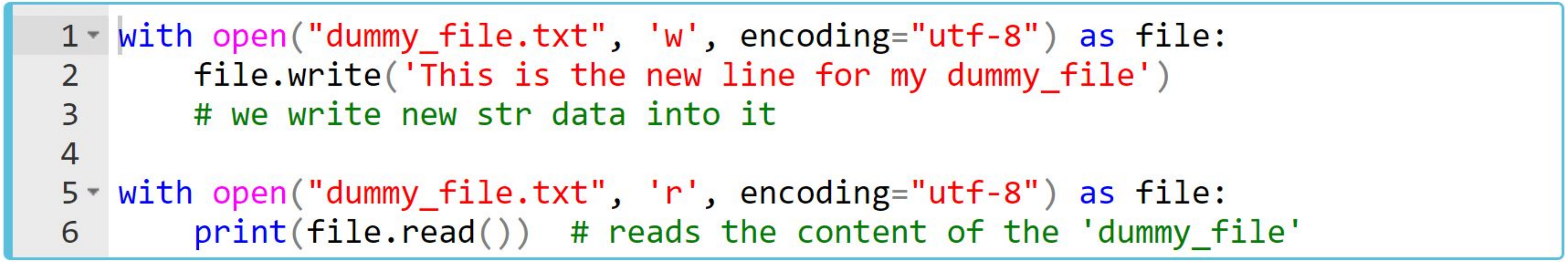
- Now let's repeat the process and see what happens. This time the file (**dummy_file**) exists :



```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
    file.write('This is the new line for my dummy_file')
    # we write new str data into it

with open("dummy_file.txt", 'r', encoding="utf-8") as file:
    print(file.read())  # reads the content of the 'dummy_file'
```

Writing to File with `.write()` Method (review)

- Now let's repeat the process and see what happens. This time the file (**dummy_file**) exists :



```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
    file.write('This is the new line for my dummy_file')
    # we write new str data into it

with open("dummy_file.txt", 'r', encoding="utf-8") as file:
    print(file.read())  # reads the content of the 'dummy_file'
```

What is the output? Try to figure out in your mind…

Writing to File with `.write()` Method (review)

- Now let's repeat the process and see what happens. This time the file (**dummy_file**) exists :

```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
    file.write('This is the new line for my dummy_file')
    # we write new str data into it

with open("dummy_file.txt", 'r', encoding="utf-8") as file:
    print(file.read())  # reads the content of the 'dummy_file'
```

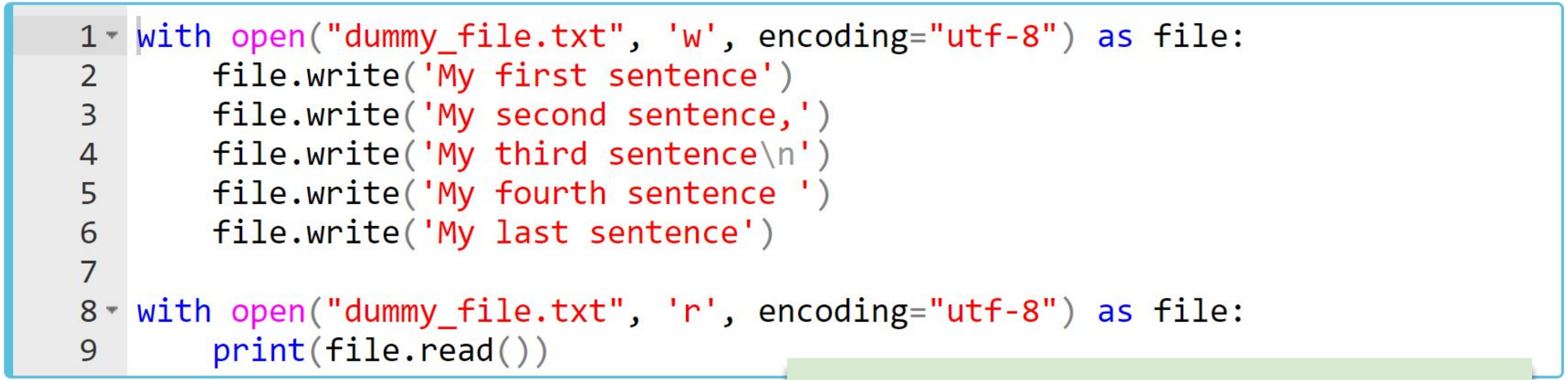
```
This is the new line for my dummy_file

```

Writing to File with `.write()` Method (review)

- When you write strings to a file using the `.write()` method, the string data is written exactly as it is.
- Whenever you use `.write()` method, each string joined together into one. Therefore you have to put newline characters (`\n`), separators or spaces, etc. manually if you want.
- Consider the following example :

Writing to File with `.write()` Method (review)

