# CS 171 - Lab 7

Professor Mark W. Boady and Professor Adelaida Medlock

*Content by Professor Lisa Olivieri of Chestnut Hill College*

Detailed instructions to the lab assignment are found in the following pages.

- Complete all the exercises and type your answers in the space provided.

What to submit:

- Lab sheet in PDF.
- **Screenshots with your code for questions 6, 5-k, 7 and 11**

Submission must be done via Gradescope

- Please make sure you have tagged all questions to PDF pages and added your lab partner if any.
- We only accept submissions via Gradescope

**Student Name:** Tony Kabilan Okeke

**User ID (abc123):** tko35

**Possible Points: 88**

**Your score out of 88**:

**Lab Grade on 100% scale:**

**Graded By (TA Signature):**

Question 1: 3 points

Create a text file named **`sports.txt`** and enter the sports listed below, one word per line. Enter and execute the program below. Be sure the saved program is in the same folder as the text file.

**FYI:** In Python, you can access data from a text file as well as from the keyboard. You can create a text file in any text editing tool. You should only have one data item per line in the file.
If you use Notepad (Windows) or TextEdit (Mac) make sure your settings are for Plain Text not Rich Text. You can also use Thonny to save a `.txt` file

**Python Program**

```
1   def main() :
2      sportsList = open('sports.txt')
3      sp = sportsList.read ()
4      print(sp)
5
6   ############ Call to main() ############
7   main()
```

**sports.txt**

```
basketball
baseball
football
volleyball
tennis
golf
soccer
hockey
swimming
rowing
```

(a) (1 point) What does the program do?
The program reads the contents of sports.txt and prints it in the command line.

(b) (1 point) In the Line 2 of code, what does the string argument for the function **`open()`** represent?
It represents the name and path of the file to be opened.

(c) (1 point) Replace the call to the function **`read()`** with the function **`readline()`**. Execute the program again. Explain the difference between the two functions: **`read()`** and **`readline()`**.
The read() function returns the whole file as a single string while readline() returns a single line of the file (in this case, it returns the first line).

Question 2: 10 points

Enter and execute the following code.

```
sportsList = open('sports.txt')
for index in range(1, 11) :
    sp = sportsList.readline ()
    print( str ( index ) + ". ", sp)
```

(a) (1 point) How does the output from this program differ from the output of the program that used the **read()** function?
This program appends "{index}. " to the beginning of each line of the file and prints that in the command window. This causes an additional linebreak ("\n") between each line.

(b) (1 point) What caused the difference?
Each line in the sports.txt file terminates with a linebreak ("\n"). The print function – in addition to printing "{index}. " in front of the line – also prints a linebreak at the end of the line. This causes the empty lines to be printed in the command line between lines from the file.

(c) (1 point) What is the subtle difference in the output if the following print statement replaced the one above? **print(index, ". ", sp)**
In this case, a space is printed between the index and the period, "."

(d) (1 point) Which is better?
print( str(index) + ". ", sp) is better.

(e) (1 point) What does **str(index)** do in the program above?
It converts the index variable to a string so that it can be concatenated to ". "

(f) (1 point) Why is the **str()** function necessary?
index is an integer. Integers cannot be added to strings. Therefore, index must first be converted to a string.

(g) (1 point) What happens when you change the arguments in the **range()** function to **1, 10**?
The indices printed will be 0 through 9 instead.

(h) (1 point) What happens when you change the arguments in the **range()** function to **0, 30**?
Indices from 0 through 29 will be printed. Also, after index = 9, no lines are printed next to the index since all the lines in the file have been read.

(i) (2 points) What do the results from (g) and (h) tell you about the arguments of the **range()** function when you are reading data from a file with a **for** loop?
The arguments to range() need to define a range of values that is shorter than, or as long as the number of lines in the file being read. If a larger range is specified, only empty strings will be returned once the lines in the file have been exhausted.

**FYI:** The purpose of the **`rstrip()`** function returns a copy of the string after all characters have been stripped from the end of the string (default whitespace characters, EOL characters, and newline characters).

Question 3: 10 points

The following program is slightly different from the program in Question 2.

```
sportsList = open('sports.txt')
for index in range(1, 11) :
   sp = sportsList.readline ()
   print( str ( index ) + ". ", sp.rstrip () )
```

(a) (2 points) Compare the output from this program to the previous program. What is the difference?
The output of this program does not include empty lines between the lines of sports.tx whereas the program in problem 2 did include the empty lines.

