MODULE ONE ASSIGNMENT

(DOCUMENTATION- TANIYA PAL [373094])

FASTAN: A Python script for parsing FASTQC text files

FastQC is a tool developed by Babraham Bioinformatics for the purpose of quality control of the Fastq files generated at the end of high throughput sequencing of DNA or RNA samples. The tool gives back a HTML file giving information about the quality of the Fastq file generated. The relevant HTML file has been divided into several modules, each module giving information about a particular criteria of quality control of sequences. It also generates figures for each module, with filter information about whether the module has passed, failed, or passed with a warning sign depending on the quality of the information in the fastq file. Along with this information, the tool gives back a FASTQC text file which has been analyzed in FASTAN.

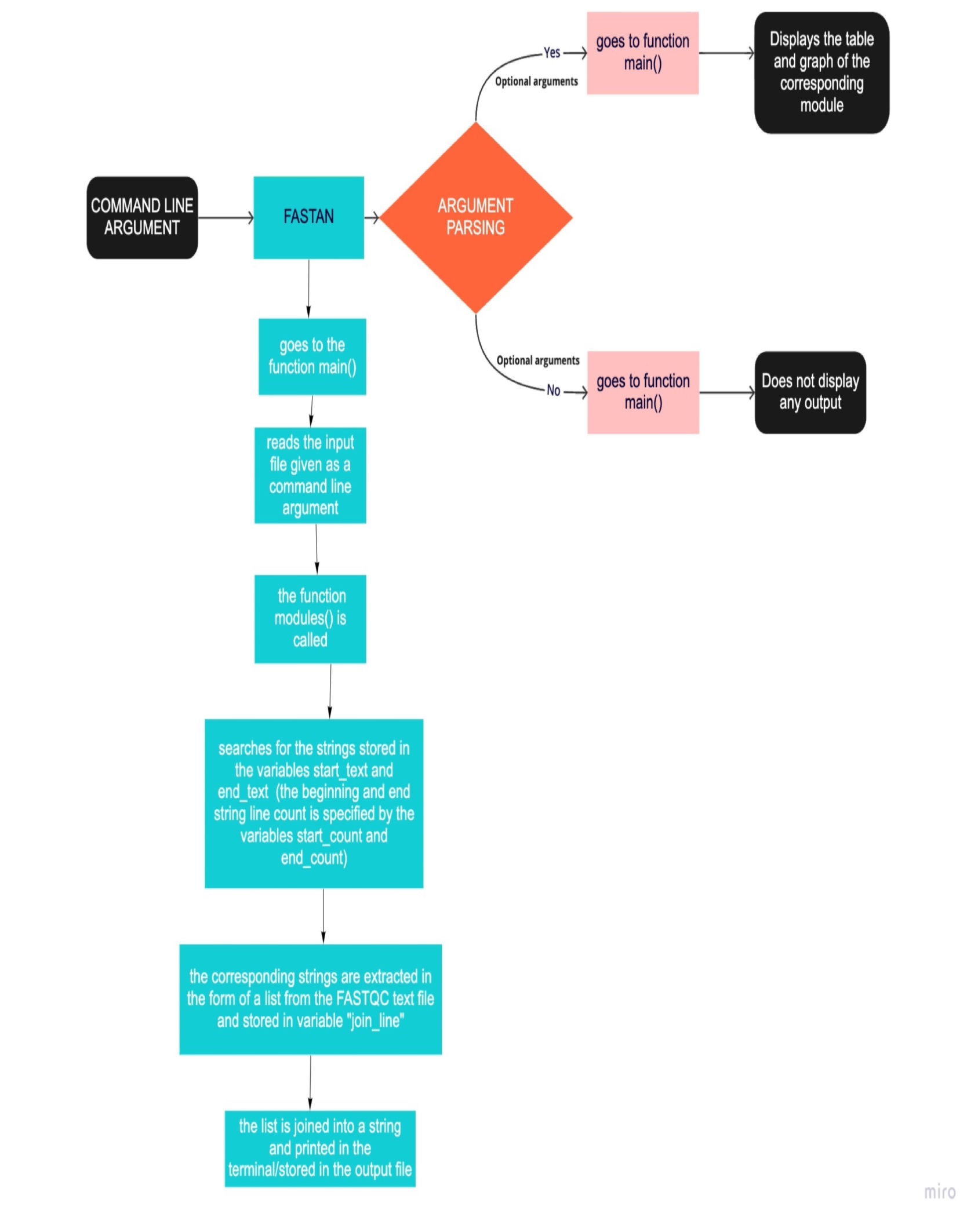
The various modules included in the FASTQC text files are:

1. Basic Statistics
2. Per Base Sequence quality
3. Per Sequence Quality Scores
4. Per Base Sequence Quality
5. Per Sequence GC Content
6. Per Base N Content
7. Sequence Length Distribution
8. Duplicate Sequence
9. Overrepresented Sequence
10. Adapter Content
11. Kmer Content

FASTAN is a python script used to parse the different modules given in the FASTQC text file and generate graph from the information given under each module. In the script, a function called “modules” is defined in the beginning. This function starts by storing each line of the FASTQC text file is stored in the variable “lines”. In this function, the string to search a respective module is taken as a default argument. By using the enumerate() function, each line of the module whose starting string(“start\_text”) has been given as an input under the function main(), is read and the count of the lines which contains the module name is stored in a variable called “start\_count”. As the string, “>>END\_MODULE” signifies the end of each module, the Python script searches for the string at the end of the module it is in currently, to mark the end of the module. The string is stored in the variable “end\_text” and the count of the line which contains the string is stored in a variable “end\_count.” The “start\_count” and “end-count” variables are initialized with “None” so that it does not count the respective strings they are searching for, twice. The variable “join\_lines” then extracts the lines with index starting from start count and ending in “end-count+1” as python indexes end with one value less than the given value. “text” joins the list of strings stored in “join\_lines” and gives back an output of string type containing the information that is present in the FASTQC text file under the name of the module given as an input under the function main().

In the next part of the script, the function “main” is called. The various arguments of the command line are parsed into Python data types using the ArgumentParser object. The input file name, output file name is stored into positional arguments. There are separate optional arguments for each module name included in the FASTQC text file. The argument “all’ displays all the modules whose information has been included in this script. Libraries such as “pandas”, “matplotlib” and “seaborn” are imported to visualize the graphs generated from each argument.

The tables for each module except Basic Statistics and Overrepresented Sequences and the corresponding graph have been generated in the Python script. For Basic Statistics and Overrepresented Sequences, only the respective tables under those modules have been included in the output. Graphs for each module has been generated with the help of seaborn and visualized with the help of matplotlib.pyplot. Though calling an argument for a particular module gives back only the text information under it, the graphs are stored in ‘.png’ format in the user’s working directory and can be visualized without the help of internet too. The graphs are stored with the module name as the file name.



**Concept Diagram**