- Let's **write** 5 sentences into the **`dummy_file.txt`** file we created before and then **read** the content of that file.



```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
    file.write('My first sentence')
    file.write('My second sentence,')
    file.write('My third sentence\n')
    file.write('My fourth sentence ')
    file.write('My last sentence')

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

What is the output? Try to figure out in your mind...

Students, write your response!

Writing to File with `.write()` Method (review)

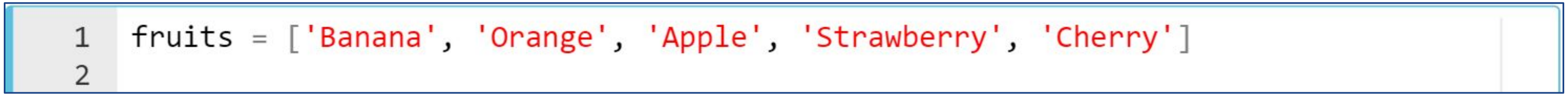
- The output is as follows :

```python
with open("dummy_file.txt", 'w', encoding="utf-8") as file:
    file.write('My first sentence')
    file.write('My second sentence,')
    file.write('My third sentence\n')
    file.write('My fourth sentence ')
    file.write('My last sentence')

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

```
My first sentenceMy second sentence,My third sentence
My fourth sentence My last sentence

```

Writing to File with `.write()` Method (review)

- **Task :**
  - Now, think of that we have a **`list`** of fruit names.
  - Let's write them to a file named **`fruits.txt`** each on separate lines one after another.
  - Read and display the entire content,
  - Read and display the content in a **`list`** form.

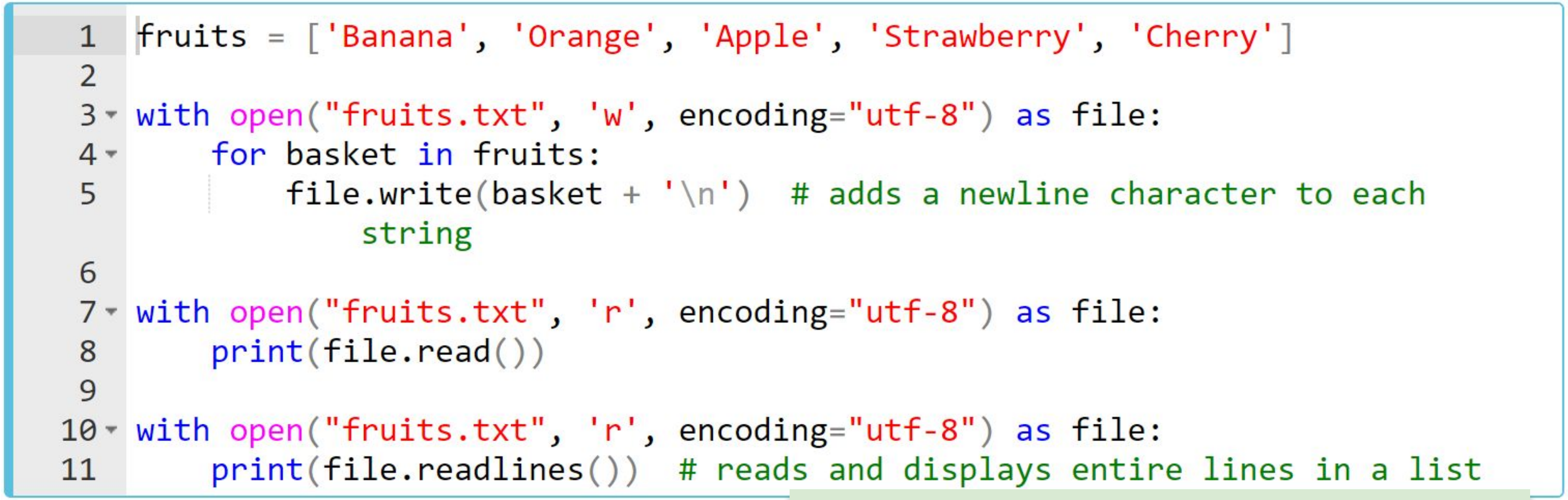


```python
fruits = ['Banana', 'Orange', 'Apple', 'Strawberry', 'Cherry']

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Writing to File with `.write()` Method (review)

- The code snippet can be as follows :



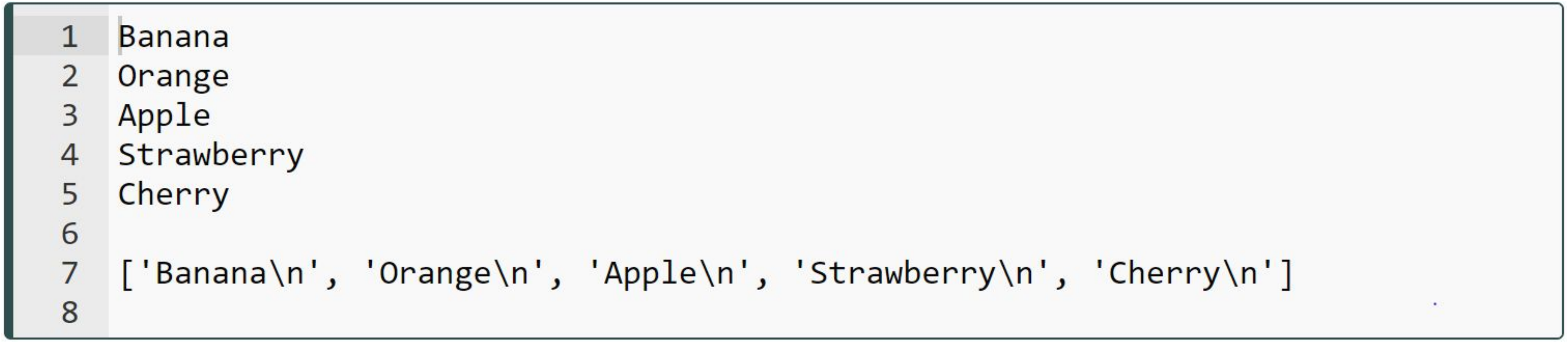
```python
fruits = ['Banana', 'Orange', 'Apple', 'Strawberry', 'Cherry']

