Programming And Scripting
Lab Topic 07-Files

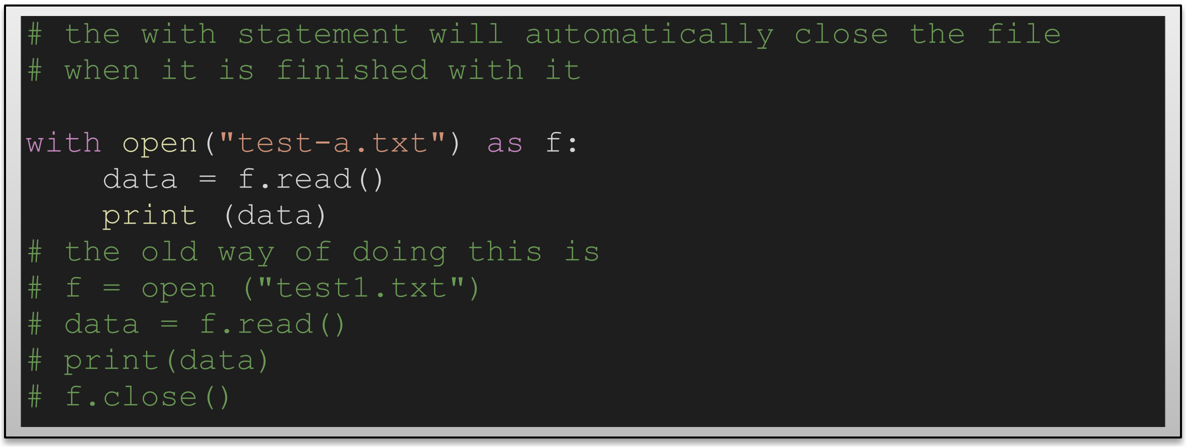
Introduction.

The first activity on this sheet is a quick quiz, answers are at the end of the sheet.
I would suggest that you create another folder in labs called **Topic07-files**, to put the programs you write for the other activities.

quiz:

1.

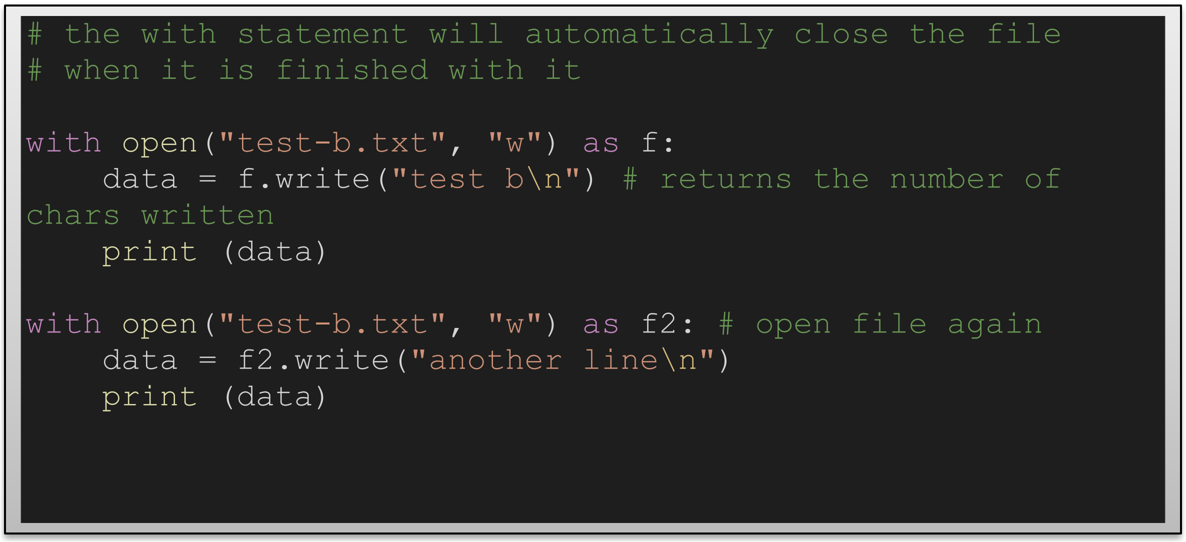
   a. Look at the program below, if the file test-a.txt does not exist. What will happen when the program runs?



```
# the with statement will automatically close the file
# when it is finished with it

with open("test-a.txt") as f:
    data = f.read()
    print (data)
# the old way of doing this is
# f = open ("test1.txt")
# data = f.read()
# print(data)
# f.close()
```

