WORKING WITH FILES

WORKING WITH FILES

- What is a file?
  - A file is a collection of data that is stored on secondary storage like a disk or a thumb drive
  - accessing a file means
    - Establishing a connection between the file and the program
    - Moving data between the file and the program

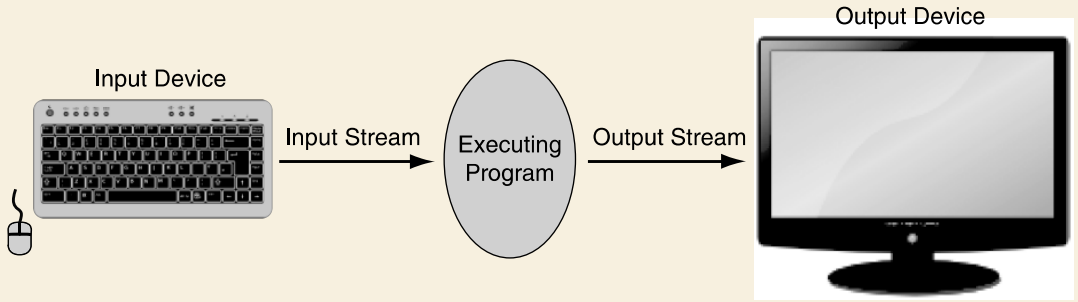
WORKING WITH FILES

Files come in two general types:

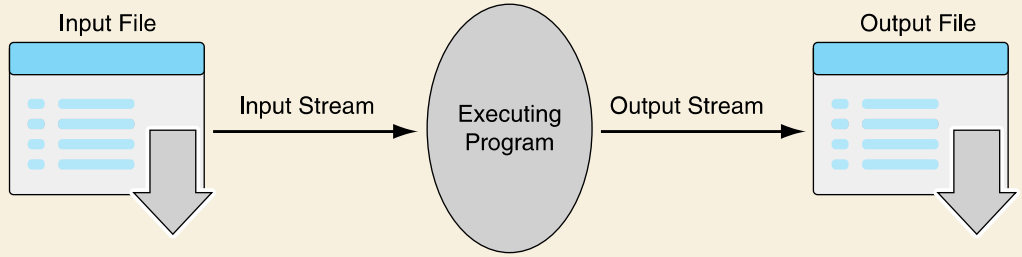
- *text files (.txt)*
  - Files where control characters such as `"/n"` are translated. These files are generally human readable
- *binary files*
  - All the information is taken directly without translation.
  - They are generally not readable and contain non-readable info

WORKING WITH FILES
FILE OBJECTS OR STREAM

- When opening a file, you create a file object(or file stream) that is a connection between the file information on disk and your program
- The file object(file stream) contains a buffer of the information from the file, and provides the information to the program



a) Standard input and output.



b) File input and output.

WORKING WITH FILES BUFFERING

- Reading data from a disk is a very slow process
  - That is why the computer will read a lot of data from a file in the hopes that, if you need the data in the future, it will be buffered in the file object.
  - A buffer is basically just a list of data
  - This means that the file object contains a copy of information from the file called a cache (pronounced "cash")

WORKING WITH FILES
FILE OBJECTS

```
1    file_content = open('names.txt', 'r')
```

- `file_content` is the file object. It contains the buffer of information. The open function creates the connection between the file on disk and the file object.
- The first parameter of the open function is the name of the file you wish to open
- the second parameter is the mode to open the file in (`"r"` means read)

WORKING WITH FILES

- The name of the file can come in one of two forms:
  - `"file.txt"` assumes the file name is file.txt and it is located in **the current program directory**
    - This is called a relative path, i.e the path is relative to the programs location on the file system
  - `"c:\bill\file.txt"` is the fully qualified file name and includes the directory information
    - This is called an absolute path, i.e the path from the root of your file system

