CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

http://www.cit.ie

Programming for Data Analytics

Input/Output

Dr. Mohammed Hasanuzzaman
Room J102, Melbourne Building (Office)
e-mail:mohammed.hasanuzzaman@cit.ie
Website: https://mohammedhasanuzzaman.github.io

Topics

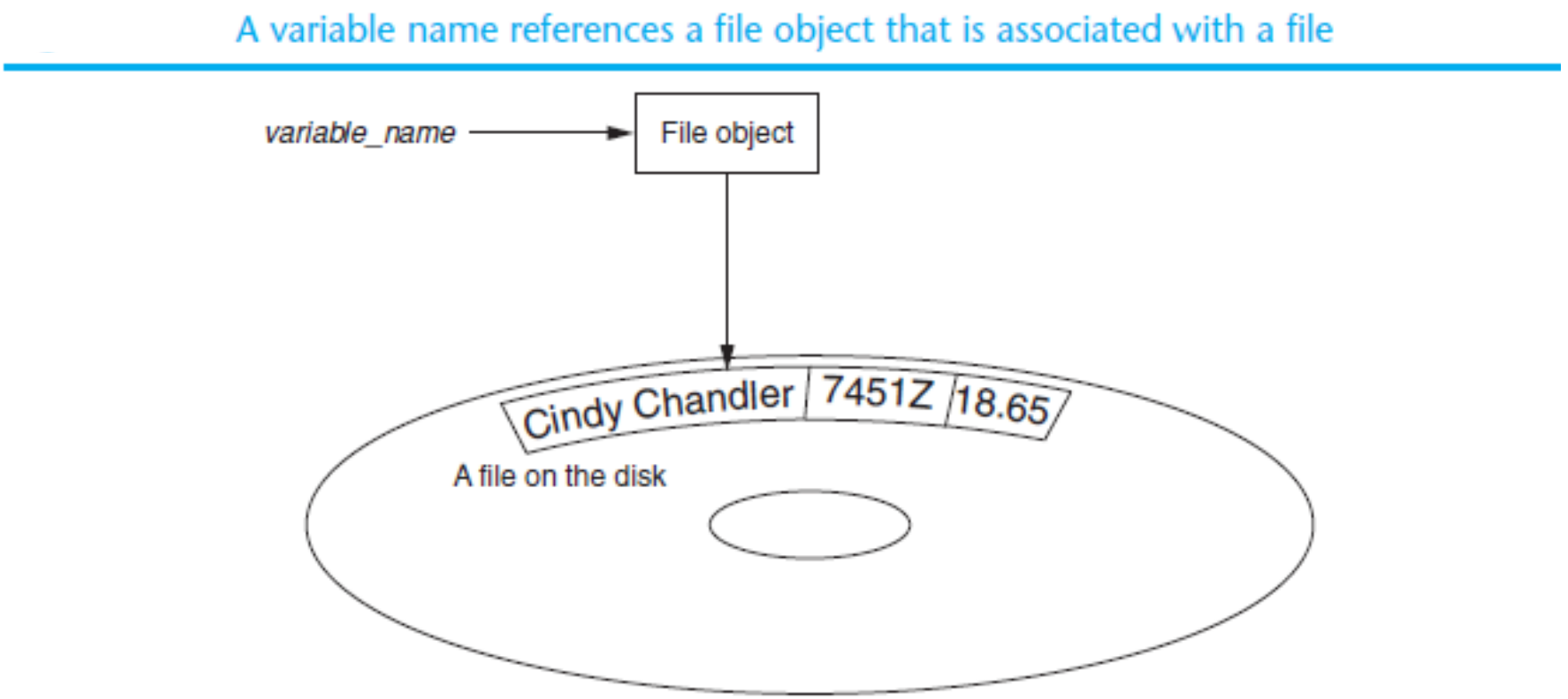
- Introduction to File Input and Output
- Using Loops to Process Files
- Using Files and Lists

Introduction to File Input and Output

- For a program to retain data between the times it is run, you must save the data
  - Data is commonly saved to a file, typically on computer disk
  - Saved data can be retrieved and used at a later time

- File object: object associated with a specific file
  - Provides a way for a program to work with the file: file object referenced by a variable

Filenames and File Objects (cont'd.)



