Intro to Programming (No Prior Experience)

Module 9 Review – Lists, Strings, Working with the Web and an Introduction to File Input & Output

Emily Zhao

T/R 4:55PM–6:10PM

Module 9

- Basic Exception Handling
  - Try/Except/Else
- Reading Data from the Web
  - Splitting Strings Into Lists
  - `urllib.request` module
- Basic File I/O
  - Programs and Memory
  - Filenames and Working with File Objects
  - Writing vs Appending Data to a File
- Additional File I/O Techniques
  - Reading Data from a file
  - Writing Numeric Data to a file
  - Reading Numeric Data from a file

Exceptions

An exception is any error that causes a program to halt while it’s running.

```
x = "Emily"
y = int(x)
```

Exceptions

- When an exception occurs Python will generate a “traceback”
- Tracebacks give information about the exception that occurred, including the line number that caused the issue and the name of the exception
- Programmers generally say that an exception has been “raised” by the program

```
>>> int("Emily")
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    int("Emily")
ValueError: invalid literal for int() with base 10: 'Emily'
```

Exceptions

Many times you can avoid exceptions all together by adjusting how we code our algorithms. For example, the following code will raise an exception if x = 0

```
x = int(input("Give me a number"))
print (100/x)
```

Yet we can avoid the issue by wrapping the potentially dangerous code inside a selection statement:

```
x = int(input("Give me a number"))

if x != 0:
    print (100/x)
```

Exceptions

Python has an exception handling statement that can be used to "catch" exceptions and prevent them from crashing your program

```
try:
    # questionable code goes here
except:
    # this block will run if the code in the
    # 'try' block above raised an exception
else:
    # this block will run if the code in the
    # 'try' block above was successful
```

Trace the Output:

```
try:
    score = int(input("Enter a number: ")) # user enters 84.5
    curved = score + 3
except:
    print("Invalid input.")
else:
    print("Your curved score is: ", curved)
```

> Invalid Input

Strings + Lists

Splitting a String

- Sometimes you are given long strings of information that need to be “parsed” in a particular way
- For example, consider the following string:

```
x = "Emily,Professor,NYU"
```

- This string contains three pieces of data – a name, a title and an employer. The data has been “packaged” into a single string in a meaningful way (i.e. we know that the comma character separates the different data points)

Splitting a String

We can use a technique called “splitting” to “unpack” a string and extract its individual values. The trick is to isolate a “separator” character that can be used to delimit the data you want to extract.

General Algorithm:

1. Obtain a string that contains data organized in a certain way
2. Split the string into a list
3. Process the list using list mechanics

Splitting a String

We can use a technique called “splitting” to “unpack” a string and extract its individual values. The trick is to isolate a “separator” character that can be used to delimit the data you want to extract.

General Algorithm:

1. Obtain a string that contains data organized in a certain way
2. Split the string into a list
3. Process the list using list mechanics

Trace the Output:

```
mystring = "Emily,Professor,NYU"
splitstring = mystring.split(",")
print(splitstring)
```

What data type will be returned?

Trace the Output:

```
mystring = "Emily,Professor,NYU"
splitstring = mystring.split(",")
print(splitstring)
```

What data type will be returned? → **a List**

```
['Emily', 'Professor', 'NYU']
```

Programming Challenge

The following string represents the test scores of a given student.

```
scores = "95,100,67,33,88"
```

Write a program that:

- Prints total # of scores
- Prints out the average score for the student
- Prints out the highest and lowest scores
- Drops the lowest score and prints out the student’s average without that score

Working with External Data

Obtaining Data from the Web

- You generally won't be "hard coding" data into your programs. You will usually be processing information that comes to you in the form of user input or via an external data source.
- One way to easily get information into Python from the "outside world" is to initiate a connection to the Web and obtain data via a URL

Obtaining Data from the Web

General Algorithm:

- Initiate a web connection to an external data source
- Obtain the data that exists at that location as a String
- “Clean up” the string into a meaningful form that your program can understand. This usually involves making sure that the character set used in the external file is one that your computer can handle.
- Parse your string into a list
- Process this list and generate output

Obtaining Data from the Web

```
import urllib.request

# where should we obtain data from?
url = "http://www.cnn.com"

# initiate request to URL
response = urllib.request.urlopen(url)

# read data from URL as a string
data = response.read().decode('utf-8')

print (data)
```

Warning!

The web is a volatile place – servers can go up and down at a moment's notice, and your connection to the network may not always be available

You need to be prepared to deal with connection errors in your programs.

You can use the try / except / else suite to help "catch" errors before they crash your program.

Programming Challenge

Write a Python program that asks the user for a name.

Determine if that name is one of the most 40 popular names for boys or girls in 2014.