(b) (2 points) What code caused the difference in the output?
Calling sp.rstrip() removed the linebreak characters at the end of the lines from sports.txt

(c) (2 points) Does the **`rstrip()`** function contain any arguments?
In the program above, no arguments are passed to the rstrip() function. When no arguments are passed, it removes whitespace chartacters.

(d) (2 points) How does it know what string to act upon?
rstrip() is a string method, as such, the object on the left hand side of the dot (.) specifies the string being acted upon.

(e) (2 points) **`lstrip()`** is a similar function. What do you think it does?
By default, the lstrip() function removes whitespace, EOL and newline characters from the beginning of a string. If an argument is passed to it, it all occurrences of the argument string from the beginning of the string.

Question 4: 6 points

The following program is slightly different from the program in Question 3.

```
sportsList = open('sports.txt')
for index in range(1, 11) :
   sp = sportsList.readline ()
   if len(sp) >= 8:
      print(sp.rstrip())
```

(a) (2 points) How is the output different from Question 3
In this case, indexes are not printed in front of the lines of sports.txt.
Also, only lines that are 8 characters or longer are printed.

(b) (2 points) Two functions use what is known as dot (.) notation. What are the two functions?
readline()
rstrip()

(c) (2 points) Examine the output and explain what the **`len()`** function does.
The len function returns the length of it's argument. In the program above, it is used in an if statement that checks whether a line is at least 8 characters long.
Only lines that satisfy this requirement are printed.

Question 5: 27 points

Enter and execute the following program.

```
1   def main() :
2      lastName = input("Enter last name: ")
3      firstName = input("Enter first name: ")
4      studentID = input("Enter ID: ")
5
6      inFile = open("studentInfo.txt", "a")
7      inFile.write ("Name: " + firstName + " " + lastName)
8      inFile.write ("\nStudentID: " + studentID )
9      inFile.write ("\n")
10     inFile.close ()
11     print("Done! Data is saved in file: studentInfo.txt")
12
13   ############ Call to main() ############
14   main()
```

(a) (4 points) Which lines print to the screen?
Lines 2, 3, 4 and 11

(b) (2 points) What does the program do?
The program prompts the user to enter a students last and first names, and student ID.
It appends this information the studentInfo.txt file, then prints a completion message.

(c) (2 points) Locate the file **studentInfo.txt** on your computer. The file is stored in the same folder as the program. What is stored in the file?
The file contains the student name and ID I entered when I first ran the program.

Name: Jane Doe
StudentID: 123456789

(d) (2 points) Run the program again. How did the **studentInfo.txt** file change?
The file now contains name and ID provided when the program was run the second time.

Name: Jane Doe
StudentID: 123456789
Name: John Doe
StudentID: 987654321

(e) (2 points) Change the argument "a" to "w" in the call to the open function. Run the program a third time. Explain what "a" to "w" do.
"a" opens a file in append mode – calling the write() function adds content to the end of the file.
On the other hand, "w" opens the file in write mode – calling the write() function overwrites the contents of the file.

(f) (2 points) Did you need create the file **studentInfo.txt** separately from the program code or did the program create it?
The file was created by the program

(g) (2 points) Notice the function **write()**. How many arguments does this function have?
The write() function takes one argument – some text to write to the file.

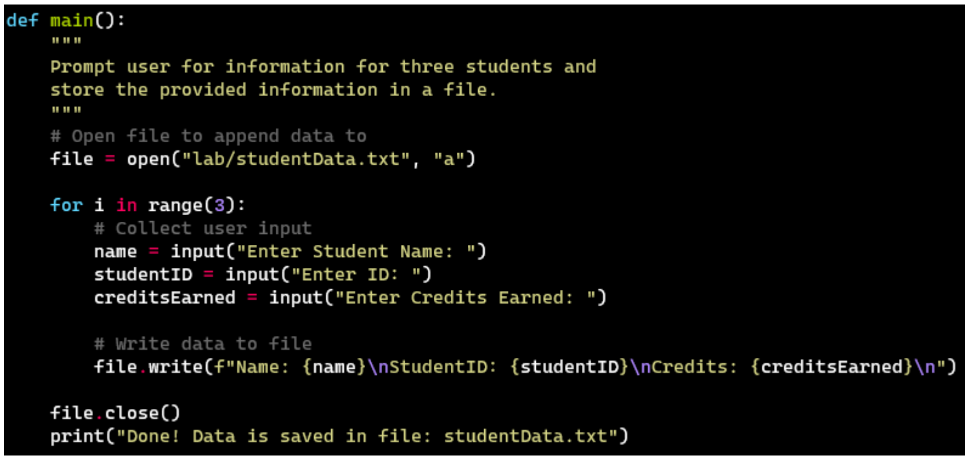
(h) (2 points) How does the **write()** function know what file to write to?
The write() method is called on the inFile object, as such it writes to the file contained in the inFile object.