with open("fruits.txt", 'w', encoding="utf-8") as file:
    for basket in fruits:
        file.write(basket + '\n')  # adds a newline character to each
            string

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.read())

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.readlines())  # reads and displays entire lines in a list
```

What is the output? Try to figure out in your mind...

CLARUSWAY©
WAY TO REINVENT YOURSELF

Writing to File with `.write()` Method (review)

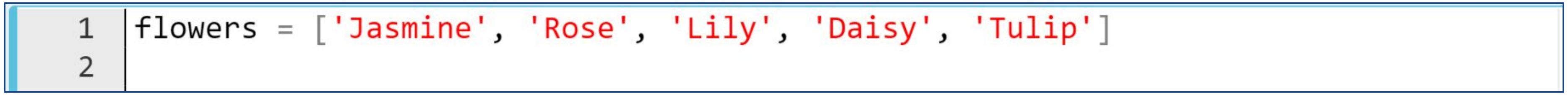
- Here the output is :



```
Banana
Orange
Apple
Strawberry
Cherry

['Banana\n', 'Orange\n', 'Apple\n', 'Strawberry\n', 'Cherry\n']

```

Writing to File with `.write()` Method

- **Task :**
  - Now, think of that we have a **`list`** of flower names.
  - Let's write them to a file named **`flowers.txt`** each on separate lines one after another and separated by an empty line.

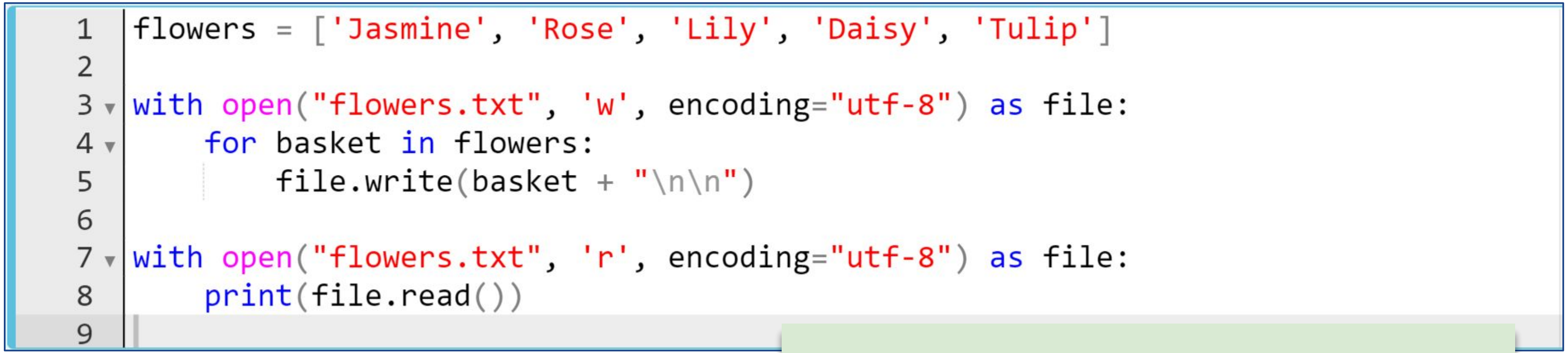


```python
flowers = ['Jasmine', 'Rose', 'Lily', 'Daisy', 'Tulip']

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Writing to File with `.write()` Method

- The code snippet can be as follows :

