Input and Files

15-110 – Friday 09/20

Learning Goals

Use **programming** to specify algorithms to computers

- Interact with **user input** through interpreter input and files

Understand how computers **organize** data at a low level

- Computers use **folders and files** to store data hierarchically
- Files store data in strings using **specific formats**

User Input

Getting Input from the User

Up until now, we've only written programs that draw input from the editor directly. But we can change this so that users can actively enter information while the program is running!

The built-in function **input(msg)** displays a message in the interpreter, lets the user type a response in the interpreter, and then **returns the response** as a string when the user presses enter.

```
name = input("Enter your name: ")
print("Hello, " + name + "!")
```

input() returns a string

Note that input() will always return a string- if we want to use a user's response as a number, we need to use type-casting to change it.

```
age = int(input("Enter your age: "))
print("You'll be", age + 1, "next year")
```

Note that the user sometimes enters unexpected whitespace at the beginning or end of a response. The built-in function `s.strip()` removes any whitespace at the beginning and end of a string, which may prove useful!

Example: Guessing Game

Now that we have input(), we can start programming interactive programs! Let's begin with a guessing game.

To let the program think of a random number between 1-10, we need to implement randomness as well. We'll do this using the built-in library **random**. We'll talk about this library a lot more at the end of the semester.

```
import random
num = random.randint(1, 10)
```

You can find the code for the guessing game posted on the website after class.

Files

Files as Input

We're not limited to receiving input in real time from users. We can also parse input from files on the computer!

To do this, we first need to understand how files are represented in a computer system.

File Representation

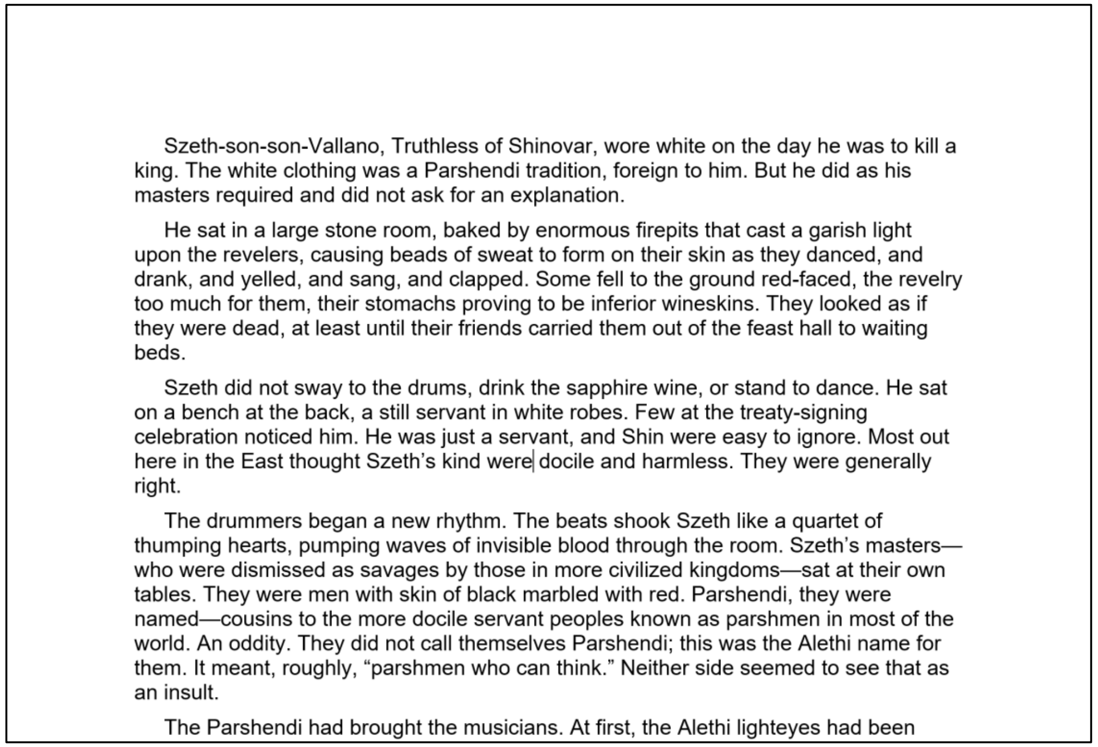
Like every other data type we've discussed, files on your computer are represented at the lowest level as **bytes**. How the computer interprets those bytes depends on the file's **filetype**.

