PYTHON IS FOR EVERYONE

Tutorial 10:

PYTHON PROGRAMMING - FILE HANDLING IN GOOGLE COLAB

Jeff Gentry

@www.linkedin.com/in/jefferycharlesgentry

Objectives

- Understand how to open, read, and write files in Python.
- Learn about different file modes.
- Explore how to handle file exceptions.
- Practice file handling through hands-on exercises.

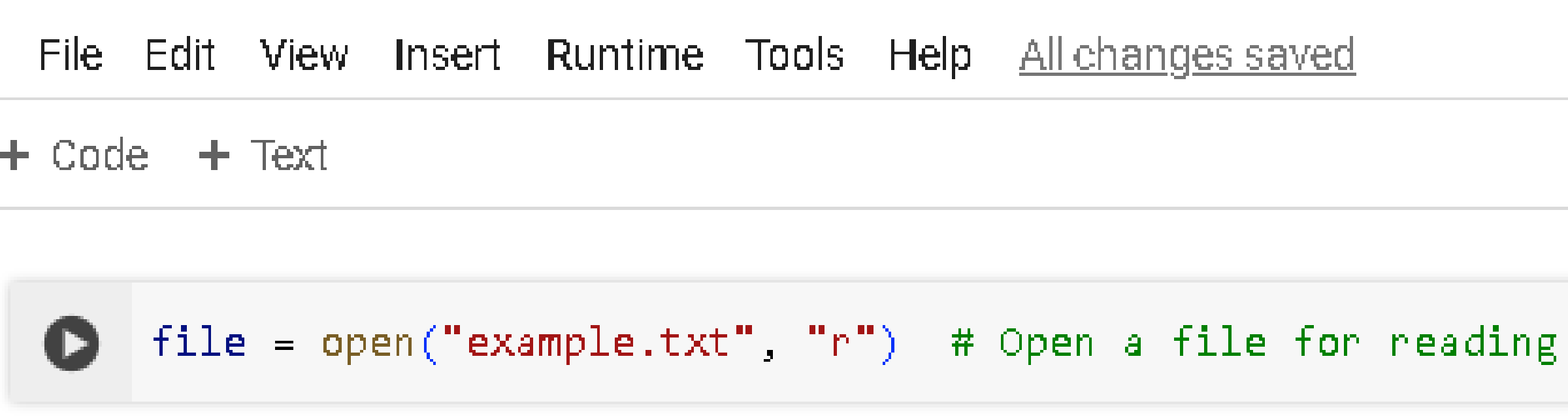
Opening a File

In Python, you can open a file using the “open()” function. The “open()” function requires at least one argument: the file name. You can also specify the mode in which to open the file.

Common File Modes

- 'r': Read (default mode). Opens a file for reading.
- 'w': Write. Opens a file for writing (creates a new file or truncates an existing file).
- 'a': Append. Opens a file for appending (creates a new file if it does not exist).
- 'b': Binary mode. Used for binary files (e.g., images).
- 't': Text mode (default). Used for text files.

Examples



You will get an error when you try to run this because we don’t currently have a file named example.txt and for now we’ll ignore this.

Writing to a File

You can write to a file using the “write()” method. If the file does not exist, it will be created. If it does exist and is opened in write mode ('w'), it will be truncated.

```
with open("output.txt", "w") as file:
    file.write("Hello, World!\n")
    file.write("This is a new file.")
```

Writing to a File

Check if the file exists (it will be displayed when you run the code snippet).

```
import os

# List files in the current directory
print(os.listdir())
```

```
['.config', 'output.txt', 'sample_data']
```

Appending to a File

To add content to an existing file without truncating it, you can open the file in append mode ('a').

```
with open("output.txt", "a") as file:
    file.write("\nAppending a new line.")
```

Handling Exceptions

When working with files, it’s important to handle exceptions that may occur, such as file not found errors. You can use “try” and “except” blocks to handle these exceptions.

```
try:
    with open("non_existent_file.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("The file does not exist.")
```

```
The file does not exist.
```

Practice Exercises

Create and Write to a File: Write a program that creates a file and writes your favorite quotes to it.

```
quotes = ["The only limit to our realization of tomorrow is our doubts of today.",
          "Life is 10% what happens to us and 90% how we react to it."]

with open("quotes.txt", "w") as file:
    for quote in quotes:
        file.write(quote + "\n")
```

Practice Exercises

Read from a File: Write a program that reads the contents of a file and prints each line.

```
with open("quotes.txt", "r") as file:
    for line in file:
        print(line.strip())
```

```
The only limit to our realization of tomorrow is our doubts of today.
Life is 10% what happens to us and 90% how we react to it.
```

Practice Exercises

Count Words in a File: Write a program that counts the number of words in a text file.

```
def count_words(filename):
    with open(filename, "r") as file:
        content = file.read()
        words = content.split()
        return len(words)

print("Number of words:", count_words("quotes.txt"))
```

```
Number of words: 27
```

Practice Exercises

Copy File Content: Write a program that copies the content of one file to another.

```
with open("quotes.txt", "r") as source_file:
    with open("quotes_copy.txt", "w") as destination_file:
        for line in source_file:
            destination_file.write(line)
```

Practice Exercises

File Existence Check: Write a program that checks if a file exists before attempting to read it.

```
import os

filename = "quotes.txt"
if os.path.exists(filename):
    with open(filename, "r") as file:
        content = file.read()
        print(content)
else:
    print("The file does not exist.")
```

```
The only limit to our realization of tomorrow is our doubts of today.
Life is 10% what happens to us and 90% how we react to it.
```

Conclusion

In this tutorial, you learned about file handling in Python, including how to open, read, write, and append to files. You also explored exception handling to manage potential errors when working with files. File handling is a vital skill for any programmer, enabling you to work with data stored in files effectively.

Next Steps

In tutorial 11, we will learn how to handle errors and exceptions gracefully in your programs.

FOLLOW ME

for more tips you didn’t know you needed

Jeff Gentry

@www.linkedin.com/in/jefferycharlesgentry