413-513 Notes- Week 5

Topics:

Random Number generating functions

-Inventory Search program

Intro to File Processing & Exceptions

**[ Functions – Random number generator ]**

**Ex. Inventory Searching**

import random

from datetime import datetime

print("Inventory Search program")

print()

#Declarations

numbers = [10, 20, 30]

#Random Number Generation

numbersGenerated = 0

while numbersGenerated < 3:

randomNumber = random.randint(10,30)

if randomNumber not in numbers and randomNumber != 27:

numbers.append(randomNumber)

numbersGenerated += 1

print("Random number generated.")

#Console Input and Output

x = 0

while x < 3:

print()

userInput = int(input("Please enter an item number to search the “

“inventory, or enter 0 to list all inventory numbers: "))

if userInput == 0:

print("Current items in the inventory: ", numbers)

elif userInput == datetime.today().day:

print("The Powers That Be have decreed you may not search for an “

“inventory number that is the same as the current day of the “

“month. Please enter another number.")

else:

if userInput in numbers:

print("Item found.")

else:

print("Item not found.")

x += 1

**[ File Processing ]**

[**http://www.tutorialspoint.com/python/python\_files\_io.htm**](http://www.tutorialspoint.com/python/python_files_io.htm)

|  |  |
| --- | --- |
| **Modes** | **Description** |
| r | Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode. |
| rb | Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| r+ | Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| rb+ | Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| w | Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| wb | Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| w+ | Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| wb+ | Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| a | Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| ab | Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| a+ | Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| ab+ | Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| file.closed | Returns true if file is closed, false otherwise. |
| file.mode | Returns access mode with which file was opened. |
| file.name | Returns name of the file. |
| file.softspace | Returns false if space explicitly required with print, true otherwise. |

# Opening a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

print "Closed or not : ", fo.closed

print "Opening mode : ", fo.mode

print "Softspace flag : ", fo.softspace

This produces the following result −

Name of the file: foo.txt

Closed or not : False

Opening mode : wb

Softspace flag : 0

The *close()* Method

The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.

Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

Syntax

fileObject.close();

Reading and Writing Files

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

The *write()* Method

The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.

The write() method does **not** add a newline character ('\n') to the end of the string −

Syntax

fileObject.write(string);

# Open a file

fo = open("foo.txt", "wb")

fo.write( "Python is a great language.\nYeah its great!!\n");

# Close opend file

fo.close()

The *read()* Method

The *read()* method reads a string from an open file. It is important to note that Python strings can have binary data. apart from text data.

syntax

fileObject.read([count]);

ex.

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print "Read String is : ", str

# Close opend file

fo.close()

display

Read String is : Python is

The *readline()* Method

This method returns the line read from the file.

File Positions

The *tell()* method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

The *seek(offset[, from])* method changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.

If *from* is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

Example

Opening a file *foo.txt*, which we created above.

# Open a file

fo = open("foo.txt", "r+")

str = fo.read(10);

print "Read String is : ", str

# Check current position

position = fo.tell();

print "Current file position : ", position

# Reposition pointer at the beginning once again

position = fo.seek(0, 0);

str = fo.read(10);

print "Again read String is : ", str

# Close opend file

fo.close()

-Program for rw file operations follows

**#write to file**

**#from sys import argv**

**#script, filename = argv**

**filename="foo.txt"**

**print ("We're going to erase %r." % filename)**

**print ("If you don't want that, hit CTRL-C (^C).")**

**print ("If you do want that, hit RETURN.")**

**input("?")**

**print ("Opening the file...")**

**target = open(filename, 'w')**

**print ("Now I'm going to ask you for three lines.")**

**line1 = input("line 1: ")**

**line2 = input("line 2: ")**

**line3 = input("line 3: ")**

**print ("I'm going to write these to the file.")**

**target.write(line1 + "\n")**

**target.write(line2)**

**target.write(line2)**

**target.write("\n")**

**target.write(line3)**

**target.write("\n")**

**print ("And finally, we close it.")**

**target.close()**

**#read from file**

**fobj = open("foo.txt") # Returns a file object**

**line = fobj.readline() # Invokes readline() method on file**

**while line:**

**print(line,end='') # Use in Python 3**

**line = fobj.readline()**

**fobj.close()**

**[ Built-in exceptions ]**

<https://docs.python.org/3/library/exceptions.html>

#Number validator

**while True:** #use as do loop logic, need break to quit

try:

**x = int(input("Please enter a number: "))**

**break**

**except ValueError:**

**print "Oops! That was no valid number. Try again..."**

-Work quick review 3 (Chapters 5-6)

**Next class**:

list and tuples!