(i) (2 points) What line of code closes the file?
Line 10: inFile.close()

(j) (2 points) Where is the line of the code that closes the file positioned in the program?
It is contained within the definition of the main() function (Line 10).

(k) (5 points) Write a program that prompts the user for information for three students: name, student ID, number of credits earned. Store the information in a file. You can use the above code as a template for your solution. Describe how you revised the code to solve this problem. Attach a screenshot showing your solution.

I added a for loop to prompt the user for student information three times and write the data to the file. I also used a single write statement with a formatted string to write all the data for a particular student at once.



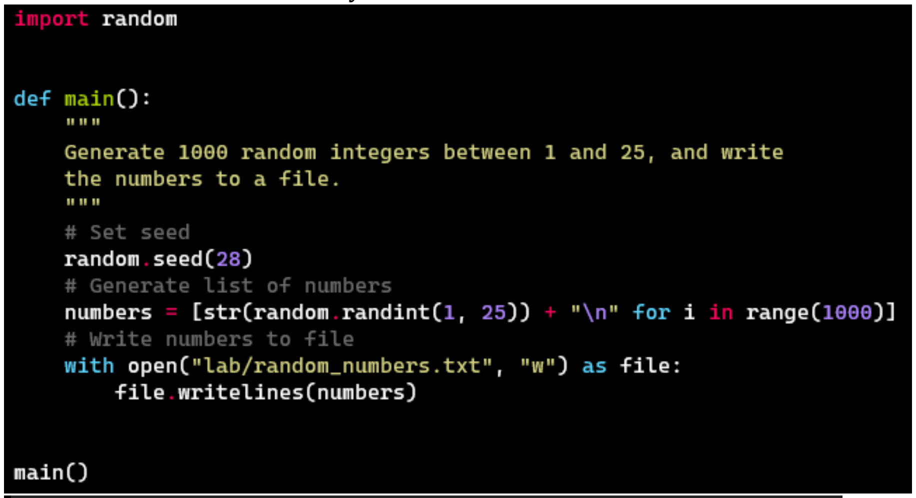
```
def main():
    """
    Prompt user for information for three students and
    store the provided information in a file.
    """
    # Open file to append data to
    file = open("lab/studentData.txt", "a")

    for i in range(3):
        # Collect user input
        name = input("Enter Student Name: ")
        studentID = input("Enter ID: ")
        creditsEarned = input("Enter Credits Earned: ")

        # Write data to file
        file.write(f"Name: {name}\nStudentID: {studentID}\nCredits: {creditsEarned}\n")

    file.close()
    print("Done! Data is saved in file: studentData.txt")
```

## Application Questions

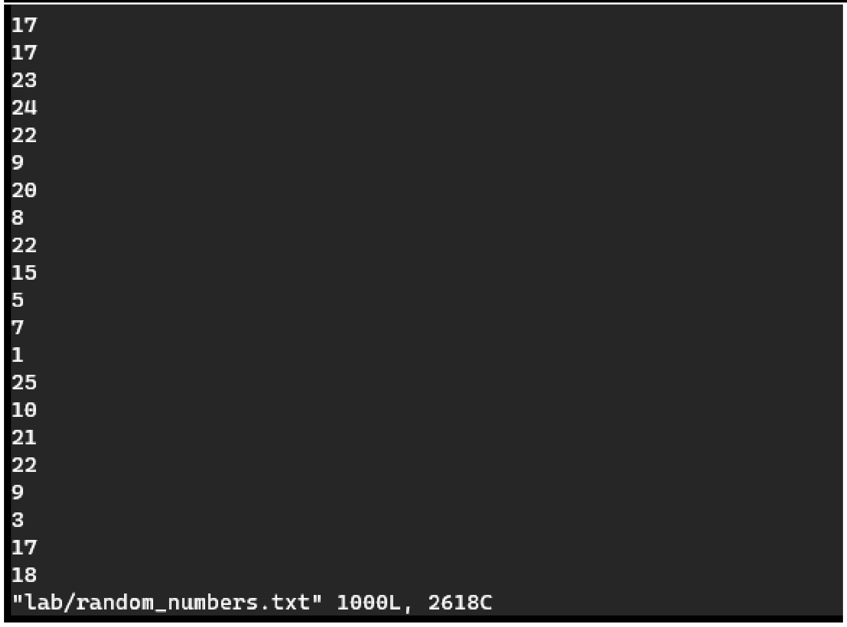
**Use the Python Interpreter to check your work.**

