

## Python Program to Read a File

This article covers a program in Python that reads a file, entered by user at run-time of the program.

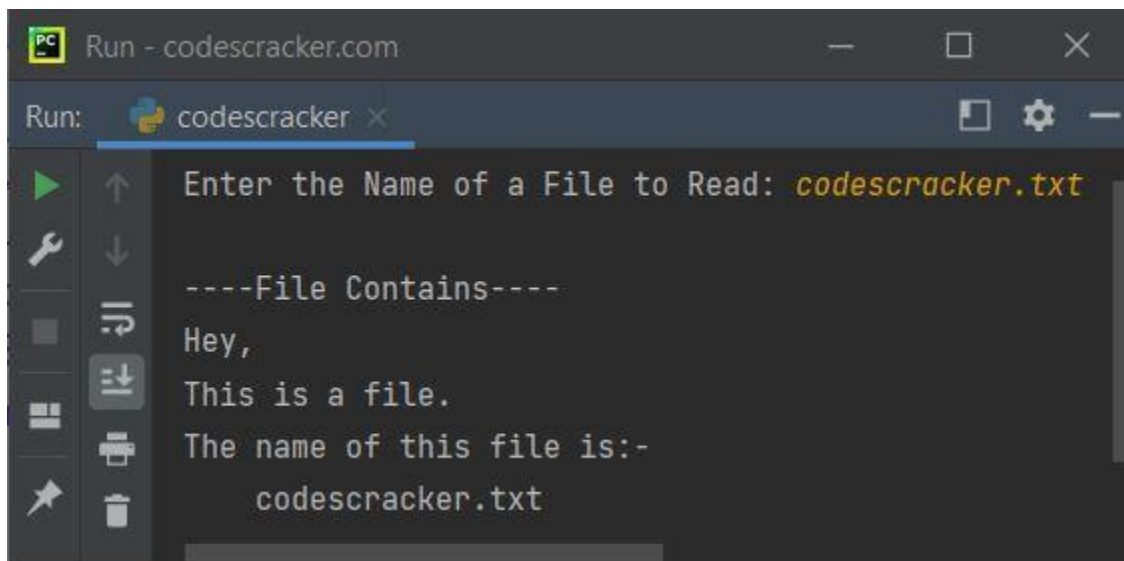
### Read a File in Python

The question is, *write a Python program to read a file. The name of file must be received by user at run-time.* The program given below is its answer:

```
print("Enter the Name of a File to Read: ", end="")
fileName = input()

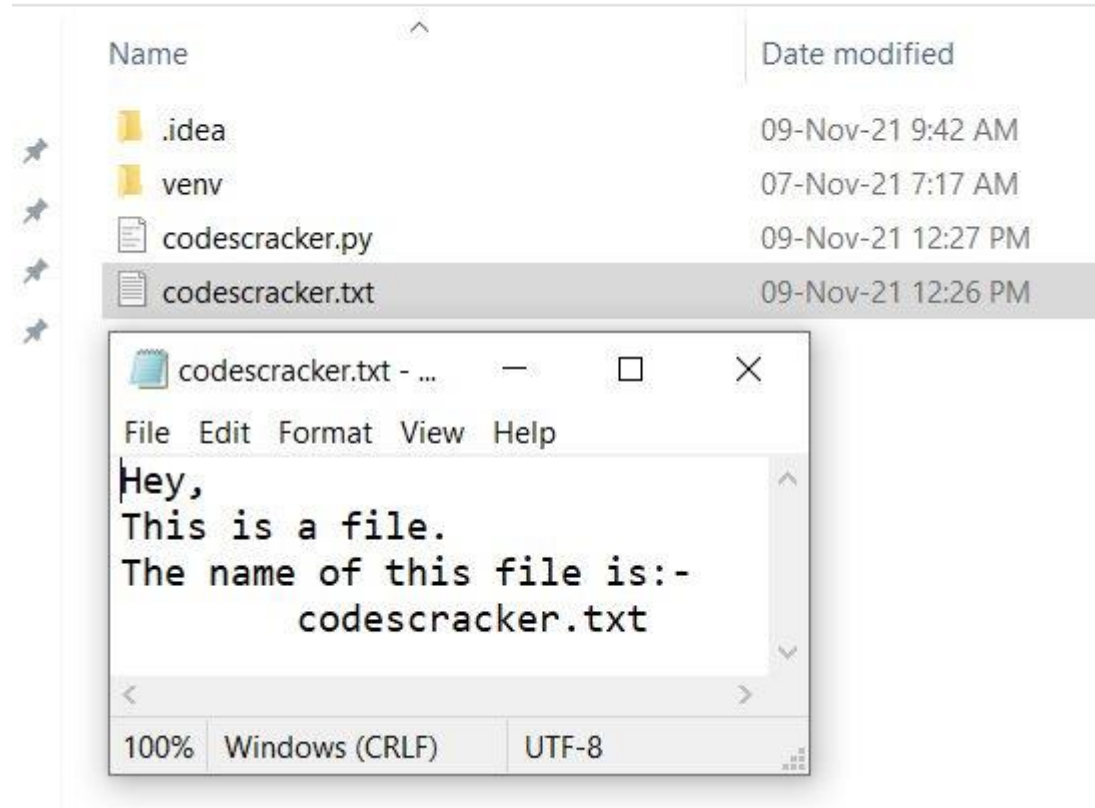
fileHandle = open(fileName, "r")
content = fileHandle.read()
print("\n----File Contains----")
print(content)
```

The snapshot given below shows the sample run of above Python program, with user input **codescracker.txt** as name of file to read



The file **codescracker.txt** has to be available inside the current directory, the directory where the above Python's source code is saved. Here is the snapshot of the current directory, with opened file, in my case:

> This PC > Local Disk (C:) > Users > DEV > codescracker.com



**Note** - The [read\(\) method](#) is used to read the whole content of a file.

**Note** - The [open\(\) method](#) is used to open a file. To learn in detail, refer to its separate tutorial.

**Note** - The [end parameter](#) used in above program, to skip insertion of an automatic newline using [print\(\)](#).

## Read Text File Line by Line in Python

This program does the same job as of previous program, that is to read a file. But this program is created in a way to read a file line by line. That is, all the lines of a file is initialized to a variable say **lines** in the form of list items or elements, using the **readlines()** method.

```
print("Enter the Name of a File to Read: ", end="")  
fileName = input()
```

```
fileHandle = open(fileName, "r")
lines = fileHandle.readlines()
print("\n----File Contains----")
for line in lines:
    print(line, end="")
```

This program produces the same output as of previous program.

**Note** - The [readlines\(\) method](#) returns the content of a file in the form of list, where each elements refers to the line of file. The detailed description about the method, is provided in its separate tutorial.

## Python Program to Write to File

This article is created to cover a program in Python that writes some text or content to a file, entered by user at run-time of the program.

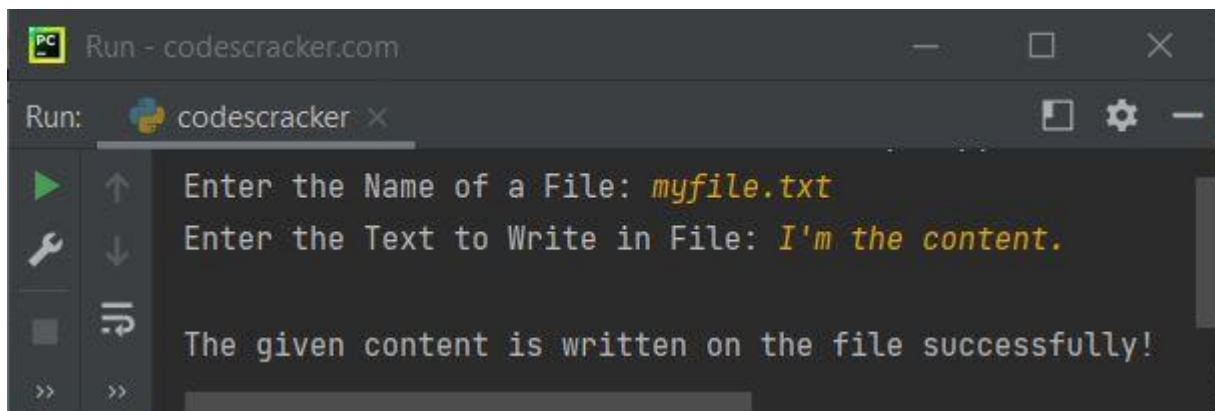
## Python Write Text to a File

The question is, *write a Python program to write some text to a file. The text and the name of file, must be received by user at run-time.* The program given below is its answer:

```
print("Enter the Name of a File: ", end="")
fileName = input()
print("Enter the Text to Write in File: ", end="")
text = input()

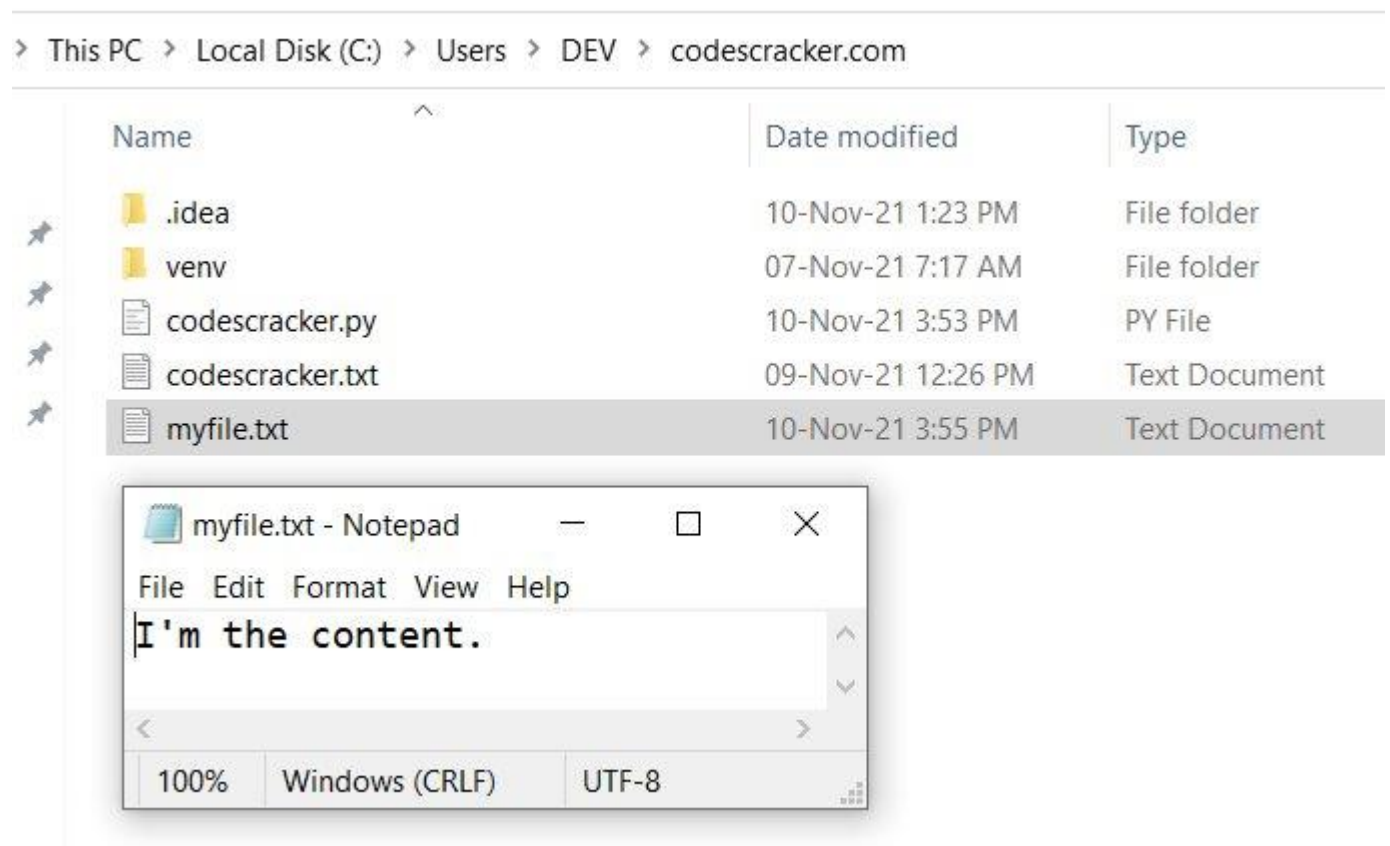
fileHandle = open(fileName, "w")
fileHandle.write(text)
fileHandle.close()
print("\nThe given content is written on the file successfully!")
```

The snapshot given below shows the sample run of above Python program, with user input **myfile.txt** as name of file, and **I'm the content.** as text to write in the file:

A terminal window titled "Run - codescracker.com" with a Python icon. The command "codescracker" is entered. The output shows three lines: "Enter the Name of a File: myfile.txt", "Enter the Text to Write in File: I'm the content.", and "The given content is written on the file successfully!".

```
Run: codescracker
Enter the Name of a File: myfile.txt
Enter the Text to Write in File: I'm the content.
The given content is written on the file successfully!
```

After executing the above program, with same sample run as given in above snapshot, if you open the current directory, then a file named **myfile.txt** will be available with the content **I'm the content.** written in it. Here is the snapshot of the current directory, in my case:



**Note** - Since the file **myfile.txt** was not available inside my current directory. Therefore the file gets created. But if the file would be available, then the file gets overwritten.

**Note** - The [open\(\) method](#) opens a file and returns as a file object.

**Note** - The [write\(\) method](#) is used to write content to a file.

**Note** - The [close\(\) method](#) is used to break the linkage of file from the program, using its object or handler, that was used to open the file.

## Python Write to File Line by Line

This program is created to allow user to write multiple lines of text or content to a desired file in the current directory, the directory where the Python program's source code is saved.

```
print("Enter the Name of a File: ", end="")
fileName = input()
fileHandle = open(fileName, "w")
print("How many lines of text to write ? ", end="")
noOfLines = int(input())
print("Enter", noOfLines, "lines of text, followed by ENTER key: ", end="")

for i in range(noOfLines):
    text = input()
    fileHandle.write(text)
    fileHandle.write("\n")

fileHandle.close()
print("\nAll lines are successfully written to the file!")

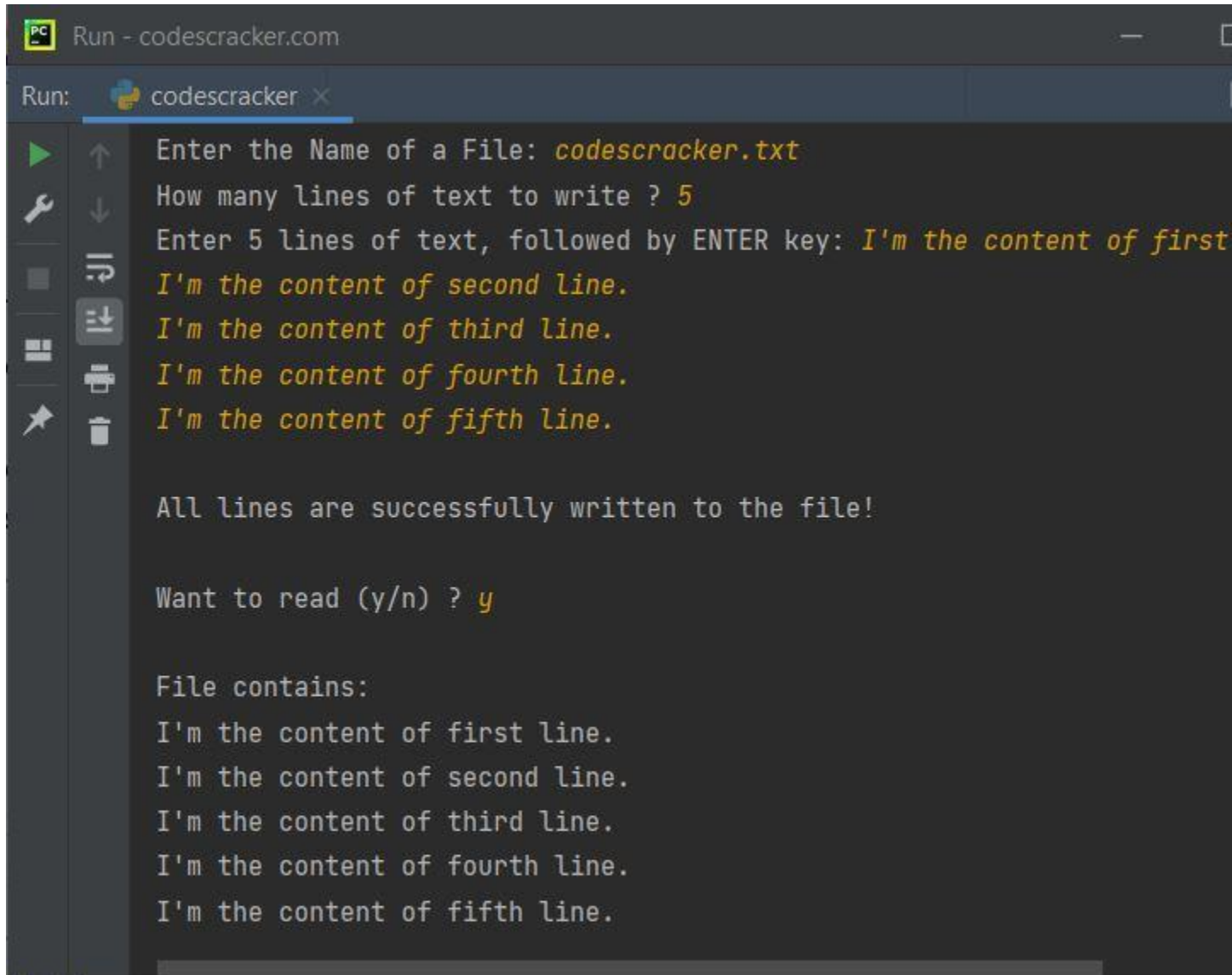
print("\nWant to read (y/n) ?", end="")
choice = input()
if choice == 'y':
    fileHandle = open(fileName, "r")
    print("\nFile contains:")
    print(fileHandle.read())
    fileHandle.close()
```

Sample run of above Python program, with user input **codescracker.txt** as name of file, **5** as total lines of text to write, and:

1. I'm the content of first line.
2. I'm the content of second line.

3. I'm the content of third line.
4. I'm the content of fourth line.
5. I'm the content of fifth line.

as five lines of text to write to the file, and finally **y** as choice to see the content of file, back on the output screen, is shown in the snapshot given below:



```
Run - codescracker.com
Run: codescracker x
Enter the Name of a File: codescracker.txt
How many lines of text to write ? 5
Enter 5 lines of text, followed by ENTER key: I'm the content of first
I'm the content of second line.
I'm the content of third line.
I'm the content of fourth line.
I'm the content of fifth line.

All lines are successfully written to the file!

Want to read (y/n) ? y

File contains:
I'm the content of first line.
I'm the content of second line.
I'm the content of third line.
I'm the content of fourth line.
I'm the content of fifth line.
```

**Note** - The [read\(\) method](#) is used to read the content of a file.

## Python Program to Append Text to a File

In this article, you will learn and get code to append some content (text) into a file using a Python program. Here are the list of programs:

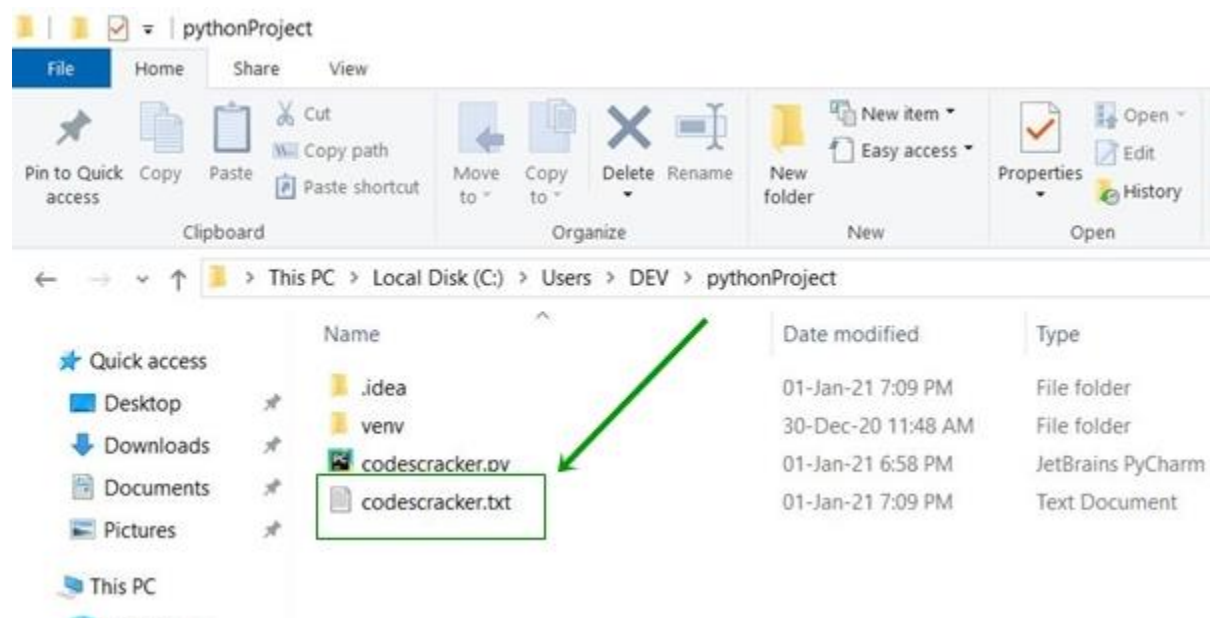
- Append Text to a File
- Append Text to a File and Display the Content of File

### Things to do Before Program

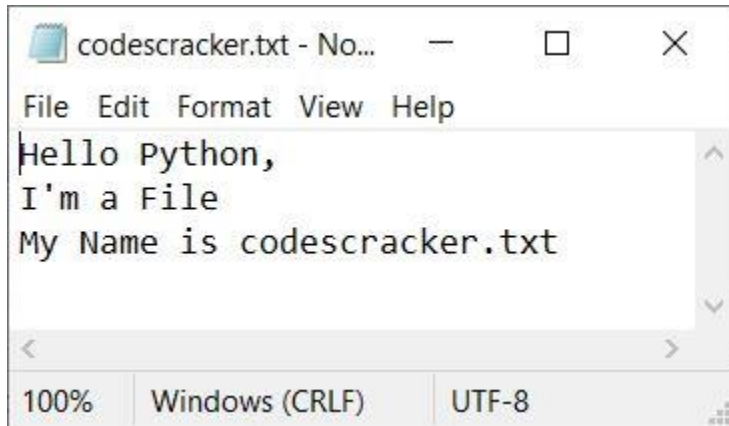
Because the program given below appends text (string or content) entered by user to a file say **codescracker.txt**. Therefore before executing the program, we've to create a file named **codescracker.txt** with some content say:

```
Hello Python,  
I'm a File  
My Name is codescracker.txt
```

Save this file in the current directory. The current directory means the directory where you are saving your Python source code. That is, the file and the Python program (to append text to the file) must be available in the same folder. Here is the snapshot of the folder where we've saved this file:



And here is the snapshot of opened file, **codescracker.txt**:



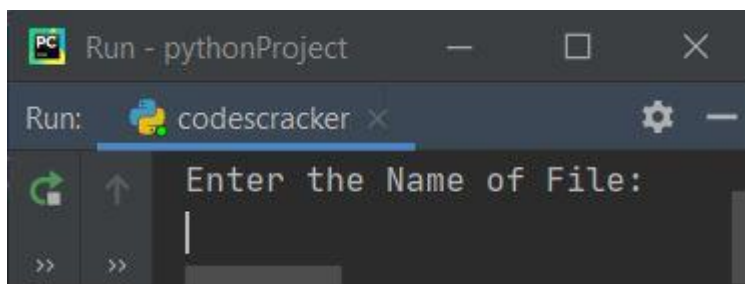
Now let's create a python program to append new content at last of this file.

## Append Text to a File in Python

This Python program asks from user to enter the name of file and then asks to enter the text (in one or more lines) to append it to the given file as shown in the program given here:

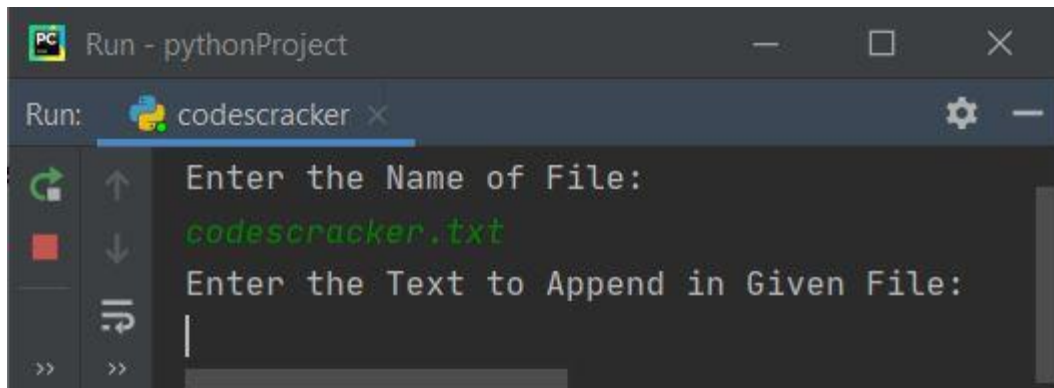
```
print("Enter the Name of File: ")
fileName = str(input())
fileHandle = open(fileName, "a")
print("Enter the Text to Append in Given File: ")
while True:
    text = str(input())
    if len(text)>0:
        fileHandle.write("\n")
        fileHandle.write(text)
    else:
        break
fileHandle.close()
```

Here is its sample run:





Now enter the name of file say **codescracker.txt** and press `ENTER`. Here is the output you will see:

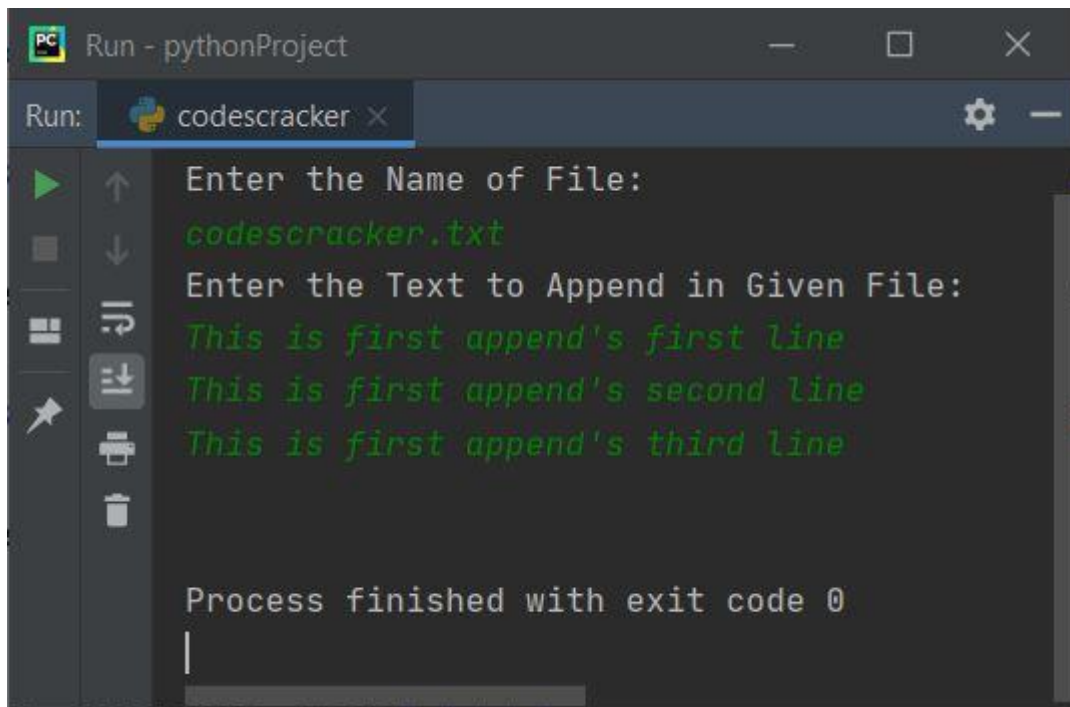


```
Run - pythonProject
Run: codescracker
Enter the Name of File:
codescracker.txt
Enter the Text to Append in Given File:
|
```

Then enter some texts (contents) say:

- This is first append's first line
- This is first append's second line
- This is first append's third line

These three lines entered in a way that, enter the first line of text, then press `ENTER`, again enter second line of text, press `ENTER` key and so on. Finally press `ENTER` key without typing anything to stop appending the content in the file. Here is the sample run with exactly same user inputs (as given above):

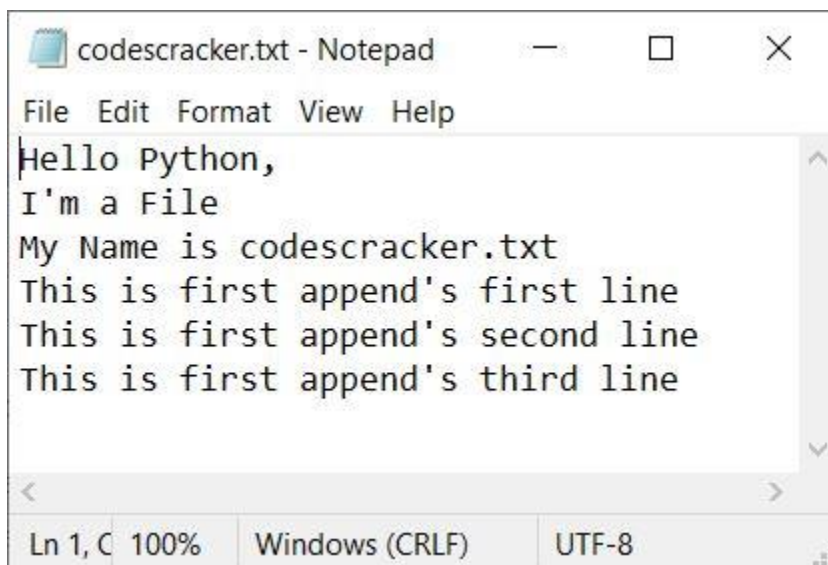


The screenshot shows a terminal window titled "Run - pythonProject" with a tab for "codescracker". The terminal displays the following text:

```
Enter the Name of File:
codescracker.txt
Enter the Text to Append in Given File:
This is first append's first line
This is first append's second line
This is first append's third line

Process finished with exit code 0
```

Now if you open the same file, **codescracker.txt**, then these content gets appended (added) to the file. Here is the snapshot of the opened file, **codescracker.txt**:



The screenshot shows a Notepad window titled "codescracker.txt - Notepad". The text inside the window is:

```
Hello Python,
I'm a File
My Name is codescracker.txt
This is first append's first line
This is first append's second line
This is first append's third line
```

The status bar at the bottom indicates "Ln 1, C 100%", "Windows (CRLF)", and "UTF-8".

