ABDELGHAFOR’S VIRTUAL INTERNSHIP

# PYTHON PROGRAM

SESSION (6)

PREPARED BY : MARK KOSTANTINE

# AGENDA OVERVIEW

03

FILE HANDLING

11

TASKS

14

QUESTIONS

# FILE HANDLING

The key function for working with files in Python is the **open()** function.

The open() function takes **two** parameters; **filename, and mode.**

There are four different methods (modes) for opening a file:

- **"r" - Read** - Default value. Opens a file for reading, error if the file does not exist
- **"a" - Append** - Opens a file for appending, creates the file if it does not exist
- **"w" - Write** - Opens a file for writing, creates the file if it does not exist
- **"x" - Create** - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as **binary** or **text** mode

- **"t" - Text** - Default value. Text mode
- **"b" - Binary** - Binary mode (e.g. images)

## SYNTAX

To open a file for reading it is enough to specify the name of the file

```
f = open("demofile.txt")
```

```
f = open("demofile.txt", "rt")
```

**Note:** Make sure the file exists, or else you will get an error.

## READ ONLY PARTS OF THE FILE

By default the **read()** method returns the whole text, but you can also specify how many characters you want to return

```
f = open("demofile.txt", "r")
print(f.read(5))
```

## READ LINES

You can return one line by using the **readline()** method

```
f = open("demofile.txt", "r")
print(f.readline())
```

By calling **readline()** two times, you can read the two first lines

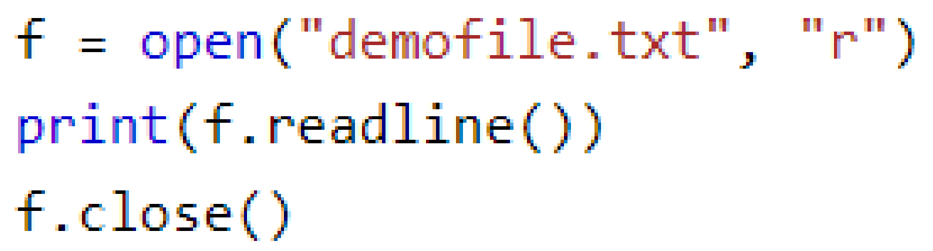
```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

By looping through the lines of the file, you can read the whole file, line by line

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

## CLOSE FILES

It is a good practice to always close the file when you are done with it



```python
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

**Note:** You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

# WRITE TO AN EXISTING FILE

To write to an existing file, you must add a parameter to the open() function

- "a" - **Append** - will append to the end of the file
- "w" - **Write** - will overwrite any existing content

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Open the file "demofile2.txt" and append content to the file

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

Open the file "demofile3.txt" and overwrite the content

## CREATE A NEW FILE

To create a new file in Python, use the open() method, with one of the following parameters:

- **"x" - Create** - will create a file, returns an error if the file exist
- **"a" - Append** - will create a file if the specified file does not exist
- **"w" - Write** - will create a file if the specified file does not exist

```
f = open("myfile.txt", "x")
```

Create a file called "myfile.txt"

```
f = open("myfile.txt", "w")
```

Create a new file if it does not exist

## DELETE A FILE

To delete a file, you must import the OS module, and run its os.remove() function

```
import os
os.remove("demofile.txt")
```

## CHECK IF FILE EXIST

To avoid getting an error, you might want to check if the file exists before you try to delete it

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

## DELETE FOLDER

To delete an entire folder, use the **os.rmdir()** method

```
import os
os.rmdir("myfolder")
```

**Note:** You can only remove empty folders.

## TASKS

**Beginner Level :**

- **Reading a File and Printing Content**
  - Write a Python function that opens a .txt file and prints each line to the console. The file should contain at least 5 lines. Use a try-except block to handle any potential errors (like file not found).
- **Writing to a File**
  - Write a Python function that takes a list of strings as input and writes each string as a new line to a .txt file. If the file already exists, it should overwrite the existing content.
- **Appending Data to a File**
  - Write a Python function that appends 5 new lines to an existing .txt file. Ensure the function adds the new lines without deleting the existing content.

## TASKS

### Intermediate Level :

- **Counting Words in a File**
    - Create a Python function that reads a .txt file and counts the total number of words in the file. Handle any potential exceptions using try-except.

- **Reading and Reversing File Content**
    - Write a Python program that reads a .txt file, stores its content in a list, reverses the order of the lines, and writes the reversed content back to the same file.

- **Searching for a Specific Word**
    - Write a Python function that searches for a specific word in a .txt file and prints the line number(s) where the word appears. If the word doesn't exist, handle it using try-except.

## TASKS

**Advanced Level :**

- **File Content Analysis**
  - Write a Python program that reads a .txt file and performs the following tasks:
    - Counts the total number of words.
    - Counts the frequency of each word (ignoring case sensitivity).
    - Displays the top 3 most frequent words along with their counts.

- **Merging Multiple Files**
  - Create a Python function that takes two .txt files, merges their content line by line, and writes the merged content into a new file. For example, line 1 of file A should be followed by line 1 of file B, and so on. Handle cases where files have different numbers of lines using error handling.

- **Creating a Log File**
  - Write a Python program that reads the content of a .txt file and tracks how many times it has been accessed (opened). Each time the file is accessed, append a new entry in a separate log file with the date, time, and number of times the file has been accessed so far. Use try-except to handle errors related to file access.

## ANY QUESTIONS ?

PYTHON PROGRAM

# THANK YOU

GOOD LUCK IN YOUR CAREER