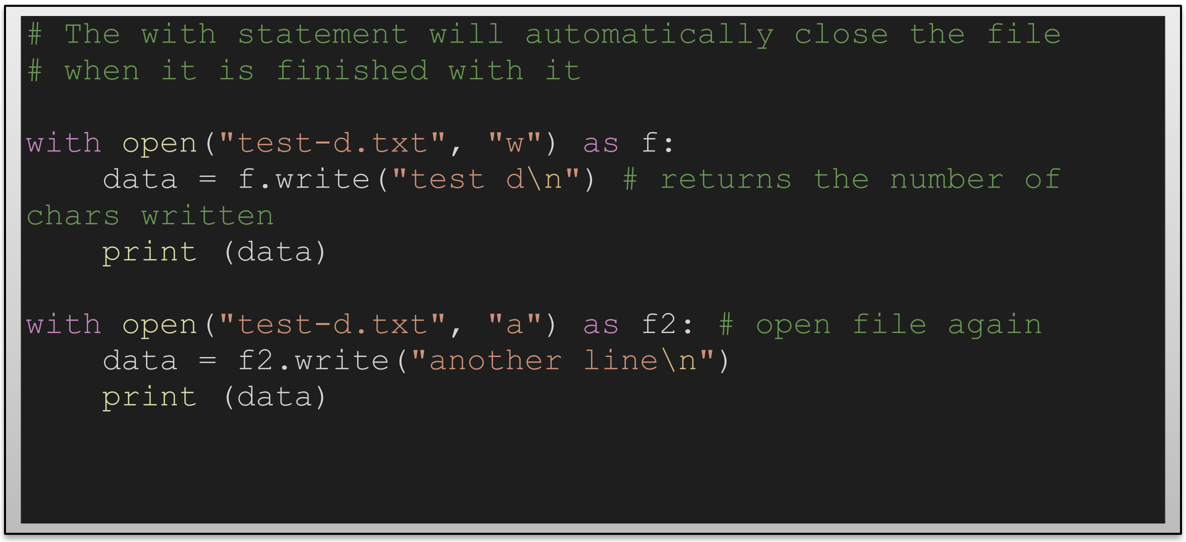
b. Look at the program below, if the file test-b.txt does not exist, what will be outputted to the console when this program is run?

c. What will the contents of the file test-b.txt be when this program is run?



```
# the with statement will automatically close the file
# when it is finished with it

with open("test-b.txt", "w") as f:
    data = f.write("test b\n") # returns the number of
chars written
    print (data)

with open("test-b.txt", "w") as f2: # open file again
    data = f2.write("another line\n")
    print (data)
```

d. Look at the modified program below, what will the contents of the file be after this program is run.



```
# The with statement will automatically close the file
# when it is finished with it

with open("test-d.txt", "w") as f:
    data = f.write("test d\n") # returns the number of
chars written
    print (data)

with open("test-d.txt", "a") as f2: # open file again
    data = f2.write("another line\n")
    print (data)
```

Lab: Messing with files.

2. Write a program that counts how many times it was run.
   For this exercise will have to store data outside of memory, and that is accessible each time the program is run, (**persistent** data). We would normally use a database for something like this, but we can use a file.
   To make life easier let's assume that the file already exists. So, we can just read the current count from it then overwrite it with the new count.
   Create a file called count.txt and put in 0 into it

```
0
```

   a. Write a function that reads in a number from a file that already exists (count.txt). test the program by calling the function and outputting the number.