**Note** - The **open()** function is used to open a file. It returns a file object. It takes two arguments. The first argument is the name of file and the second argument is its opening mode.

**Note** - The **break** keyword is used (in above program) to exit from the **while** loop

**Note** - The **write()** function is used to write content to a file using its object say **fileHandle**

**Note** - Don't forgot to close the file's object using **close()** function

## Append Text to File and Display the File

This program is similar to previous program with an extra feature. The extra feature is, this program appends the content entered by user to the given file and then asks from user whether they wants to see the new content of file or not.

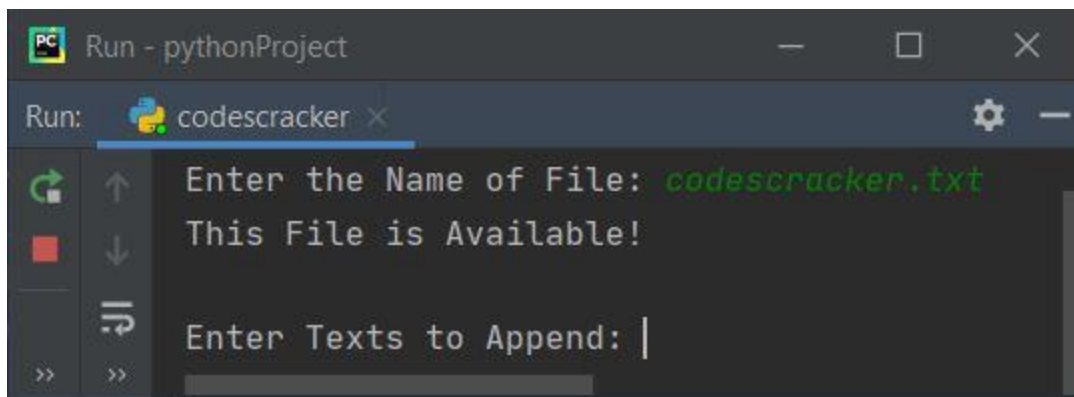
```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    print("This File is Available!")
    fileHandle.close()
    fileHandle = open(fileName, "a")
    print(end="\nEnter Texts to Append: ")
    while True:
        text = str(input())
        if len(text)>0:
            fileHandle.write("\n")
            fileHandle.write(text)
        else:
            break
    fileHandle.close()
    print("Texts Appended to the File Successfully!")
    print(end="Want to see the Content of File (y/n): ")
    ch = input()
    if ch=='y':
        fileHandle = open(fileName, "r")
        for content in fileHandle:
            print(end=content)
    else:
        print("Exiting...")
except IOError:
    try:
        fileHandle = open(fileName, "a")
        print("File Created Successfully!")
        print(end="\nEnter Texts to Append (Add): ")
        while True:
            text = str(input())
            if len(text)>0:
                fileHandle.write(text)
```

```

        fileHandle.write("\n")
    else:
        break
fileHandle.close()
print("Texts Appended to the File Successfully!")
print(end="Want to see the Content of File (y/n): ")
ch = input()
if ch=='y':
    fileHandle = open(fileName, "r")
    for content in fileHandle:
        print(end=content)
else:
    print("Exiting...")
except IOError:
    print("Error Occurred!")
print()

```

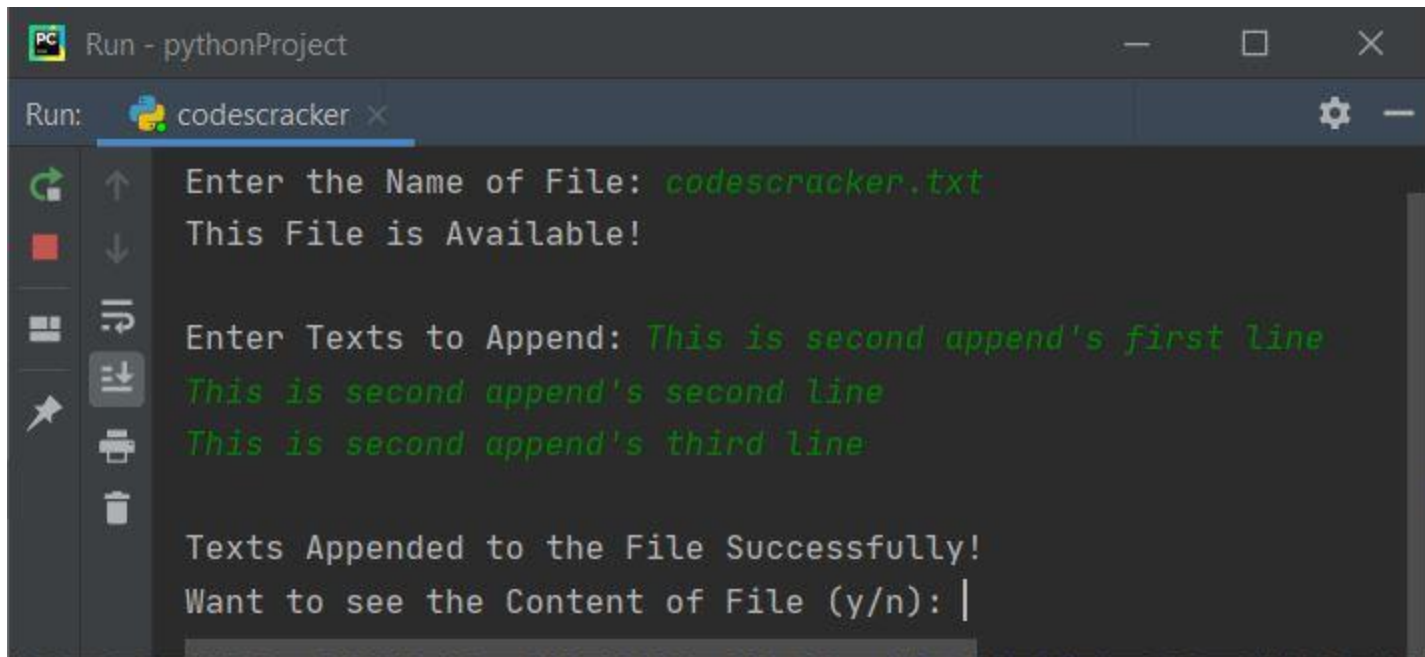
Now enter the name of same file, that is **codescracker.txt** and press `ENTER`. Here is the output:



Enter the following text, one by one:

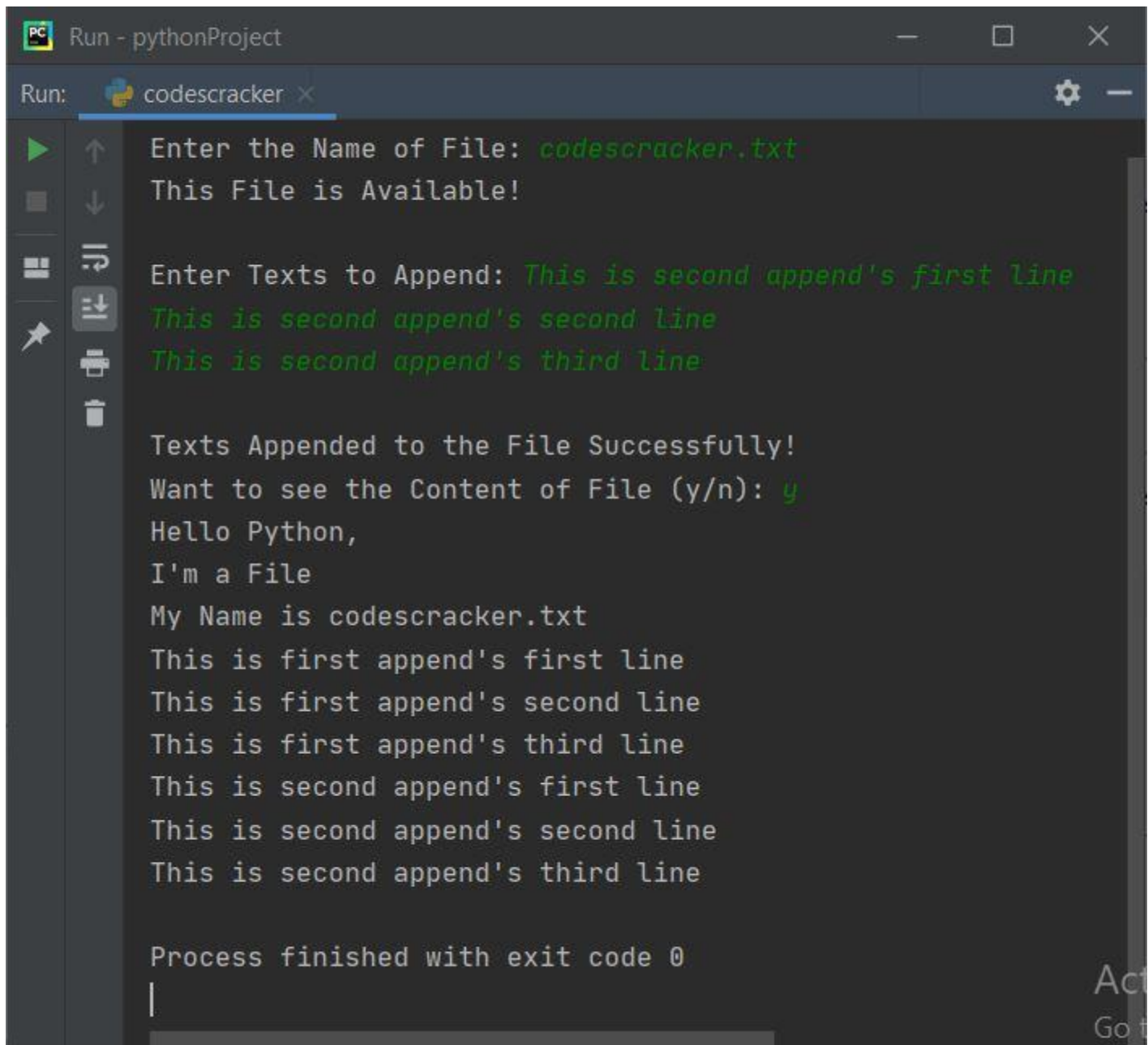
- This is second append's first line
- This is second append's second line
- This is second append's third line

Supply exactly these inputs and press `ENTER` key without typing any text. Here is the output you will see:

A screenshot of a Python IDE terminal window titled "Run - pythonProject". The terminal shows the execution of a program named "codescracker". The program prompts the user to enter a file name, which is "codescracker.txt". It then confirms the file is available. Next, it prompts for text to append, and three lines of text are entered: "This is second append's first line", "This is second append's second line", and "This is second append's third line". The program then confirms the text was appended successfully and asks if the user wants to see the file content (y/n). The cursor is positioned at the end of the prompt.

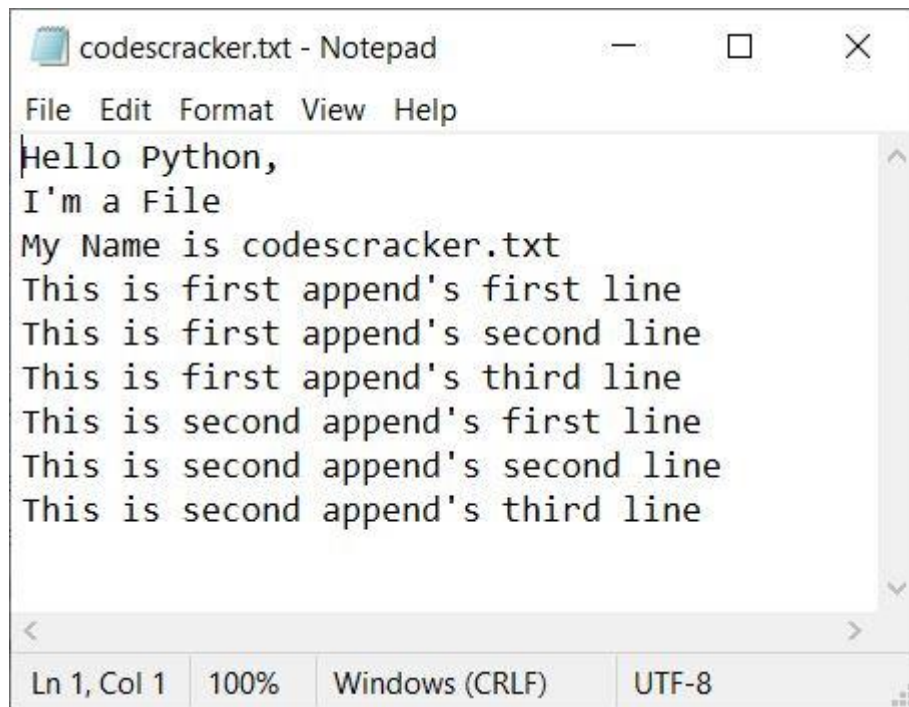
```
Run - pythonProject
Run: codescracker
Enter the Name of File: codescracker.txt
This File is Available!
Enter Texts to Append: This is second append's first line
This is second append's second line
This is second append's third line
Texts Appended to the File Successfully!
Want to see the Content of File (y/n): |
```

Now type **y** and press `ENTER` key to see the content of the file, **codescracker.txt**. Otherwise press **n** to exit from the program. Here is the sample output with **y** as choice:

A screenshot of a Python IDE terminal window titled "Run - pythonProject". The terminal shows the execution of a Python script named "codescracker". The script prompts the user to enter a file name, which is "codescracker.txt", and confirms it is available. It then prompts for text to append, with three lines of input: "This is second append's first line", "This is second append's second line", and "This is second append's third line". After appending, it confirms success and asks if the user wants to see the file content (y/n), with 'y' entered. The file content is then displayed, showing the original text and the three appended lines. The process finishes with exit code 0.

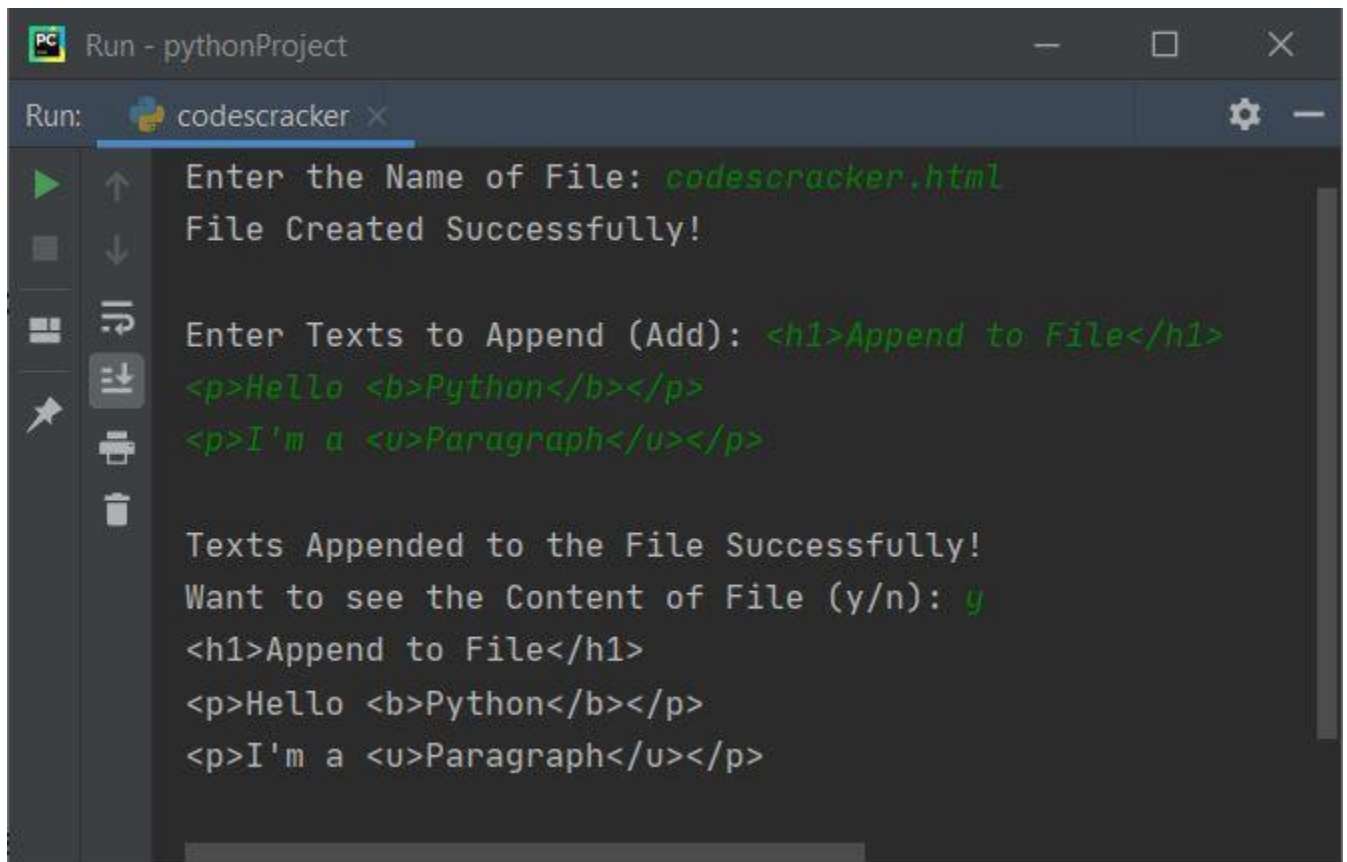
```
Run - pythonProject
Run: codescracker x
Enter the Name of File: codescracker.txt
This File is Available!
Enter Texts to Append: This is second append's first line
This is second append's second line
This is second append's third line
Texts Appended to the File Successfully!
Want to see the Content of File (y/n): y
Hello Python,
I'm a File
My Name is codescracker.txt
This is first append's first line
This is first append's second line
This is first append's third line
This is second append's first line
This is second append's second line
This is second append's third line
Process finished with exit code 0
|
```

And here is the file, **codescracker.txt** after executing all the things given above:



```
codescracker.txt - Notepad
File Edit Format View Help
Hello Python,
I'm a File
My Name is codescracker.txt
This is first append's first line
This is first append's second line
This is first append's third line
This is second append's first line
This is second append's second line
This is second append's third line
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Here is another sample run with user input say **codescracker.html**, the file that doesn't exist in the current directory, with some content and then **y** as choice to see the content of file:



```
Run - pythonProject
Run: codescracker x
Enter the Name of File: codescracker.html
File Created Successfully!

Enter Texts to Append (Add): <h1>Append to File</h1>
<p>Hello <b>Python</b></p>
<p>I'm a <u>Paragraph</u></p>

Texts Appended to the File Successfully!
Want to see the Content of File (y/n): y
<h1>Append to File</h1>
<p>Hello <b>Python</b></p>
<p>I'm a <u>Paragraph</u></p>
```

**Note** - The **a** (append) file opening mode opens a file. If file doesn't exist, then a new file with same name gets created automatically.

Now if you open this newly created file say **codescracker.html** through previous program's sample run, in a web browser like **Google Chrome**, here is the output you will see:





**Note** - Because the file is of **HTML** type, therefore this output is produced. To learn more about it, refer to [HTML Tutorial](#) to get its in-depth detail.

## Python Program to Copy Content of One File to Another

This article is created to cover some programs in Python, that copies the content of one file to another. Name of both files must be entered by user at run-time. Here are the list of programs:

- Copy Content of One File to Another without using **copyfile()** Method
- Using **copyfile()** Method
- Using **with** and **as** Keywords
- Shortest Python Code to Copy File's Content

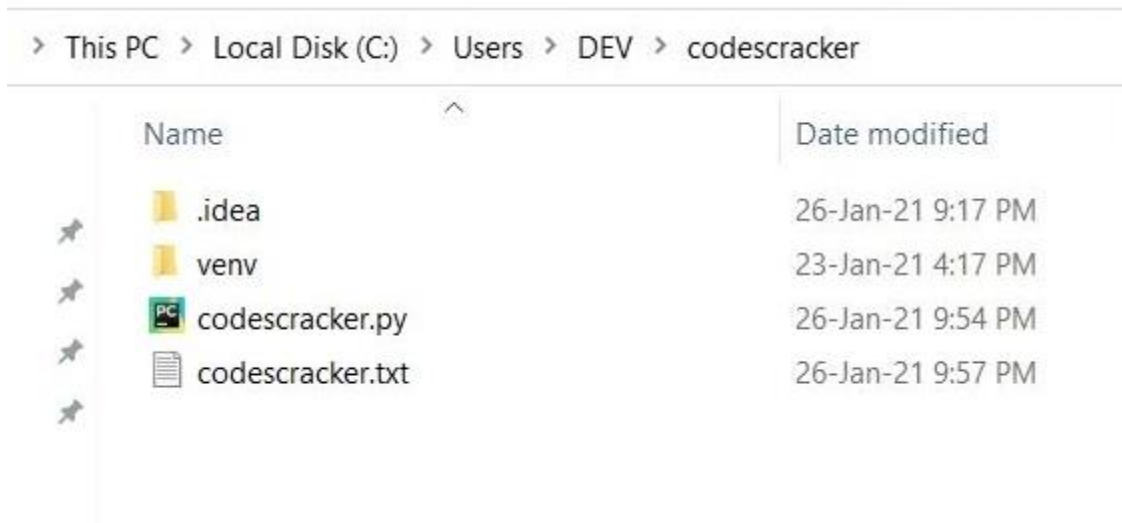
### Things to Do before Program

Since the program given below copies the content of one file to another, entered by user. Therefore we've to create a file inside the current directory. Current directory means, the directory where the Python source code to copy one file to another is saved. Therefore create a file named **codescracker.txt** with following content:

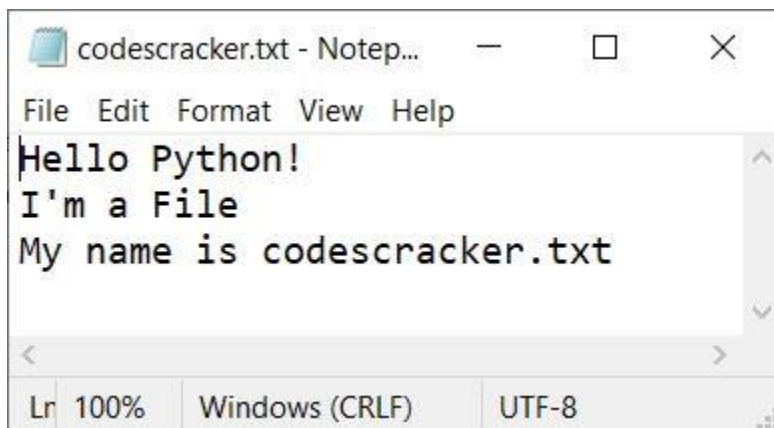
```
Hello Python!  
I'm a File
```

```
My name is codescracker.txt
```

Here is the snapshot of folder where the file and the Python source code (to copy file's content) is saved:



And here is the snapshot of opened file **codescracker.txt**:



Now let's move on to the program that copies the content of file entered by user (codescracker.txt here) to any other file, also entered by user.

## Copy Content of One File to Another in Python

To copy the content of one file to another in Python, you have ask from user to enter the name of two files. The first file referred as a source, whereas the

second file referred as a target file. That is, the content of source file gets copied to the target file as shown in the program given below:

The question is, **write a Python program to copy one file to another entered by user at run-time**. Here is its answer:

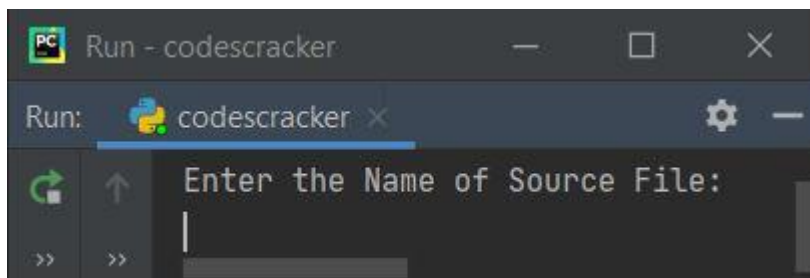
```
print("Enter the Name of Source File: ")
sFile = input()
print("Enter the Name of Target File: ")
tFile = input()

fileHandle = open(sFile, "r")
texts = fileHandle.readlines()
fileHandle.close()

fileHandle = open(tFile, "w")
for s in texts:
    fileHandle.write(s)
fileHandle.close()

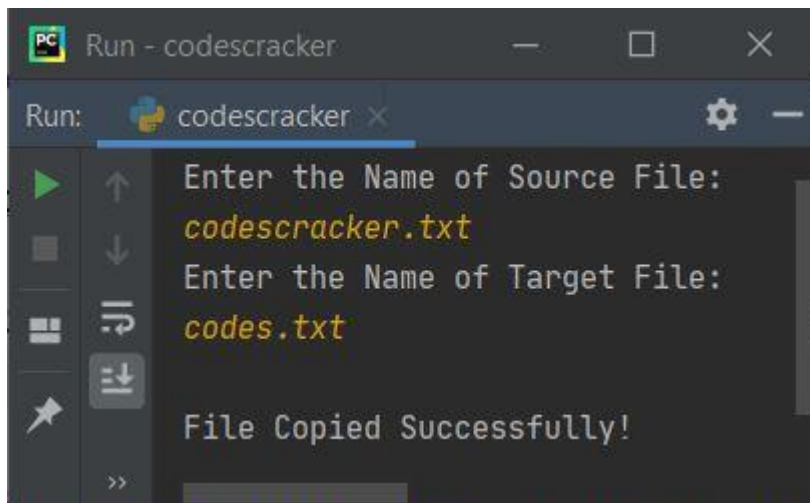
print("\nFile Copied Successfully!")
```

This is the initial output produced by this Python program, asking from user to enter the name of source file. Source file is the file, of which the content gets copied to another (target) file:

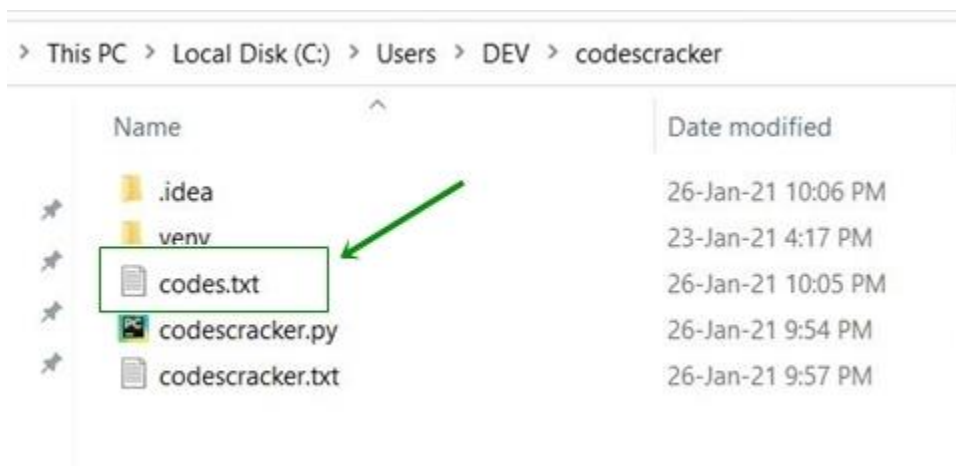


Now supply inputs say **codescracker.txt** (newly created file) as name of source file, press `ENTER` key and then type **codes.txt** as name of target file, and again press `ENTER` key to copy the content of source file to the target file.

If target file doesn't exist in the current directory, then a new file with same name gets created and the content of source file gets copied. Here is its sample output:



After supplying exactly these inputs as shown in this sample run. A file named **codes.txt** gets created inside the same folder, where the source code (above program) and the file (codescracker.txt) is saved. Here is the snapshot of the folder:



As you can see that a new file with same name as entered for the name of target file, gets created. And if you open the file **codes.txt**, it contains the same content as of **codescracker.txt** file.

### ***Modified Version of Previous Program***

This is the modified version of previous program. This program uses **try-except** to print error message when anything strange happens like the source file entered by user doesn't exist.

The **end=** is used to skip inserting newline using **print()**. This program also displays the copied content if user enters **y** as choice.

```
print("Enter Source File's Name: ", end="")
sfile = input()

try:
    filehandle = open(sfile, "r")

    print("Enter Target File's Name: ", end="")
    tfile = input()
    texts = filehandle.readlines()
    filehandle.close()

    try:
        filehandle = open(tfile, "w")

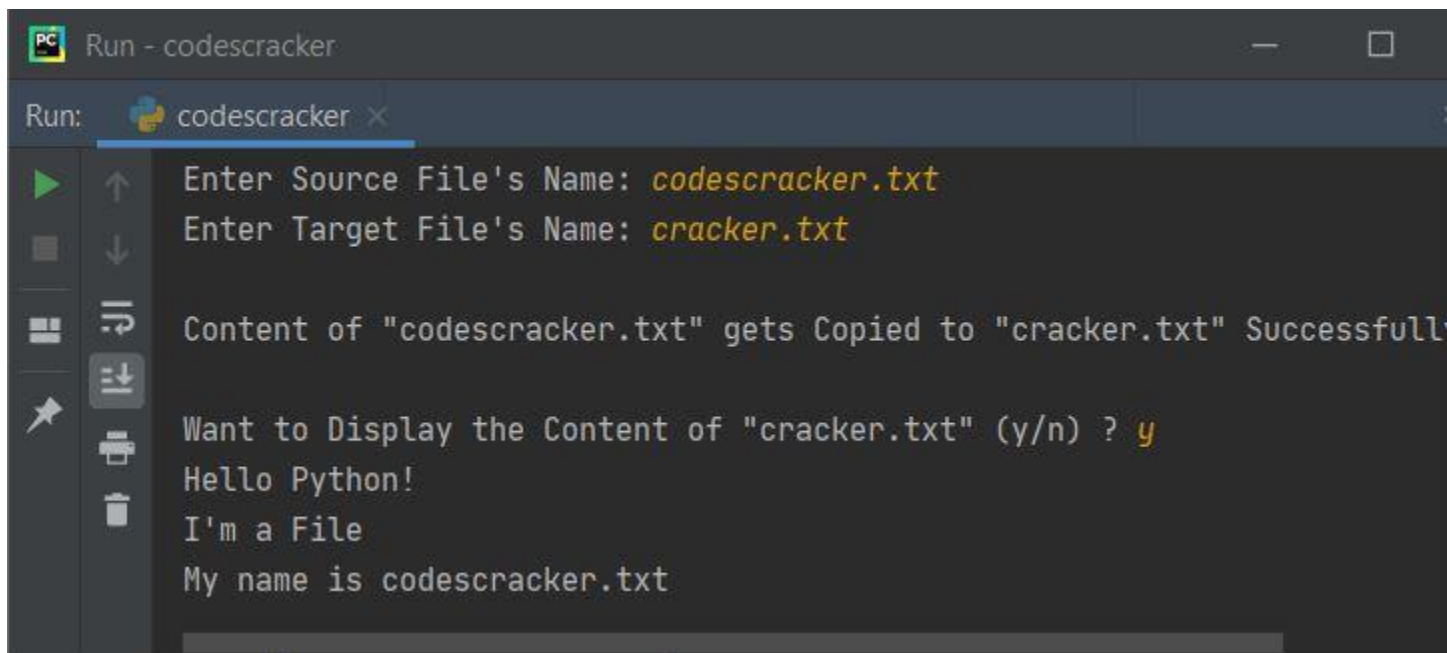
        for s in texts:
            filehandle.write(s)
        filehandle.close()

        print("\nContent of \"" + sfile + "\" gets Copied to \"" + tfile + "\"
Successfully!")
        print("\nWant to Display the Content of \"" + tfile + "\" (y/n) ? ",
end="")
        chk = input()
        if chk.lower()=='y':
            try:
                filehandle = open(tfile, "r")
                contents = filehandle.readlines()
                for s in contents:
                    print(s, end="")
                filehandle.close()
                print()
            except IOError:
                print("\nError occurred while opening the file!")

    except IOError:
        print("\nError occurred while opening/creating the file!")

except IOError:
    print("\nThe file doesn't exist!")
```

Here is its sample run with users inputs, **codescracker.txt** as name of source file, **cracker.txt** as name of target file and **y** as input to see the copied content of target file:



```
Run - codescracker
Run: codescracker x
Enter Source File's Name: codescracker.txt
Enter Target File's Name: cracker.txt
Content of "codescracker.txt" gets Copied to "cracker.txt" Successfully
Want to Display the Content of "cracker.txt" (y/n) ? y
Hello Python!
I'm a File
My name is codescracker.txt
```

**Note** - If you'll open the folder, you'll see another new file named **cracker.txt** gets created with same content.

In above program, if the statement:

```
filehandle = open(sfile, "r")
```

written as first statement inside the body of **try**, raises an error like when file doesn't exist, then its counterpart, that is **except**'s statement(s) gets executed.

## Copy File using **copyfile()** of **shutil**

This program uses **copyfile()** method to copy the content of one file to another. But before using the **copyfile()** method, we've to import **copyfile** from **shutil** module like shown in the program given below:

```
from shutil import copyfile

print("Enter Source File's Name: ", end="")
sfile = input()
print("Enter Target File's Name: ", end="")
tfile = input()

copyfile(sfile, tfile)
print("\nContent of \"" + sfile + "\" gets Copied to \"" + tfile + "\" Successfully!")
```

```

print("\nWant to Display the Content of \"" + tfile + "\" (y/n) ? ", end="")
chk = input()

if chk.lower()=='y':
    filehandle = open(tfile, "r")
    print(filehandle.read())
    filehandle.close()

```

This program works in similar way as of previous program. Above program can also be created by replacing the following two statements:

```

from shutil import copyfile
copyfile(sfile, tfile)

```

with

```

import shutil
shutil.copyfile(sfile, tfile)

```

## Copy File using with and as Keywords

This program uses **with** and **as** keywords to do the same job of copying the content of one file to another.