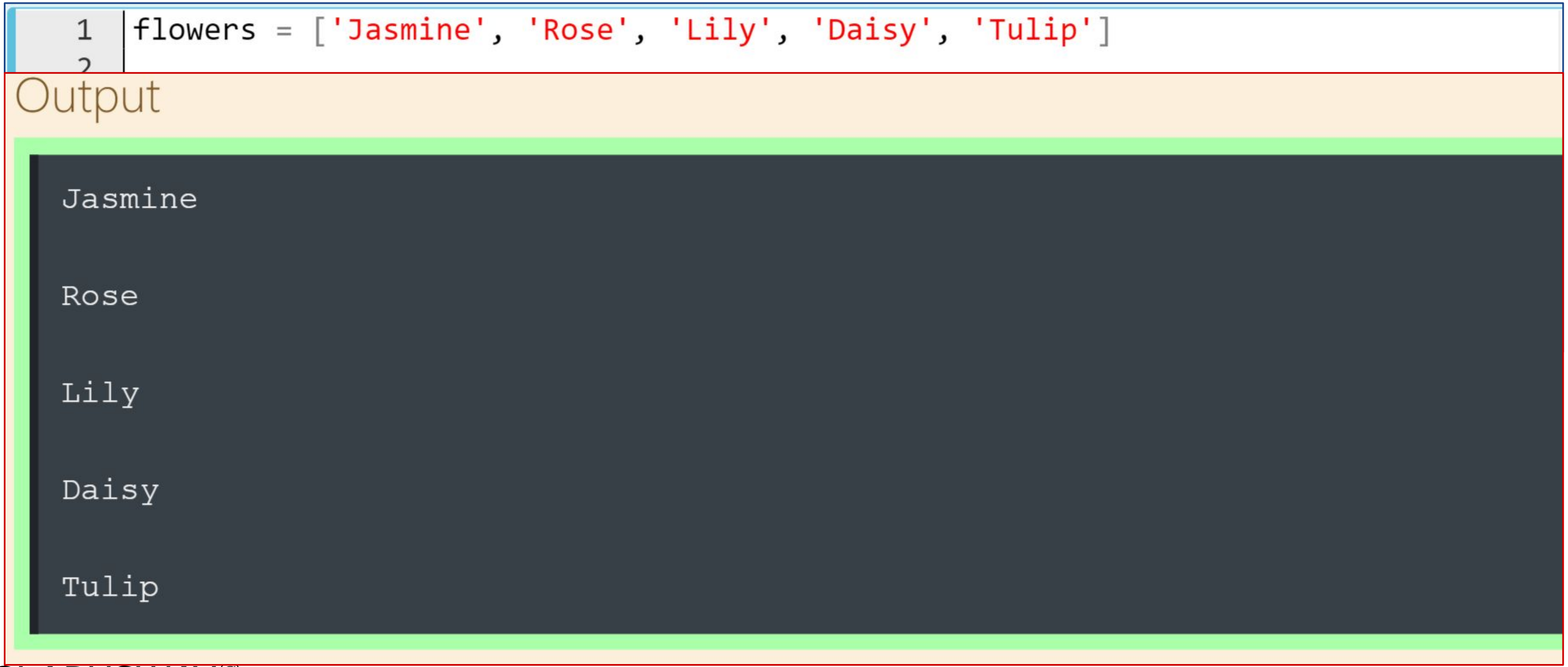


```python
flowers = ['Jasmine', 'Rose', 'Lily', 'Daisy', 'Tulip']

with open("flowers.txt", 'w', encoding="utf-8") as file:
    for basket in flowers:
        file.write(basket + "\n\n")

with open("flowers.txt", 'r', encoding="utf-8") as file:
    print(file.read())

```

What is the output? Try to figure out in your mind...

Writing to File with `.write()` Method

- The output is as follows :



```
flowers = ['Jasmine', 'Rose', 'Lily', 'Daisy', 'Tulip']
```

Output

```
Jasmine

Rose

Lily

Daisy

Tulip
```

2

Writing to File with `.writelines()` Method

Make connections

How are these two methods connected?

Type your answers.

