File Input & Output in Python

October 19, 2022

Administrative Notes

Some homework 5 notes

Remember it’s due next Monday, the 24th

Number bases

As humans, we tend to count things by powers of 10: 10, 100,1000,..

We have only ten symbols, or digits, used to count

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Why? Because most humans have ten digits - fingers, or toes - and that made it easy to keep track of things

While that's the best way for humans to count, it's not the best way for machines to count

Why not?

Most computers (and similar machines) are made up of electrical switches.

Electrical switches have two states: ***ON*** and ***OFF***

We can say that if something is ON it has the value 1 and if it is OFF it has the value 0.

This is called “binary” numbering. We only need two digits, 0 and 1, to represent all numbers, using powers of 2. So, there are only two binary digits

“***Bi***nary digi***t***” was a pain to say all the time, so it got shortened to … “bit”

There are 10 types of people in the world - those who understand binary and those who don't.

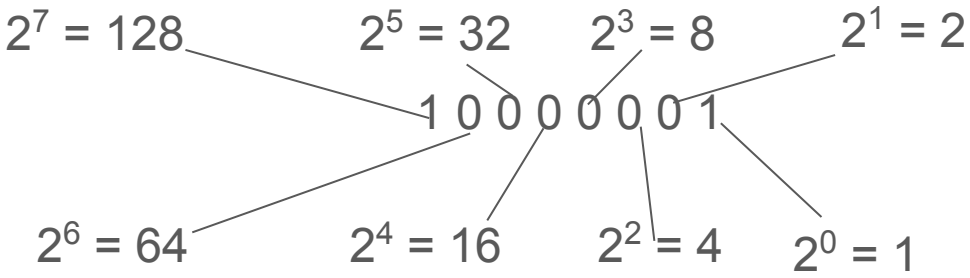
```
147355  - 5 * 10**0 + 5 * 10**1 + 3 * 10**2 + 7 * 10**3
+ 4 * 10**4 + 1 *10**5
```

A binary number

1 0 0 0 0 0 0 1 in base 10 = 129

What's it's value in human-speak i.e. decimal?

Places mean the same as in decimal - the "least significant" is on the right and each place to the left is one power of two greater



Awkward! Binary numbers get big and hard to deal with

Luckily, there's another numbering system: base 16, or "hexadecimal"

- "hex" for short

We need 16 different symbols to represent hexadecimal "digits"

We have 0,1,2,3,4,5,6,7,8,9 - that's 10. We need six more. So we use the first six letters of the alphabet.

A in hex = 10 in decimal; B = 11; C = 12; D = 13; E = 14 and F = 15. That's all we need, because 16 in decimal is 10 in hex. We go by powers of 16

Note: you will find people who prefer upper case A, B, C D, E, and F in their hex numbers; and others who want lower case a ,b, c, d, e and f. For Homework 5, it will be upper case.

HW5 Problem 1

You’re going to be given a string and you’ll need to ensure that each character in it is a valid hex digit. That is, is each character you encounter in [0,1,2,3,4,5,6,7,8.9, ‘A’. ‘B’, ‘C’, ‘D’, ‘E’, ‘F’]

Read each character from the string. If it’s in that list - if it’s a valid Hex digit - print that out. Otherwise print that it’s not.

HW5 Problem 2

Now you have to evaluate a hex number - you have to say what it's value is in decimal.

Start with the least significant digit. That's the "ones" place. Its value is whatever number is there.

Now move one place to the left. Multiply whatever's there by 16 - or 16 to the 1st power

Move left one place again. Multiply whatever's there by 16**2 or 256.

Keep going until you're done.

0xABCD = 13 + (12 * 16) + (11 + 16**2) + (10 * 16**3)
= 43,981

0x1234 to decimal = 4 +(3 * 16) + (2 * 16**2) + (1 * 16**3)
= 4,660

From last time - create a 2D list from scratch

Create a 2D list with the first 50 squares in order. Make there be 5 rows of 10 numbers.

You have to do this by adding a row at a time

Now new material

Input and Output in Python

Up until now, all input has been from the keyboard and all output has been to the screen

X = input(“kindly hand over the data I need.”)

- Everything comes in as a string; you have to cast it to another type if you need integers, floats, etc.

print (“the output is”, output)

That has limited usefulness

It’s a pain in the neck to ask the user to type in data, over and over and over…

You can’t easily preserve your program results for later use, unless you do a whole bunch of screenshots

It would be far better if you could have all the data stored in a file that was saved on a computer, so it didn’t have to be typed in

And it would help if you could save your output in a file that you could reference later on, or even use for further processing

Fortunately, you can!!!