You can look at your own computer's files to see many different filetypes! .docx, .csv, .png, .py, .txt, etc., are all different **encodings** that can be used to interpret the contents of a file.

Your computer uses the **file extension** to determine which application should open it. The application also uses the extension to determine how it should read the file. If you use the wrong file extension or open with the wrong application, the application might show you nonsense instead!

Application Comparison

.docx in Microsoft Word



Szeth-son-son-Vallano, Truthless of Shinovar, wore white on the day he was to kill a king. The white clothing was a Parshendi tradition, foreign to him. But he did as his masters required and did not ask for an explanation.

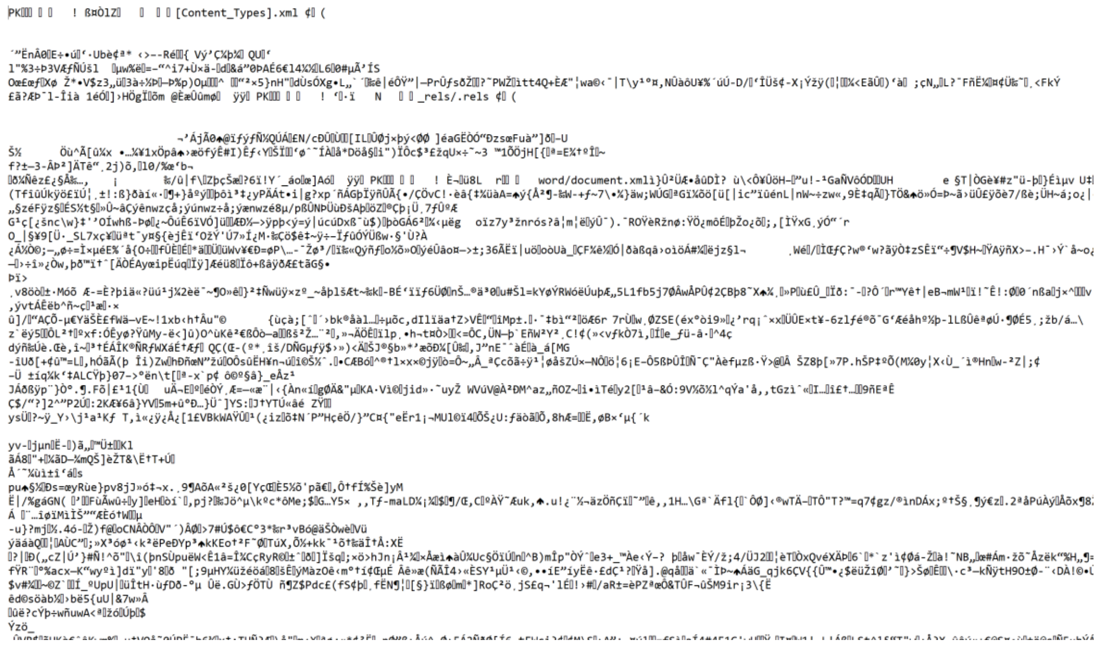
He sat in a large stone room, baked by enormous firepits that cast a garish light upon the revelers, causing beads of sweat to form on their skin as they danced, and drank, and yelled, and sang, and clapped. Some fell to the ground red-faced, the revelry too much for them, their stomachs proving to be inferior wineskins. They looked as if they were dead, at least until their friends carried them out of the feast hall to waiting beds.

Szeth did not sway to the drums, drink the sapphire wine, or stand to dance. He sat on a bench at the back, a still servant in white robes. Few at the treaty-signing celebration noticed him. He was just a servant, and Shin were easy to ignore. Most out here in the East thought Szeth's kind were docile and harmless. They were generally right.

The drummers began a new rhythm. The beats shook Szeth like a quartet of thumping hearts, pumping waves of invisible blood through the room. Szeth's masters—who were dismissed as savages by those in more civilized kingdoms—sat at their own tables. They were men with skin of black marbled with red. Parshendi, they were named—cousins to the more docile servant peoples known as parshmen in most of the world. An oddity. They did not call themselves Parshendi; this was the Alethi name for them. It meant, roughly, "parshmen who can think." Neither side seemed to see that as an insult.