`.readlines()`

`.writelines()`

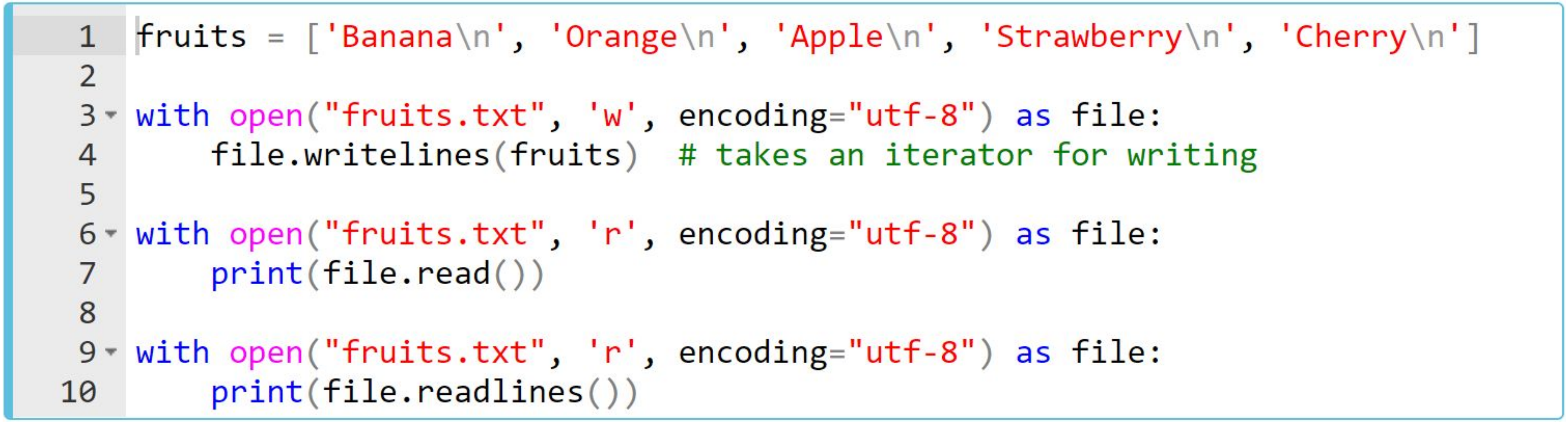
Students, write your response!

Writing to File with `.writelines()` Method (review)

- There is another method for writing data to the files. It is `.writelines()` method. Unlike `.write()` method, `.writelines()` takes the iterable sequence of strings and writes them to the file.
- The difference between these two methods is just similar to the logic of difference between `.read()` and `.readlines()`.

Writing to File with `.writelines()` Method (review)

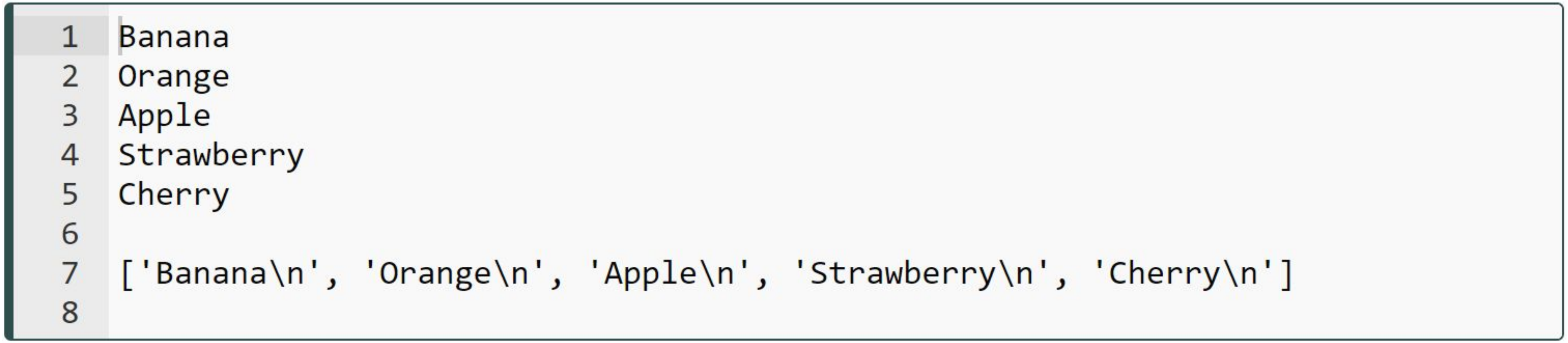
- Let's use the same **`list`** of the fruits again but this time in a little bit different way. We should choose the line separators ourselves without using **`for`** loop. Let's see how we do it :



```python
fruits = ['Banana\n', 'Orange\n', 'Apple\n', 'Strawberry\n', 'Cherry\n']

with open("fruits.txt", 'w', encoding="utf-8") as file:
    file.writelines(fruits)  # takes an iterator for writing

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.read())

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.readlines())
```

Writing to File with `.writelines()` Method (review)

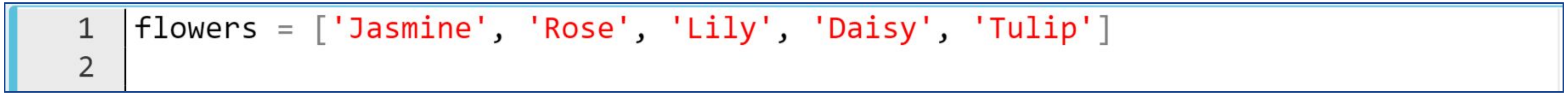
- The output looks like :



```
Banana
Orange
Apple
Strawberry
Cherry

['Banana\n', 'Orange\n', 'Apple\n', 'Strawberry\n', 'Cherry\n']