Question 6: 5 points

Write a program that randomly generates 1000 integers with values between 1 and 25. Writes the numbers to a file, one number per line. Seed the random number generator with the value 28.
Put a screenshot of your source code and output text file. Only show part of your output file, you do not need to show all 1000 lines. You may submit more than one screenshot if needed to show all your source code.



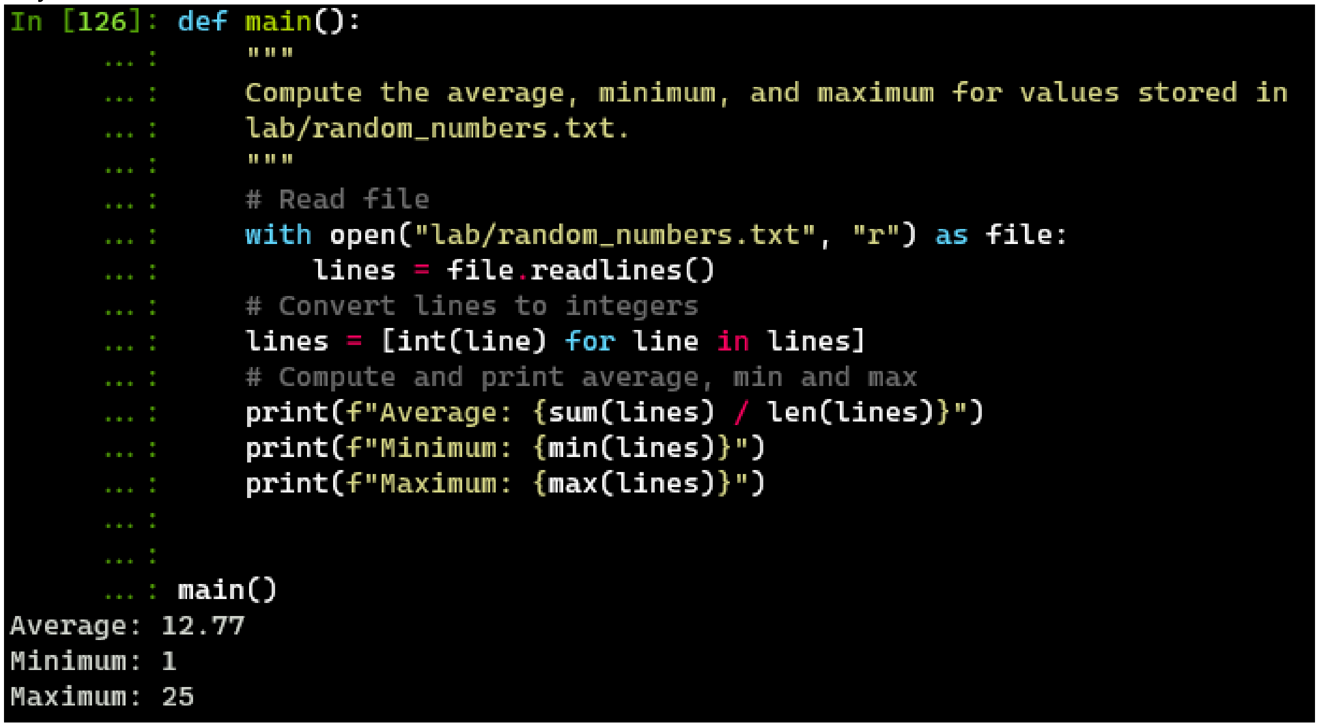
```python
import random


def main():
    """
    Generate 1000 random integers between 1 and 25, and write
    the numbers to a file.
    """
    # Set seed
    random.seed(28)
    # Generate list of numbers
    numbers = [str(random.randint(1, 25)) + "\n" for i in range(1000)]
    # Write numbers to file
    with open("lab/random_numbers.txt", "w") as file:
        file.writelines(numbers)


main()
```



```
17
17
23
24
22
9
20
8
22
15
5
7
1
25
10
21
22
9
3
17
18
"lab/random_numbers.txt" 1000L, 2618C
```