Opening a File

- `open` function: used to open a file
  - Creates a file object and associates it with a file on the disk
  - General format:

    ```
    file_object = open(filename, mode)
    ```

- Mode: string specifying how the file will be opened
  - Example: reading only ('r'), writing ('w'), and appending ('a')

Opening a File

- Example: The variable customerFile will reference a file object that is associated with the file customers.txt on disk.

```
customerFile = open('cusomters.txt', 'r')
salesFile = open('sales.txt', 'w')
```

When opening a file to write it will delete all previous contents before commencing the write

Writing Data to a File

- Method: a function that belongs to an object
  - Performs operations using that object

- File object's `write` method used to write data to the file
  - Format: *`file_variable`*`.write(`*`string`*`)`

- File should also be closed using file object `close` method
  - Format: *`file_variable`*`.close()`

Writing Data to a File

- Example: The variable customerFile will reference a file object that is associated with the file customers.txt on disk.

```
def main():


    customerFile = open('cusomters.txt', 'w')
    customerFile.write("Java \n")
    customerFile.write("Perl \n")
    customerFile.close()
main()
```

Notice the write command doesn't automatically insert a newline. We do that using \n

Reading Data From a File

- `read` method: file object method that reads entire file contents into memory
  - Only works if file has been opened for reading
  - Contents returned as a string

- `readline` method: file object method that reads a line from the file
  - Line returned as a string, including '`\n`'

Reading Data from a File - Example


```
def main():

    customerFile = open('cusomters.txt', 'w')
    customerFile.write("Java \n")
    customerFile.write("Perl \n")
    customerFile.close();

    updatedCustomerFile = open('cusomters.txt', 'r')
    fileContents = updatedCustomerFile.read();
    updatedCustomerFile.close();
    print(fileContents)

main()
```

Will output:
Java
Perl

Reading Data from a File - Example

This program uses readLine method as opposed to read. It only reads a single line

```
def main():

    customerFile = open('cusomters.txt', 'w')
    customerFile.write("Java \n")
    customerFile.write("Perl \n")
    customerFile.close();

    updatedCustomerFile = open('cusomters.txt', 'r')
    fileContents = updatedCustomerFile.readline();
    updatedCustomerFile.close();
    print(fileContents)

main()
```

Will output:
Java

Writing and Reading Numeric Data

- Strings can be written directly to a file with the write method, but numbers must be converted to strings before they can be written.

- Python has a built-in function named str that converts a value to a string.

- For example, assuming the variable num is assigned the value 99, the expression *str(num)* will return the string ' 99 '.

Writing and Reading Numeric Data


```
def main( ) :
    # Open a file for writing.
    outfile = open('numbers.txt', 'w')
    # Get three numbers from the user.
    num1 = int(input("Enter a number:"))
    num2 = int(input("Enter another number :"))
    outfile.write(str(num1) + '\n')
    outfile.write(str(num2) + '\n')
    outfile.close()



main()
```

Converts numeric data type to string before writing to file

Writing and Reading Numeric Data

- When we read a value from a file it is read as a String.
- Therefore, we need a means of converting that value from a String to the relevant numerical type.
  - int()
  - float()

Writing and Reading Numeric Data

```
def main( ) :
    # Open a file for writing.
    outfile = open('numbers.txt', 'w')
    outfile.write(str(12.3) + '\n')
    outfile.write(str(10) + '\n')
    outfile.close()

    updatedFile =   open('numbers.txt', 'r')
    number1 = float(updatedFile.readline());
    number2 = int(updatedFile.readline());
    updatedFile.close();

main()
```

Program writes a float and integer value to file. When we read the numeric values in the file we have to convert them from a string to an int or float

Convert from Strings to int and float

Using Loops to Process Files

- Files typically used to hold large amounts of data
- Loop typically involved in reading from and writing to a file
  - Use a for loop for writing to a file
  - Use for and while loop for reading from a file

Writing to a File using Loops

```
def main( ) :

    salesFile = open('sales.txt', 'w')
    numDays = 5
    for count in range( 1, numDays + 1 ) :
        # Get the sales for a day.
        sales = int(input( 'Enter the sales for day ' + str(count) + '
'))
        salesFile.write(str(sales) + '\n')

    salesFile.close()

main()
```

Iteratively write the sales amount to a file using a for loop. Sales is a numeric so it must be converted to a string

Read Values from a File using Loops

- Often the number of items stored in file is unknown
- The `readline` method uses an empty string as a sentinel when end of file is reached
  - Can write a while loop with the condition

    `while` *`line`* `!= ''`

Writing to a File using While Loops


```
def main( ) :

    salesFile = open('sales.txt', 'r')
    line = salesFile.readline()

    while line != '':
        amount = float(line)
        print(amount)
        line = salesFile.readline()

    salesFile.close()

main()
```

This programs read from a file that contains a single column of sales figures (float values)

When line is equal to the empty string '' we have reached the end of the file and the loop exits

