בס"ד

**A little bit about handles**:

Open IPython, and run the line:

opened\_file = open(r’c:\file.txt’, ‘w’)

If you open your C drive, you should see a file called file.txt.

Open the file with notepad, type something, and try to save. (Keep IPython opened all this time). It should not work...

This is because when you opened the file with 'w' access, it requests exclusive access to writing the file - no other program can open the file for writing, even notepad. (When you tried to save the file, notepad.exe requested write access to the file, in order to save your changes, and was denied.

This behavior is useful when you intend to write information to a file, and obviously you don’t want another process to write at the same time, causing the file to have corrupted information. However, imagine what happens if even after finishing the desired usage the file stays open. Any other program will not be able to save the file. And what if your program runs endlessly?

The operating system grants a type of pointer (called a handle) to the required resource, this is what happens with the open method. This stays open, until the program marks the resource as not required, by closing this pointer. Afterwards, the handle is invalid, and cannot be used to access the resource – accessing the resource requires the opening of a new handle. For example, try running the following code:

file\_path = r'c:\temp\some\_file.txt'

inner\_file = open(file\_path, 'r') # Request access to the resource

inner\_file.close() # Free the resource

content = inner\_file.read() # Try to read the closed file

You would have to open the file again:

file\_path = r'c:\temp\some\_file.txt'

inner\_file = open(file\_path, 'r') # Request access to the resource

inner\_file.close() # Free the resource

inner\_file.open(file\_path, ‘r’) # Reopen handle

content = inner\_file.read() # Read the file

inner\_file.close() # Close the file!!!

Many resources that the operating system provides are used this way – request access to the resource, and need to be closed afterwards.

Returning to the original IPython\Notepad example, try closing the file you opened, and try to save. Works?

Python offers a more comfortable way to use resources that require open\close methods - the ‘with’ statement, which enables the following usage:

with open(r'c:\file.txt', 'r') as handle:

print(handle.read())

This opens a context in which the file is 'active', and known as handle, and once the context is exited the close method is executed behind the scenes. (You will see how this is achieved in the rest of the exercise.)

\* The name ‘handle’ used is my design choice, it is not important to the statement. The same code would run if instead of the name ‘handle’ I used ‘my\_file’, or ‘bla’, or any other name of my choosing. Handle is just an accurate description of what this code snippet is trying to demonstrate.

Bonus:

Go back to the first stage - try opening a file for exclusive access, opening the same file in notepad, writing content (check that you can't save it), exactly like in the first section.

Now, instead of closing the file, close the IPython prompt. Can you save now?

This is because Python itself takes care of freeing resources when it is closed (part of its garbage collecting). However relying on this is bad practice!