Question 7: 5 points

Write a program that reads the file you created in Question 6. Open the file and read in the values. Compute the Average, Minimum, and Maximum number from the file.
Put a screenshot of your source code and output. You may submit more than one screenshot if needed to show all your source code.



```
In [126]: def main():
     ...:     """
     ...:     Compute the average, minimum, and maximum for values stored in
     ...:     lab/random_numbers.txt.
     ...:     """
     ...:     # Read file
     ...:     with open("lab/random_numbers.txt", "r") as file:
     ...:         lines = file.readlines()
     ...:     # Convert lines to integers
     ...:     lines = [int(line) for line in lines]
     ...:     # Compute and print average, min and max
     ...:     print(f"Average: {sum(lines) / len(lines)}")
     ...:     print(f"Minimum: {min(lines)}")
     ...:     print(f"Maximum: {max(lines)}")
     ...:
     ...:
     ...: main()
Average: 12.77
Minimum: 1
Maximum: 25
```

Question 8: 4 points

Examine the following program. It includes a function that takes a list as an argument. Enter and execute the code

```
import random

def orderList ( newList ) :
    newList.sort ()
    newList.reverse ()
    return newList

myList = []
for y in range(0, 100) :
    myList.append(random.randint (1, 100) )
print( orderList (myList) )
```

(a) (2 points) What is the name of the function defined in this program?
orderList

(b) (2 points) What does the function do?
The function sorts a list and then reverses it. Therefore, it returns a list sorted in descending order.

Question 9: 4 points

Enter and execute the following code.

```
userInput = input("Enter a string that contains only letters: ")
if userInput.isalpha () :
   print("Your string is valid.")
else :
   print("Your string does not contain all letters.")
```

(a) (2 points) Execute the program with different inputs and examine the output. Determine what the program does experimentally. Describe what this program does.
The program checks if the user input contains only letters (no numbers or other characters). If it does, it prints that the string is valid. Otherwise, it tells the user the string doesn't contain only letters.

(b) (2 points) What does the **isalpha()** function do?
It checks whether a string contains only alphabetic characters (upper and lowercase letters)

Question 10: 4 points

Enter and execute the following code.

```
numString = input("Enter a number: ")
if numString.isdigit () :
    num = int (numString)
    print(num, "to the fourth power is", num ** 4)
else :
    print("Your input is not a valid number.")
    print("Program terminated!")
```

(a) (2 points) Execute the program with at least five different types of input. Examine the output for each input. What does the program do?
The program checks if the user input (numString) contains only digits (numbers from 0-9). If it does, the input is converted to an integer, and prints the number raised to the fourth power in a formatted string. Otherwise, it tells the user the input is not a valid number, and prints "Program terminated!" in the command window.

(b) (2 points) What does the **isdigit()** function do?
It checks whether a string contains only digits (numbers from 0 to 9).