```

print("Enter Name of Source and Target File: ", end="")
sfile = input()
tfile = input()

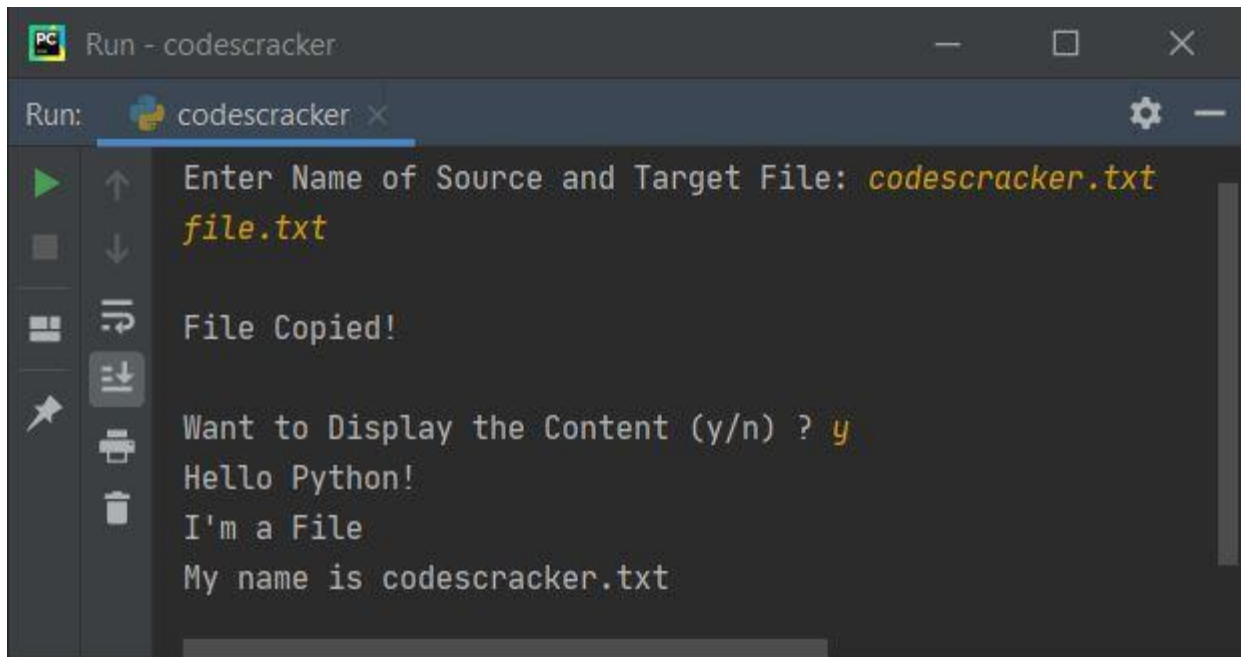
with open(sfile, "r") as shandle:
    with open(tfile, "w") as thandle:
        for line in shandle:
            thandle.write(line)

shandle.close()
thandle.close()

print("\nFile Copied!")
print("\nWant to Display the Content (y/n) ? ", end="")
chk = input()
if chk.lower()=='y':
    with open(tfile, "r") as fhandle:
        for line in fhandle:
            print(line, end="")
    fhandle.close()
    print()

```

Here is its sample run with user inputs, **codescracker.txt** (source file's name), **file.txt** (target file's name), and **y** (choice to see the content):



```
Run - codescracker
Run: codescracker x
Enter Name of Source and Target File: codescracker.txt
file.txt
File Copied!
Want to Display the Content (y/n) ? y
Hello Python!
I'm a File
My name is codescracker.txt
```

## Shortest Python Code to Copy File's Content

This is the shortest Python code to copy file's content with user-input.

```
sfile = input()
tfile = input()

with open(sfile, "r") as shandle:
    with open(tfile, "w") as thandle:
        for line in shandle:
            thandle.write(line)
```

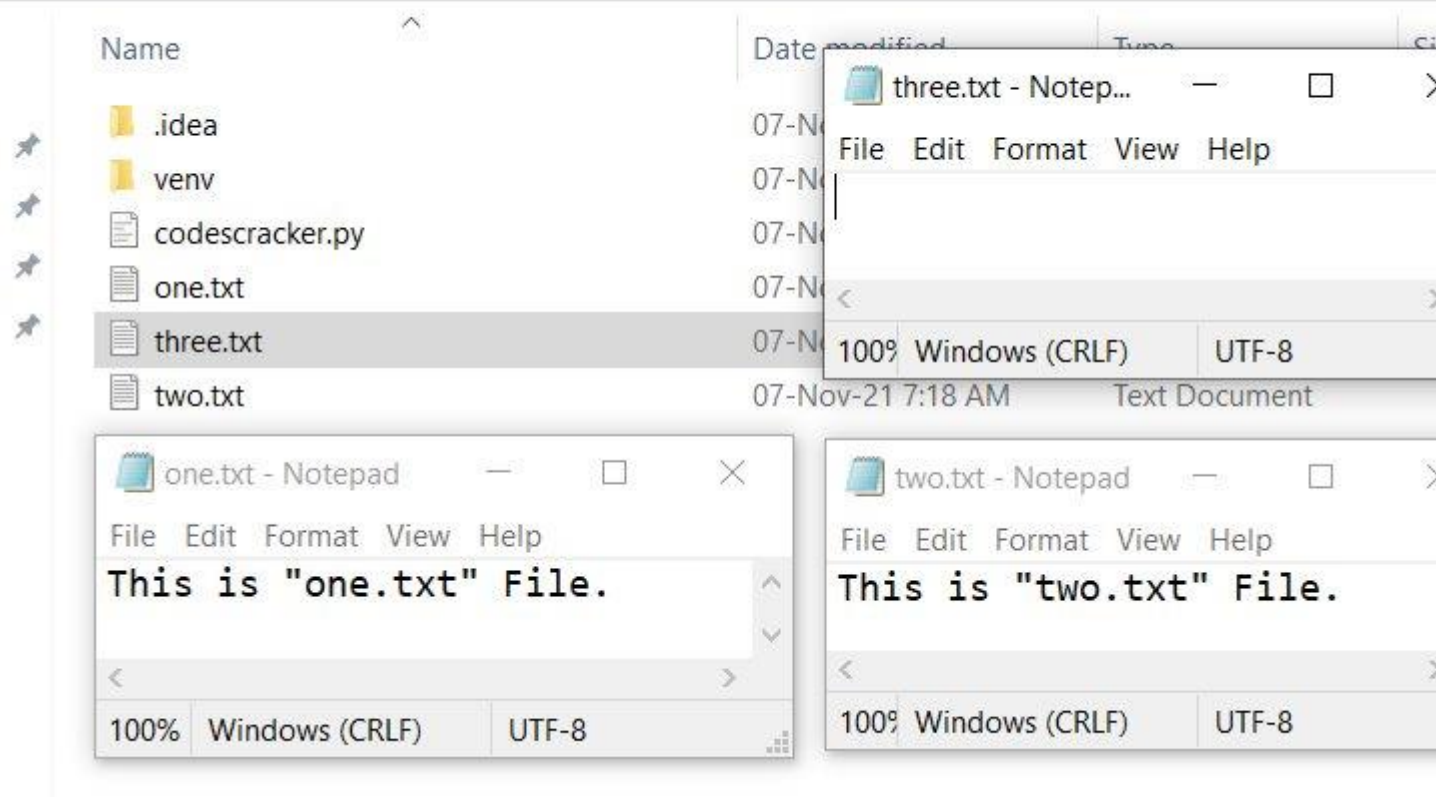
## Python Program to Merge Two Files

To merge the content of two files into a third file using a Python program. Then, first of all, we need to create two files in the current directory, the directory where the Python source code is saved.

Therefore, let's create two files namely **one.txt** and **two.txt** with some contents. And a third file named **three.txt** with or without any content. Here is the snapshot of the current directory, along with all these three files, opened:



> This PC > Local Disk (C:) > Users > DEV > codescracker.com



Now let's create the program, to merge the content of two text files into the third. It is not compulsory to create the third file, because while writing the content using **w** mode, the file automatically gets created, if not available.

## Merge Two Files into Third File in Python

The question is, *write a Python program to merge the content of two files into a third file. The name of all the three files must be received by user at run-time of the program.* Answer to this question, is the program given below:

```
print("Enter the Name of First File: ", end="")
fileOne = input()
print("Enter the Name of Second File: ", end="")
fileTwo = input()
print("Enter the Name of Third File: ", end="")
fileThree = input()

content = ""
fh = open(fileOne, "r")
for line in fh:
```

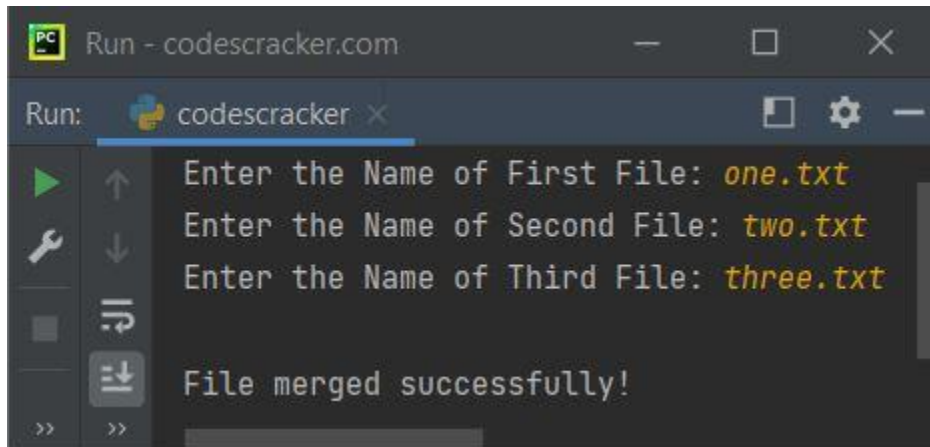
```
        content = content + line + '\n'
    fh.close()

    fh = open(fileTwo, "r")
    for line in fh:
        content = content + line + '\n'
    fh.close()

    fh = open(fileThree, "w")
    fh.write(content)

    print("\nFile merged successfully!")
```

The snapshot given below shows the sample run of above program, with user input **one.txt** and **two.txt** as name of first and second file whose content is going to merge into the third file named **three.txt**, also entered by user as name of third file:

A screenshot of a web-based code runner interface. The window title is "Run - codescracker.com". The code editor shows a Python script for merging two files into a third. The output console on the right shows the program's execution: it prompts for the first file name, the second file name, and the third file name, with user inputs "one.txt", "two.txt", and "three.txt" respectively. The final output is "File merged successfully!".

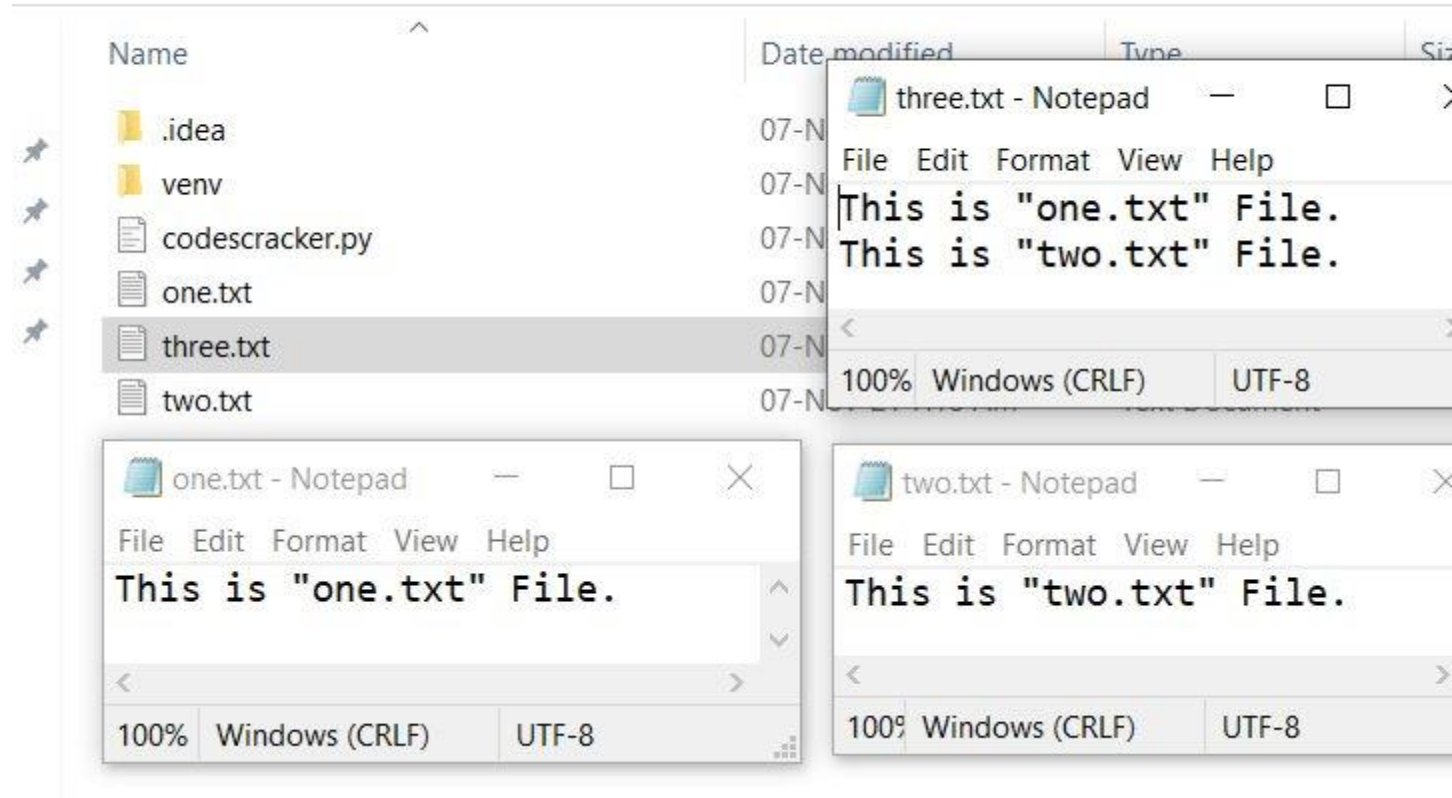
```
Run: codescracker x

Enter the Name of First File: one.txt
Enter the Name of Second File: two.txt
Enter the Name of Third File: three.txt

File merged successfully!
```

Here is the new snapshot of the three files, after executing the above program, using the sample run shown in the snapshot:

> This PC > Local Disk (C:) > Users > DEV > codescracker.com



**Note** - The [end parameter](#) used in above program, in a way to skip an automatic insertion of a newline.

**Note** - Use **a** in place of **w** mode, so that the previous content of third file does not gets deleted or overwritten.

Now the question is, what if user enters the name of file that does not available in the directory ?

therefore in that case, wrap the [open\(\) method](#), that opens a file, into **try** block, so that we can **catch** the exception named **FileNotFoundError**.

Here is the modified version of above program, that handles with file names, that does not available in the current directory:

```
print("Enter the Name of First File: ", end="")
fileOne = input()
try:
```

```

fhOne = open(fileOne, "r")
print("Enter the Name of Second File: ", end="")
fileTwo = input()
try:
    fhTwo = open(fileTwo, "r")
    print("Enter the Name of Third File: ", end="")
    fileThree = input()

    content = ""
    for line in fhOne:
        content = content + line + '\n'
    for line in fhTwo:
        content = content + line + '\n'
    fhOne.close()
    fhTwo.close()

    fh = open(fileThree, "w")
    fh.write(content)
    print("\nFile merged successfully!")

except FileNotFoundError:
    print("\nFile not found!")
except FileNotFoundError:
    print("\nFile not found!")

```

To learn about [File Handling](#) in detail, refer to its separate tutorial.

## Python Program to Count Vowels, Lines, Characters in Text File

This article is created to cover many programs in Python related to counting of characters such as vowels, consonants, newline, blank space etc. in a given or entered text file by user at run-time. Here are the list of programs covered in this article:

- Count vowels in a text file
- Count consonants in a text file
- Count new lines in a text file
- Count blank spaces in a text file
- Count total/all characters in a text file

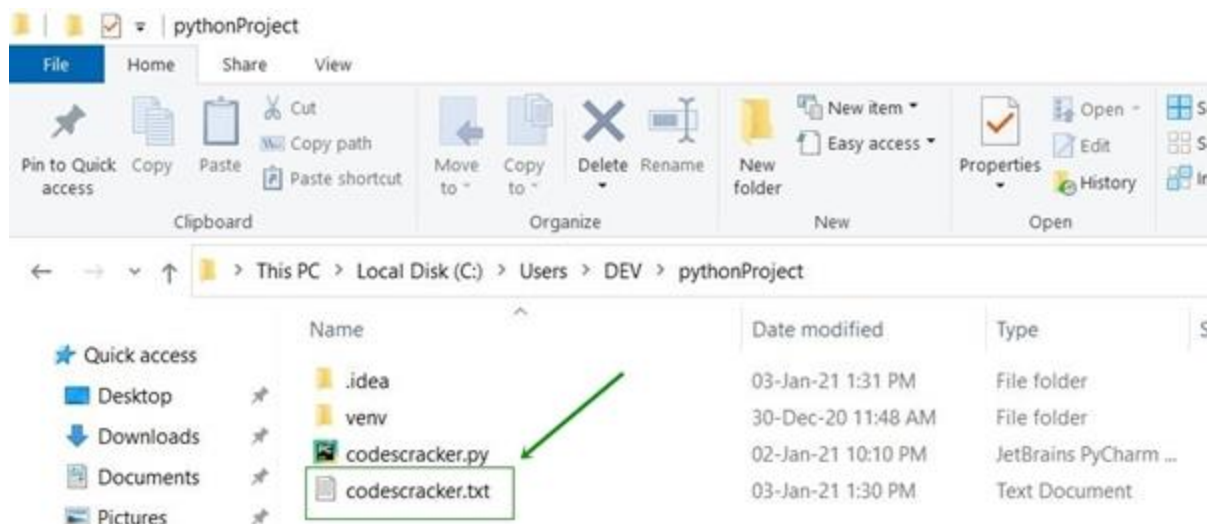
Before moving on to the programs, let's do some important things to implement and execute the program based on text file.

## Things to do Before Program

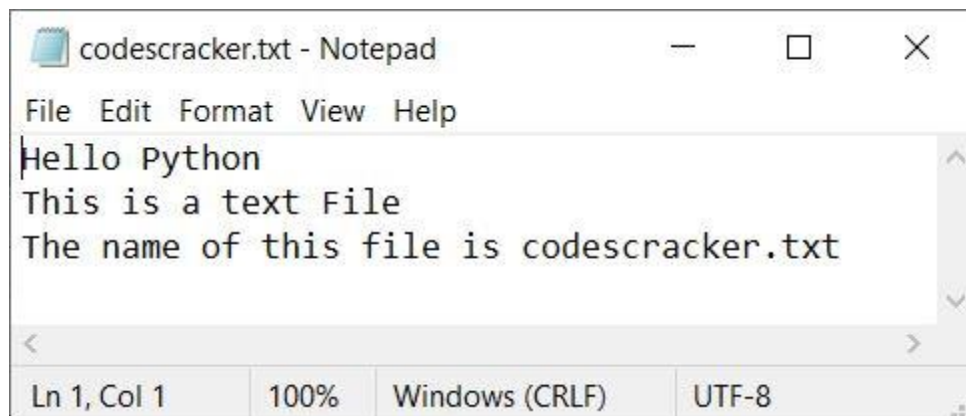
Because the program given below is used to count characters in a text file. Therefore first we've to create a text file say **codescracker.txt** with some contents say:

```
Hello Python
This is a text File
The name of this file is codescracker.txt
```

Save this file inside the current directory. The current directory is the directory where the Python code to count characters of this file is saved. Here is the snapshot of the folder where the file **codescracker.txt** is saved:



And here is the snapshot of opened file **codescracker.txt**:



Now let's create some Python programs to do the task like counting of characters, vowels, spaces etc. of this text file.

## Count Vowels in Text File

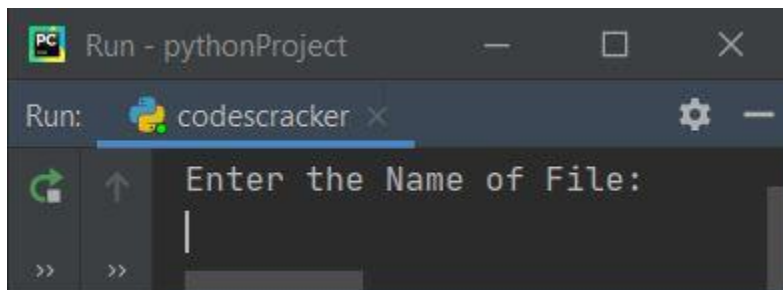
The question is, **write a Python program to count number of vowels present in a file.** The program given below is the answer to this question:

```
print("Enter the Name of File: ")
fileName = str(input())
fileHandle = open(fileName, "r")
tot = 0
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

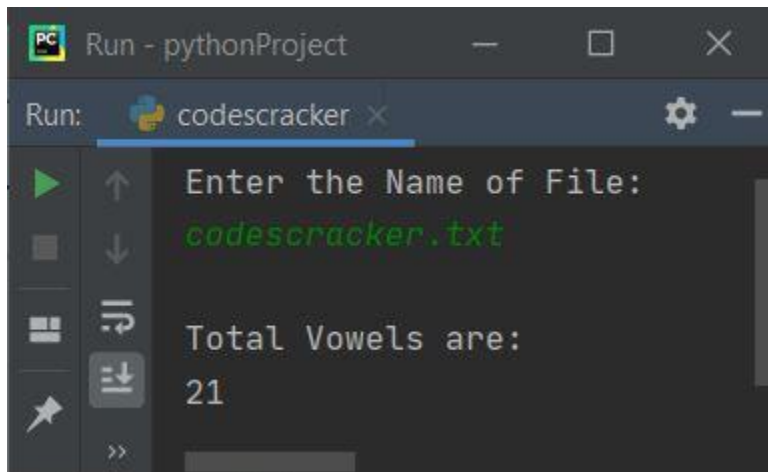
for char in fileHandle.read():
    if char in vowels:
        tot = tot+1
fileHandle.close()

print("\nTotal Vowels are:")
print(tot)
```

Here is its sample run:



Now enter the name of file say **codescracker.txt** (a newly created file as early of this article) and press `ENTER` to count and print total number of vowels present in the content of this file as shown in the snapshot given below:



### ***Modified Version of Previous Program***

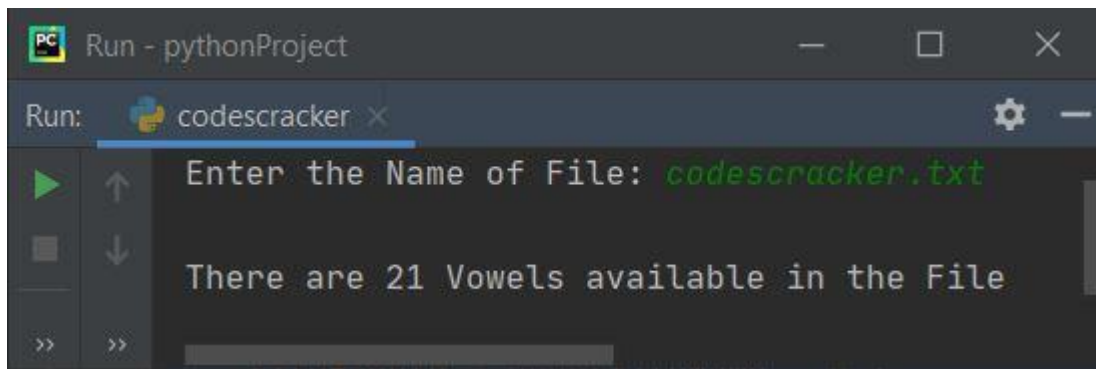
Let's modify the previous program. This program uses **end** to skip printing of an automatic newline. The **try-except** block is used for exception handling.

```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0
    vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

    for char in fileHandle.read():
        if char in vowels:
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " Vowels available in the File")
    elif tot==1:
        print("\nThere is only 1 Vowel available in the File")
    else:
        print("\nThere is no any Vowel available in the File!")
except IOError:
    print("\nError Occurred!")
    print("Either File doesn't Exist or Permission is not Allowed!")
```

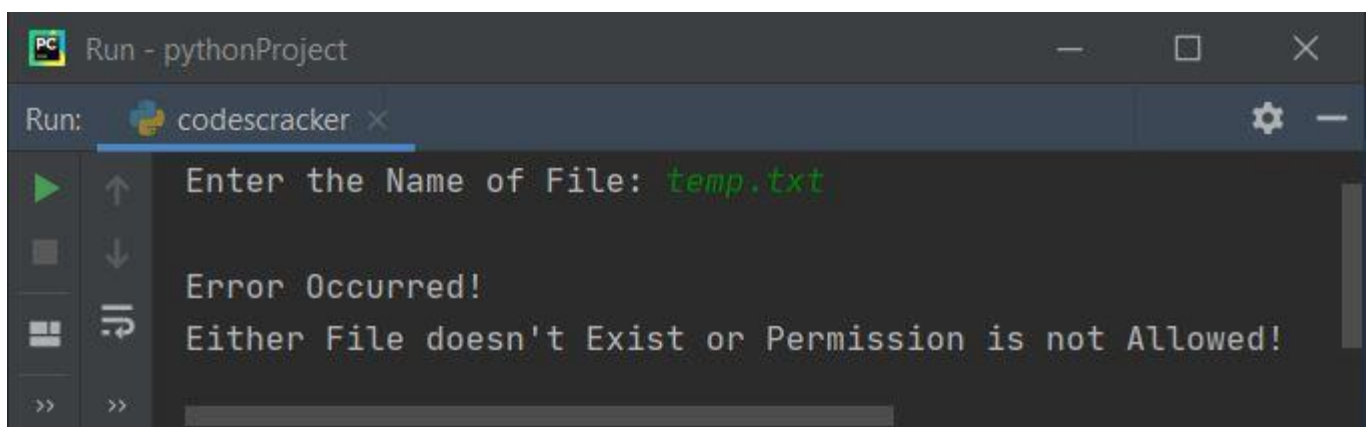
Here is its sample run with same user input say **codescracker.txt** as file's name:



The screenshot shows a terminal window titled "Run - pythonProject". The prompt "Run:" is followed by a Python icon and the text "codescracker". The terminal displays the following interaction: a green prompt "Enter the Name of File:" followed by the user input "codescracker.txt" in green, and then the output "There are 21 Vowels available in the File".

```
Run: codescracker
Enter the Name of File: codescracker.txt
There are 21 Vowels available in the File
```

Here is another sample run with user input say **temp.txt** (a non-existing file):



The screenshot shows a terminal window titled "Run - pythonProject". The prompt "Run:" is followed by a Python icon and the text "codescracker". The terminal displays the following interaction: a green prompt "Enter the Name of File:" followed by the user input "temp.txt" in green, and then the output "Error Occurred! Either File doesn't Exist or Permission is not Allowed!".