There are two data files on the web which represent the most popular boy and girl baby names from 2014. These names are sorted in popularity order (i.e. the first item is the most popular name in each file)

- http://002-text-files.glitch.me/girl-names.txt
- http://002-text-files.glitch.me/boy-names.txt

Issues with urllib.request

- In IDLE you will need to explicitly install a "security certificate" to allow your program to make a connection to a website. You can set this up by doing the following (note that you only need to do this once):
- Click on Finder on a Mac, or File Explorer if on Windows
- Navigate to the directory where Python is installed. On a Mac it will be under the Applications folder. On a PC it will most likely be under C:\Program Files
- Look for the the "Install certificate.command" file -- double click on this file to run it. You should now be able to make web connections via the urllib.request library in your programs

Programming Challenge

This data file contains all World Series winning teams up until a few years ago:

— http://002-009-text-files.glitch.me/world-series.txt

Write a program that reads in this data and finds the team that won the most games

File Input and Output (I/O)

Working with Files

Step 1: Open up a connection between your program and the operating system. Define the name and location of the file you wish to work with as well as what you want to do to that file (read or write)

Step 2: Process the data

- **Writing**: send information in the form of variables from your program into the file to be written
- **Reading**: extract information into variables in your program

Step 3: Close the file, telling the OS to release the file for use by other programs

Reading Data from a File

```
file_object = open(filename, mode)

# File Modes
# 'r' = read
# 'w' = write   - if the file does not exist, create it
#               - if the file does exist, overwrite it
# 'a' = append  - if the file does not exist, create it
#               - if the file does exist, append data to the end of it
```

Opening a file in Python

Filenames are expressed as strings. If you type in a simple filename (i.e. “myfile.txt”) you are telling Python to open up a file with that name in the current directory

You can ask Python to open up a file that exists outside of the current directory by typing in an absolute path. For example:

Windows: ‘C:\temp\testfile.txt’
Mac: ‘/var/local/testfile.txt’

When providing an absolute path you should preface the string with the ‘r’ directive – this tells Python to treat the text in the string as “raw” text and not try and add in any escape characters. Example:

```
file_object = open(r’C:\temp\testfile.txt’, ‘w’)
```

Writing Data to a File

You can write data into a file once you have created a file object and have opened a file for writing using the write() function. Here's an example:

```
file_object = open('myfile.txt', 'w')

file_object.write('Emily')
file_object.write('hello')

# close the file when you're done
file_object.close()
```

Delimiters + Files

You want to try and avoid writing files that concatenate all of your data into one long line of unintelligible text

This is bad practice since it can be very difficult – or impossible – to extract out your data later on

One way to separate data in a text file is by splitting out your data into multiple lines using the “\n” escape character. This allows you to store a single value on each line of your text file, which will make things a lot easier for you later on when you need to read your file and perform some kind of operation on the data contained within

Reading Data from a File

You can read data contained inside a file once you have created a file object and have opened a file for reading using the read() function. The read() function extracts all data from a file as a string and returns a string.

Usually, tt returns one large string that must be further processed before it can actually be used.

Here's an example:

```
file_object = open('myfile.txt', 'r')

alldata = file_object.read()
print (alldata)
# > Emilyhello

file_object.close()
```

Programming Challenge

Part 1:

– Prompt the user for a username and a password

– Store the username and password into a file named “security.txt”

Part 2:

– Prompt the user for a username and password

– If they match the values stored in the previous file, allow them to continue. Otherwise present an error message.

Writing Numeric Data to a File

The write() function only accepts strings – the following code will generate a runtime error:

```
myvar = open("test2.txt", "w”)
myvar.write(55)
myvar.close()
```

You need to convert any data that will be stored in a file into a string by using the str() conversion function. Example:

```
myvar = open("test2.txt", "w”)
myvar.write(str(55))
myvar.close()
```

Programming Challenge

Write a program that opens up a file named “testscores.txt”. This file contains the following information in the following format:

student name
score1
score2
score3

Read in the values and print out the average score for the student specified in the file along with the student’s name.

Using Loops to Write Data to a File

You can use the write() function inside a repetition structure to write multiple values to a file

Programming Challenge

Write a program that asks the user to enter in a series of questions in the following format:

```
Enter a question:  What color is the sky?
Enter an answer:  Blue
Would you like to enter another question? (y/n):  n
```

Write the questions and answers to a text file

Programming Challenge

Write a program that interfaces with your question text file and prompts the user for the answer to each question

If they get a question right you can give them a point. Otherwise they do not earn a point.

At the end of the program you can display the total number of points earned as well as their average score (i.e. 5/10 questions correct = 50%)

Extension: randomize the quiz for the user