The Parshendi had brought the musicians. At first, the Alethi lighteyes had been

.docx in Notepad



PK ! ß¤Ò1Z [Content_Types].xml ¢ (

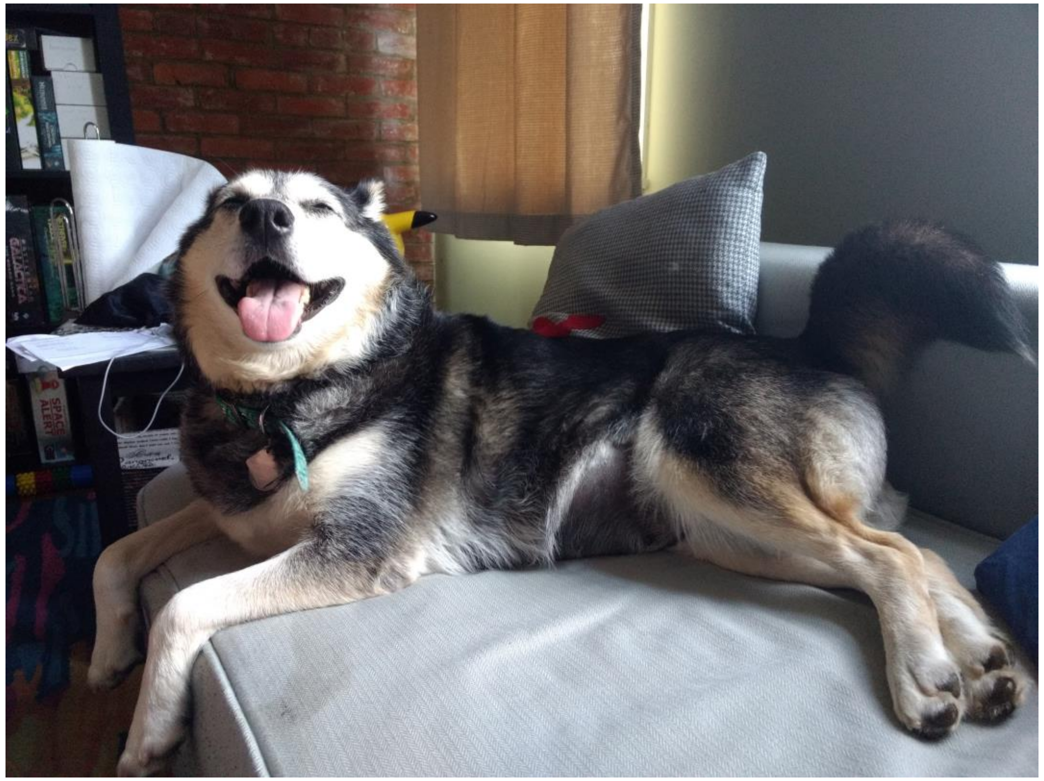
[illegible] _rels/.rels ¢ (

[illegible] word/document.xml [illegible]

[illegible]