```
Run: codescracker
Enter the Name of File: temp.txt
Error Occurred!
Either File doesn't Exist or Permission is not Allowed!
```

## Count Consonant in Text File

The question is, **write a program in Python that counts total consonants available in a text file.** Here is its answer:

```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0
    vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

    for char in fileHandle.read():
        if char>='a' and char<='z':
            if char not in vowels:
                tot = tot+1
        elif char>='A' and char<='Z':
            if char not in vowels:
                tot = tot+1

    fileHandle.close()
```

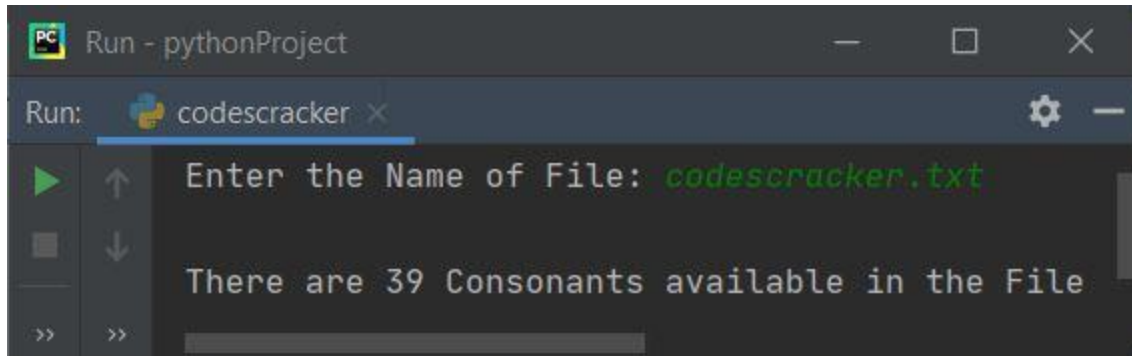


```

if tot>1:
    print("\nThere are " + str(tot) + " Consonants available in the File")
elif tot==1:
    print("\nThere is only 1 Consonant available in the File")
else:
    print("\nThere is no any Consonant available in the File!")
except IOError:
    print("\nError Occurred!")

```

Here is its sample run with same file name say **codescracker.txt**:



This program is similar to the previous program. The only difference is in logical code, we've changed the following code:

```

if char in vowels:
    tot = tot+1

```

with the block of code given below:

```

if char>='a' and char<='z':
    if char not in vowels:
        tot = tot+1
elif char>='A' and char<='Z':
    if char not in vowels:
        tot = tot+1

```

## Count New Lines in Text File

To count the number of new lines in a text file, use following Python program:

```

print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

    for char in fileHandle.read():
        if char=='\n':

```

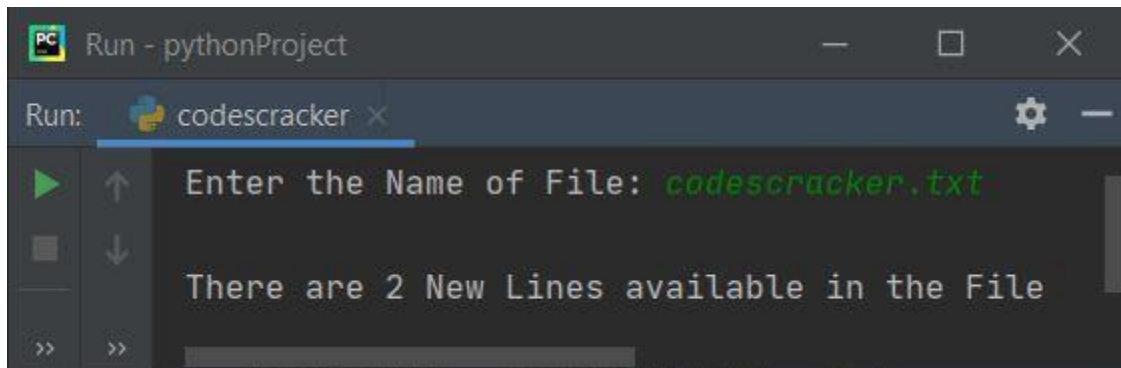
```

        tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " New Lines available in the File")
    elif tot==1:
        print("\nThere is only 1 New Line available in the File")
    else:
        print("\nThere is no any New Line available in the File!")
except IOError:
    print("\nError Occurred!")

```

Here is its sample run with same file name as created earlier:



```

PC Run - pythonProject
Run: codescracker x
>>> Enter the Name of File: codescracker.txt
>>> There are 2 New Lines available in the File
>>>

```

The only difference with this program to the program given to count vowels in text file is, we've changed the following block of code:

```

if char in vowels:
    tot = tot+1

```

with the block of code given below:

```

if char=='\n':
    tot = tot+1

```

## Count Blank Spaces in Text File

This program counts total number of blank spaces available in a text file entered by user at run-time.

```

print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

    for char in fileHandle.read():

```

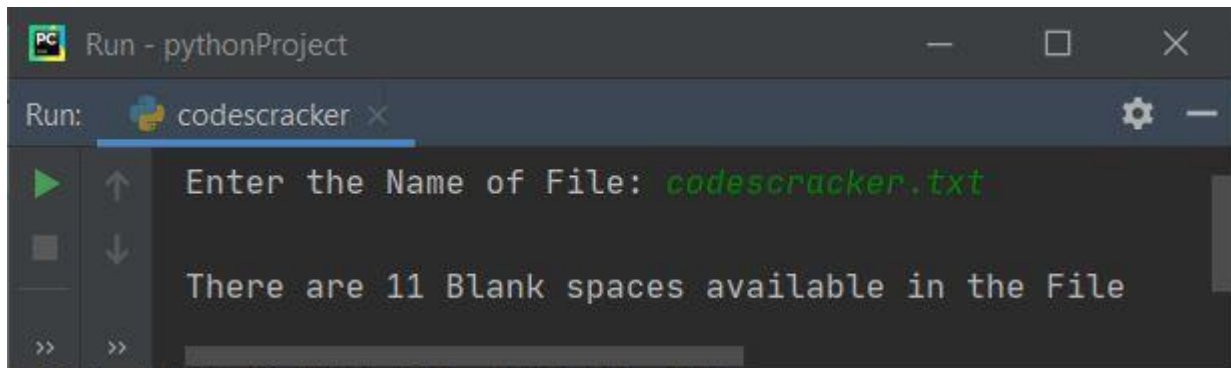
```

        if char==' ':
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " Blank spaces available in the File")
    elif tot==1:
        print("\nThere is only 1 Blank space available in the File")
    else:
        print("\nThere is no any Blank space available in the File!")
except IOError:
    print("\nError Occurred!")

```

The snapshot given below shows the sample run of this program with again same file name, **codescracker.txt**:



## Count Total Characters in Text File

This is the last program of this article. This program is created to count all the characters or total number of characters available in a text file.

```

print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

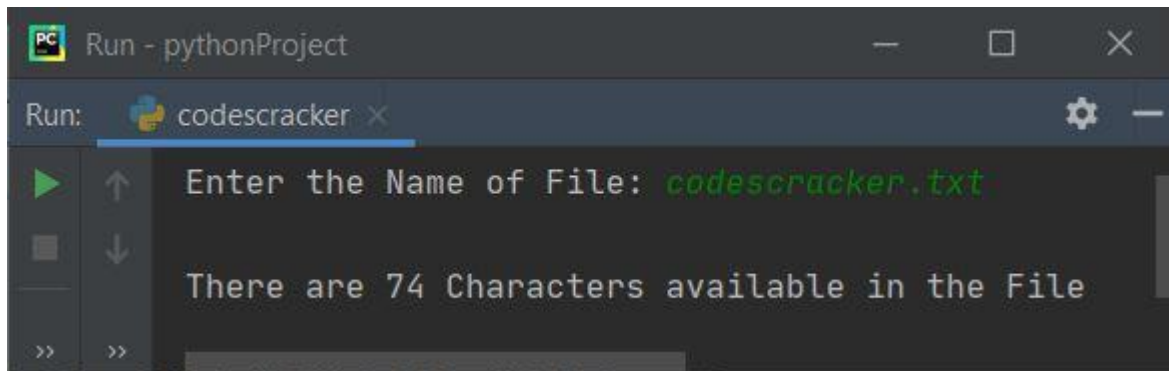
    for char in fileHandle.read():
        if char:
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " Characters available in the File")
    elif tot==1:
        print("\nThere is only 1 Character available in the File")
    else:
        print("\nThe File is empty!")
except IOError:

```

```
print("\nError Occurred!")
```

Here is its sample run:

A screenshot of a Python IDE window titled "Run - pythonProject". The window has a dark theme. At the top, there's a tab labeled "Run: codescracker". Below the tab, there's a terminal area. The terminal shows a prompt "Enter the Name of File: codescracker.txt" and the output "There are 74 Characters available in the File". The terminal has a scrollbar on the right side.

**Note** - Out of **74** characters, there are **21** vowels, **39** consonants, **11** blank spaces, **2** new lines and **1** dot (.)

## Python Program to Count Words in Text File

This article is created to cover some programs in Python, that counts total number of words available in a given text file by user at run-time. Here are the list of programs covered in this article:

- Count Words in a Text file using **for** Loop
- Count Words in a Text file using **for** Loop, **split()** and **len()** Method

Since the program created below is based on text file, therefore we must have to do some task (creating the file with some contents) before proceeding the program. Let's do this first.

### Things to do Before Program

As directed above, we must have to create a text file (in the current directory) before executing the program given below. So create a file named **codescracker.txt** with following content:

```
Hello World
This is a text File
My name is codescracker.txt
```

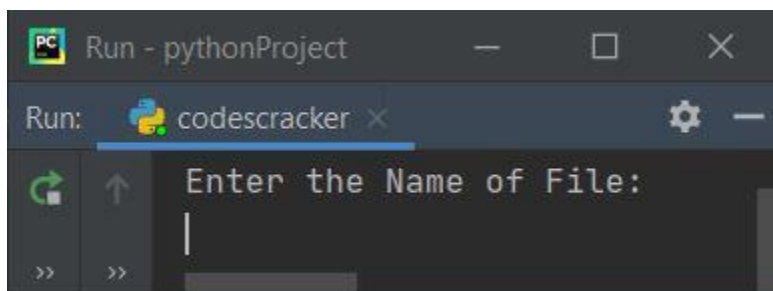
Save this file to the folder, where the source code of Python program to count all the words of this file, is saved. Now let's move on to the program.

## Count Total Words in a Text File

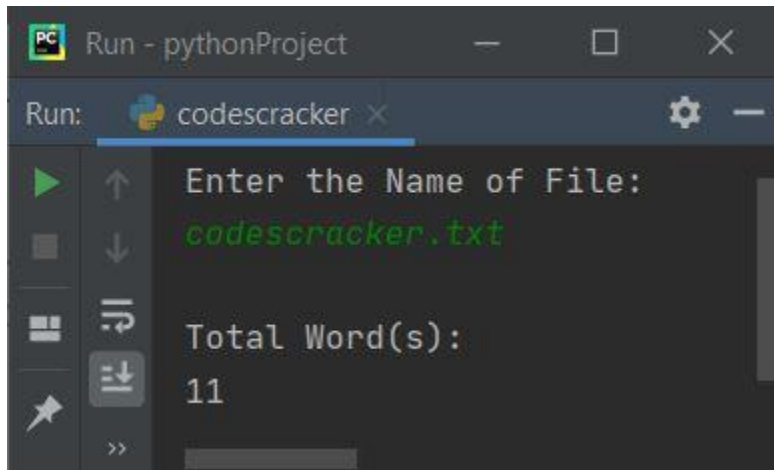
This is the simplest program that counts total number of words in a text file entered by user. Later on, we've modified this program:

```
print("Enter the Name of File: ")
fileName = input()
fileHandle = open(fileName, "r")
countWord = 0
for content in fileHandle:
    chk = 0
    contentLen = len(content)
    for i in range(contentLen):
        if content[i]==' ':
            if chk!=0:
                countWord = countWord+1
            chk = 0
        else:
            chk = chk+1
    if chk!=0:
        countWord = countWord+1
print("\nTotal Word(s): ")
print(countWord)
```

Here is its sample run:



Now enter the name of newly created file **codescracker.txt** and press **ENTER** key to count and print the number of words available in this file:

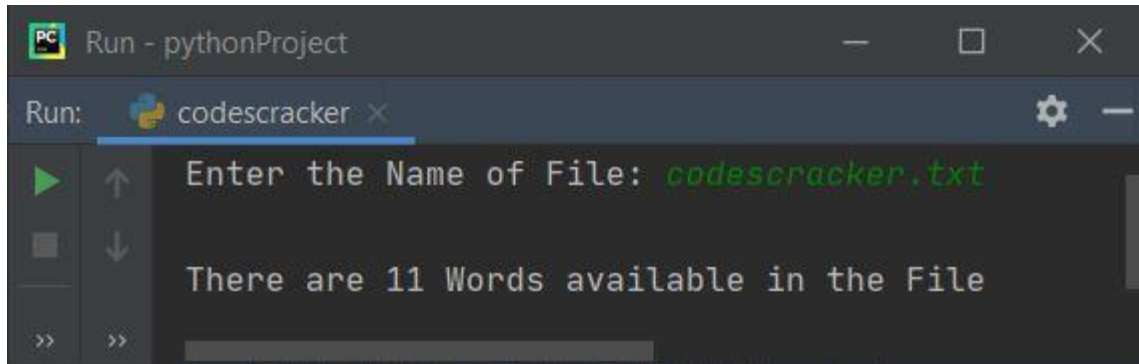


### ***Modified Version of Previous Program***

This program uses **end**, that skip insertion of an automatic newline. The **try-except** block is used for exception handling. That is, if file entered by user doesn't exist or anything other error gets thrown by **open()** method inside the **try** block, then program flow goes to **except** block and execute the code inside that block.

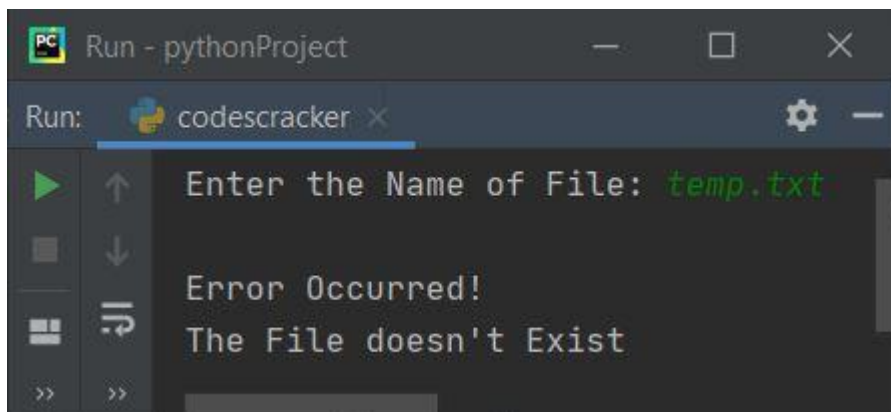
```
print(end="Enter the Name of File: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    countWord = 0
    for content in fileHandle:
        chk = 0
        contentLen = len(content)
        for i in range(contentLen):
            if content[i]==' ':
                if chk!=0:
                    countWord = countWord+1
                chk = 0
            else:
                chk = chk+1
        if chk!=0:
            countWord = countWord+1
    if countWord>1:
        print("\nThere are " +str(countWord)+ " Words available in the File")
    elif countWord==1:
        print("\nThere is only 1 word available in the File")
    else:
        print("\nThe File is empty!")
except IOError:
    print("\nError Occurred!")
    print("The File doesn't Exist")
```

Here is its sample run with same user input as of previous program:

A screenshot of a Python IDE terminal window titled "Run - pythonProject". The terminal shows the prompt "Enter the Name of File:" followed by the user input "codescracker.txt" in green. The output of the program is "There are 11 Words available in the File". The terminal interface includes a left sidebar with icons for running, debugging, and other functions, and a top bar with a tab labeled "codescracker".

```
Run: codescracker x
Enter the Name of File: codescracker.txt
There are 11 Words available in the File
```

Here is another sample run with user input say **temp.txt** (a non-existing file):

A screenshot of a Python IDE terminal window titled "Run - pythonProject". The terminal shows the prompt "Enter the Name of File:" followed by the user input "temp.txt" in green. The output of the program is "Error Occurred!" followed by "The File doesn't Exist". The terminal interface is similar to the previous screenshot, with a left sidebar and a top bar with a tab labeled "codescracker".

```
Run: codescracker x
Enter the Name of File: temp.txt
Error Occurred!
The File doesn't Exist
```

**Note** - To learn more about the user-based code (as given in previous program) that how to count words, refer to [Count Words in String](#) article to get every required things.

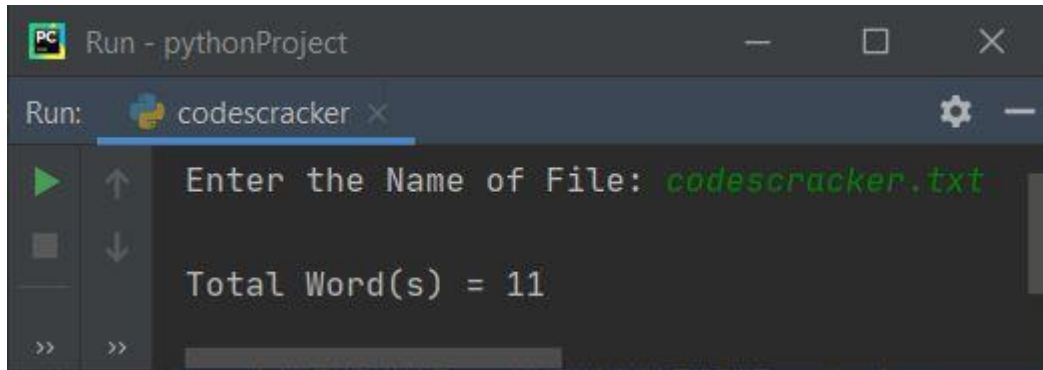
## Count Words in Text File using split()

This program uses **split()** method to split string into a list. And the **len()** method counts total elements available in a list.

```
print(end="Enter the Name of File: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    countWord = 0
    for content in fileHandle:
        wrd = content.split()
        countWord = countWord+len(wrd)
    print("\nTotal Word(s) = " + str(countWord))
```

```
except IOError:
    print("\nFile doesn't Exist!")
```

Here is its sample run with same user input as of very first program in this article:

A screenshot of a Python IDE window titled "Run - pythonProject". The window has a dark theme. At the top, there's a tab labeled "Run: codescracker". Below the tab, there's a terminal area. The terminal shows the prompt "Enter the Name of File: codescracker.txt" and the output "Total Word(s) = 11". The terminal also has a scrollbar on the right side.

## Python Program to Print Contents of a File in Reverse Order

This article is created to cover some programs in Python that prints the content of a file (entered by user at run-time) in reverse order. For example, if a file say *abc.txt* contains:

```
Hello,  
This is Python
```

And if user enters the name of file as input say **abc.txt**, after executing the program given below. Then the content of this file gets printed on output in reverse order. That looks like:

```
nohtyP si sihT  
,olleH
```

**Condition** - Both file and program's source code must be saved in same folder (current directory).

### Things to do Before Program

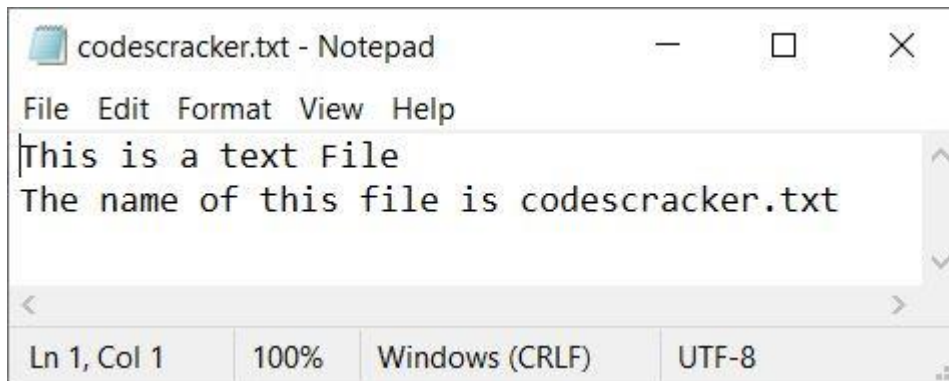
Because the program given below is used to print the content of a file (entered by user) in reverse order. Therefore we've to create a file say **codescracker.txt** with some content to print the content of this file in



reverse order using the Python program given below. Create a file with following content:

```
This is a text File  
The name of this file is codescracker.txt
```

Save the file with name **codescracker.txt** inside the current directory (the directory where the python program to print file's content in reverse order is saved). Here is the snapshot of the opened file, *codescracker.txt*.



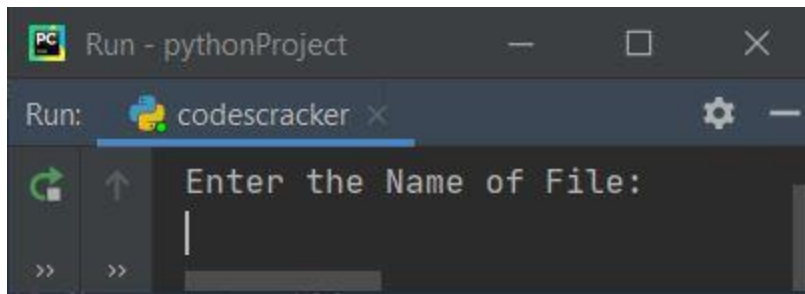
Now let's move on and create a Python program to read the content of this file and print in reverse order.

## Print File's Content in Reverse Order

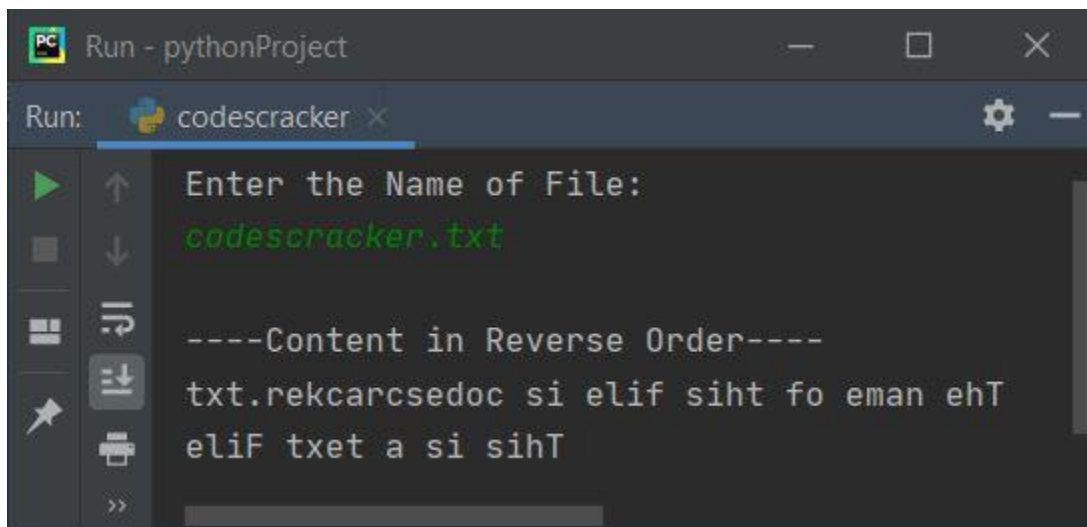
The question is, *write a Python program that prints file's content in reverse order*. The upcoming program is answer to this question:

```
print("Enter the Name of File: ")  
fileName = input()  
  
fileHandle = open(fileName, "r")  
fileContent = ""  
for content in fileHandle:  
    fileContent = fileContent+content  
  
print("\n----Content in Reverse Order----")  
fileContent = fileContent[::-1]  
print(fileContent)
```

Here is the initial output of this program's sample run:



Now enter the name of newly created file and press `ENTER` key to read the file and print its content in reverse order as shown in the snapshot given below:



### ***Modified Version of Previous Program***

What if user enters a file that doesn't exist in the current directory ?

What if the file entered by user is not accessible by the program ?

if any problem like these two gets occurred, then here we've another program, that is the modified version of previous program.

This program uses **try-except** block to handle with these errors happened while operating with the file. Let's have a look at the program and its sample run given below for clear understanding about it.

```
print(end="Enter File's Name: ")
fname = input()

try:
    fhand = open(fname, "r")
```

```

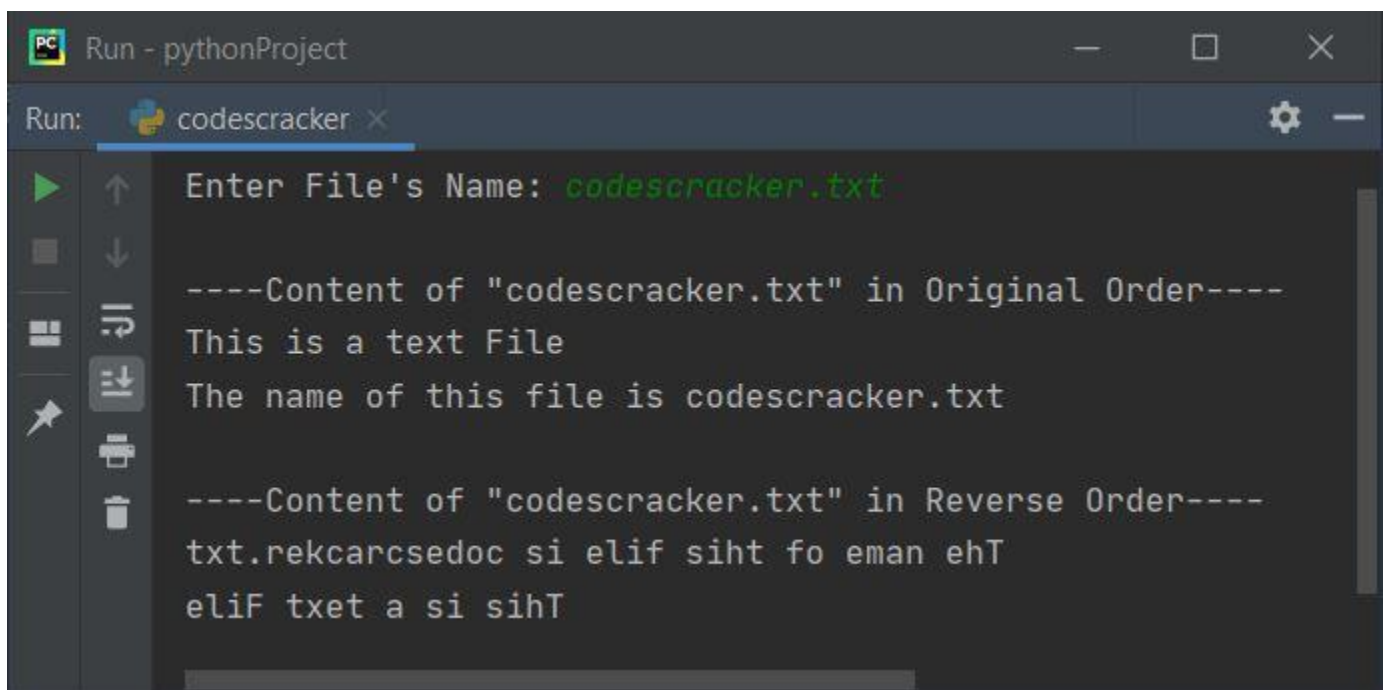
fcont = ""
for cont in fhand:
    fcont = fcont + cont

print("\n----Content of \"" +str(fname)+ "\"" in Original Order----")
print(fcont)
print("\n----Content of \"" +str(fname)+ "\"" in Reverse Order----")
fcont = fcont[::-1]
print(fcont)

except IOError:
    print("\nThe file doesn't exist!")

```

Here is its sample run with same user input as of previous program:

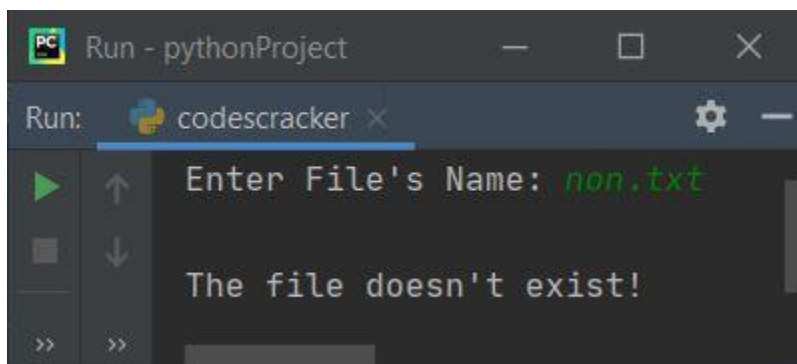


```

Run - pythonProject
Run: codescracker x
Enter File's Name: codescracker.txt
----Content of "codescracker.txt" in Original Order----
This is a text File
The name of this file is codescracker.txt
----Content of "codescracker.txt" in Reverse Order----
txt.rekcarcsedoc si elif siht fo eman ehT
eliF txet a si siht

```

Here is another sample run with user input **non.txt** (non-existing file):



```

Run - pythonProject
Run: codescracker x
Enter File's Name: non.txt
The file doesn't exist!

```

Since, the file **non.txt** does not exist in the current directory, therefore program flow goes to **except** block and throws an error saying **The file doesn't exist!**. You can also modify the program according to your need.

## Python Program to Print Lines Containing Given String in File

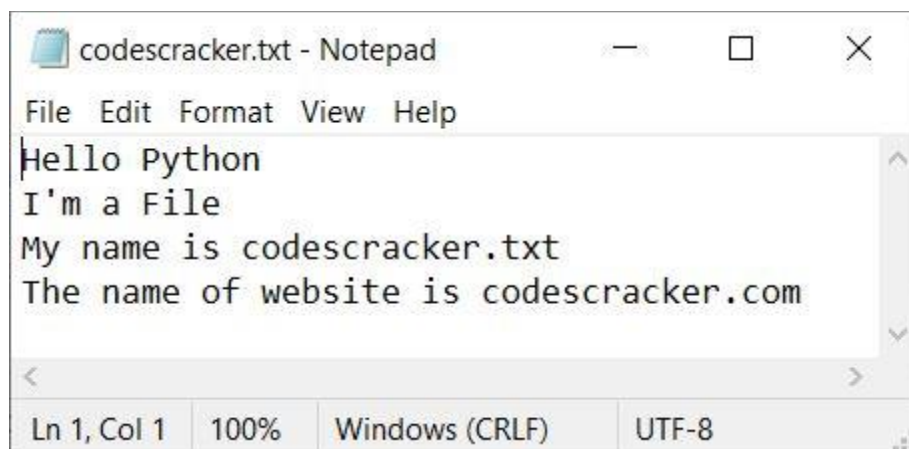
This article is created to cover some programs in Python that find and prints lines containing any given string (entered by user at run-time) in a given file (also entered by user at run-time). Let's do some task before proceeding the program.

### Things to do Before Program

Because the program given below is used to list and print all lines that contains the string entered by user in a text file. Therefore a text file say **codescracker.txt** must be created and saved inside the current directory. So create a file with following content:

```
Hello Python
I'm a File
My name is codescracker.txt
The name of website is codescracker.com
```

Save this file with name **codescracker.txt** in the folder where the python program to print lines containing given string in a text file is saved. Here is the snapshot that shows the content of newly created file:

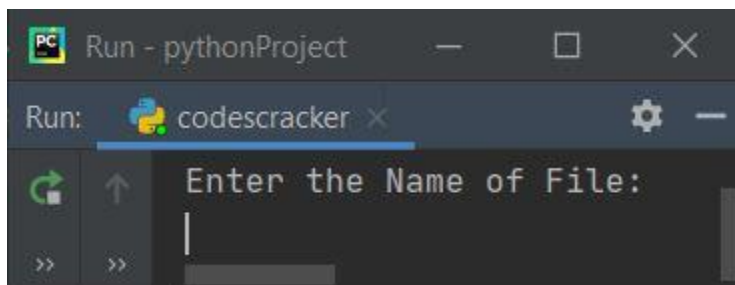


## Print Line that Contains Given String in a File

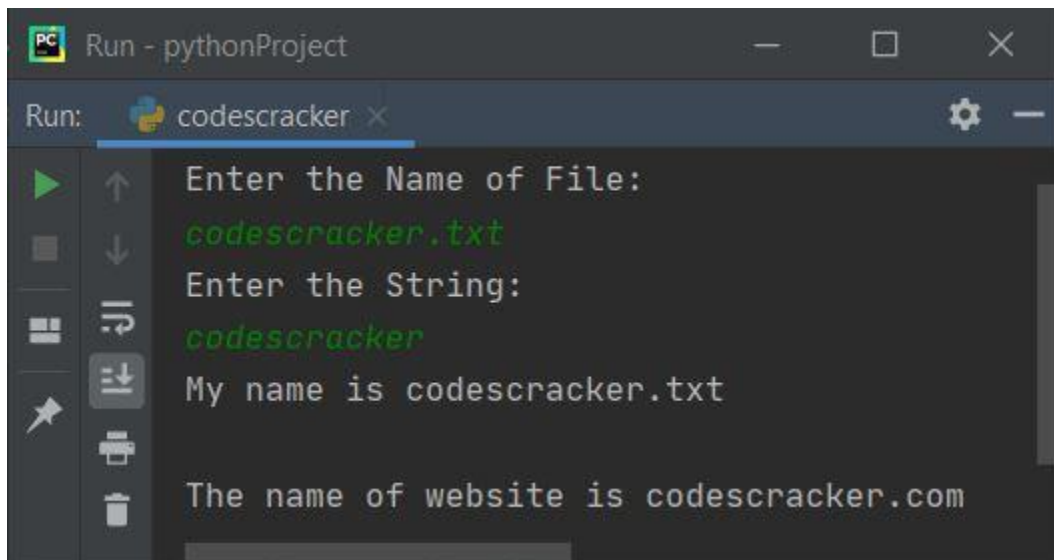
The question is, *write a Python program to print lines that contains the string entered by user.* The program given below is answer to this question:

```
print("Enter the Name of File: ")
fileName = input()
print("Enter the String: ")
text = input()
fileHandle = open(fileName, "r")
lines = fileHandle.readlines()
for line in lines:
    if text in line:
        print(line)
fileHandle.close()
```

Here is its sample run:



Now supply inputs say **codescracker.txt** as name of file, then **codescracker** as string to print all the lines that contains *codescracker*, the given string, in given file (codescracker.txt):

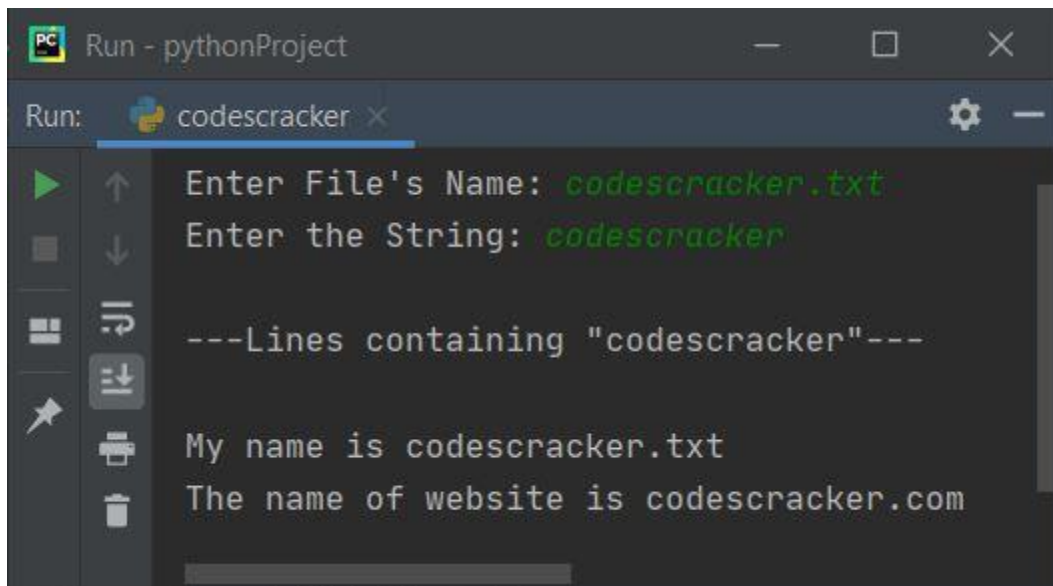


## Modified Version of Previous Program

This program uses *try-except*, an exception handling code to handle the exception such as file doesn't exist, the directory is not accessible etc. like things. Let's have a look at the program and its sample run for clear understanding.

