Lecture 7

Error handling and Files

Announcements

- There is going to be a “final” project that will be assigned soon which will be much longer and you will be given approximately a week and a half to complete it. This project will incorporate everything that we have learned so far into one comprehensive project.
- There will also be a short assignment on files which covers today’s lecture

Working with txt files

Although I don’t do this very often or at all it is another concept that is taught in 31.

You can think of this as an intro or a general idea of how to work with outside files since in real world applications we would read from / write to databases which isn’t taught in lower division courses

Functions to use with text files:

open()

read()

readline()

readlines()

write()

close()

Files: open()

The open() function takes in two parameters as strings.

The first parameter is the file that you want to open. The second parameter is a letter that represents what you want to do with the opened file. (I don't recommend using w+ or r+ can be a little tricky)

```
#Open a file to read from it and will raise an exception if
#the file does not exist
file = open("myFile.txt", "r")

#Open a file to read and write from it and will raise an exception if
#the file does not exist
file2 = open("myFile.txt", "r+")

#Open a file to write to it. Creates a new file if it does not exist
file3 = open("myFile.txt", "w")

#Open a file to read and write from it. Creates a new file if it does not exist
file4 = open("myFile.txt", "w+")
```

Files: .close()

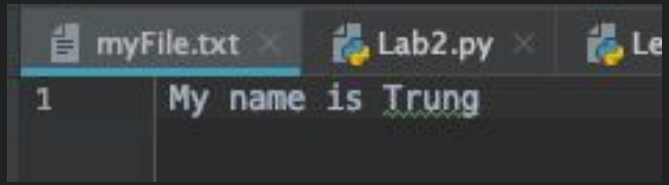
When ever we open a file we have to close it when we are done using it or if we want to reopen the file using a different mode: read/write

```
file.close()
file2.close()
file3.close()
file4.close()
```

Files: .write()

We use this method to write a line in the desired text file that we opened. In the example myFile.txt was created since it didn't exist before I ran the program. After running the program successfully the file: myFile.txt will be created and found in the same project folder as your program.

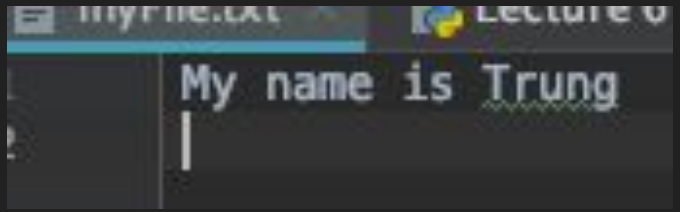
```
file = open("myFile.txt", "w")
file.write("My name is Trung")
file.close()
```



Files: write() notice

Look what happens when I try to write to file more than once

```
file = open("myFile.txt", "w")
file.write("My name is Trung\n")
file.close()
file = open("myFile.txt", "w")
file.write("My name is Trung\n")
file.close()
```



“My name is Trung” is not written twice since every time we open to write to a file without going to the end existing data is either truncated or overwritten

Files: .read()

This method can take in an integer: n and will return the first n characters in the file as a string. If you don’t pass in an integer it will return the entire file as a string exactly like how it is formatted in the file.

```
file = open("myFile.txt", "r")
print(file.read())
file.close()
```

```
/Users/trungdao/PycharmProjects/Lab2/venv
My name is Trung
```

Files: .read() notice

Look what happens when I try to read the file multiple times.

```
file = open("myFile.txt", "r")
print(file.read())
print(file.read(2))
print(file.read())
file.close()
```

```
My name is Trung
```

“My name is Trung” only printed once. That is because after reading the entire file on line 2, the cursor has reached the end of the file and has nothing more to read. It does not reset.

