open and close

```
myfile = open("filename.txt")
"r"read mode
"w"write mode
"a"append mode
"r+"write/read mode
"wb"write binary mode
....
-----------------
myfile.close()
```

The argument of the open function is the path to the file.

working with files

It is good practice to avoid wasting resources by making sure that files are always closed after they have been used.

```
try:
    f = open("filename.txt")
  print(f.read())
finally:
    f.close()
```
....
An alternative way of doing this is using with statements. This creates a temporary variable (often called f), which is only accessible in the indented block of the with statement.
The file is automatically closed at the end of the with statement, even if exceptions occur within it.

```
with open("filename.txt") as f:
    print(f.read())
```

reading

```
file = open("filename.txt", "r")
cont = file.read()
print(cont)
file.close()
```
....
```
file = open("filename.txt", "r")
print(file.read(16))
```
This determines the number of bytes that should be read.
....
To retrieve each line in a file, you can use the readlines method
```
file = open("filename.txt", "r")
print(file.readlines())
file.close()
>>>
['Line 1 text \n', 'Line 2 text \n', 'Line 3 text']
>>>
```
....
You can also use a for loop to iterate through the lines in the file:
```
file = open("filename.txt", "r")
for line in file:
    print(line)
file.close()
>>>
Line 1 text
Line 2 text
Line 3 text
>>>
```

writing

```
file = open("newfile.txt", "w")
file.write("This has been written to a file")
file.close()
```
....
When a file is opened in write mode, the file's existing content is deleted.
....
The write method returns the number of bytes written to a file, if successful.
```
msg = "Hello world!"
file = open("newfile.txt", "w")
amount_written = file.write(msg)
print(amount_written)
file.close()
>>>
12
>>>
```