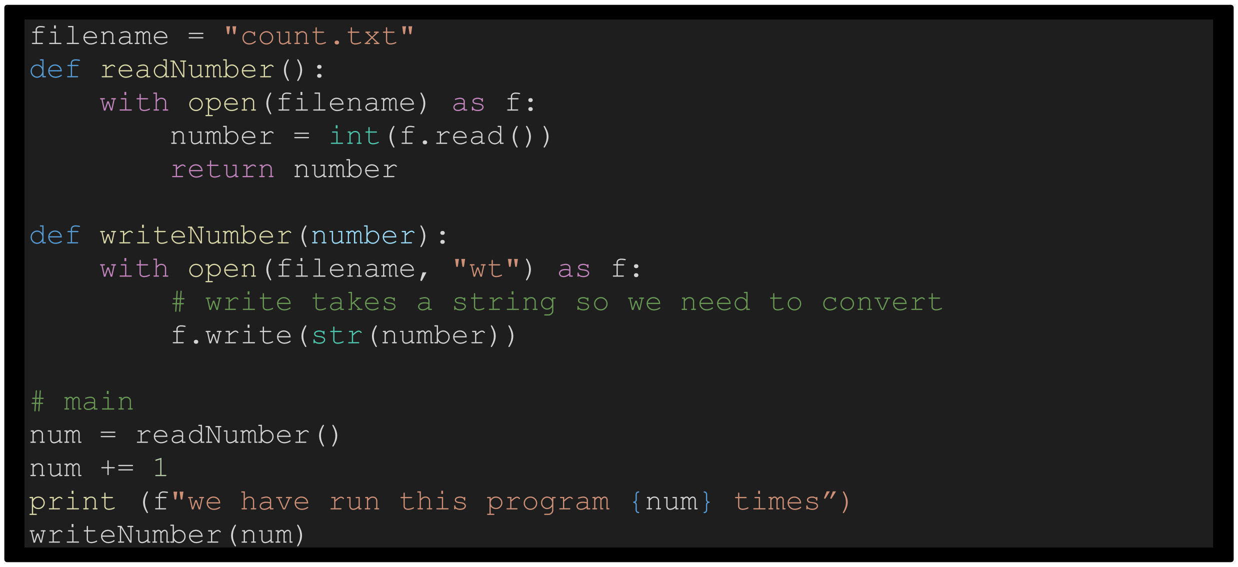
Remember last week I said, in functions, only use data that was passed in as an argument. Here. I am breaking that rule. I am using a variable that I want to treat as a constant (FILENAME) so I am putting all in capitals. I do not change this variable.

```
FILENAME = "count.txt"
def readNumber():
    with open(FILENAME) as f:
        number = int(f.read())
        return number
# test it
num = readNumber()
print (num)
```

   b. Write a function that takes in a number and overwrites a file with that number (count.txt). test it and check that the file has been changed

```
def writeNumber(number):
    with open(FILENAME, "wt") as f:
        # write takes a string so we need to convert
        f.write(str(number))

# test it
writeNumber(3)
```

c. Write a program that, uses these two functions, to count how many times the program has been run. Test it, check to see that the number goes up each time.



```python
filename = "count.txt"
def readNumber():
    with open(filename) as f:
        number = int(f.read())
        return number

def writeNumber(number):
    with open(filename, "wt") as f:
        # write takes a string so we need to convert
        f.write(str(number))

# main
num = readNumber()
num += 1
print (f"we have run this program {num} times”)
writeNumber(num)
```

Discussion:

In this question we assume that the file count.txt exists, what happens the first time you run this program on a new machine where count.txt does not exist?
(answer: The program will throw an error, so)
Should we:

- Make the user create the file "by hand" before they run the program, (this is easy of the user in this case but more difficult for more complicated data structures)
- Create an "init" program that initializes the file, in this case it will just put 0 into the file.
  (The function will need to be written into the file as well of course

```
writeNumber(0)
```