```

Writing to File with `.writelines()` Method

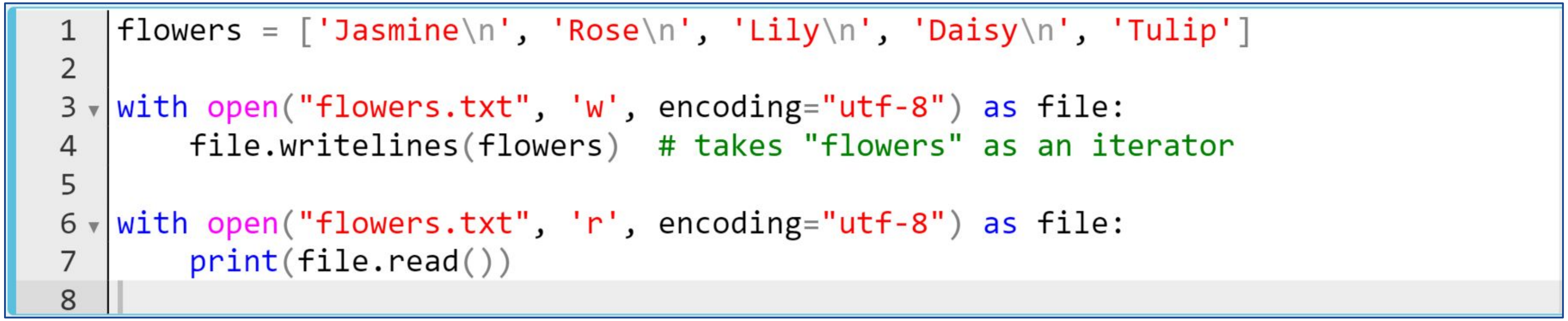
- **Task :**
  - Use the same **`list`** of flower names,
  - Modify the **`list`** for use,
  - Overwrite them to the same **`flowers.txt`** file each on separate lines one after another.



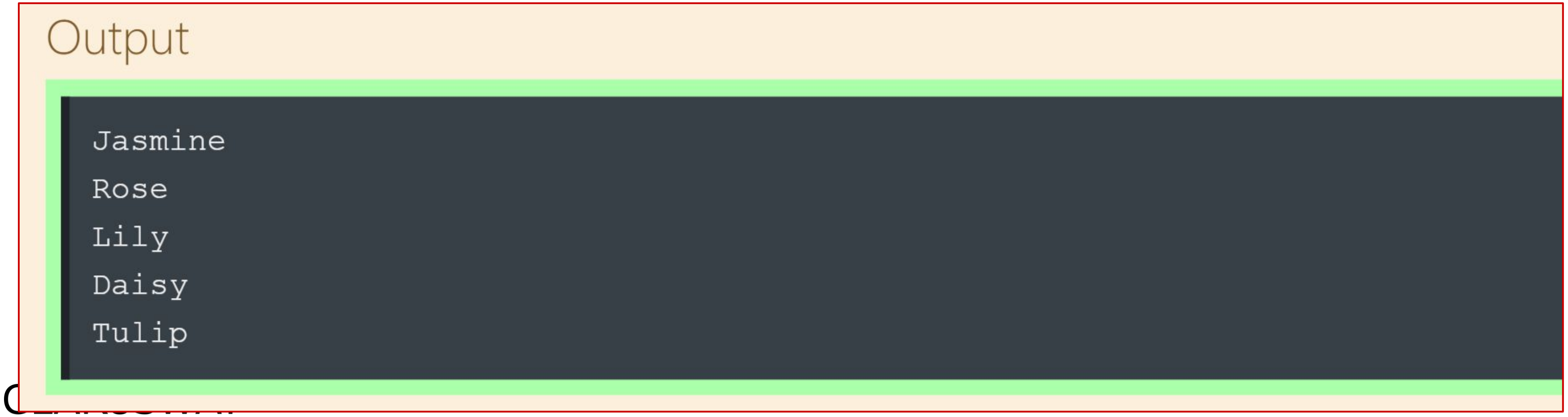
```python
flowers = ['Jasmine', 'Rose', 'Lily', 'Daisy', 'Tulip']

```

Writing to File with `.writelines()` Method

- The code snippet and the output can be as follows :



```python
flowers = ['Jasmine\n', 'Rose\n', 'Lily\n', 'Daisy\n', 'Tulip']

with open("flowers.txt", 'w', encoding="utf-8") as file:
    file.writelines(flowers)  # takes "flowers" as an iterator

with open("flowers.txt", 'r', encoding="utf-8") as file:
    print(file.read())