```
print(end="Enter File's Name: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    print(end="Enter the String: ")
    text = input()
    lines = fileHandle.readlines()
    lineList = []
    i = 0
    for line in lines:
        if text in line:
            lineList.insert(i, line)
            i = i+1
    fileHandle.close()
    if i==0:
        print("\n\"" +text+ "\"" is not found in "\"" +fileName+ "\"!")
    else:
        lineLen = len(lineList)
        print("\n---Lines containing \"" +text+ "\"---\n")
        for i in range(lineLen):
            print(end=lineList[i])
        print()
except IOError:
    print("\nThe file doesn't exist!")
```

Here is its sample run with exactly same user input as of previous program's sample run:

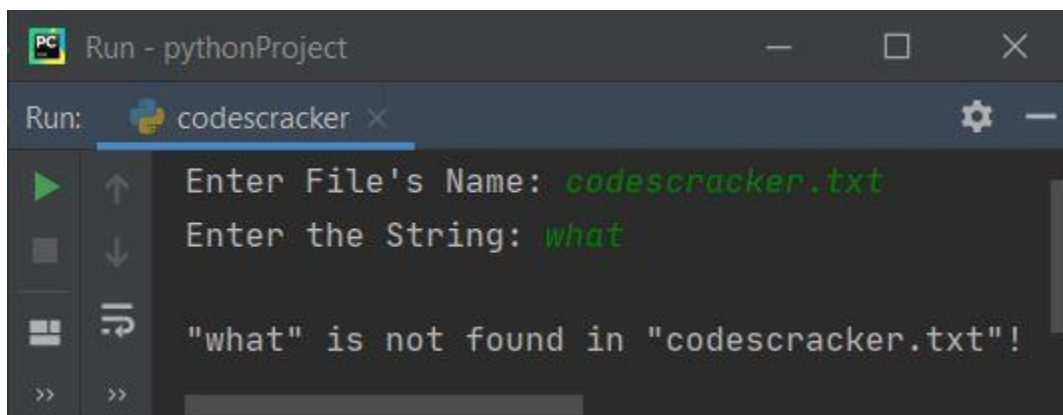


```
Run - pythonProject
Run: codescracker x
Enter File's Name: codescracker.txt
Enter the String: codescracker

---Lines containing "codescracker"---

My name is codescracker.txt
The name of website is codescracker.com
```

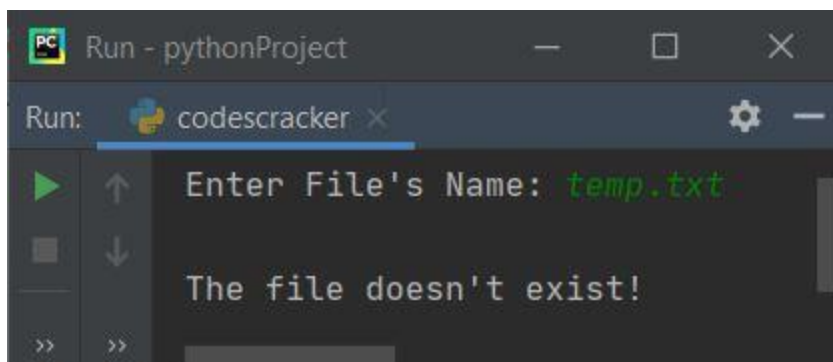
Here is another sample run with user input, **codescracker.txt** as file name, and **what** as string:



```
Run - pythonProject
Run: codescracker x
Enter File's Name: codescracker.txt
Enter the String: what

"what" is not found in "codescracker.txt"!
```

And here is the last sample run of this program, with user input **temp.txt** (a non existing file) as file name:



```
Run - pythonProject
Run: codescracker x
Enter File's Name: temp.txt

The file doesn't exist!
```

As you can see with the above (last sample run), the exception handling code gets the exception and prints the error message.

## Python Program to Delete Specific Line from File

This article is created to cover some programs in Python, that deletes any particular (specific) line from a given file. Here both line's content and file's name must be entered by user.

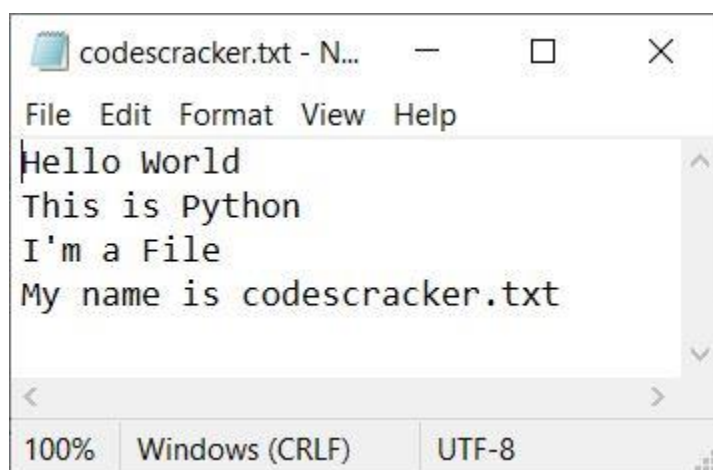
### Things to do Before Program

Because the program given below is used to delete a specific line entered by user from a given file. Therefore we must have to create a file with some content inside it.

For example, create a file say **codescracker.txt** and put the following content in it:

```
Hello World
This is Python
I'm a File
My name is codescracker.txt
```

Save this file in current directory. The directory where the Python program given below (to delete specific line from a file) is saved. Here is the snapshot of opened file, **codescracker.txt**:





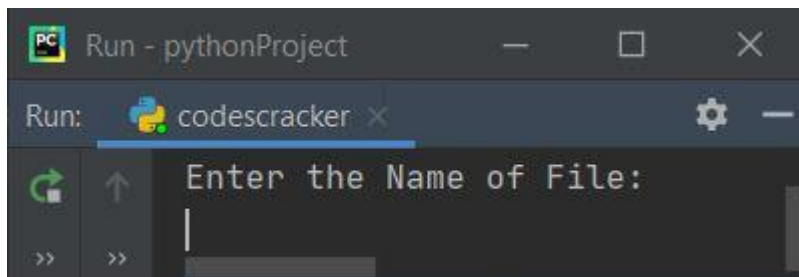
Now let's move on to create a Python program that deletes any specific line from this file through the program.

## Delete a Specific Line from File

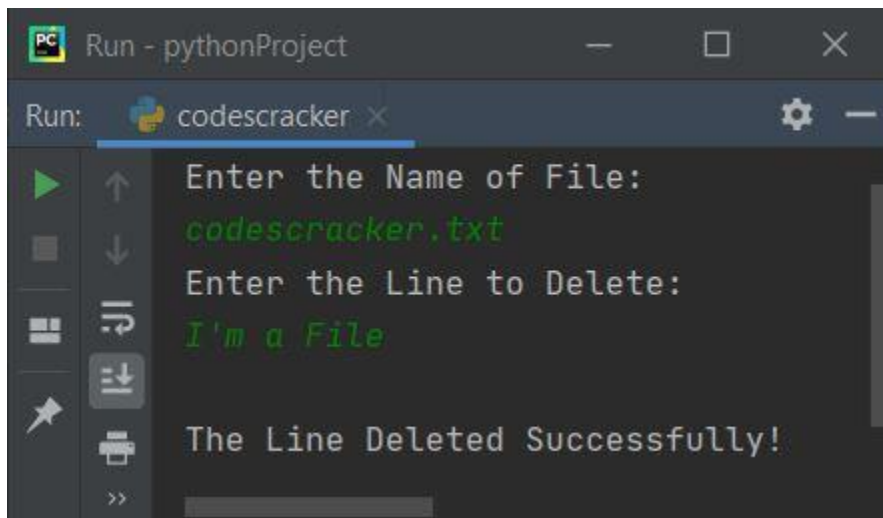
This is very basic version of the Python program to delete any required line from a file. Later on, we've modified the program:

```
print("Enter the Name of File: ")
fileName = input()
print("Enter the Line to Delete: ")
lineText = input()
fileHandle = open(fileName, "r")
lines = fileHandle.readlines()
fileHandle.close()
fileHandle = open(fileName, "w")
for line in lines:
    if line.strip("\n") != lineText:
        fileHandle.write(line)
fileHandle.close()
print("\nThe Line Deleted Successfully!")
```

Here is its sample run:

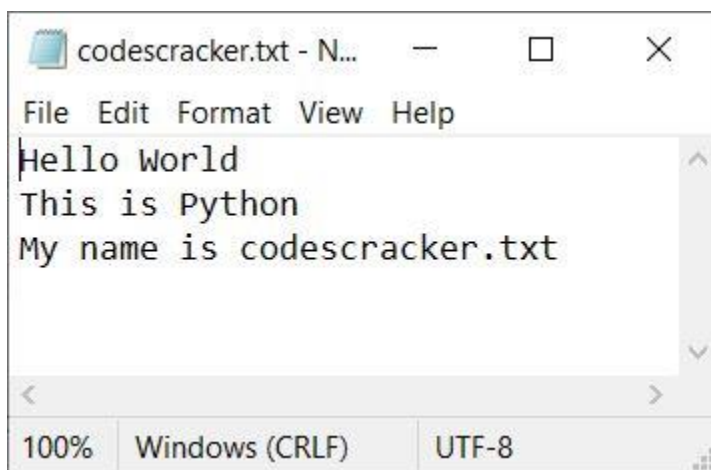


Now enter the name of newly created file say **codescracker.txt** and then enter the line say **I'm a File** to delete this line from the given file:



```
Run - pythonProject
Run: codescracker x
Enter the Name of File:
codescracker.txt
Enter the Line to Delete:
I'm a File
The Line Deleted Successfully!
```

Now if you see the same file (as created earlier), the line gets deleted. Here is the new snapshot of opened file, **codescracker.txt**:



### ***Modified Version of Previous Program***

This program is the modified version of previous program. In this program, I've used **try-except** block to handle exception (if any) thrown by **open()** method while opening the file. The **end** is used to skip inserting an automatic newline.

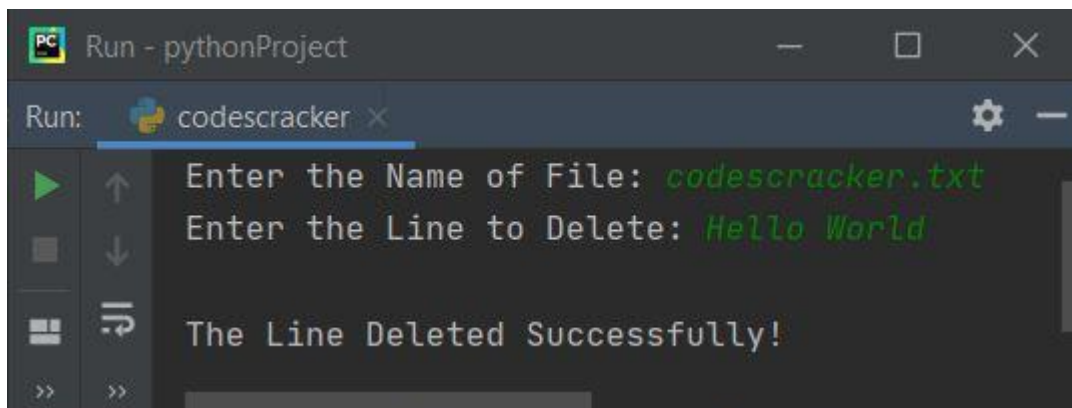
```
print(end="Enter the Name of File: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    lines = fileHandle.readlines()
    fileHandle.close()
try:
    fileHandle = open(fileName, "w")
```

```

print(end="Enter the Line to Delete: ")
lineText = input()
chk=0
for line in lines:
    if line.strip("\n") != lineText:
        fileHandle.write(line)
    else:
        chk=1
fileHandle.close()
if chk==1:
    print("\nThe Line Deleted Successfully!")
else:
    print("\nThe line doesn't exist!")
except IOError:
    print("\nError Occurred while Opening the File (Writing Mode)!")
except IOError:
    print("\nThe File doesn't exist!")

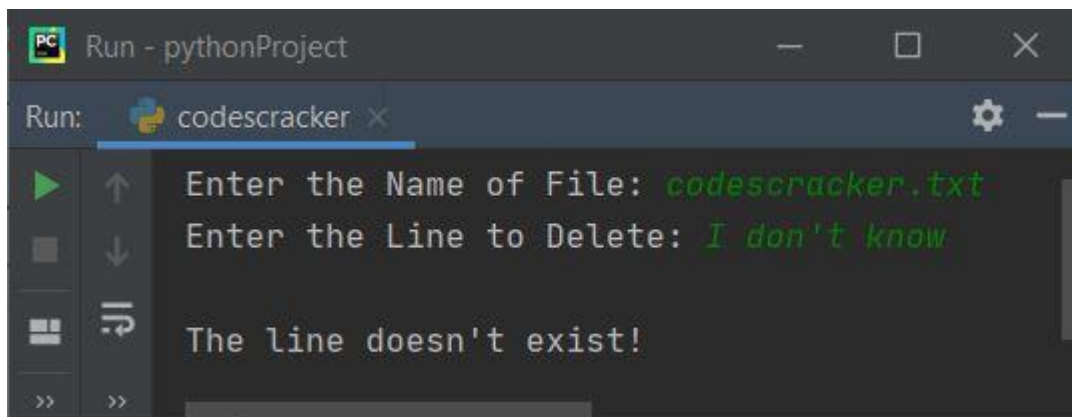
```

Here is its sample run with user input, **codescracker.txt** as file's name and **Hello World** as line to delete:



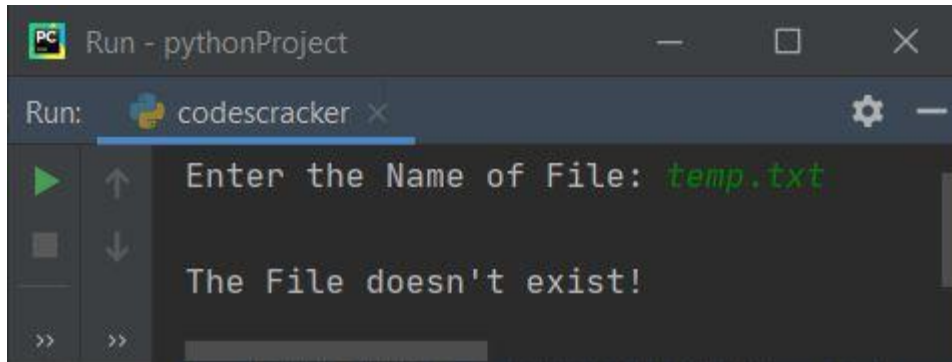
The screenshot shows a terminal window titled "Run - pythonProject" with a tab for "codescracker". The terminal displays the following text: "Enter the Name of File: codescracker.txt", "Enter the Line to Delete: Hello World", and "The Line Deleted Successfully!". The user input is shown in green text.

Here is another sample run:



The screenshot shows a terminal window titled "Run - pythonProject" with a tab for "codescracker". The terminal displays the following text: "Enter the Name of File: codescracker.txt", "Enter the Line to Delete: I don't know", and "The line doesn't exist!". The user input is shown in green text.

And here is the last sample run with user input, **temp.txt** a non-existing file's name:

A screenshot of a Python IDE terminal window. The window title is "Run - pythonProject". The terminal shows a prompt "Enter the Name of File:" followed by the user input "temp.txt" in green. Below that, the output "The File doesn't exist!" is displayed in white. The terminal has a dark background and a light-colored border.

```
Run - pythonProject
Run: codescracker x
Enter the Name of File: temp.txt
The File doesn't exist!
```

## Python Program to Capitalize each Word in a File

In this article, you will learn and get code in Python, to capitalize each and every (all) words in a text file entered by user at run-time.

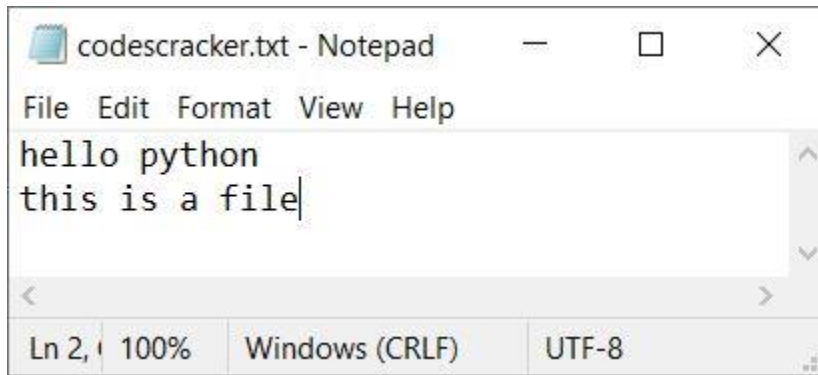
### Things to do Before Program

Because the program given below is used to capitalize each word in a File. That is, after executing the program given below, all words of a file entered by user gets capitalized. Capitalize means, the first letter of a word becomes capital letter.

Therefore, first create a file named **codescracker.txt** with following content:

```
hello python
this is a file
```

Save this file inside the current directory, the directory where the python program to capitalize each word in a file is saved. Here is the snapshot of the opened file, **codescracker.txt**:



Now let's create a Python program to capitalize all words of this file.

## Capitalize Each Word in File

This Python receives the name of file from user at run-time and capitalizes all of its words. The question is, **write a Python program to capitalize each word in a file**. Here is its answer:

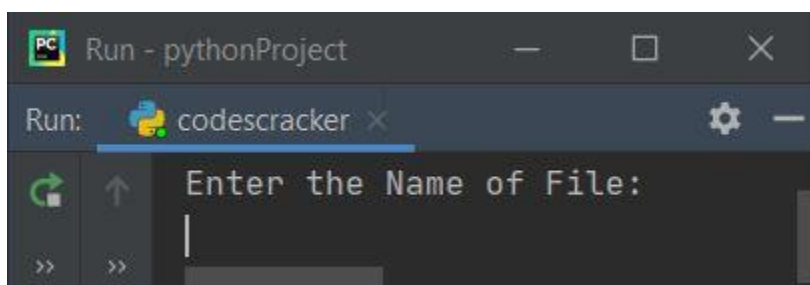
```
totContent = ""
print("Enter the Name of File: ")
fileName = str(input())
fileHandle = open(fileName, "r")

for content in fileHandle:
    newContent = content.title()
    totContent = totContent + newContent

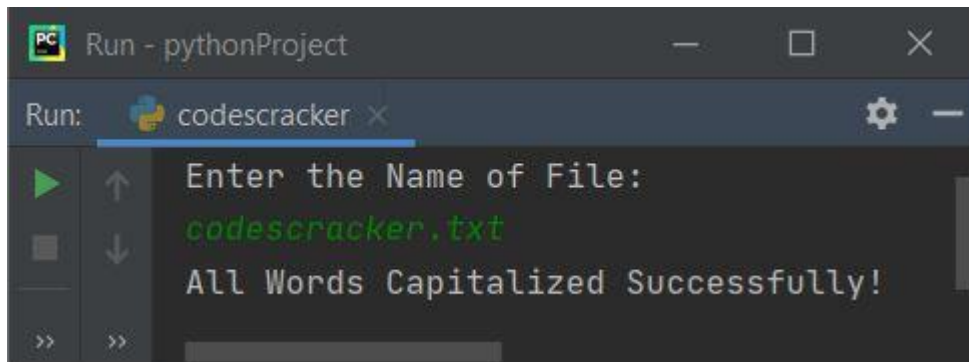
fileHandle.close()
fileHandle = open(fileName, "w")
fileHandle.write(totContent)

print("All Words Capitalized Successfully!")
fileHandle.close()
```

Here is its sample run:

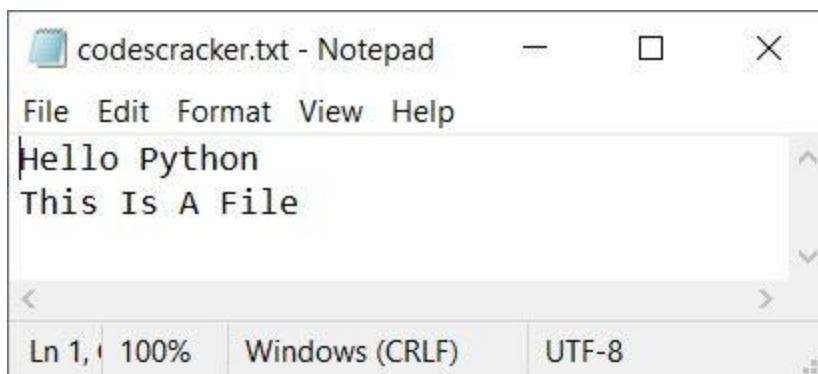


Now enter the name of file say **codescracker.txt** and press `ENTER` key to capitalize first letter of every word in this file. Here is the sample run:



```
Run - pythonProject
Run: codescracker x
Enter the Name of File:
codescracker.txt
All Words Capitalized Successfully!
```

And here is the opened file **codescracker.txt** after executing the above program:



```
codescracker.txt - Notepad
File Edit Format View Help
Hello Python
This Is A File
Ln 1, 100% Windows (CRLF) UTF-8
```

**Note** - The **title()** method converts first letter of each word in capital letter or uppercase.

**Note** - To capitalize each word without using **title()**, refer to [capitalize each word in string](#) article to implement the code manually.

The program works in a way that, the file is opened in reading mode and all of its content gets read. All words gets capitalized and initialized to **totContent** variable. Now file gets closed using **close()** method. The file again gets opened, but in writing mode this time, to put the content of **totContent** in the same file. In this way, we've capitalized all words in a file.

## Modified Version of Previous Program

This program is the modified version of previous program. This program includes error handling code while operating with files. That is, it handles the error when the file entered by user doesn't exist in the directory. Let's have a look at the program:

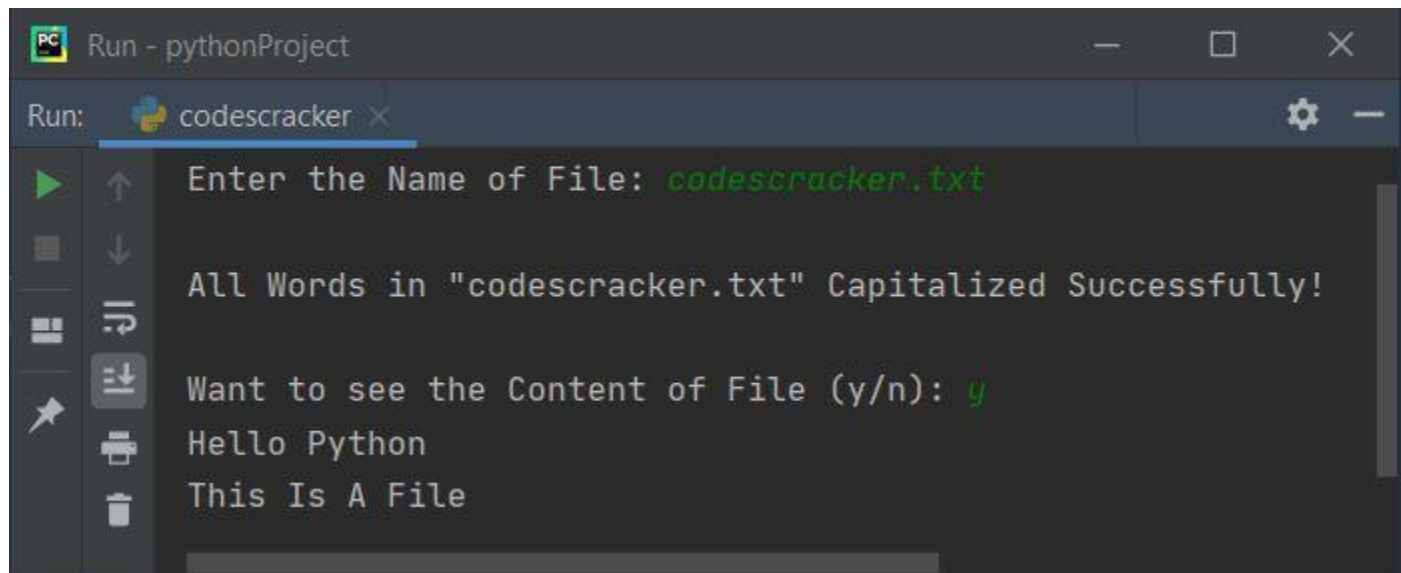
```
totContent = ""
print(end="Enter the Name of File: ")
fileName = str(input())

try:
    fileHandle = open(fileName, "r")
    for content in fileHandle:
        newContent = content.title()
        totContent = totContent + newContent
    fileHandle.close()

    try:
        fileHandle = open(fileName, "w")
        fileHandle.write(totContent)
        print("\nAll Words in \"" + fileName + "\" Capitalized
Successfully!")
        print(end="\nWant to see the Content of File (y/n): ")
        ch = input()
        if ch=='y':
            fileHandle = open(fileName, "r")
            for content in fileHandle:
                print(end=content)
        else:
            print("Exiting...")
            fileHandle.close()
            print()
    except IOError:
        print("Error Occurred!")