- Write some code to check if the file exists. To do this we will need to import os.path and use the isfile() function. You would need to look this up.

```
import os.path
filename = "count.txt"
if not os.path.isfile(filename):
    print ("File does not exist")
    #initialise file here
    writeNumber(0)
```

- Use a try catch loop on the read (I think the best way).
  We will be covering exception handling later in the course, so don't worry about this yet.

```
def readNumber():
    try:
        with open(filename) as f:
            number = int(f.read())
            return number
    except IOError:
        # this file will be created when we write back.
        # no file assumes first time running
        # ie 0 previous runs
        return 0
```

Using json module to save a data structure (Dict or List)

JavaScript
Object
Notation

If we want to store a more complicated data structure to a file, we should use either:

a. JSON: Which will store the data structure in a human readable way. JSON is a standard way of storing objects, you will see more on this later in the course.
   Python has a module called json, which has two functions:
   - dump(obj, fp); which writes a Dict or list to a file

   And
   - load(fp): which loads a data structure (Dict or list) from a file

Or

b. Pickle: Which will store the data structure in binary format (not human readable).

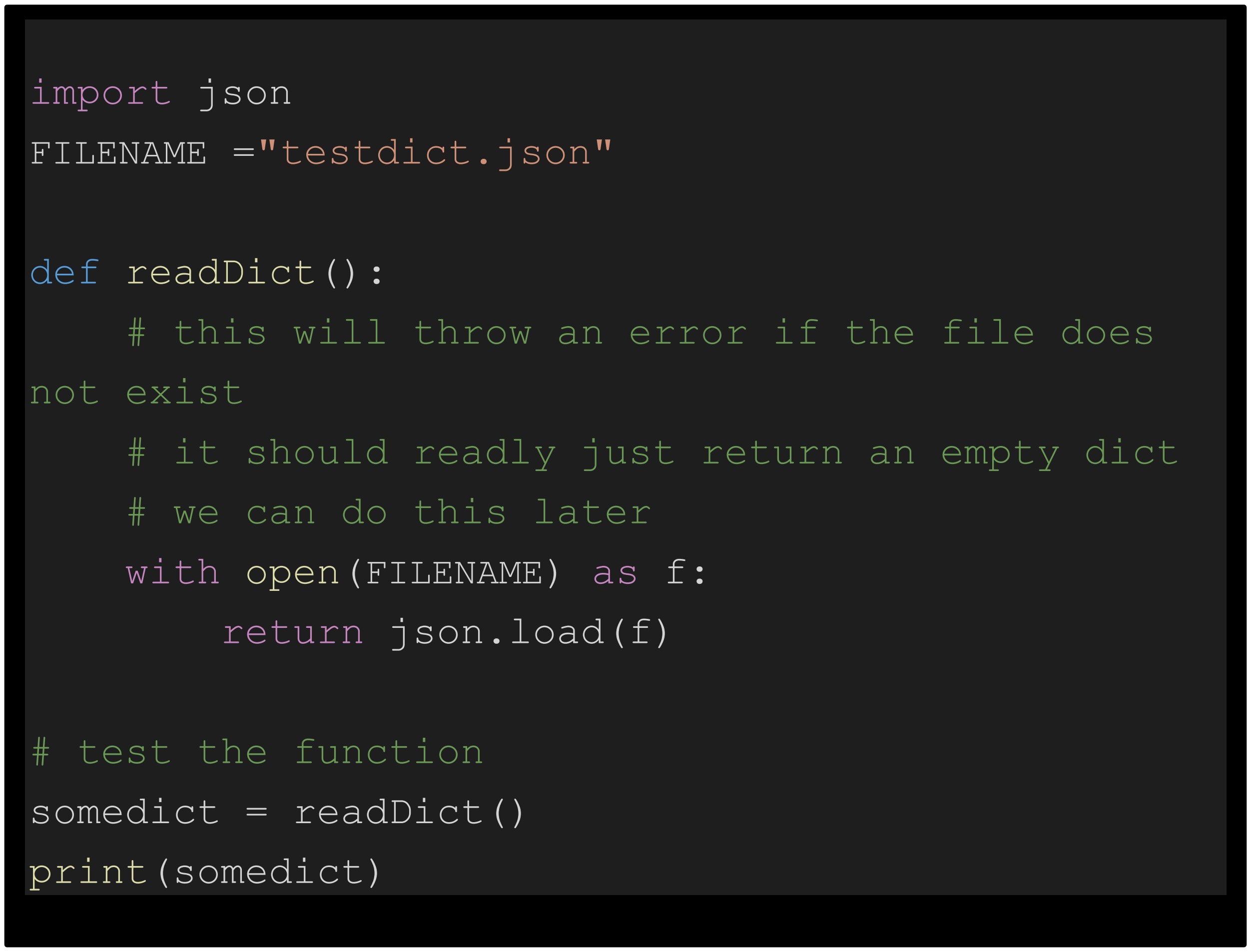
3. Write a function that will store a simple Dict to a file as JSON. TEST IT
   Answer

```
import json
FILENAME="testdict.json"
sample= dict(name='fred', age=31, grades=[1,34,55])

def writeDict(obj):
    with open(FILENAME, 'wt') as f:
        json.dump(obj,f)

#test the function
writeDict(sample)
```