Files: .readline()

This method reads and returns an entire line as a string by default. Just like read() you can pass in an integer: n and it will read n characters from that line. It does NOT matter if the line has more more than n characters, .readline() will not read more than 1 line at a time.

```
file = open("myFile.txt", "r")
print(file.readline())
file.close()
```

```
Hello my name is Trung
```

```
file = open("myFile.txt", "r")
print(file.readline(3))
file.close()
```

```
Hel
```

```
file = open("myFile.txt", "r")
print(file.readline(100))
file.close()
```

```
Hello my name is Trung
```

Files: .readlines()

This method will read all the lines and return a list with each line being an item in the list.

```
Hello my name is Trung
Hello my name is Brian
Hello my name is Karan
```

```
file = open("myFile.txt", "r")
print(file.readlines())
file.close()
```

```
['Hello my name is Trung\n', 'Hello my name is Brian\n', 'Hello my name is Karan']
```

Files: .readlines() notice

Notice how if I were to call readlines() multiple times it returns an empty list. This is because with the first function call the whole file was already read and now there is nothing to read.

```
file = open("myFile.txt", "r")
print(file.readlines())
print(file.readlines())
print(file.readlines())
file.close()
```

```
['Hello my name is Trung\n', 'Hello my name is Brian\n', 'Hello my name is Karan']
[]
[]
```

The “cursor” when working with files

Notice how I tried to open a file with w+ to read and write from the file but after writing to it I am not able to print anything. As stated before this is something that we need to be very careful about since after we wrote to the file the cursor has moved so read is no longer able to read from the line that was just written.

The “cursor”

For the most part you won’t be using anything other than readline() or write() when reading or writing from/to a text file.

Every time you read or write to a file, the cursor moves forward.

Notice how there is a space in between each line. This is because of the “\n” string that exists when we make a new line in the text file. Also the 4th readline function doesn't print anything since there is nothing left to read. The file is only 3 lines long.

```
file = open("myFile.txt", "r")
print(file.readline())
print(file.readline())
print(file.readline())
print(file.readline())
file.close()
```

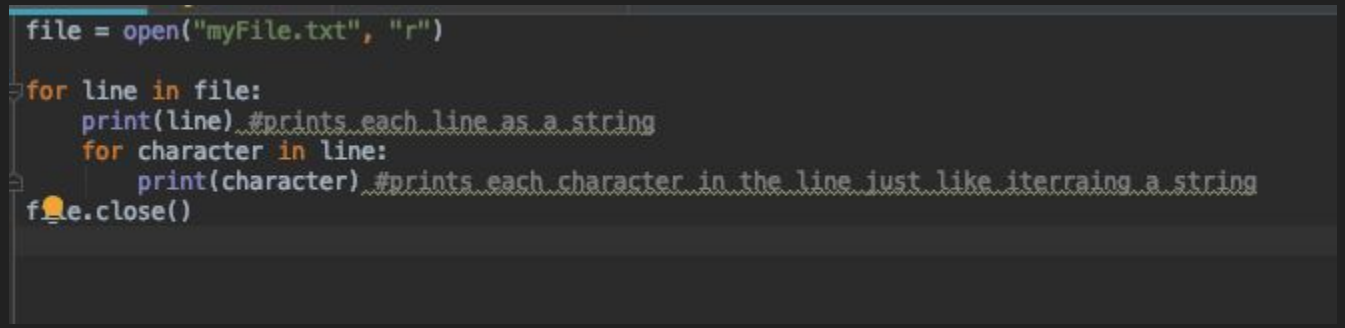
```
Hello my name is Trung

Hello my name is Brian

Hello my name is Karan
```

Iterating with files

Iterating through a text file is very similar to how you would loop through anything else. As you can see by now the text file basically seen a large string formatted by lines, because of this you can loop through them in a similar way to strings.



You can use while loops as well

My while condition is checking if the line is empty or the end of the file. Notice how i have to reassign what the variable line is at each iteration by calling readline() again. This is because when readline was first called on line 2 it read the first line and that's it. I want to be calling readline after I read a line to allow the cursor to move to the next line.

```
file = open("myFile.txt", "r")
line = file.readline()
while line != '':
    print(line)
    line = file.readline()
file.close()
```

```
Hello my name is Trung

Hello

hi

howdy
```

What if we try to read from a file that doesn’t exist?

If we try to do this the open() function will raise an IOError and our program will crash. Even if we did everything correctly we need to consider this fact as a possibility as we are coding for others not for ourselves.

We will discuss how to do this in the following lecture.

What is an Error?

