CHAPTER 10 – FILE ACCESS AND ERROR HANDLING

10.1

To access data stored in an external file:

1. Open the file
2. Use the file
3. Close the file

Types of file access:

* Sequential access 🡪 software accesses the file and reads the data beginning to end without the possibility of jumping from one point to another
* Direct access 🡪 possibility to go directly to the desired data without having to read all data preceding it

To open a file use the open function:

filevariablename = open (file, mode)

File specifies the complete path of the file (with a slash \ and not a backslash /) in the disk, mode is an optional argument specifying how to open the file, if omitted it is read-only

Main values for the mode argument:

Simple modes:

‘r’ 🡪 reading only mode

‘w’ 🡪 writing only mode (cancels all previous data in the file)

‘a’ 🡪 append mode (appends automatically at the end of the file)

Double modes:

‘r+’ 🡪 read and write mode (python positions itself at the beginning of the file)

‘w+’ 🡪 read and write mode (automatically deletes all data)

‘a+’ 🡪 read and append mode (python positions itself at the end of the file)

Methods for files:

Filevariablename.method()

To close a file:

Filevariablename.close()

Attributes for files allow us to retrieve basic information about the file:

Filevariablename.name 🡪 complete path and name of the file

Filevariablename.mode 🡪 mode with which the file was opened

Filevariable.close 🡪 True if the file is closed, False otherwise

10.2

When creating an object for a file, the current position is plotted and stored by a pointer.

Moving the pointer methods:

.seek(offset, [,whence]) 🡪 moves the pointer to the offset byte starting from the whence position (0=beginning,1=current position, 2=end) – only works on binary files

.tell() 🡪 returnes the current position in terms of bytes from the beginning

Reading methods:

.read(size) 🡪 reads the entire document from the position of the pointer, if size is specified it reads the number of bytes indicated by size

.readline(size) 🡪 reads the entire line of the position of the pointer, if size is specified it reads the number of bytes indicated by size

.readlines(size) 🡪 reads the entire document from the position of the pointer returming a list with each line as element, if size is specified it reads the number of bytes indicated by size

10.3

Writing methods:

.writable() 🡪 returns True if it is possible to wrtir data to the file, False otherwise

.writew¡(string) 🡪 writes to the file the contents of the string and returns the number of characters written

.writelines(lines) 🡪 writes to the file the sequence of lines specified in the lines argument

10.4

An error which occurs during the execution of a program is called an exception.

Three types of errors:

1. Syntax errors 🡪 error in how the program is written or its structure
2. Runtime errors 🡪 error in the code even if the syntax is correct
3. Semantic errors 🡪 results are incorrect even though the program is executed correctly

10.5

The try except statement allows to deal with errors, its syntax is:

try:

statement that may cause the error

except:

statement on what to do if any error occurs

Its more specific variation is:

try:

statement that may cause the error

except exceptionname1:

statement on what to do if exceptionname1 occurs

except exceptionname2:

statement on what to do if exceptionname2 occurs

…

Main exception names:

* AttributeError 🡪 method or attribute assigned to incorrect object or data type
* IndexError 🡪 index is greater than the length minus 1
* KeyError 🡪 key of a dictionary not found
* ModuleNotFoundError 🡪 module not found
* NameError 🡪 variable not found
* SyntaxError
* TypeError 🡪 operation or function assigned to incorrect object or data type
* ValueError 🡪 function or method receives an argument of the correct data type but of wrong value
* ZeroDivisionError 🡪 dividend is 0

10.6

Debugging consists in removing errors to optimize the program