Look at the contents of the file testdict.json, you will see that it is very similar to a Dict, this format is called JSON (javaScript Object Notation), as I said we will be doing more on this later.

Reading a dict from a file

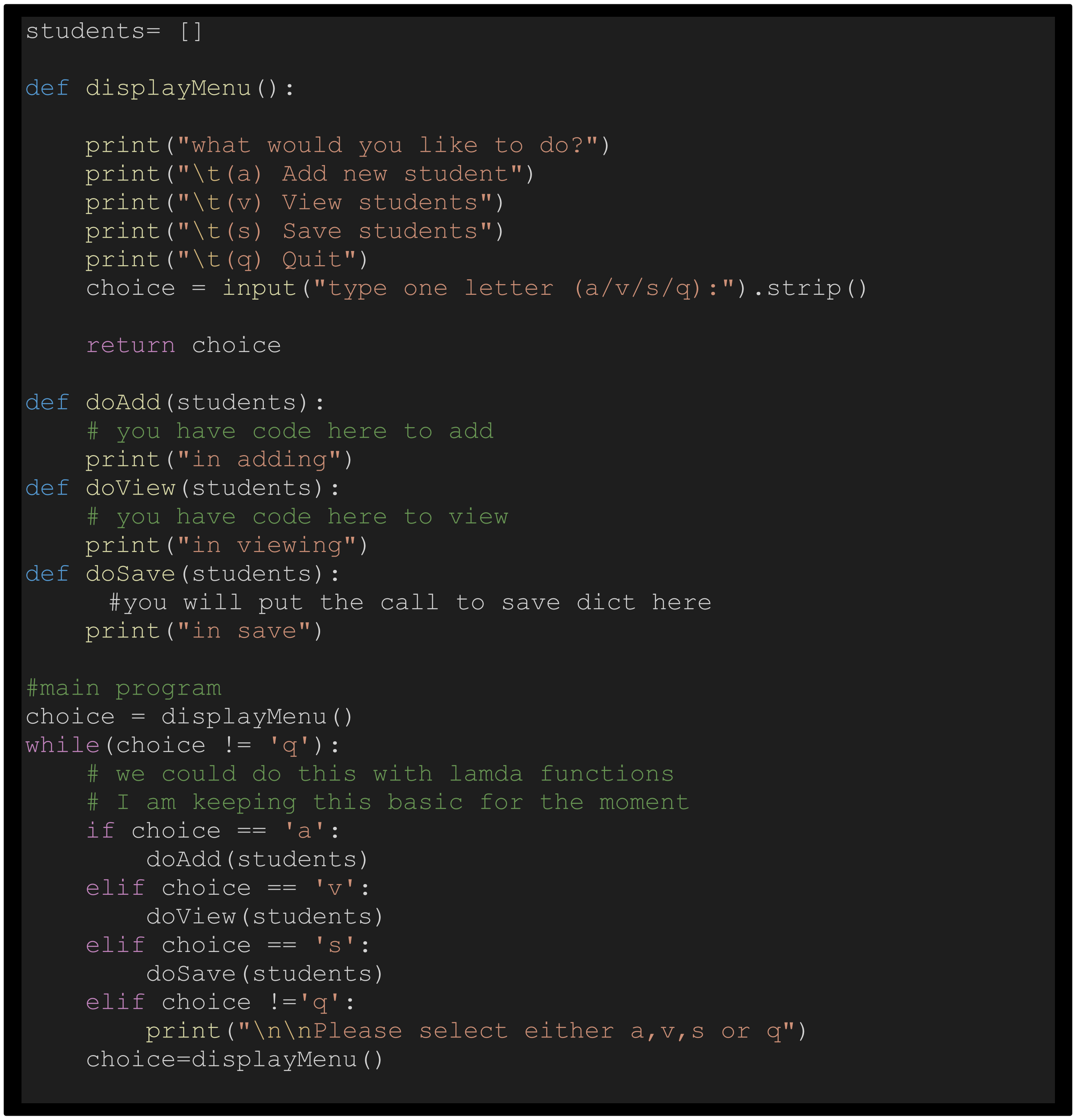
4. Write a function that will read in a dict object from file. TEST IT



```python
import json
FILENAME ="testdict.json"

def readDict():
    # this will throw an error if the file does
not exist
    # it should readly just return an empty dict
    # we can do this later
    with open(FILENAME) as f:
        return json.load(f)

# test the function
somedict = readDict()
print(somedict)
```