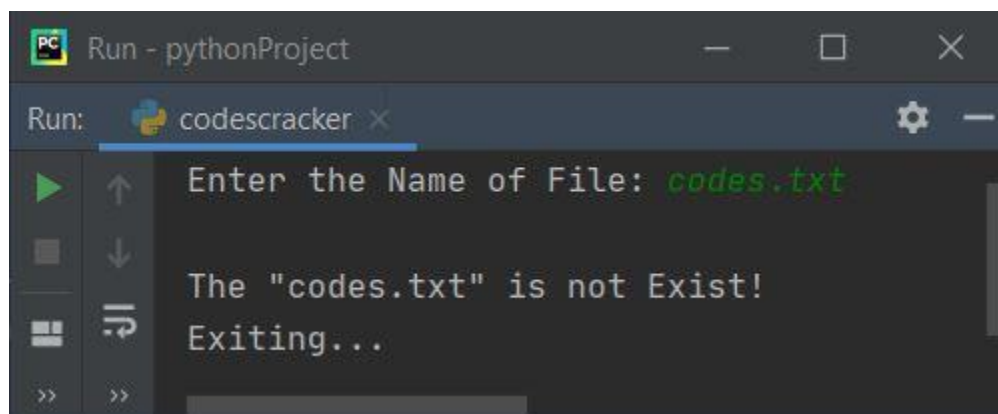
except IOError:
    print("\nThe \"" + fileName + "\" is not Exist!")
    print("Exiting...")
```

Here is its sample run with user input say **codescracker.txt** (existing file):



```
Run - pythonProject
Run: codescracker x
Enter the Name of File: codescracker.txt
All Words in "codescracker.txt" Capitalized Successfully!
Want to see the Content of File (y/n): y
Hello Python
This Is A File
```

And here is another sample run with user input say **codes.txt** (non-existing file):



```
Run - pythonProject
Run: codescracker x
Enter the Name of File: codes.txt
The "codes.txt" is not Exist!
Exiting...
```

## Python Program to Replace Text in a File

This article is created to cover some programs in Python that replaces text in file. Here are the list of programs available in this article:

- Replace specific text with new text in file
- Replace specific text in file and print new content

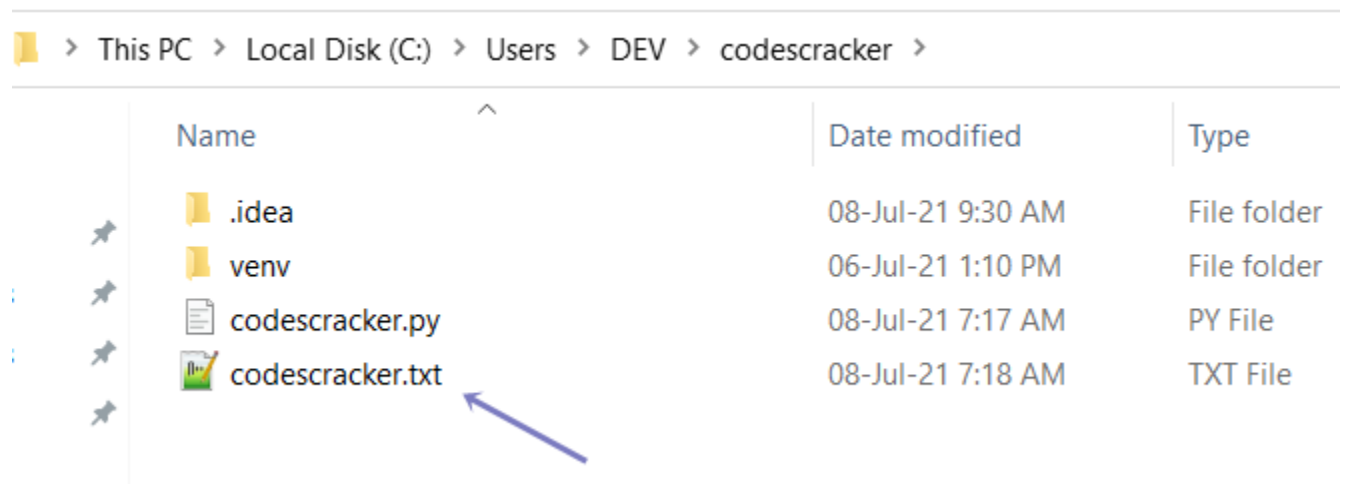


## Things to do before Program

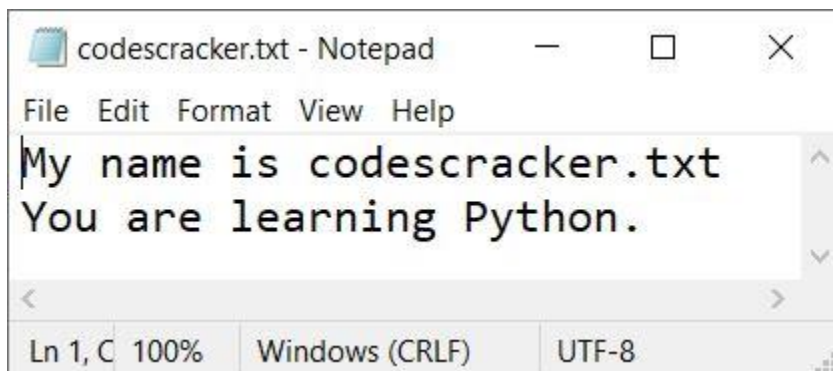
Since the program given below operates on file. Therefore a text file say **codescracker.txt** must be available in the current directory (the folder where the Python program's source code is saved). Therefore create a file with following content:

```
My name is codescracker.txt  
You are learning Python.
```

and save this as **codescracker.txt**. Here is the snapshot of the file created in the current directory:



And here is the snapshot of the opened file named **codescracker.txt**:



Now let's move on and create a program that replaces some text with new text as provided by user at run-time, from the file.

## Replace Text with New Text in File

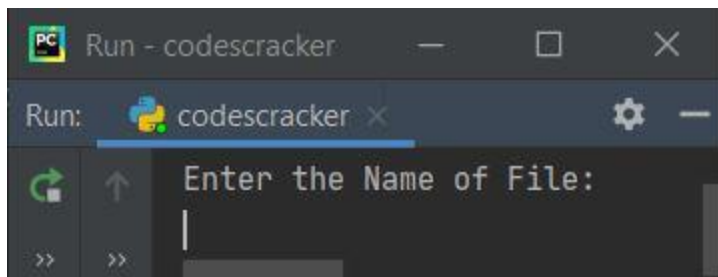
The question is, *write a Python program that replaces any particular text available in a file with a new text. Here the name of file, old text to search, and new text to replace with, must be entered by user at run-time.* The answer to this question is the program given below:

```
print("Enter the Name of File: ")
filename = input()
filehandle = open(filename, "r")
content = filehandle.read()
filehandle.close()

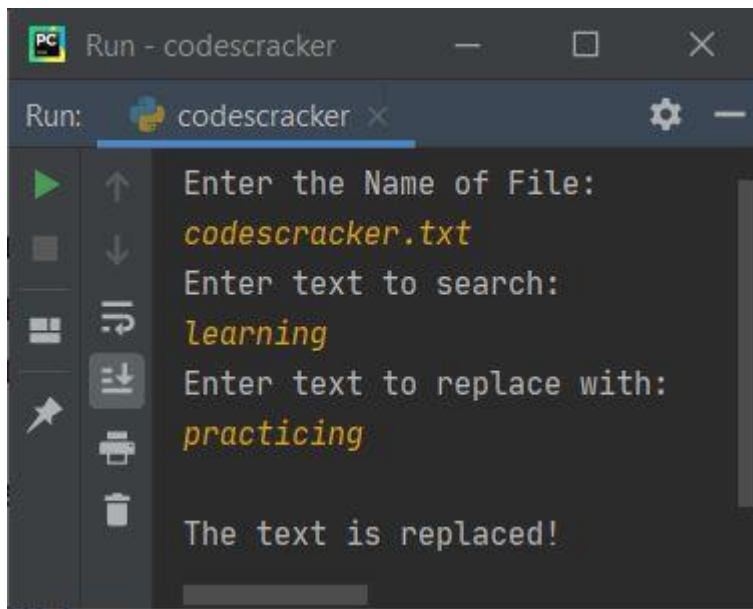
print("Enter text to search: ")
text = input()
print("Enter text to replace with: ")
replace = input()

if text in content:
    content = content.replace(text, replace)
    filehandle = open(filename, "w")
    filehandle.write(content)
    filehandle.close()
    print("\nThe text is replaced!")
else:
    print("Not Found!")
```

The snapshot given below shows the initial output produced by this Python program:



Now supply all the inputs as asked by program and required by you. After providing inputs say **codescracker.txt** as file name, **learning** as text to search, **practicing** as text to replace, here is the sample run with exactly same inputs:

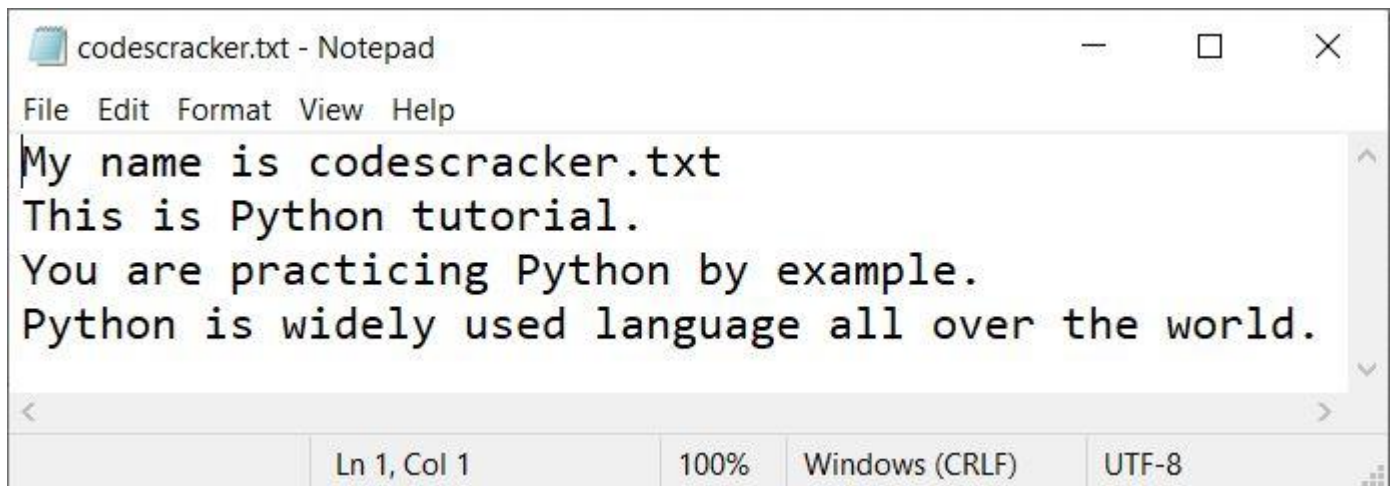


Now if you see the same file as created above, the content of it gets changed. That is, in place of **learning**, you will see the text **practicing** will be available. Here is the new snapshot of the opened file named **codescracker.txt**:



## Replace Text from File and See New Content

Now I've changed the content of file, **codescracker.txt** with content as shown in the following snapshot:



Now let's create the modified version of previous program, that replaces text from given file and prints old and new content of the file:

```
print("Enter File's Name: ", end="")
filename = input()

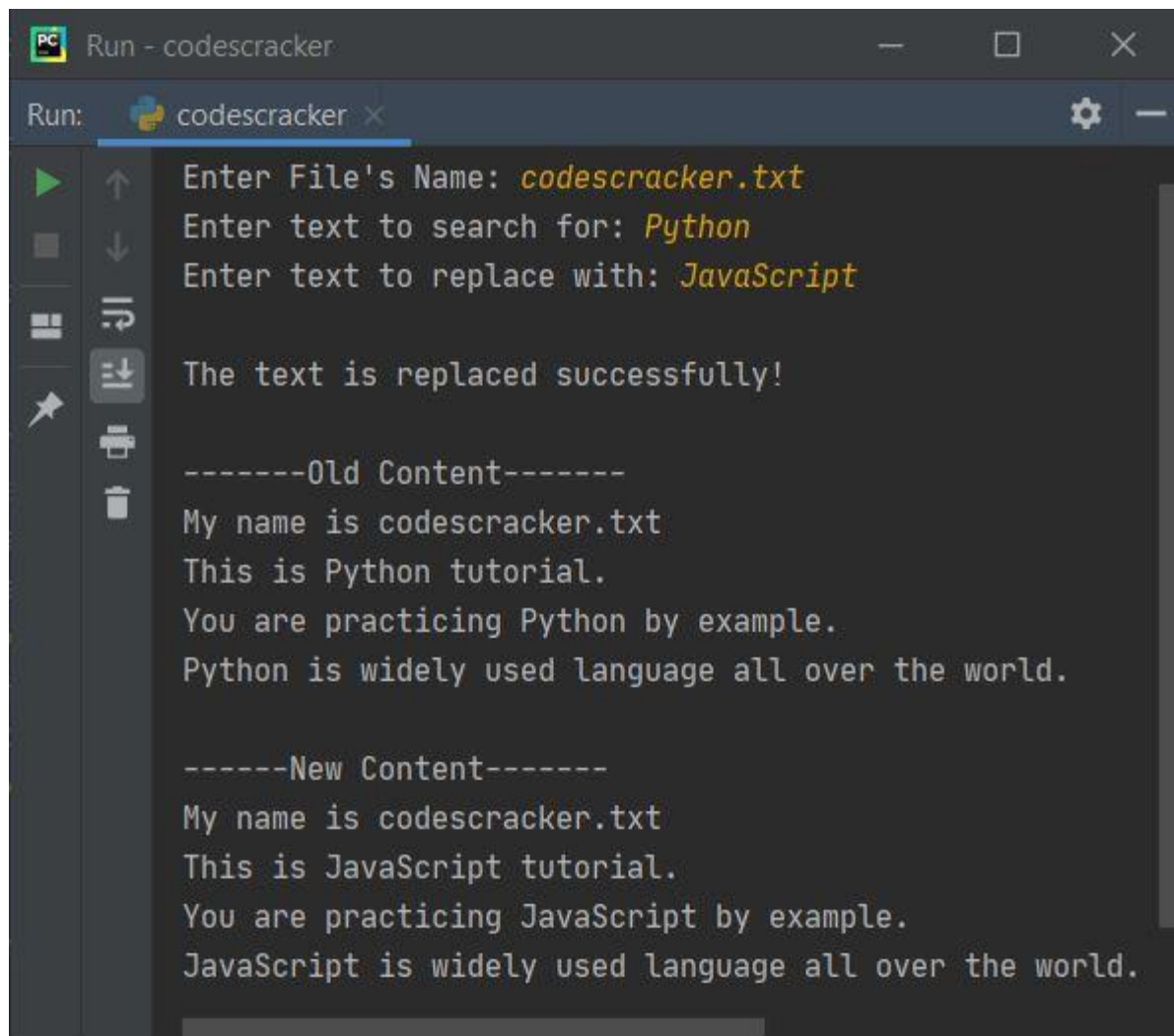
filehandle = open(filename, "r")
content = filehandle.read()
filehandle.close()

print("Enter text to search for: ", end="")
text = input()
print("Enter text to replace with: ", end="")
replace = input()

oldcontent = content

if text in content:
    content = content.replace(text, replace)
    filehandle = open(filename, "w")
    filehandle.write(content)
    filehandle.close()
    print("\nThe text is replaced successfully!")
    print("\n-----Old Content-----")
    print(oldcontent)
    print("\n-----New Content-----")
    print(content)
else:
    print("\nNot Found!")
```

Here is its sample run with user input, **codescracker.txt** as file's name, **Python** as text to search for, **JavaScript** as text to replace with:



```
Run - codescracker

Run: codescracker x

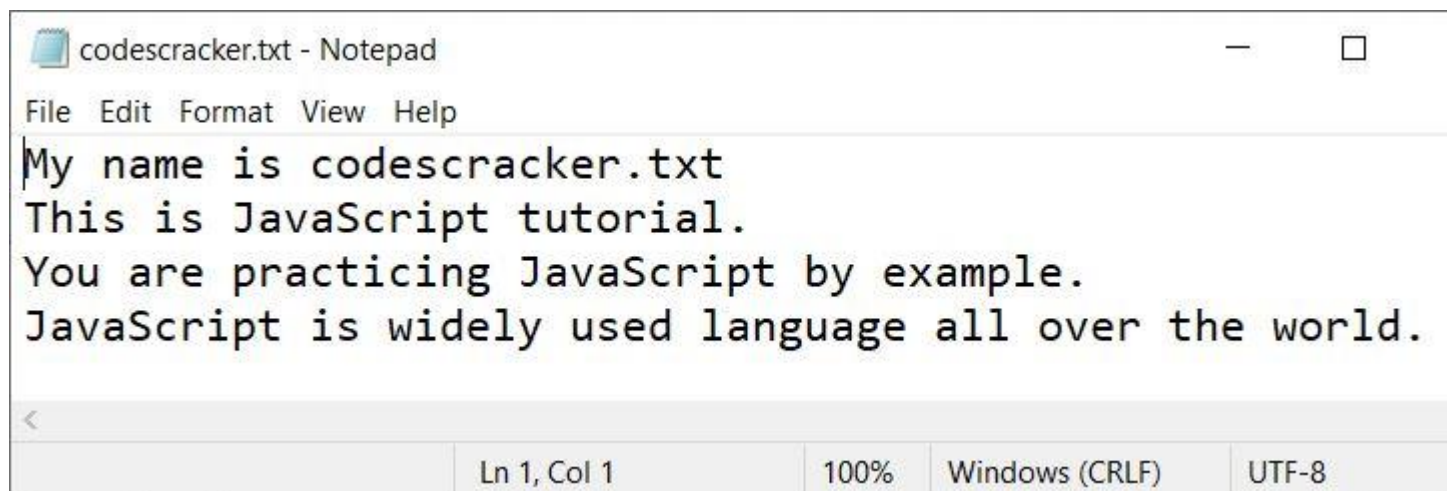
Enter File's Name: codescracker.txt
Enter text to search for: Python
Enter text to replace with: JavaScript

The text is replaced successfully!

-----Old Content-----
My name is codescracker.txt
This is Python tutorial.
You are practicing Python by example.
Python is widely used language all over the world.

-----New Content-----
My name is codescracker.txt
This is JavaScript tutorial.
You are practicing JavaScript by example.
JavaScript is widely used language all over the world.
```

And here is the new snapshot of the same file after executing the above program:



```
codescracker.txt - Notepad
File Edit Format View Help
My name is codescracker.txt
This is JavaScript tutorial.
You are practicing JavaScript by example.
JavaScript is widely used language all over the world.

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

## Python Program to Replace Specific Line in File

This article deals with some programs in Python that replaces specific line from a file. Here are the list of programs covered in this article:

- Replace specific line in a file
- Replace specific line in a file and print the new content of file

### Things to do before Program

Since the program given below operates on file, therefore a file must required before executing the program given below. Therefore, create a file name **codescracker.txt** and put the following content:

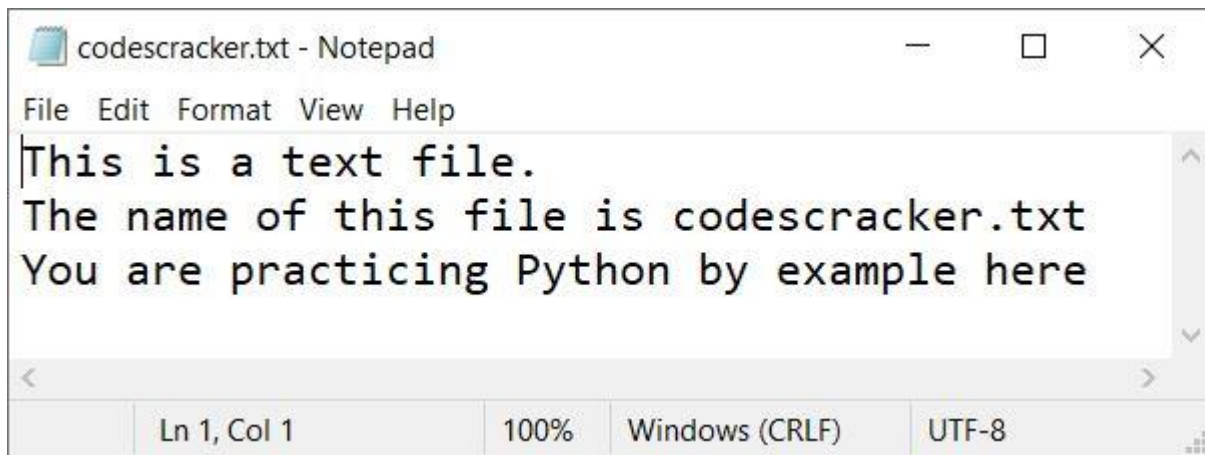
```
This is a text file.  
The name of this file is codescracker.txt  
You are practicing Python by example here
```

inside that file. Save the file inside the current directory. Here current directory means, the folder where the program's source code is saved. Here is the snapshot of the file stored in current directory:

This PC > Local Disk (C:) > Users > DEV > codescracker >

Name	Date modified	Type
.idea	08-Jul-21 12:11 PM	File folder
venv	06-Jul-21 1:10 PM	File folder
codescracker.py	08-Jul-21 12:11 PM	PY File
codescracker.txt	08-Jul-21 12:12 PM	TXT File

and here is the snapshot of opened file named **codescracker.txt**:



Now let's move on and create the program to operate on this file to replace any line with new line.

**Note** - To replace any line, you've to enter the line number and then the content to replace that line number with given content.

## Replace Specific Line from File

The question is, *write a Python program that replaces any specific line from a file. The name of file, line number, and new content for the line number must be entered by user at run-time.* The program given below is answer to this question:

```
print("Enter the name of file: ")
filename = input()

filehandle = open(filename, "r")
listOfLines = filehandle.readlines()
filehandle.close()

print("Enter line number to replace for: ")
lineNo = int(input())
print("Enter new content for line number", lineNo, ": ")
newline = input()
listOfLines[lineNo] = newline

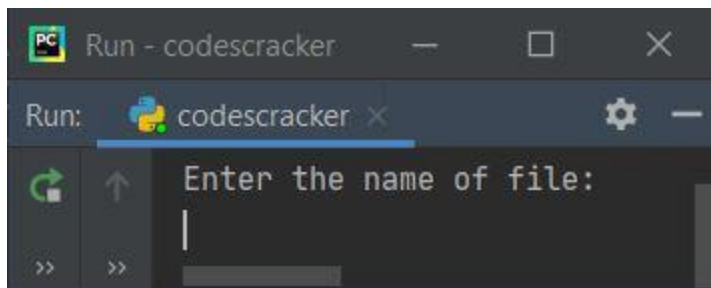
filehandle = open(filename, "w")
filehandle.writelines(listOfLines)
filehandle.close()

print("\nLine replaced successfully!")
```

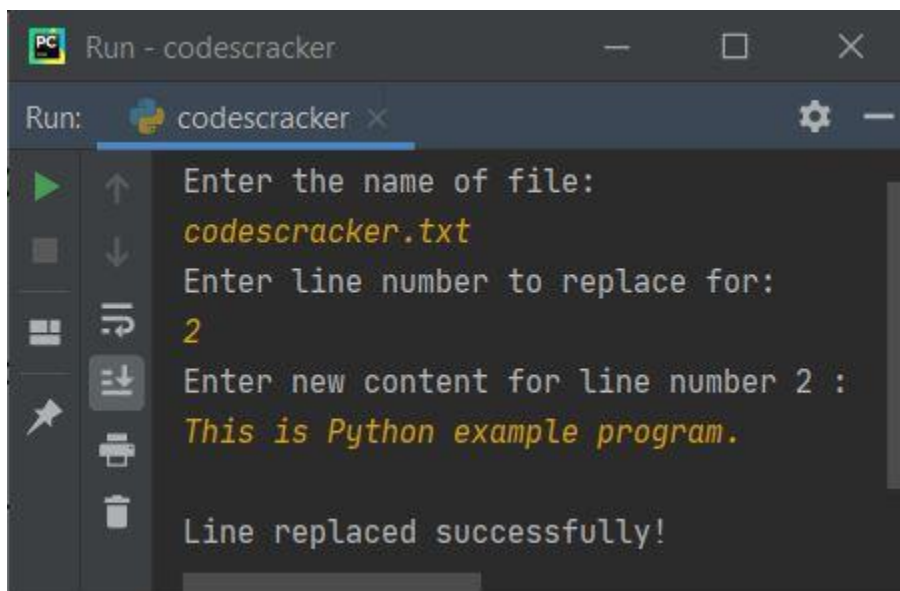


**Note** - Use indexing while entering the line number. That is, if you want to replace first line of text file, then enter 0 as input. If you want to replace second line, then enter 1, and so on.

The snapshot given below shows the initial output produced by above Python program. Let's have a look on it:

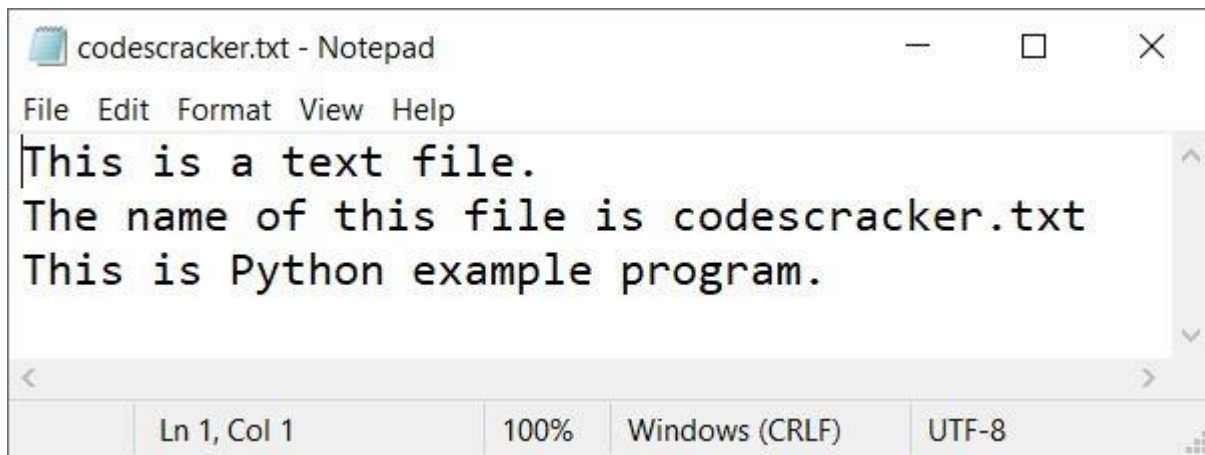


Now provide the inputs, **codescracker.txt** as name of file, **2** as line number, and **This is Python example program.** as content to replace with. Here is its sample run with exactly these inputs:



Now if you'll see the content of file, **codescracker.txt**, its third line gets replaced with new line as entered above. Here is the new snapshot of the same file:





## Replace any Line in File and Print New Content

In this program, I've implemented the code that prints both the old and new content of file after replacing the line. As it looks more interactive.

```
print("Enter File's Name: ", end="")
fname = input()

fhandle = open(fname, "r")
lines = fhandle.readlines()
fhandle.close()

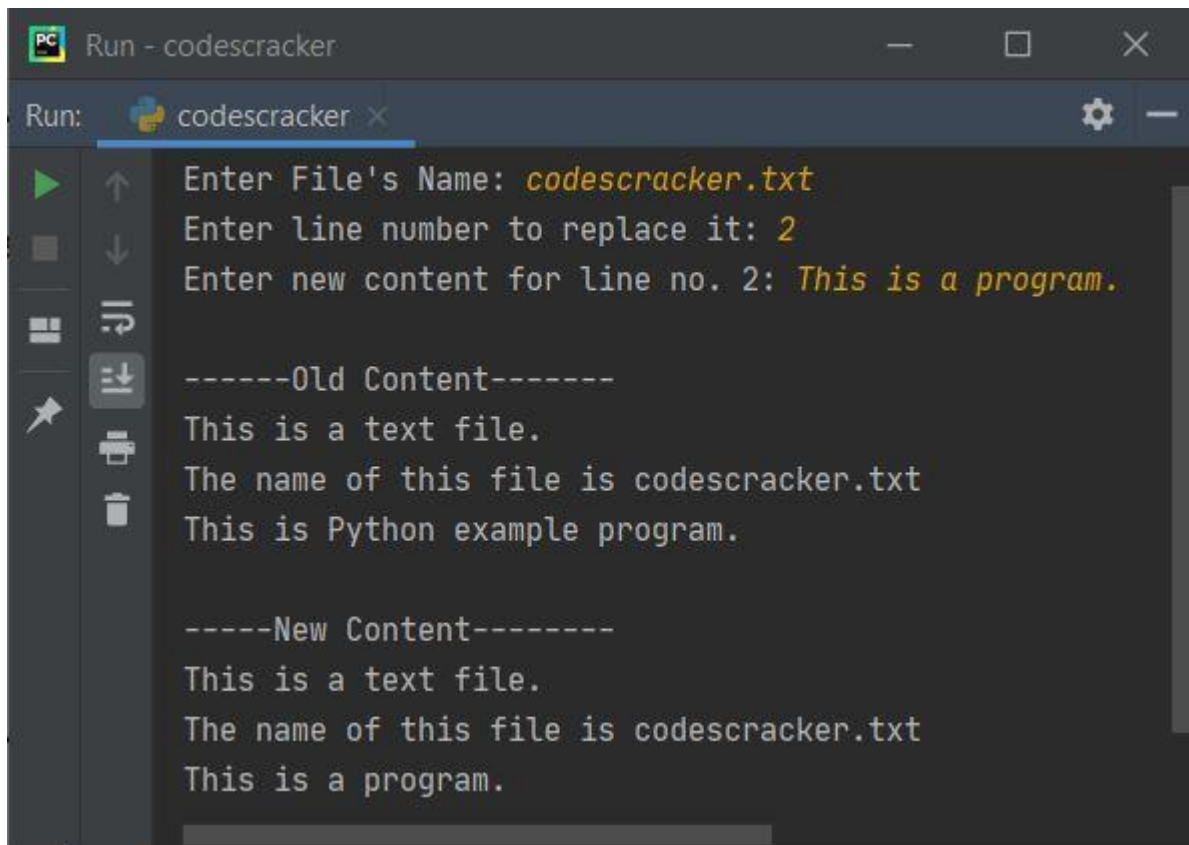
print("Enter line number to replace it: ", end="")
lno = int(input())
print("Enter new content for line no.", lno, "\b: ", end="")
nline = input()

print("\n-----Old Content-----")
content = ""
content = content.join(lines)
print(content)

lines[lno] = nline
fhandle = open(fname, "w")
fhandle.writelines(lines)
fhandle.close()

print("\n-----New Content-----")
content = ""
content = content.join(lines)
print(content)
```

Here is its sample run with user input, **codescracker.txt** as file's name, **2** as line number, **This is a program.** as new line to replace with:



```
Run - codescracker
Run: codescracker x
Enter File's Name: codescracker.txt
Enter line number to replace it: 2
Enter new content for line no. 2: This is a program.

-----Old Content-----
This is a text file.
The name of this file is codescracker.txt
This is Python example program.

-----New Content-----
This is a text file.
The name of this file is codescracker.txt
This is a program.
```

## Python Program to Find Size of a File

This article is created to cover some programs in Python, that find and prints the size of a file entered by user at run-time. Here are the list of programs covered in this article:

- Find size of file using **getsize()** function
- Find size of file without using any pre-defined function
- Find size of file in KB, MB, GB, TB

### Find Size of File using getsize()

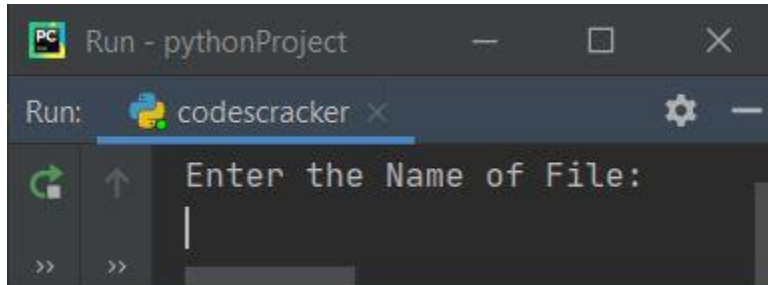
The question is, *write a Python program that finds size of file using **getsize()** method.* The program given below is answer to this question:

```
import os

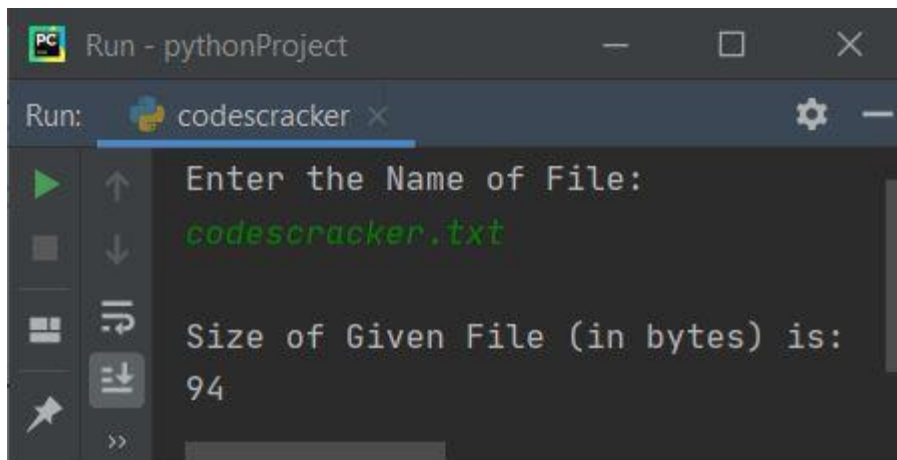
print("Enter the Name of File: ")
fileName = input()
```

```
sizeOfFile = os.path.getsize(fileName)
print("\nSize of Given File (in bytes) is:")
print(sizeOfFile)
```