Application Comparison

.png in Image Viewer



.png in Notepad

```
‰PNG
▯

IHDR   ▯
€▯     =▯ãW         pHYs   ▯6   ▯6▯¿▯Øz    ▯tEXtTitle PDF CreatorA^¼(
 ▯tEXtAuthor PDF Tools AG▯Ïw0    -zTXtDescription   ▯™Ë
())°Ò×///×+HIÓ-ÉÏÏ)ÖKÎÏ▯ nŸ▯ñ—
¯,¸▯▯÷ŠIDATxÚ”½YdÉ•&vw_ÃcË½*‹Åb‘Ó= ^5ƒ~ë×þç▯▯’€▯šb7›Ã"³*÷Ø|÷»é
[Ì,,z› Yˆôp¿~¯Ù±³~ß9ùõ÷¿ëº®,ë¬È7›MQ▯Óé´®ë¾ï÷û}{<TE±
\.·ëÍd2¹¾¾>lÖŸ¿~}ûöm>ŽíñtﾠIs;ÉËyÕd]?-
k¼³lêûÝÃÝú¡Í†▯×™4mÛžN§"Ë²¡ŸàÒe…ÿ•y^â[ëúáááÕ›7ï?
~¸½¿ËŠ"k¦³Ét>ŸWy‘c•g‡ýþîîîB?¸Îf»½}¸ŸÍfe]åy¾X-p…
b▯ýTUµÛíªªÆ/c‘ó{‹b¿;Žã¸Z-N]ÛµG|
/òFp▯E9⮝ý~·Û><dyþæÛoÏÎÎV‹%îù—_~Á·ìw▯x[Q•▯.Ç▯VX™ÃáÐö▯▯
k2⮝YÖ▯¸ÚñxüŠ•ùþ-Þ†%ÅÛ>~üøÿøøå¯▯ýë~▯œ/▯øÁWã¯¸>n    KrÊxÛø—
*ó onª▯k»œ/pÃ‡ín³^s¹Ê2Füµ,p›9>8©°G#▯Ë
,z÷Ë/Óùl¶ZÖMƒ•ßmßám¸sÜ'–▯_±ßîðmø–¡íp‰ª(ç“i>fØšÃnGÀ”“&/ù¤Ý8
´Cÿ°Ýlv[lG?Žx▯n▯×
„œà¿Ó†;;ÃJ▯e>▯¸TÓ4WÏŸmN‡¯ëõ▯,èa7].öÇCßvËù_1výÅò o▯Ûîþë͡óÕåùbùùç
÷/Ÿ=ºÏØ▯▯–yÈÆC×aup▯§aÄëXðlèð▯MC▯Ës.Ýt1Ç
c▯V▯ç?ÿû¿▯ÿã]7¼ûË_Š¦ùÝï~wq±Ún·x'>xsssyy‰%ÅÍ▯þü▯B,Û€äœÖßþð▯®
‰W†¡›Ö
Ö▯▯Š=Â¢A pKØ;Hæ˜g¸ÈÅ³▯øåÝ‡ŸO}7™ÏŠIÝÀrÍ›Éj1ÏŽm…·¶ín½
ce&Õ▯;ŽûÄzâ▯!´§S‡ïÂ»ðS5µ¥▯+Ù÷▯ƒvðÊá°ŸLkHËçÏqÛx(¬@žA^
+È'>ˆ5Á¶B>ýòŸýöÍ7¸Õõý▯▯òáÃ‡ï▯ø5d¯ûÅÙ▯«ÿåîö4ô”a,iïelž={†à
```

Byte Interpretation

When we try to open a .docx or .png file in Notepad, Notepad interprets it as if it were **plaintext**; in other words, it interprets each byte in the file as an ASCII (or Unicode) character. Since it isn't actually plaintext, most of the result looks like nonsense.

Most filetypes involve sophisticated encoding, which requires parsing. However, there are a few which are just represented as plaintext, including .txt, .csv, and .py. We can read and manipulate these types of files directly!

Files and Folders

It's important to note that all files are stored in **folders,** or **directories**. Folders can contain files, but they can also contain other folders. This makes a folder a **hierarchical structure**. We'll talk about this more in the Data Structures unit.

When you're working with files, always make sure you know which folder your file is located in! A sequence of folders from the top-level of the computer to a specific file is called a **filepath**.

Using Files in Python

To interact with a file in Python, we'll need to access its contents. We can do this by using the built-in function **open(filepath)**. This will create a **File object**, which we can read from or write to.

```
f = open("sample.txt")
```

open() can either take a full filepath, or it can take a **relative path** from the location of the python file. It's usually easier to just put the file you want to read/write from in the same directory as the python file, so you can simply refer to the filename directly.

Reading and Writing

When we open a file, we need to specify whether we plan to **read from** or **write to** the file. This will change the **mode** we use to open the file.

```
f = open("sample.txt", "r") # read mode
text = f.read() # returns a string
```

```
f = open("sample2.txt", "w") # write mode
f.write(text) # writes a string to the file
```

Only one instance of a file can be kept open at a time, so you should always **close** a file once you're done with it.

```
f.close()
```

Example: Script Manipulation

Let's write some code with files! Say you've been given a script for a play you're in, and you want to identify all of the lines that your character has.

Task 1: given a filename of a script, count the number of lines a character has in the script. Assume lines are always of the format

Name: line that they speak

Task 2: given a filename of a script, generate a new file containing only the lines and cue lines for a given character.

Check the course website after class for the solution.

Learning Goals

Use **programming** to specify algorithms to computers

- Interact with **user input** through interpreter input and files

Understand how computers **organize** data at a low level

- Computers use **folders and files** to store data hierarchically
- Files store data in strings using **specific formats**