WORKING WITH FILES DIFFERENT MODES

- Here is a list of the possible modes you can use when working with a file

Mode	How Opened	File Exists	File Does Not Exist
'r'	read-only	Opens that file	Error
'w'	write-only	Clears the file contents	Creates and opens a new file
'a'	write-only	File contents left intact and new data appended at file's end	Creates and opens a new file
'r+'	read and write	Reads and overwrites from the file's beginning	Error
'w+'	read and write	Clears the file contents	Creates and opens a new file
'a+'	read and write	File contents left intact and read and write at file's end	Creates and opens a new file

WORKING WITH FILES

- You must be careful with write modes when you are working with files
  - If you open a file in `'w'` mode. It sets an existing file's contents to be empty, destroying any existing data.
  - The `'a'` mode is nicer, it allows you to write to the end of an existing file without changing the existing contents
    - This is called appending data to the file

WORKING WITH FILES

- Text files use strings
  - This means that if you are interacting with a text file, all the content of the file *is a string!*
  - everything read from a file is a string!
  - if you write to a file, you can only write a string!

WORKING WITH FILES
READING FROM A FILE

- We can read the content from a file object using a for loop
  - This works as follows
    - Each **line** of the file is read as a single string
      - That means we can use string methods on each line

Note thate the file names.txt contains one name in each line

```
1   file_content = open('names.txt', 'r')
2
3   for name in file_content:
4       print(name)
5
6   file_content.close()
```

WORKING WITH FILES
READING FROM A FILE

- A common use of reading data from a file is to store the data in a list

We start of by creating an empty list

We iterate over each name in the txt file. Clean each name with the strip() method. Then we add the cleaned up name to the list

```
1  file_content = open('names.txt', 'r')
2
3  name_list = []
4
5  for name in file_content:
6      cleaned_name = name.strip()
7      name_list.append(cleaned_name)
8
9  file_content.close()
```

WORKING WITH FILES
READING FROM A FILE

- Do note that you might have to tell Python what kind of encoding to use
  - There might be icelandic letters in the file which we want to print appropriately
  - We can do this by adding the value utf-8 to the named parameter called encoding in the open function

```
1   file_content = open('names.txt', 'r', encoding="utf-8")
2
3   name_list = []
4
5   for name in file_content:
6       cleaned_name = name.strip()
7       name_list.append(cleaned_name)
8
9   file_content.close()
10
11  print(name_list)
```

WORKING WITH FILES CLOSING THE FILE

When the program is finished working with a file, we must `close` the file

- Closing a file will:
  - flush the buffer contents from the computer to the file
  - tear down the connection to the file
- `close()` is a method of every file object

Here we are closing the connection to the file because we don´t need it anymore

```
1  file_content = open('names.txt', 'r')
2
3  for name in file_content:
4      print(name)
5
6  file_content.close()
```

WORKING WITH FILES
WRITING TO A FILE

Once you have created a file object, opened for reading, you can use the print command to write the content to a file!

- you add `file=<file reference>` as a parameter to the print function

We start by opening a file. If the file does not exist it will be created(note the w node)

Here we are printing the content to the file (note the file parameter)

Remember closing the connection to the file!

Note that we use utf-8 encoding because one of the strings holds an Icelandic character

```
1  output_file = open('output.txt', 'w', encoding="utf-8")
2
3  print("hello wörld!", file=output_file)
4  print("hello world again", file=output_file)
5
6  output_file.close()
```

WORKING WITH FILES

- There is another way of working with files
  - That is by using the with statement
    - One of the main benefits of using the with statement is that it closes the connection to the file for you ☺

This is an example of the with statement

.read() is a method on all file objects which returns the entire file contents as a single string

We can reference the file object with this name

```
1  with open('names.txt', 'r', encoding="utf-8") as file_content:
2      lines = file_content.read()
3
4  print(lines)
```

WORKING WITH FILES

- Other file object methods are:
  - .readline()
  - .readlines()
  - .write()
  - .writelines()
  - And more ...