Save the students we made last week

5. With the program we made last week, create a new menu item called save. When the user selects the doSave() function should be called (the do save can do nothing but printout doSave for the moment.



```
students= []

def displayMenu():

    print("what would you like to do?")
    print("\t(a) Add new student")
    print("\t(v) View students")
    print("\t(s) Save students")
    print("\t(q) Quit")
    choice = input("type one letter (a/v/s/q):").strip()

    return choice

def doAdd(students):
    # you have code here to add
    print("in adding")
def doView(students):
    # you have code here to view
    print("in viewing")
def doSave(students):
      #you will put the call to save dict here
    print("in save")

#main program
choice = displayMenu()
while(choice != 'q'):
    # we could do this with lamda functions
    # I am keeping this basic for the moment
    if choice == 'a':
        doAdd(students)
    elif choice == 'v':
        doView(students)
    elif choice == 's':
        doSave(students)
    elif choice !='q':
        print("\n\nPlease select either a,v,s or q")
    choice=displayMenu()
```

6. Put the savedict() function into the program above, and call it from the dosave()
(I changed the name of the file to studentData.json)

In do save I added the lines
(NOTE: students is the array that stores the data)
A copy of the program is on GitHub.

```
    writeDict(students)
    print("students saved")
```

Read the data

7. Modify the program so that there is a load menu item, and a doLoad() function. The do load function should call the readDict() function return it so that is can be stored in the students list.

```
def doLoad():

    return readDict()


# in the menu
students = doLoad()
```

Discussion:

- You will notice that students is a list and not a dict but the program still works, the studentsData.json file has square brackets [].
- You could add a lot more to this program eg
    - Auto load the data
    - Ask for the save/load file name
    - Error checking

  But I don't think it is worth getting carried away with it!!!!

Quiz answers

a. The program will throw an error, the default mode is 'r' ie read, and read will throw an error if the file does not exist.

b. 7
   13

   This is because the write function returns the number of characters writing to the file this includes the new line character, I have not tested this on a windows machine(it may be one more).

c. Another line

   The file will be overwritten when we open the file in write mode (again)

d. test d
   another line

   This time we open the file in append mode, so the file is not overwritten