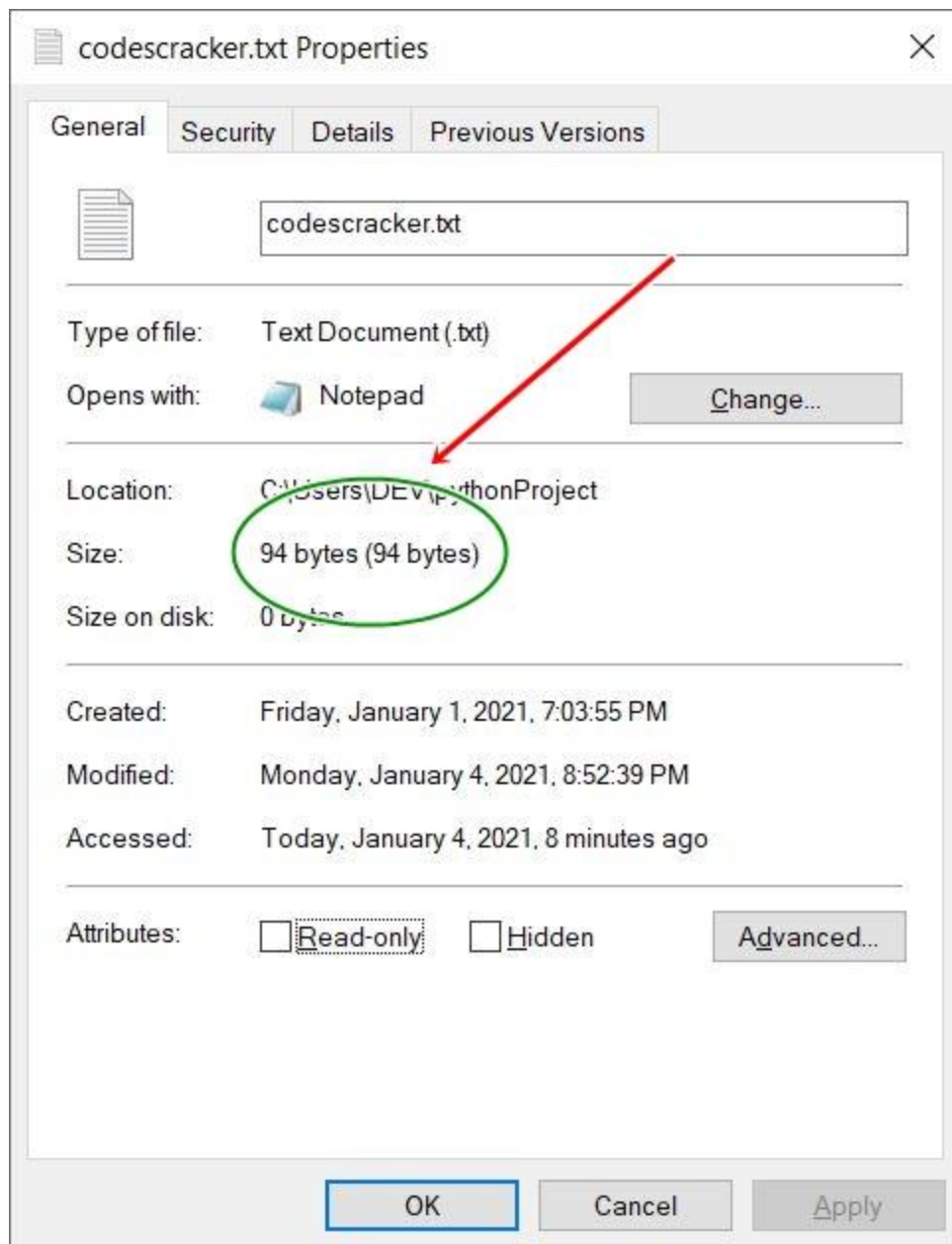
Here is its sample run:



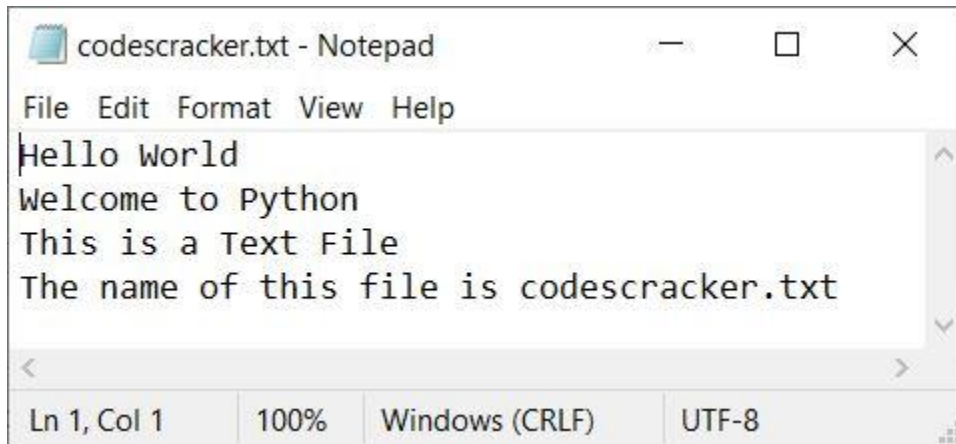
Now supply the input as name of file to find and print its size. Here is its sample output with user input **codescracker.txt**. This file already exists in the current directory. That is, the folder where the above program (source code) is also saved:



Here is the window that shows the details along with size of the same file as provided in sample run:



And the snapshot given below shows the content of same file:



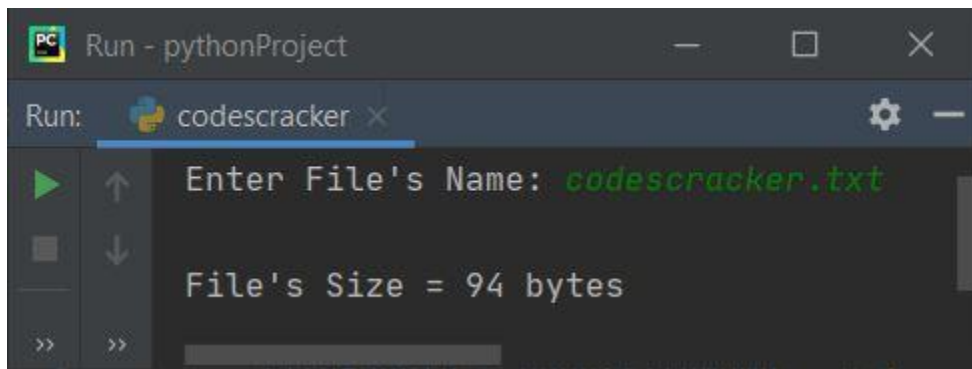
**Note** - As you can see from the content of file, **codescracker.txt**. There are **92** characters (with spaces) and 2 newlines (that can not be seen). The second and third line of texts has newlines. That is after first line, there is a newline, and then after second line, there is a newline.

## Find Size of File with User-based Code

The question is, *write a program in Python that find and prints the size of file without using any pre-defined function*. The following program is the answer to this question:

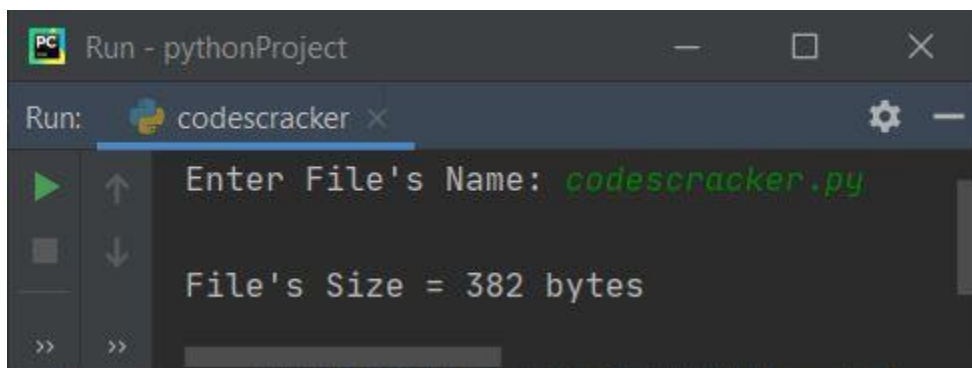
```
print(end="Enter File's Name: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    sizeOfFile = 0
    for line in fileHandle.readlines():
        sizeOfFile = sizeOfFile + (len(line)+1)
    sizeOfFile = sizeOfFile-1
    print("\nFile's Size = " +str(sizeOfFile)+ " bytes")
except IOError:
    print("\nThe File doesn't exist!")
```

Here is its sample run with same file name say **codescracker.txt**:



```
Run - pythonProject
Run: codescracker x
Enter File's Name: codescracker.txt
File's Size = 94 bytes
```

Here is another sample run with user input, **codescracker.py** (the name of current Python program source code's file):



```
Run - pythonProject
Run: codescracker x
Enter File's Name: codescracker.py
File's Size = 382 bytes
```

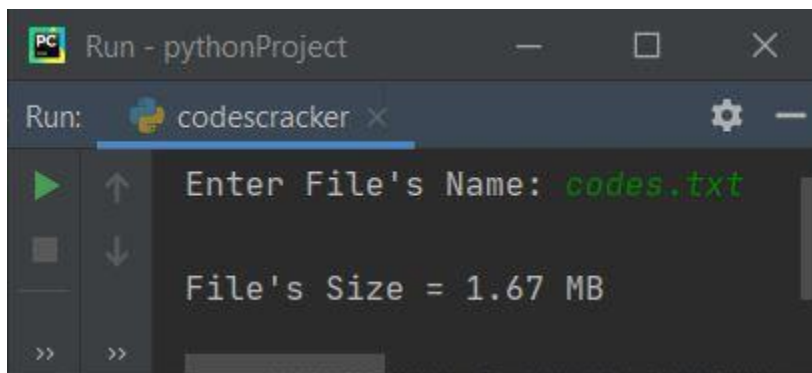
## Find Size of File in KB, MB, GB, TB

Now this is the final program of this article, that find and prints the size of file in KB, MB, GB, TB. Let's have a look at the program and its sample output:

```
print(end="Enter File's Name: ")
fileName = input()
try:
    fileHandle = open(fileName, "r")
    sizeOfFile = 0
    for line in fileHandle.readlines():
        sizeOfFile = sizeOfFile + (len(line)+1)
    sizeOfFile = sizeOfFile-1
    kb = sizeOfFile/1024
    if kb>1:
        mb = kb/1024
        if mb>1:
            gb = mb/1024
            if gb>1:
                tb = gb/1024
                if tb>1:
                    tb = "{0:.2f}".format(tb)
                    print("\nFile's Size = " +str(tb)+ " TB")
```

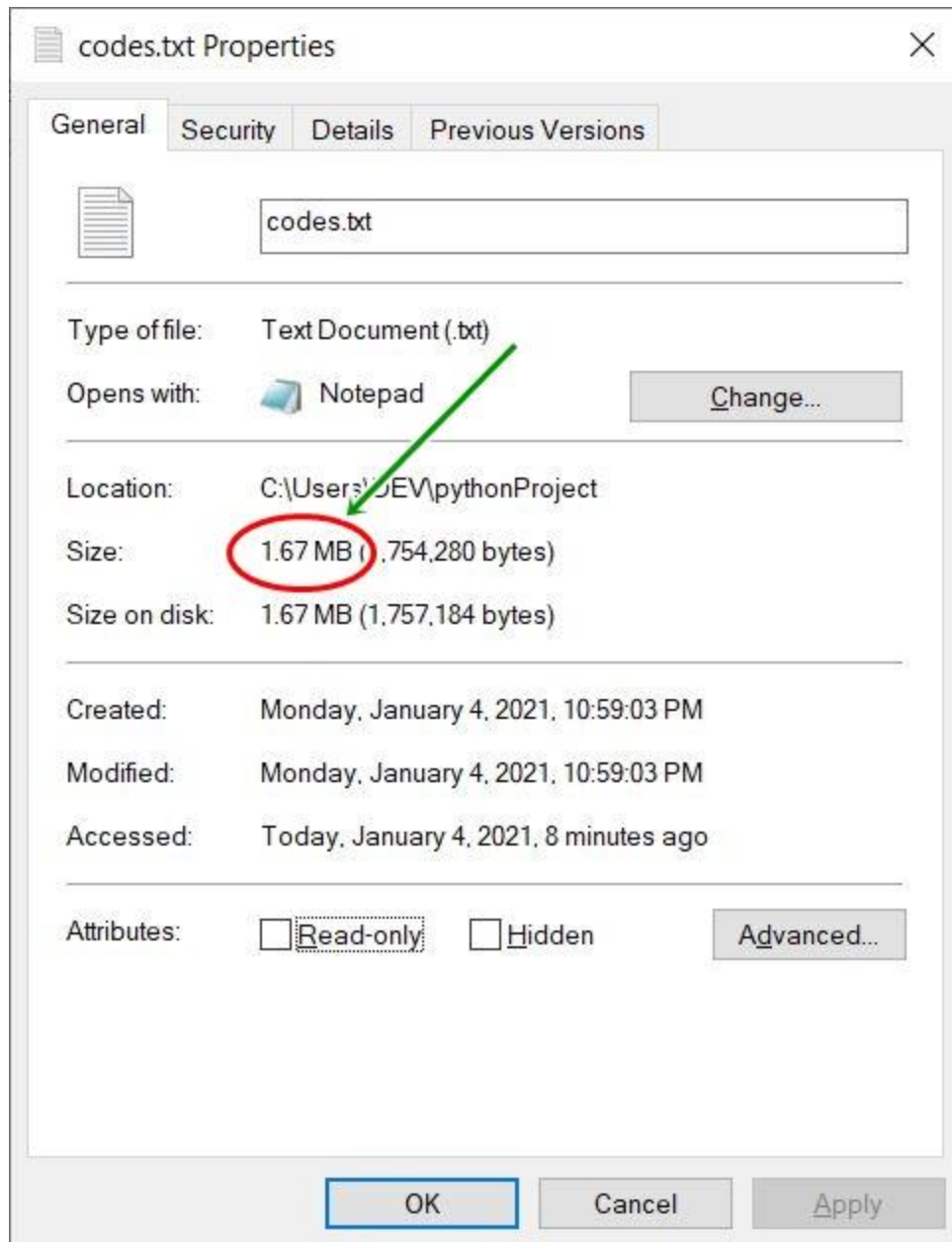
```
    else:
        gb = "{0:.2f}".format(gb)
        print("\nFile's Size = " +str(gb)+ " GB")
    else:
        mb = "{0:.2f}".format(mb)
        print("\nFile's Size = " +str(mb)+ " MB")
    else:
        kb = "{0:.2f}".format(kb)
        print("\nFile's Size = " +str(kb)+ " KB")
    else:
        print("\nFile's Size = " +str(sofarFile)+ " bytes")
except IOError:
    print("\nThe File doesn't exist!")
```

Here is its sample run with user input, **codes.txt**:



```
Run - pythonProject
Run: codescracker x
> Enter File's Name: codes.txt
File's Size = 1.67 MB
```

Here is the snapshot of properties of **codes.txt** file:



## Python Program to List Files in Directory

This article is created to cover some programs in Python, that list and prints files from directory. Here are the list of programs available in this article:

- List and print all files in a current directory
- List files with only particular extension from current directory
- List files from any directory provided by user



## List and Print all Files in Current Directory

To list all files present in current directory, we've to use **glob()** method. This method is defined in **glob** module, therefore before using it, it must be imported. So that, we can call the **glob()** method defined in that module using module name followed by **dot (.)** operator and then method name, that is **glob.glob()**

The current directory means the directory where the source code of Python program is saved.

The argument of **glob.glob()** defines what type of file to list. That is, if user provides **\*.\***, then it indicates to list all files. The **\*** indicates to all type, that is the star (**\***) before dot (**.**) indicates to fetch all files and the star (**\***) after dot (**.**) indicates to fetch all extension.

Now let's create the program to do the task. But before going through program, here is the snapshot of the current directory folder, in my case, that contains three files:

> This PC > Local Disk (C:) > Users > DEV > codescracker		
Name	Date modified	Type
 .idea	19-Feb-21 2:34 PM	File f
 venv	23-Jan-21 4:17 PM	File f
 codes.txt	26-Jan-21 11:00 PM	Text I
 codescracker.py	19-Feb-21 2:34 PM	PY Fil
 cracker.txt	27-Jan-21 12:00 AM	Text I

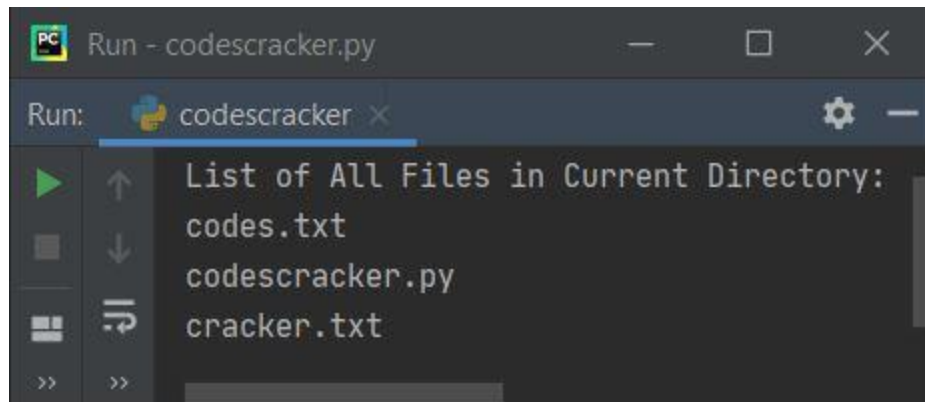
Now let's create the Python program to list and print all these files through program:

```
import glob

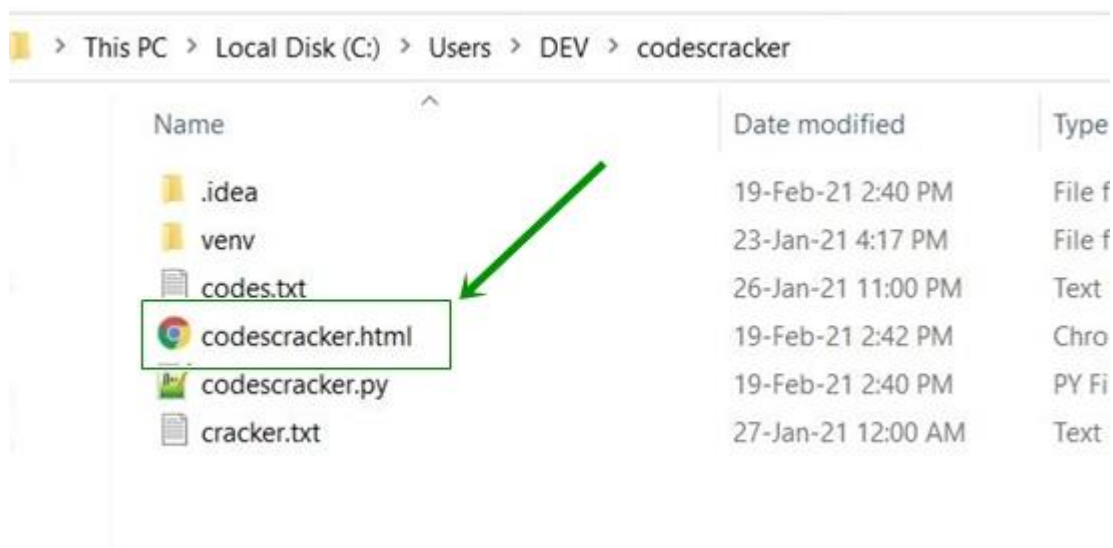
print("List of All Files in Current Directory:")
```

```
for file in glob.glob("*."):
    print(file)
```

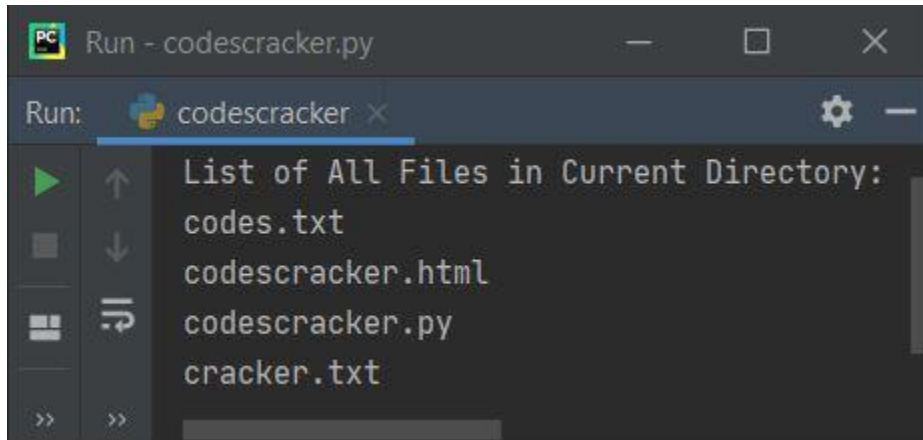
This program produces the output like shown in the snapshot given below. Whatever (files) stored in your current directory (the directory where this program is saved) gets listed and printed on output:



**Note** - Now if you create or add another file say **codescracker.html** inside the same directory. Here is the snapshot of the current directory folder after creating another file:



Now if you re-run the above program, then this time, the output includes a new file name, that is **codescracker.html** as its output like shown in the snapshot given below:



```
Run - codescracker.py
Run: codescracker
List of All Files in Current Directory:
codes.txt
codescracker.html
codescracker.py
cracker.txt
```

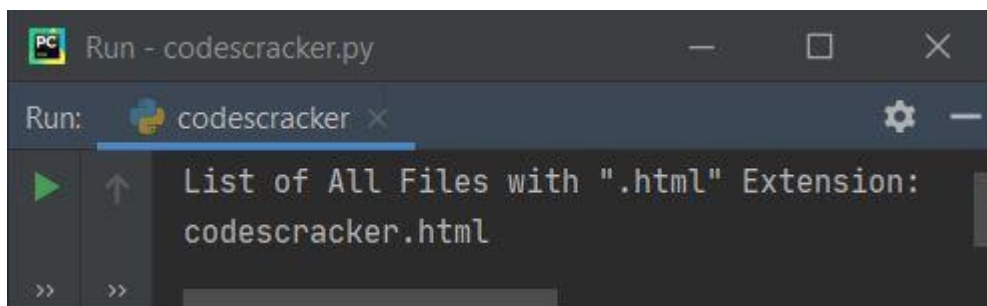
## List Files in Directory with Extension

To list and print files having particular extension say **.html**, then replace the second star (\*) with **html**. Or to list only textual files (all files with **.txt** extension), replace **\*.\*** with **\*.txt**. Here is the complete program to list files with only particular extension (.html):

```
import glob

print("List of All Files with \".html\" Extension:")
for file in glob.glob("*.html"):
    print(file)
```

Here is its sample output:



```
Run - codescracker.py
Run: codescracker
List of All Files with ".html" Extension:
codescracker.html
```

There is only one file available in the current directory with **.html** extension.

## List Files with Extension Provided by User

This is the modified version of previous program. This program receives input from user as extension to list required extension's files. Enter .\* to list all files without mattering about extension:

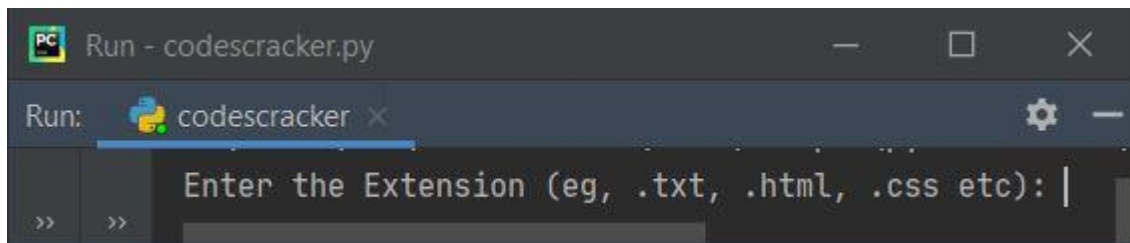
```
import glob

print("Enter the Extension (eg, .txt, .html, .css etc): ", end="")
e = input()

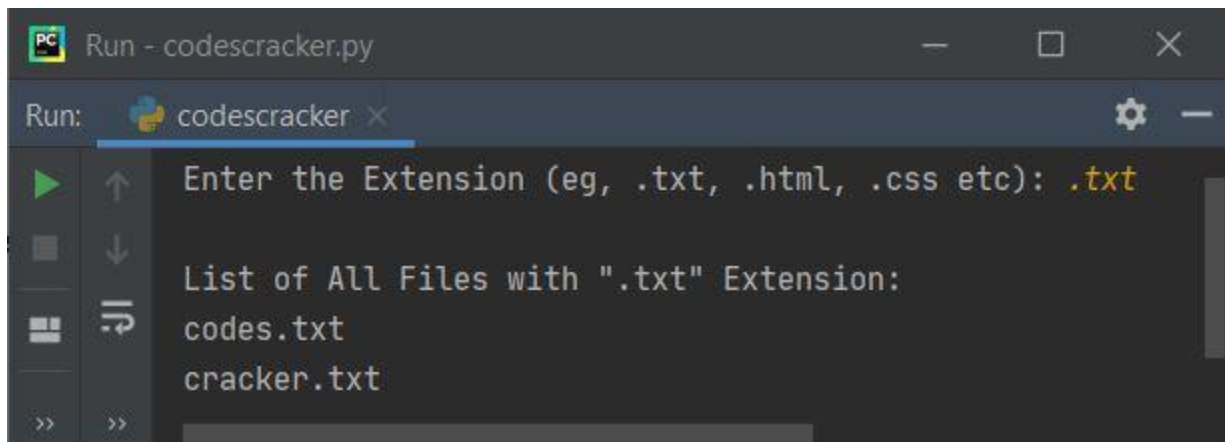
fileslist = []
for file in glob.glob("*"+e):
    fileslist.append(file)

if len(fileslist)>0:
    print("\nList of All Files with \"" + e+ "\"" Extension:")
    for f in fileslist:
        print(f)
else:
    print("\nNot found with \"" + e+ "\"" extension!")
```

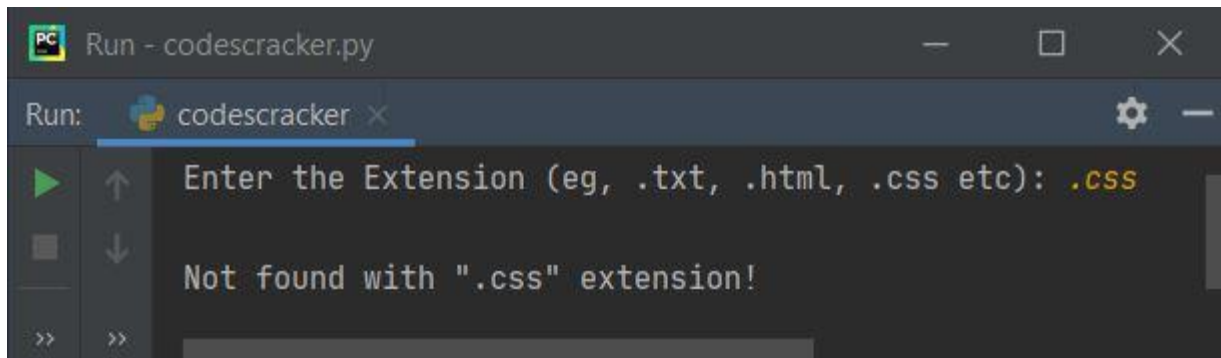
Here is its sample run. This is the initial output:



Now supply the input say .txt as extension to list and print all files having .txt extension like shown in the snapshot given below:



And here is another sample run with user input **.css** (no file available with this extension, in current directory):



```
Run - codescracker.py
Run: codescracker x
Enter the Extension (eg, .txt, .html, .css etc): .css
Not found with ".css" extension!
```

## List and See Files from any Directory by User

Now let's modify the above program and allow user to enter their own required directory to list all the files from it. The **chdir()** (**change directory**) is used to change the directory. That is, whatever the directory is provided as its argument. The current working directory gets changed to that directory. This method is defined in **os** module.

```
import glob, os

print("Enter Full Path of a Folder: ", end="")
path = input()
os.chdir(path)

print("\n1. List all Files ?")
print("2. List all Files with Particular Extension ?")
print("Enter Your Choice (1 or 2): ", end="")
try:
    ch = int(input())
    if ch==1:
        print("\nList of All Files:")
        for file in glob.glob("*."):
            print(file)
    elif ch==2:
        print("\nEnter the Extension (eg, .txt, .html, .css etc): ", end="")
        e = input()

        fileslist = []
        for file in glob.glob("*" + e):
            fileslist.append(file)

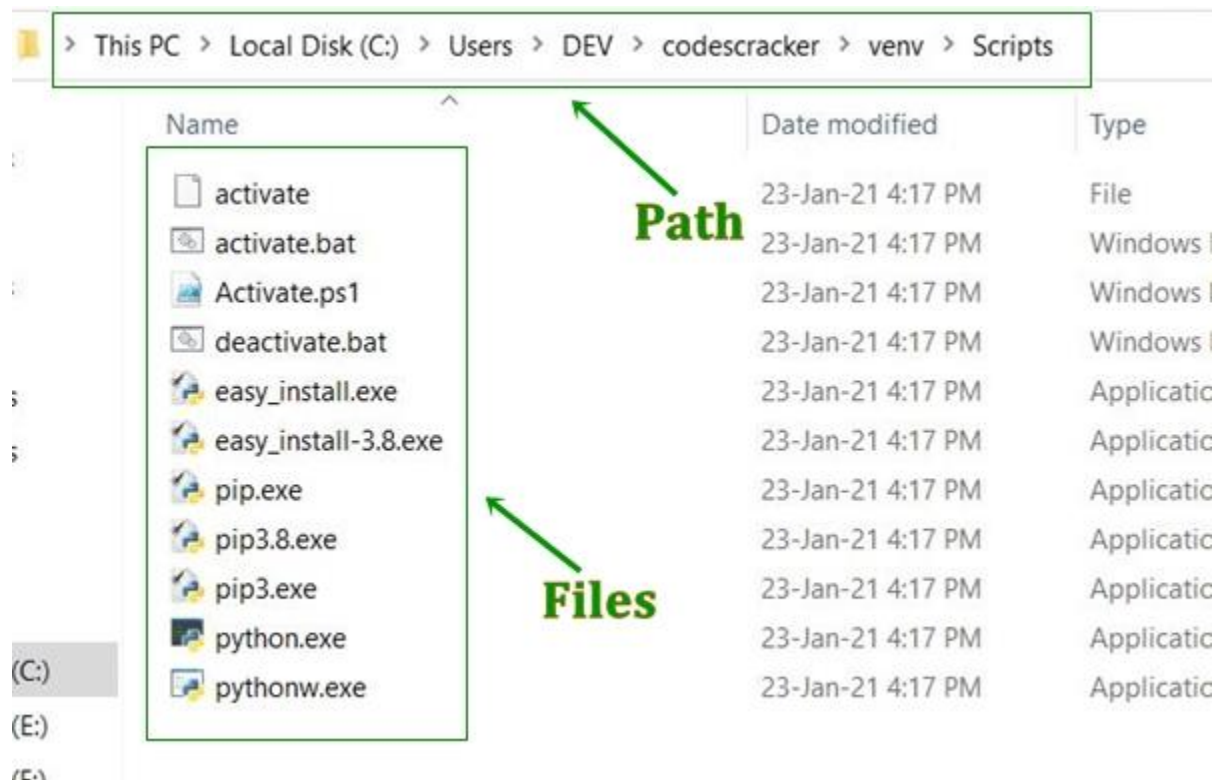
        if len(fileslist) > 0:
            print("\nList of All Files with \"" + e + "\" Extension:")
            for f in fileslist:
```

```

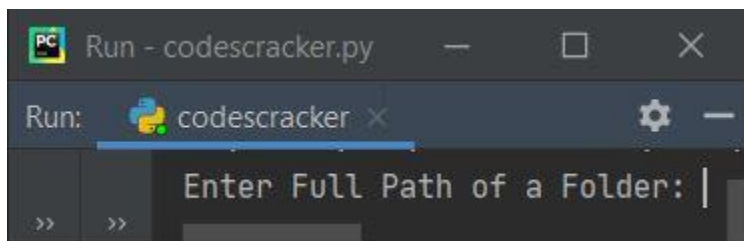
        print(f)
    else:
        print("\nNot found with \"" + e + "\" extension!")
    else:
        print("\nInvalid Choice!")
except ValueError:
    print("\nInvalid Input!")