## Application Questions

**Use the Python Interpreter to check your work.**

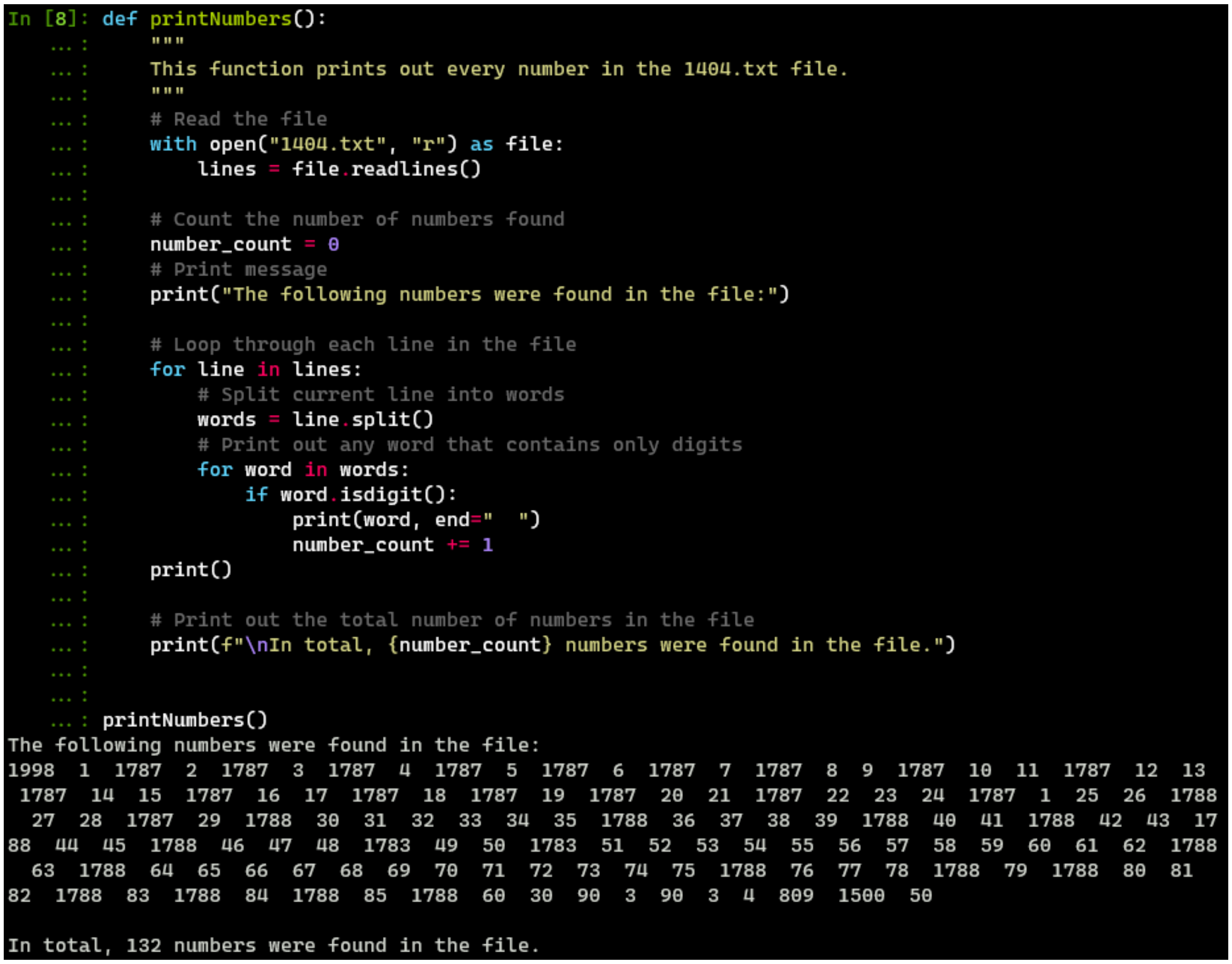
Question 11: 10 points

Download the following book from **Project Gutenberg**: File 1404.txt from:

http://www.gutenberg.org/files/1404/

Write a program that prints out every number in the file. Look at every line, then split into words. Print out any word where **`isdigit`** returns **`True`**. At the end of the program, print the total number of numbers found in the file.

Submit screenshots of your code and its output. You may submit multiple screenshots if required to show all the source code and output.



```
In [8]: def printNumbers():
   ...:     """
   ...:     This function prints out every number in the 1404.txt file.
   ...:     """
   ...:     # Read the file
   ...:     with open("1404.txt", "r") as file:
   ...:         lines = file.readlines()
   ...:
   ...:     # Count the number of numbers found
   ...:     number_count = 0
   ...:     # Print message
   ...:     print("The following numbers were found in the file:")
   ...:
   ...:     # Loop through each line in the file
   ...:     for line in lines:
   ...:         # Split current line into words
   ...:         words = line.split()
   ...:         # Print out any word that contains only digits
   ...:         for word in words:
   ...:             if word.isdigit():
   ...:                 print(word, end="  ")
   ...:                 number_count += 1
   ...:     print()
   ...:
   ...:     # Print out the total number of numbers in the file
   ...:     print(f"\nIn total, {number_count} numbers were found in the file.")
   ...:
   ...:
   ...: printNumbers()
The following numbers were found in the file:
1998  1  1787  2  1787  3  1787  4  1787  5  1787  6  1787  7  1787  8  9  1787  10  11  1787  12  13
 1787  14  15  1787  16  17  1787  18  1787  19  1787  20  21  1787  22  23  24  1787  1  25  26  1788
  27  28  1787  29  1788  30  31  32  33  34  35  1788  36  37  38  39  1788  40  41  1788  42  43  17
88  44  45  1788  46  47  48  1783  49  50  1783  51  52  53  54  55  56  57  58  59  60  61  62  1788
  63  1788  64  65  66  67  68  69  70  71  72  73  74  75  1788  76  77  78  1788  79  1788  80  81
82  1788  83  1788  84  1788  85  1788  60  30  90  3  90  3  4  809  1500  50

In total, 132 numbers were found in the file.
```