File Input in Python

You can read from any file that can be found on the computer - that is in the “Python path”

- Exactly how this works varies by operating system, but there is an environment variable that can be set that tells the Python interpreter where to look for any files you reference

The easiest thing to do is just make sure that your data files are in the directory in which you’re working

- Or you can specify the full path in the **open** statement

Opening a file

Before you can read from or write to a file in Python, you have to open it

There are several different open statements; we'll use

```
with open("filename", "mode") as f:
        #rest of your code indented one tab
```

"

"Mode" is one of:

- "r" if you just want to read from the file
- "r+" if you want to both read from and write to the file - modify existing contents
- "w" if you want to write to the file - ERASE EXISTING CONTENTS!!
- "a" if you just want to append to the file - add new content without modifying existing content

Modes

- If you leave off the mode value, it defaults to “r” - you can read from the file but you can’t modify it
- Using “w” gives you a new, empty file!! If the file previously existed, any content that was there before will be erased.
- Using “a” does not erase existing content; it just moves the “stream pointer” to the end of the existing file and adds from there

Text mode

For our purposes, all files are just text. They consist of one or more strings and lines of characters. All read commands return strings.

There are “binary” files in Python that are different, but we’re not going to talk about them right now.

Reading from a file

File name in the current directory or in the path

```
with open("test_data.txt", "r") as f:
    data = f.read()  #reads in all data from the file as a single string
    data_lines = f.readlines() # reads in all data from the file
                               # returns a list of strings - each element in the
                               #list is one line from the file
    next_line = f.readline() #reads in one line from the file; returns it as a string
```

Use one of these three

If you want to give a path name for the file:

On a Mac:

with open (“/Users/alfredarsenault/PycharmProjects/Fall_2021/next_test.py”, “r”) as f:

This will work fine because MacOS uses the forward slash / which is not a reserved character in Python.

On Windows:

```
with open("C:\Users\Dad\PycharmProjects\next_test.py", "r") as f:
    s = f.read()
```

Results in:

```
C:\Users\Dad\PycharmProjects\exam_qs\venv\Scripts\python.exe C:/Users/Dad/PycharmProjects/exam_qs/new_dem.py
  File "C:/Users/Dad/PycharmProjects/exam_qs/new_dem.py", line 1
    with open("C:\Users\Dad\PycharmProjects\next_test.py", "r") as f:
              ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

Process finished with exit code 1
```

You have to escape the ‘\’ character

```
with open("C:\\Users\\Dad\\PycharmProjects\\next_test.py",
"r") as f:
    s = f.read()
```

Linux is like Mac; it uses / which doesn’t have to be escaped. But make sure your code runs on gl!!

How it works

There is a ***file pointer*** for any open file, that points to the next character to be read or the place where the next item will be written.

When you open a file for reading, the file pointer starts at the beginning of the file.

Each time you read a line, the file pointer is advanced to the start of the next line.

When you open a file for writing, the file pointer starts at the beginning of a new file.

When you open a file for appending, the file pointer starts at the end of the current content

Reading a file - an example

```
# read in a file of integers, and print the largest integer

with open(“integers.txt”, “r”) as f:
        #read the whole file in as a single string
        data = f.read()
        #split the string into components at whitespace
        numbers = data.split()
        #we have a list of strings, each of which must be converted to integers
        for i in range(len(numbers)):
                numbers[i] = int(numbers[i])
        #now find the largest integer. Start by
        #setting the “largest” to the first value
        largest = numbers[0]
        #now see what’s bigger
        for i in range(len(numbers)):
                if numbers[i] > largest:
                        largest = numbers[i]
        # now print out the largest value
        print(“The largest integer in the file was”, largest)
```

What if we wanted to write the result to a file?

“print” puts the output on the screen. What if we wanted it to go to a file?

The file.write() method will work

```
with open(“results.txt”, “w”) as out:
        out.write(“The largest integer in the file is” + largest)
```

Next example: read from & write to the same file

Read data from a file

Manipulate the data - replace every occurrence of the word “dog” with the word “cat”

Write the modified data back out to the same file we started with

Be careful how you do this! You don’t want to erase all the data in the file before you’ve finished reading it!!!

IF YOU’RE GOING TO READ AND WRITE THE SAME FILE; OPEN FOR READING; CLOSE; THEN OPEN FOR WRITING - less dangerous than opening for r+

If we have time..

Using .csv (comma-separated value) files

Using .json (JavaScript Object Notation) files

(if no time today, we'll briefly cover these Wednesday)