You get errors when you do something wrong and the interpreter will let you know with a message such as the following:

```
/Users/trungdao/PycharmProjects/IntroToICS/venv/bin/python /Users/trungdao/PycharmProjects/IntroToICS/errortest.py
Traceback (most recent call last):
  File "/Users/trungdao/PycharmProjects/IntroToICS/errortest.py", line 1, in <module>
    print(a)
NameError: name 'a' is not defined

Process finished with exit code 1
```

How do we read error messages?

```
Traceback (most recent call last):
  File "/Users/trungdao/PycharmProjects/IntroToICS/errortest.py", line 1, in <module>
```

This tells us exactly what line of code in our program caused the error

```
print(a)
```

This tells us what part of the line caused the error

```
NameError: name 'a' is not defined
```

This tells us what kind of error was raised and why it was raised

Common Errors

Syntax related Errors

These kinds of errors happen when the interpreter isn’t able to understand our code.

This will usually prevent the program from running or cause it to crash.

Logic related Errors

These kinds of errors are more difficult to find and are caused by flaws in your logic.

Most of the time this will still run you just won’t get the answer that you were expecting

Errors

There are many kinds of built in errors that you will run into things such as syntax error, value error, key error, type error, index error, name error, etc.

There are too many to list but they are all very self explanatory and with the traceback you should be able to see what you did wrong

General Examples:

Index error: Indexing position 10 for a list that only has 2 items

Syntax error: forgetting a :

Key error: trying to access a key that doesn't exist

Name Error: using a variable name that doesn't exist

As you can see they are all pretty self explanatory

Exceptions

Exceptions are still bad to have cause they can still cause our program to crash.

They are usually the result of the interpreter knowing what to do but is not able to do it.

Example:

I want to open a file and read / write to it but the file isn't able to be opened.

I want to connect to a network to get information but I have no internet access

Most likely there isn't anything wrong with our code but these things will still cause our program to crash so we have to learn how to handle it just incase it does happen.

try/except

A try statement will always come with an except statement.

The general format is try: (code block), if that code block fails except will “catch” the exception/error and will prevent the program from crashing. Otherwise the except code block will do nothing.

(IOError is the exception that is raised when a file fails to open)

```
#Part 9
#Open and loop through TheReVeFestivalFinale.txt and print every song(line)
#Write your code here
try:
    file = open("TheReVeFestivalFinale.txt", "r")
    for line in file:
        print(line)
    file.close()
except IOError:
    print("The file 'TheReVeFestivalFinale.txt does not exist")
```

In this example, we “try” to open the file and do our task as usual but if the file were to fail when opening an IOError will be raised which will then be caught by the except statement and the program will let us know that the file does not exist.

The program never crashes.

try/except

You can have multiple except statements to handle different errors that might happen

```
try:
    file = open("TheReVeFestivalFinale.txt", "r")
    for line in file:
        print(line)
    file.close()
    a = 10
    b = 0
    print(a/b)
except IOError:
    print("The file 'TheReVeFestivalFinale.txt does not exist")
except ZeroDivisionError:
    print("denominator is 0")
```

Finally

When you are using try/except statements you can, but don't have to, use a finally statement. A finally statement will always run despite the error happening or not.

```
try:
    file = open("TheReVeFestivalFinale.txt", "r")
    for line in file:
        print(line)
    file.close()
except IOError:
    print("The file 'TheReVeFestivalFinale.txt does not exist")
finally:
    print("This statement will always run")
```

Don’t over do it

I don’t find myself using Try/Except very often unless im working with something that requires downloading/network connections/etc.

Although you CAN implement a try/except into all your functions to catch possible exceptions doesn’t mean you SHOULD.

Write your programs so that errors don’t happen instead of letting them happen and dealing with it.

When it comes to properly using functions, a type annotation or comment is good enough. You don’t need to implement checks for all possible inputs, ex: a function that takes in an int and returns an int, you don’t need to account for strings/lists/etc

Congrats

We have covered all of the material from ICS 31
From now on I will mainly be discussing ideas and concepts that you can apply to what you have learned to make better programs

Final Project?

In addition to a written final there will also be a final project that is currently in the works and will be released later tonight or tomorrow afternoon.

In a typical programming course a project/lab is generally only assigned once every 2 weeks (4-5 projects per quarter) and are much longer than the assignments that we have been doing.

For the last week ish left we will be working on a project similar to this which will incorporate everything that we have learned so far. We will continue to learn new ideas and concepts during lecture but I won't require you to use them and if they show up they will be done for you already so all you have to do is apply it just so you can see how they work.