```



Output

```
Jasmine
Rose
Lily
Daisy
Tulip
```

WAY TO REINVENT YOURSELF

3

Appending to File with `' a '`

Appending to File with `'a'` (review)

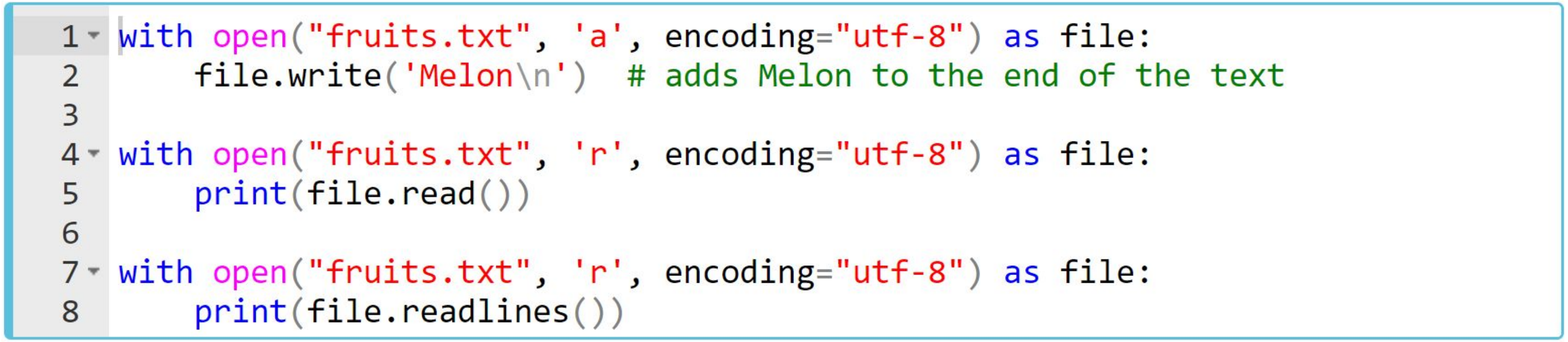
- Unlike the previous writing mode (`'w'`), in most cases when we want to add new content to a file, deleting the existing content is undesirable.
- Therefore, there is a need for another mode that both keeps the existing content of the file and saves the new content to the continuation of the file. **In Python, we meet this need with** `'a'` mode which stands for **append**.

Appending to File with `'a'` (review)

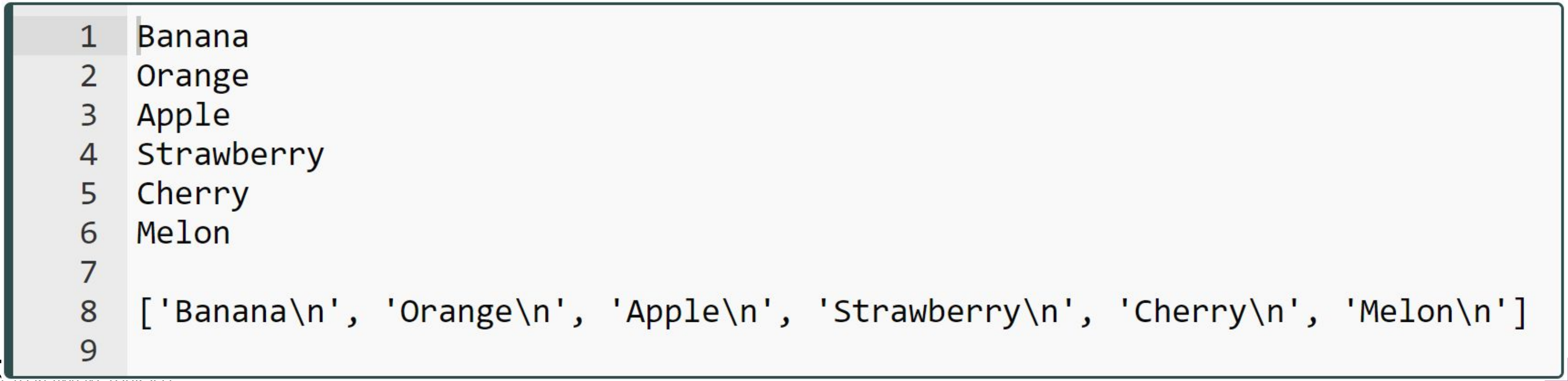
- **Task :**
  - Let's add `'melon'` to our existing **`fruits.txt`** file as the last line,
  - Read and display the entire file content,
  - Read and display the entire file content line by line in a **`list`** form.