Notice every time we read a string from the file we convert it to a float

Using Python's `for` Loop to Read Lines

- Python allows programmer to write a `for` loop that automatically reads lines in a file and stops when end of file is reached

  - Format: `for` *`line`* `in` *`file_object`*`:`

    `statements`

  - The loop iterates once over each line in the file

Writing to a File using For Loops


```
def main( ) :

    salesFile = open('sales.txt', 'r')

    for line in salesFile:
        amount = float(line)
        print(amount)

    salesFile.close()

main()
```

Notice no read operation required and for loop automatically exits when the end of the file is reached

Reading Strings from a File

```
nameFile = open("Names.txt", "w")

nameFile.write('Ted'+'\n')
nameFile.write('John'+'\n')
nameFile.write('Frank'+'\n')

nameFile.close()

nameFile = open("Names.txt", "r")
print(nameFile.readline())
print(nameFile.readline())
print(nameFile.readline())

nameFile.close()
```

When we run this program we get the following output(notice the blank line between the names):

Ted

John

Frank

Reading Strings from a File

```
nameFile = open("Names.txt", "w")

nameFile.write('Ted'+'\n')
nameFile.write('John'+'\n')
nameFile.write('Frank'+'\n')

nameFile.close()

nameFile = open("Names.txt", "r")
print(nameFile.readline().rstrip('\n'))
print(nameFile.readline().rstrip('\n'))
print(nameFile.readline().rstrip('\n'))

nameFile.close()
```

rstrip function is used to strip the '\n' from the string we read from the file.

When we run this program we get the following output(notice no blank line between the names):

Ted
John
Frank

Programming Task

- Write a program that will ask the user to enter employee details. The program will first ask the user for the number of employees they wish to record. It should then obtain the name, ID and department of each employee. The program should store this information in a file.

- Write a second program that will open and read the file and write the employee information to the screen.

```
def main ( ) :

    numEmps = input ( 'How many employee records? ' )
    # Open a file for writing.
    empFile = open('employees.txt', 'w')

    for count in range( 1, numEmps + 1):
        print('Enter data for employee #' + str(count))
        name = input ( 'Name: ' )
        idNum = input('1D number: ')
        dept = input('Department: ')
        # Write the data as a record to the file.
        empFile.write(name + '\n')
        empFile.write(idNum + '\n')
        empFile.write(dept + '\n')

    # Close the file.
    empFile.close()
    print('Employee records written to employees.txt.')
```

For each iteration of the for loop we ask the user for the details of an employee and we write those to the file

```
def main ( ) :

    empFile= open('employees.txt', 'r')
    name = empFile.readline()
    while name != '' :
        idNum = empFile.readline()
        dept = empFile.readline()

        name = name.rstrip('\n')
        dept = dept.rstrip('\n')
        idNnum = idNum.rstrip('\n')

        print(' Name : ' , name)
        print(' ID : ' , idNum)
        print(' Dept : ' , dept)

        # Read the name field of the next record.
        name = empFile.readline()

   # Close the file.
   empFile.close()
```

rstrip function strips the \n from the string value we read in the from the file

Read and Writing a List To/From Files

- Some tasks may require you to save the contents of a list to a file so the data can be used at a later time (Use the write method in Python file object).

- Likewise, it may be necessary to read the data from a file into a list (Use the readLines method in Python file object).

- The next two slides show an example of writing a list to a file and reading a list from a file.

Writing a List to a File

```
def main():
    cities = ['New York', 'Boston', 'Atlanta', 'Dallas']
    # Open a file for writing.
    outfile = open('cities.txt', 'w')
    # Write the list to the file.
    for item in cities:
        outfile.write(item + '\n')
    # Close the file.
    outfile.close()

main()
```

cities.txt

```
New York
Boston
Atlanta
Dallas
```

Use of write method to write a list item to a file

Reading a List from a File

```
def main():
    infile = open('cities.txt', 'r')
    cities = []
    city = infile.readline().rstrip('\n')

    while (city != ''):
        cities.append(city)
        city = infile.readline().rstrip('\n')

    infile.close()

    print(cities)
    print(len(cities))
```

Reads each line in file into a String element in the list

rstrip method removes the newline '\n' character from each string item

Reading a Line Containing Multiple Entries

- Assume we have a file called numbers.txt that contains a single line the sequence of numbers

- 55 75 87

- We want to write a program that will read this line of code and add up all three numbers

```
numbers = numberFile.readline().rstrip('\n')

numberList = numbers.split()

total = int(numberList[0])+int(numberList[1])+int(numberList[2])

print(total)

numberFile.close()
```

Notice that the list numberList is a list of Strings. So to sum up the numbers we must convert each string to an int