```

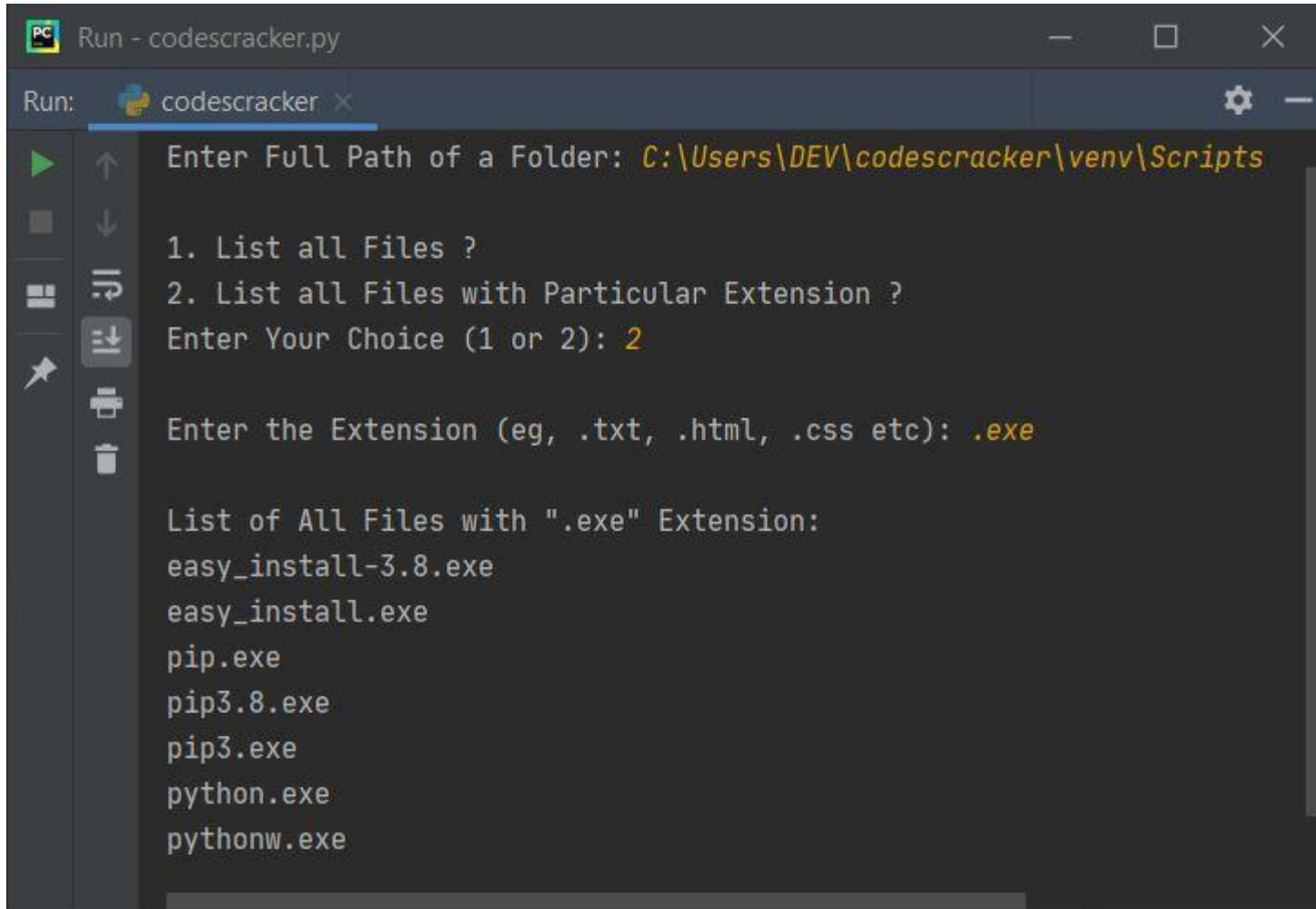
I'm going to operate with a directory, where the folder named **Scripts** contains these files:



Copy the full path of this folder (or any folder you want), then execute the above program and just paste the path as first input asked by the program like shown in the snapshot given below:



Now paste the folder's path say **C:\Users\DEV\codescracker\venv\Scripts** in my case, then entered the choice as **2**, and then **.exe** as extension to list all files from that directory with **.exe** extension. Here is the complete sample run:



```
Run - codescracker.py
Run: codescracker
Enter Full Path of a Folder: C:\Users\DEV\codescracker\venv\Scripts
1. List all Files ?
2. List all Files with Particular Extension ?
Enter Your Choice (1 or 2): 2
Enter the Extension (eg, .txt, .html, .css etc): .exe

List of All Files with ".exe" Extension:
easy_install-3.8.exe
easy_install.exe
pip.exe
pip3.8.exe
pip3.exe
python.exe
pythonw.exe
```

## Python Program to Delete a File

This article deals with some programs created in Python, that deletes a file entered by user at run-time. Here are the list of programs:

- Delete a File from Current Directory
- Delete File if Exists
- Delete any File from any Directory. The Name of File and Directory must be entered by user

Since the program given below deletes a file from the current directory. The current directory means, the folder where the Python source (given below) to delete file is saved. Here is the snapshot of the folder (current directory) before executing the program given below:

> This PC > Local Disk (C:) > Users > DEV > codescracker			
	Name	Date modified	Type
★	.idea	29-Jan-21 10:43 AM	Fi
★	venv	23-Jan-21 4:17 PM	Fi
★	codes.txt	26-Jan-21 11:00 PM	Fi
★	codescracker.py	29-Jan-21 10:43 AM	Je
★	codescracker.txt	26-Jan-21 9:57 PM	Fi
	cracker.txt	27-Jan-21 12:00 AM	Fi

As you can see that a file named **codescracker.txt** is available in this folder. So I'll delete this file using the program given below. Let's get started.

## Delete a File from Current Directory

To delete a file from current directory in Python, you have to ask from user to enter the name of file, then use **os.remove()** to do the job like show in the program given below:

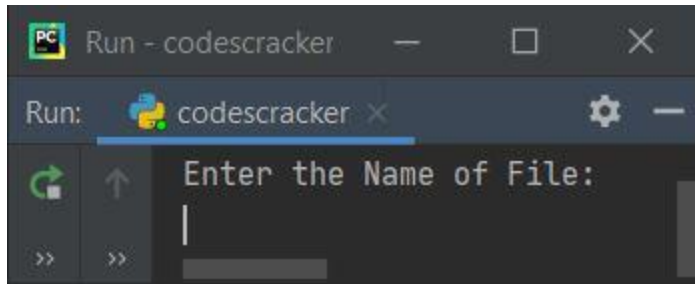
```
import os

print("Enter the Name of File: ")
filename = input()

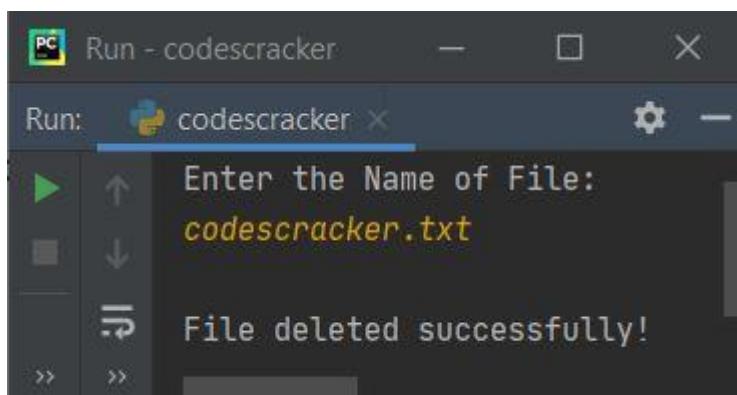
os.remove(filename)
print("\nFile deleted successfully!")
```

Here is the initial output produced by this Python program:





Now supply the input say **codescracker.txt** as name of file to delete, press `ENTER` key to delete it from the current directory. Here is the output produced after providing these inputs:



Here is the snapshot of the current directory after executing the program:

> This PC > Local Disk (C:) > Users > DEV > codescracker		
Name		Date modified
	.idea	29-Jan-21 10:48 AM
	venv	23-Jan-21 4:17 PM
	codes.txt	26-Jan-21 11:00 PM
	codescracker.py	29-Jan-21 10:48 AM
	cracker.txt	27-Jan-21 12:00 AM

As you can see from the above screenshot, the file **codescracker.txt** is not present inside the directory, as the file gets deleted using the execution and sample run of above program.

**Note** - The **os** module allows us to work with operating system using its predefined functions.

**Note** - The **os.remove()** is used to remove or delete a file provided as its argument.

## Delete File if Exists

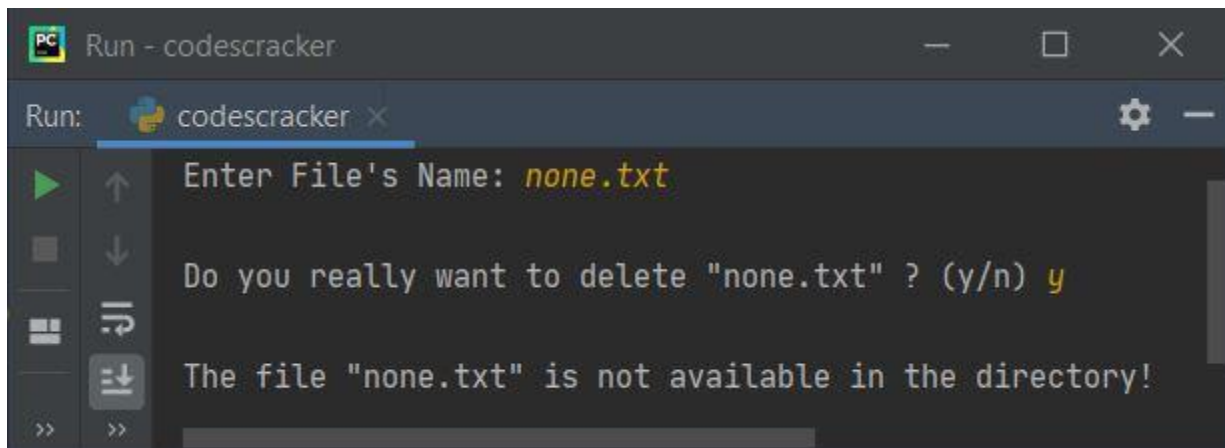
This is the modified version of previous program. This program displays confirmation message to user, whether he/she actually wants delete the file entered by him/her or not. This program also uses **try-except** to handle the error produced when deleting the file. Let's have a look:

```
import os

print("Enter File's Name: ", end="")
filename = input()

print("\nDo you really want to delete \"" + (filename) + "\" ? (y/n) ", end="")
ch = input()
if ch=='y':
    try:
        os.remove(filename)
        print("\nThe File, \"" + (filename) + "\" deleted successfully!")
    except IOError:
        print("\nThe file \"" + (filename) + "\" is not available in the directory!")
    else:
        print("\nExiting...")
```

Here is its sample run with input **none.txt** (a non-existing file):



```
Run - codescracker
Run: codescracker x
Enter File's Name: none.txt
Do you really want to delete "none.txt" ? (y/n) y
The file "none.txt" is not available in the directory!
```

The first statement inside the **try** block gets executed. That is, **os.remove(filename)**. When this statement produces an error, or something strange happened while deleting the file, then using **except**, I've processed the **IOError** (input/output error) to produce or print a message regarding it.

## Delete any File from any Folder (Directory)

This program is the complete version of deleting a file from a directory in Python. That is, this program allows user to delete any file from any directory.

```
import os, glob

print("Enter Full Path of Directory to Operate on: ", end="")
dirpath = input()
os.chdir(dirpath)
print("\nList of Files: ")
chk = 0
for file in glob.glob("*."):
    chk = 1
    print(file)

if chk == 0:
    print("Empty Folder!")
else:
    print("\nEnter File's Name to Delete: ", end="")
    filename = input()

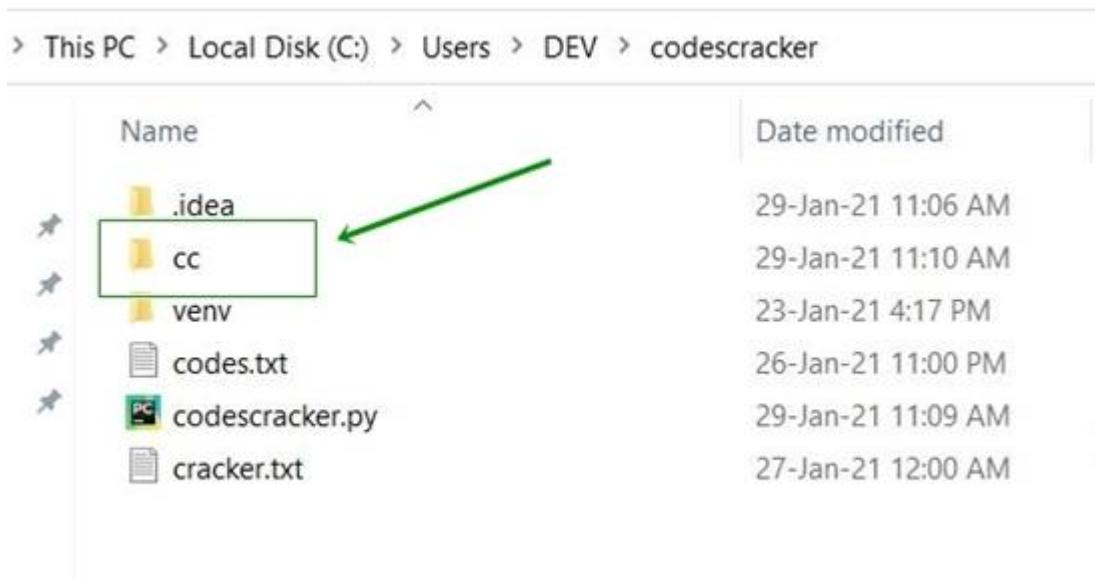
    print("\nDo you really want to delete \"" + (filename) + "\" ? (y/n) ",
end="")
    ch = input()
    if ch=='y':
        try:
            os.remove(filename)
```

```

print("\nThe File, \"" + (filename) + "\" deleted successfully!")
print("\nDo you want to list remaining files ? (y/n) ", end="")
ch = input()
if ch=='y':
    os.chdir(dirpath)
    print("\nList of Files: ")
    for file in glob.glob("*."):
        print(file)
except IOError:
    print("\nThe file \"" + (filename) + "\" is not available!")
else:
    print("\nExiting...")

```

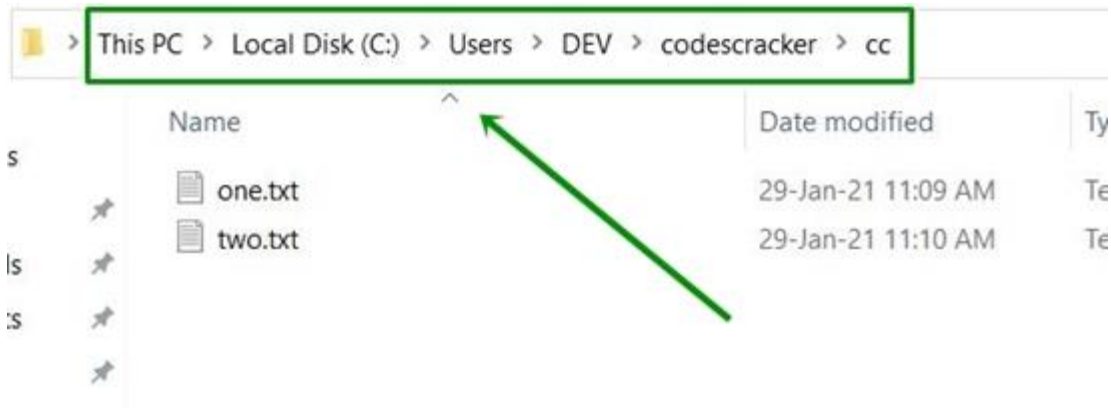
before executing the program, create another folder say **cc** like shown in the snapshot given below:



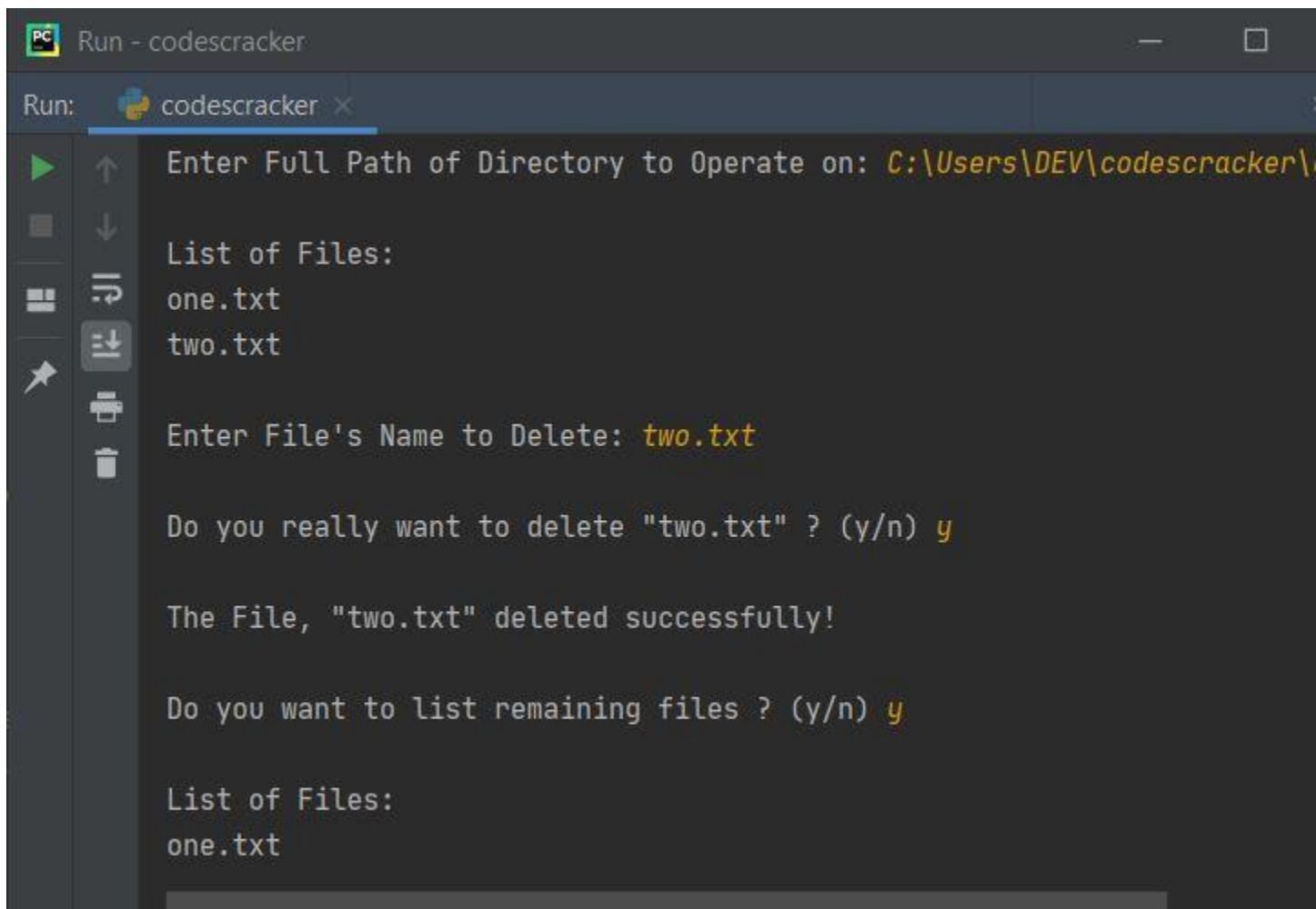
create any two files say **one.txt** and **two.txt** inside this newly created folder. Here is the snapshot of the folder that contains two newly created files:



Now copy the full path of this folder, like shown in the snapshot given below:



That is, the full path of newly created folder named **cc** in my case is **C:\Users\DEV\codescracker\cc**. Now execute the above program, and just copy and paste (or type) the full path and press `ENTER` key, the list of files gets displayed. Then enter the name of file to delete. Here is the snapshot that shows complete sample runs of above program, one by one:



```
Run - codescracker
Run: codescracker x
Enter Full Path of Directory to Operate on: C:\Users\DEV\codescracker
List of Files:
one.txt
two.txt
Enter File's Name to Delete: two.txt
Do you really want to delete "two.txt" ? (y/n) y
The File, "two.txt" deleted successfully!
Do you want to list remaining files ? (y/n) y
List of Files:
one.txt
```

**Note** - The **glob** module of Python allows us to match specified pattern as per rules related to Unix shell. Basically this module (in above program) is used to print all files with all extensions using \*.\*. \* before and after dot (.) represents all file's name and all file's extension.

**Note** - The **os.chdir()** method is used to change the current working directory. The directory passed as its argument becomes the current directory

The dry run of above program goes like:

- Using **input()**, the complete path of folder entered by user gets initialized to **dirpath**. Therefore **dirpath = "C:\Users\DEV\codescracker\cc"**

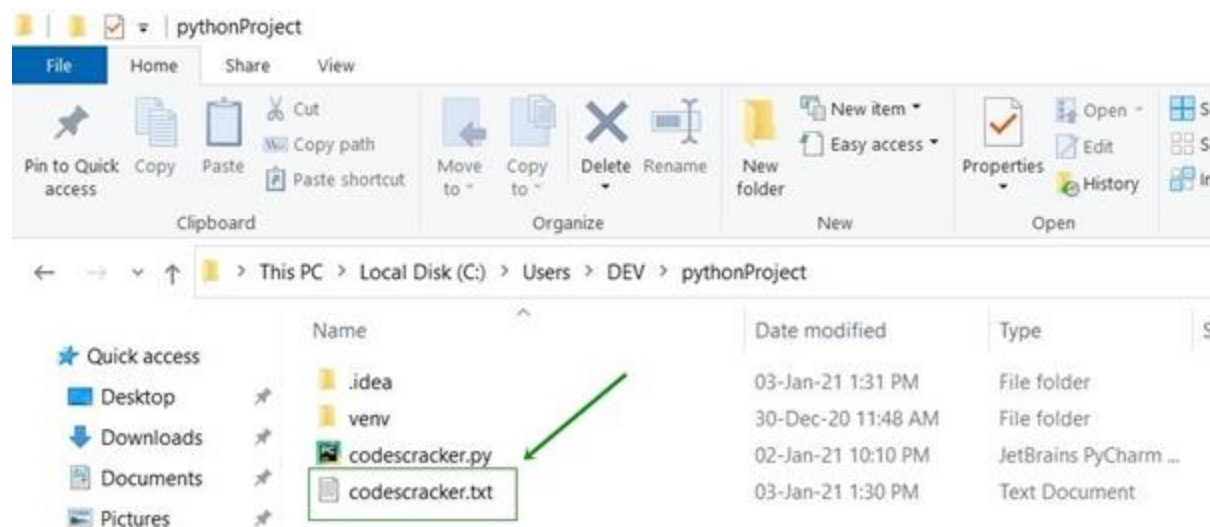
- The execution of **for** loop begins, that is the name of first file inside the current directory gets initialized to **file** and program flow goes inside the loop
- Inside the loop, using **print()**, the file's name gets printed. I've initialized **1** to **chk** to check its value after exiting from the loop, that whether any file is available inside the directory or not
- Now at second time, the second file gets initialized to **file**, and the value of it gets printed from inside the loop
- This process continues until and unless, each and every file gets printed
- Now after exiting from the loop, the condition **chk==0** gets evaluated. That is, if program flow goes inside the loop, means **1** gets initialized to **chk** and folder has some files. Or when program flow doesn't goes inside the loop, means there is no any file available in the folder. Therefore **chk**'s value remains 0 (as initialized initially)
- Now program flow goes to **else**'s body (if **chk**'s value is 1, or not equal to 0). All the statements gets executed one by one.
- Inside its body, the name of file gets received and initialized to **filename**
- Again using the **input()**, I've received another input from user, whether he/she really wants to delete the file or not. That is, if entered character is **'y'**, then the condition (of **if**) **ch=='y'** or **'y'=='y'** evaluates to be true, and program flow goes inside its body
- Now the first statement of **try** gets executed. That is, **os.remove(filename)** gets executed. If it raises an **IOError** (input/output error), then the program flow goes to **except**'s body and print any message written by programmer like **file is not available** or anything you want to provide
- But if file is available, then the file gets deleted and using another **input()**, I've displayed the list of files if user enters **'y'**

## Things to do Before Program

Because the program given below is used to count characters in a text file. Therefore first we've to create a text file say **codescracker.txt** with some contents say:

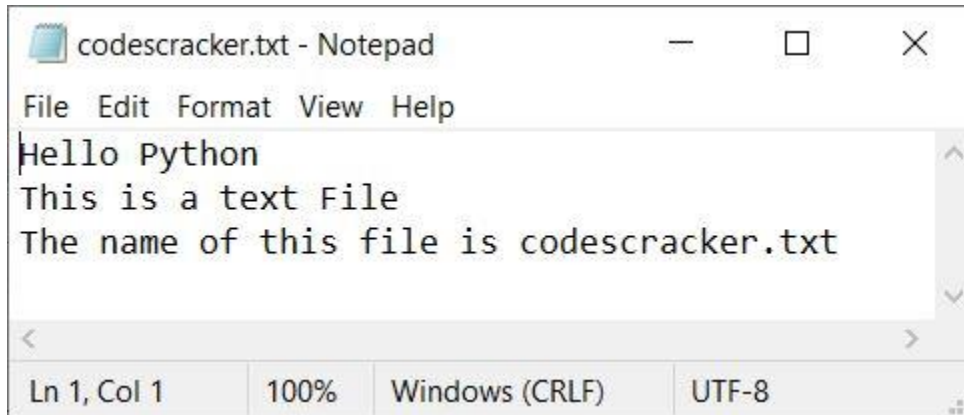
```
Hello Python  
This is a text File  
The name of this file is codescracker.txt
```

Save this file inside the current directory. The current directory is the directory where the Python code to count characters of this file is saved. Here is the snapshot of the folder where the file **codescracker.txt** is saved:



And here is the snapshot of opened file **codescracker.txt**:





Now let's create some Python programs to do the task like counting of characters, vowels, spaces etc. of this text file.

---

## Count Vowels in Text File

The question is, **write a Python program to count number of vowels present in a file.** The program given below is the answer to this question:

```
print("Enter the Name of File: ")
fileName = str(input())
fileHandle = open(fileName, "r")
tot = 0
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

for char in fileHandle.read():
    if char in vowels:
        tot = tot+1
fileHandle.close()

print("\nTotal Vowels are:")
print(tot)
```

## Count Consonant in Text File

The question is, **write a program in Python that counts total consonants available in a text file.** Here is its answer:

```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
```

```

fileHandle = open(fileName, "r")
tot = 0
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

for char in fileHandle.read():
    if char>='a' and char<='z':
        if char not in vowels:
            tot = tot+1
    elif char>='A' and char<='Z':
        if char not in vowels:
            tot = tot+1

fileHandle.close()

if tot>1:
    print("\nThere are " + str(tot) + " Consonants available in the File")
elif tot==1:
    print("\nThere is only 1 Consonant available in the File")
else:
    print("\nThere is no any Consonant available in the File!")
except IOError:
    print("\nError Occurred!")

```

## Count New Lines in Text File

To count the number of new lines in a text file, use following Python program:

```

print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

    for char in fileHandle.read():
        if char=='\n':
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " New Lines available in the File")
    elif tot==1:
        print("\nThere is only 1 New Line available in the File")
    else:
        print("\nThere is no any New Line available in the File!")
except IOError:
    print("\nError Occurred!")

```

## Count Blank Spaces in Text File

This program counts total number of blank spaces available in a text file entered by user at run-time.

```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

    for char in fileHandle.read():
        if char==' ':
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " Blank spaces available in the File")
    elif tot==1:
        print("\nThere is only 1 Blank space available in the File")
    else:
        print("\nThere is no any Blank space available in the File!")
except IOError:
    print("\nError Occurred!")
```

## Count Total Characters in Text File

This is the last program of this article. This program is created to count all the characters or total number of characters available in a text file.

```
print(end="Enter the Name of File: ")
fileName = str(input())
try:
    fileHandle = open(fileName, "r")
    tot = 0

    for char in fileHandle.read():
        if char:
            tot = tot+1
    fileHandle.close()

    if tot>1:
        print("\nThere are " + str(tot) + " Characters available in the File")
    elif tot==1:
        print("\nThere is only 1 Character available in the File")
    else:
        print("\nThe File is empty!")
except IOError:
    print("\nError Occurred!")
```