Appending to File with `'a'` (review)

- The code snippet and the output are as follows:



```python
with open("fruits.txt", 'a', encoding="utf-8") as file:
    file.write('Melon\n')  # adds Melon to the end of the text

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.read())

with open("fruits.txt", 'r', encoding="utf-8") as file:
    print(file.readlines())
```



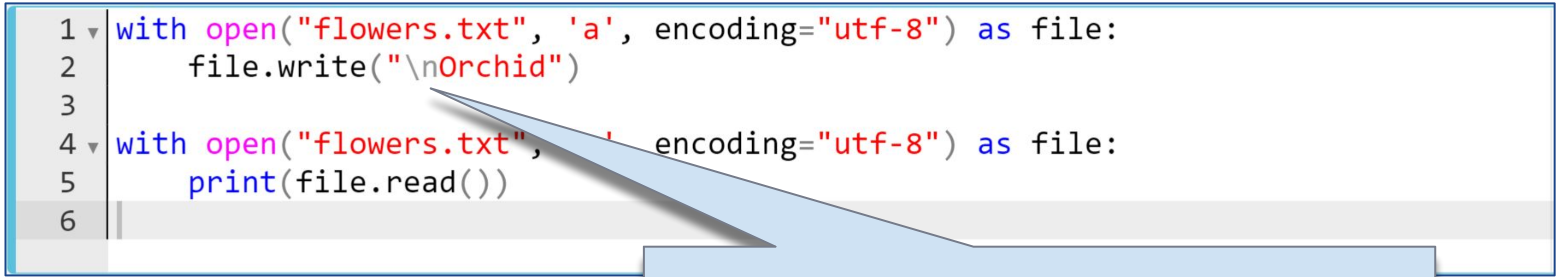
```
Banana
Orange
Apple
Strawberry
Cherry
Melon

['Banana\n', 'Orange\n', 'Apple\n', 'Strawberry\n', 'Cherry\n', 'Melon\n']

```

Appending to File with `'a'`

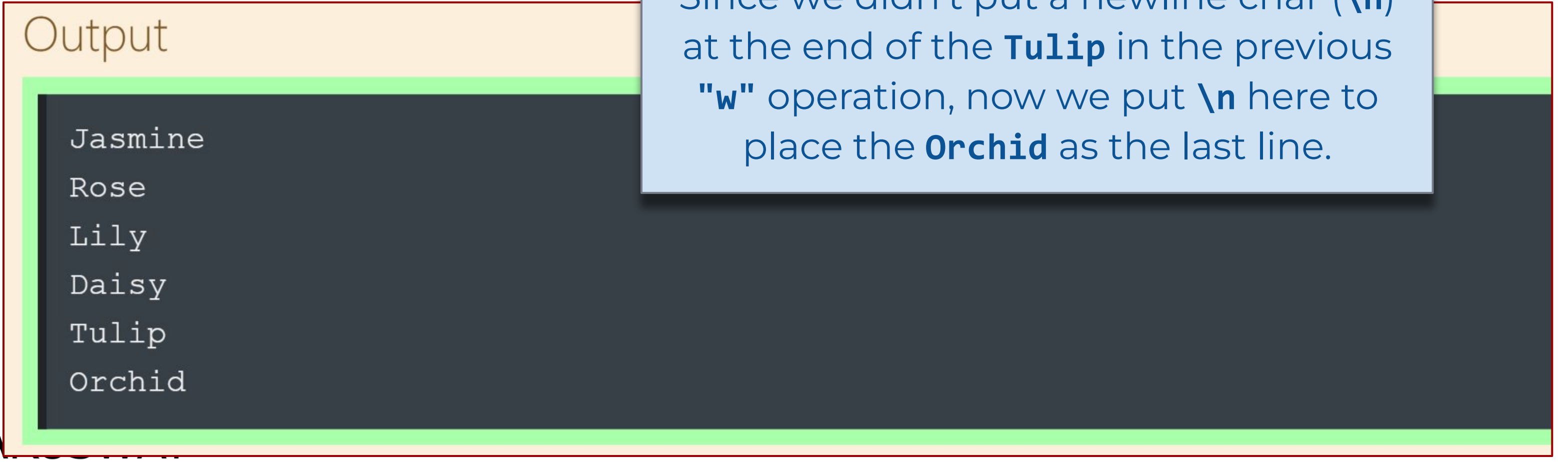
- **Task :**
  - Let's add `'orchid'` to our existing `flowers.txt` file as the last line,
  - Read and display the entire file content.

Appending to File with `'a'`

- The code snippet and the output are as follows:



```python
with open("flowers.txt", 'a', encoding="utf-8") as file:
    file.write("\nOrchid")

with open("flowers.txt", 'r', encoding="utf-8") as file:
    print(file.read())

```



Output

```
Jasmine
Rose
Lily
Daisy
Tulip
Orchid
```

Since we didn't put a newline char (**`\n`**) at the end of the **`Tulip`** in the previous **`"w"`** operation, now we put **`\n`** here to place the **`Orchid`** as the last line.

Did you find this lesson interesting and challenging?

Too hard

Just right

Too easy

THANKS!

Any questions?