# **Programming with Python**

# Module 7:



* Working with Text files
* Error handling
* Working with Binary files

## Working with Text files

We have already learned how to write data to a file using code that look like this:

objFile = open("C:\\\_PythonClass\\test.txt", "**w**")

objFile.write("test" + "\n")

objFile.close()

Data can be read from the file using code like this:

objFile = open("C:\\\_PythonClass\\test.txt", "**r**")

strData = objFile.read()# read() reads all the data at once

print(strData)

objFile.close()

You can read a single line at a time like this:

objFile = open("C:\\\_PythonClass\\test.txt", "**r**")

strData = objFile.readline()# readline() reads a line of the data

print(strData)

objFile.close()

Each time you call the readline() method, it advances one line:

objFile = open("C:\\\_PythonClass\\test.txt", "**r**")

strData = objFile.readline()# readline() also act like a "cursor"

print("line 0: "+ strData)

strData = objFile.readline()

print("line 1: "+ strData)

objFile.close()

Data is often collected from the user and stored on the hard drive like this.

objFile = open("C:\\\_PythonClass\\Customers.txt", "**a**")

print("Type in a Customer Id and Name you want to save in a file")

print("(Enter 'Exit' to quit!)")

while(True):

strUserInput = input("Enter the Id and Name (ex. 1,Bob Smith): ")

if(strUserInput.lower() == "exit"): break

else: objFile.write(strUserInput + "\n")

objFile.close()

Combining reading and writing will give you a simple, but useful application:

# Display the current data

# The file must be created first before this will work,

#(and a+ does not work here!)

objFile = open("C:\\\_PythonClass\\Customers.txt", "**r+**")#see page 193

print("Here is the current data:")

print(objFile.read())

print("Type in a Customer Id and Name you want to add to the file")

print("(Enter 'Exit' to quit!)")

while(True):

strUserInput = input("Enter the Id and Name (ex. 1,Bob Smith): ")

if(strUserInput.lower() == "exit"): break

else: objFile.write(strUserInput + "\n")

print("Here is this data was saved:")

print(objFile.read())

objFile.close()



**LAB 7-1: Working with text files**

1. Create a script that allows you to store product ids, names, and prices.

The code will be very similar to the last example.



## Error Handling

You may have noticed that if you do not create a file before you use the read option (r) you will get an error. You can trap errors in your programs using a try-except construct.

#Using try

try:

decValue = 5/0

print(decValue)

except:

print("There was an error!")

#Now, without try

decValue = 5/0

print(decValue)

You can also capture the Exception object python puts into memory and extract the error messages like this:

try:

decValue = 5/0

print(decValue)

except Exception as e:

print("There was an error!")

print("pythons error info: ")

print(e)

Variables declared within a try block are not available outside of the block.

try:

decValue = 5/0

print(decValue)

except Exception as e:

print("There was a error!")

print("pythons error info: ")

print(e)

print(decValues) #This will not work!

"Exception" is a built in python class, that other classes inherit from. If you want to be more specific in catching an error you need to indicate that you want to catch one of these "Child" class objects.

try:

decValue = 5/0

print(decValue)

except ZeroDivisionError as e:

print("do not divide by zero!")

except Exception as e:

print("There was a unexpected error!")

print("pythons error info: ")

print(e)

You can have as many of this as you think you need.

v2 = int(input("Enter a number: "))

try:

decValue = 5/v2

print(decValue)

except ValueError as e:

print("Input must be a whole number!")

except ZeroDivisionError as e:

print("do not divide by zero!")

except Exception as e:

print("There was a unexpected error!")

print("pythons error info: ")

print(e)

Now that you know how to use a try block, you would add it to your programs whenever you think an error is likely to occur.

# Display the current data

# The file must be created first before this will work,

#(and a+ does not work here!)

try:

objFile = open("C:\\\_PythonClass\\Customerzzzzz.txt", "r+")#Bad file name

print("Here is the current data:")

print(objFile.read())

print("Type in a Customer Id and Name you want to add to the file")

print("(Enter 'Exit' to quit!)")

while(True):

strUserInput = input("Enter the Id and Name (ex. 1,Bob Smith): ")

if(strUserInput.lower() == "exit"): break

else: objFile.write(strUserInput + "\n")

print("Here is this data was saved:")

print(objFile.read())

objFile.close()

except Exception as e:

print("Python reported the following error: " + str(e))

finally:

print("The program has stopped")



**LAB 7-2: Working with errors**

1. Add error handling code to the code you created in Lab 7-1



## Saving Lists of data

List data can be saved to a file with the writelines method and read back again with the readlines method.

# Make a list

# To save it to a txt file

# it must be converted to a string

Id = str(1)

Name = "Bob Smith"

lstRow1 = [Id, Name]

print(lstRow1)

# output to file

objFile = open("C:\\\_PythonClass\\Customers.txt", "a")

objFile.writelines(lstRow1)

objFile.close()

objFile = open("C:\\\_PythonClass\\Customers.txt", "r")

lstFileData = objFile.readlines()

objFile.close()

print(lstFileData)

Note: Since the list data is not stored or retrieved in a friendly format, you may just want to extract the string values and write these to the file.

## Saving Data from an object

Data can be saved to a file from an object by extracting the string data as well.

#-- Data --

strData = None # Holds returned data

objFile = None # Handle for a text file

#--- create data classes ---

class Customer(object):

Id = None

Name = None

def ToString(self):

return str(self.Id) + "," + str(self.Name)

#End of class

# --- Use the class ----

# Create an object

objCustomer1 = Customer()

# Write data to the object's fields

objCustomer1.Id = input("Enter an Id: ")

objCustomer1.Name = input("Enter a Name: ")

# Read data from the object's fields

strData = objCustomer1.ToString()

# Use the data

# output to file

objFile = open("C:\\\_PythonClass\\Customers.txt", "a")

objFile.write(strData + "\n")

objFile.close()

# read from the file

objFile = open("C:\\\_PythonClass\\Customers.txt", "r")

print(objFile.read())

objFile.close()

## Working with Binary files

Data can be saved in binary code instead of just "plain" text using a technique called ***pickling***:

**import pickle**

intId = int(input("Enter an Id: "))

strName = str(input("Enter a Name: "))

lstCustomer = [intId, strName]

print(lstCustomer)

#Now we store the data with the pickle.dump method

objFile = open("C:\\\_PythonClass\\Customers.dat", "ab")

**pickle**.dump(lstCustomer, objFile)

objFile.close()

#And, we read the data back with the same pickle.load method

objFile = open("C:\\\_PythonClass\\Customers.dat", "rb")

objFileData = **pickle**.load(objFile) #Note that load() only load one row of data.

objFile.close()

print(objFileData)