OS, Files, and CSV

In this lecture we will cover

- Operating Systems
- Using Python to read
  - Plain Text Files
  - Comma Separated Value Files (CSV)
    - spoiler this is also plain text, but in a common format!

Opening Question

What operating systems do you use

- A. Linux (including Chromium / Chromebooks)
- B. iOS
- C. MacOS
- D. Windows
- E. Android

Your future in CS

I used to include this on my slides, but since these slides have changed - going to just leave it up here for every notebook. I get a lot of questions about more programming courses, the concentrations, and minors in computer science. Here is a brief reminder.

CS 164 – Next Course In Sequence, also consider CS 220 (math and stats especially)

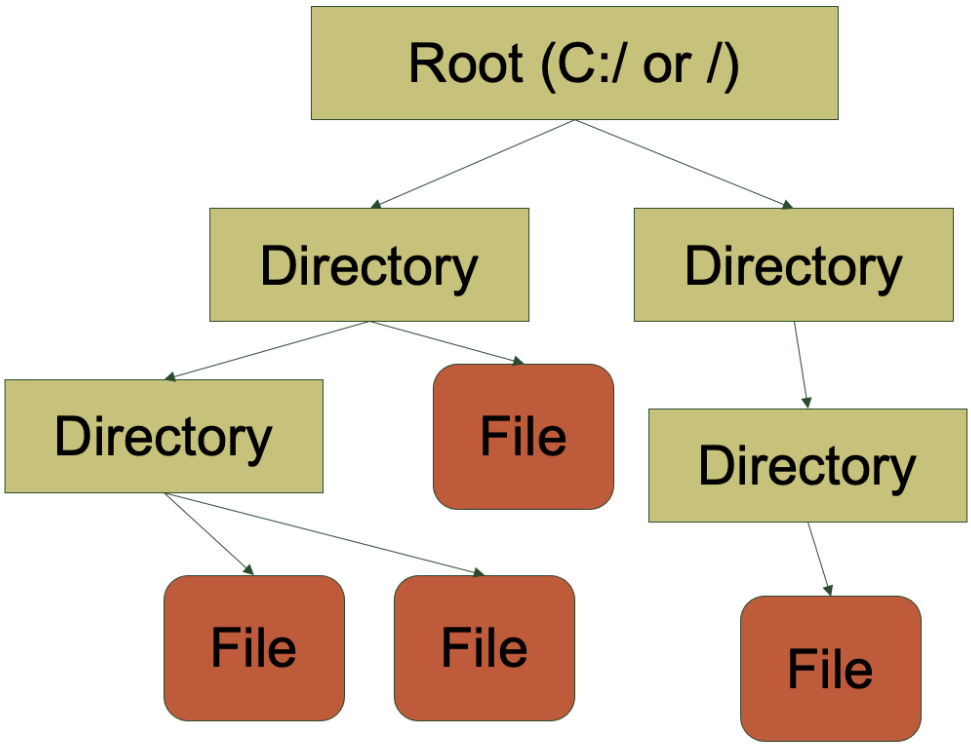
- CO Jobs Report 2021 – 77% of *all* new jobs in Colorado require programming
- 60% of all STEM jobs requires *advanced* (200-300 level)
- 31% of all Bachelor of Arts degree titled jobs also required coding skills
- 2016 Report found on average jobs that require coding skills paid $22,000 more

- Concentrations in CS:
  - Computer science has a number of concentrations.
    - General concentration is the most flexible, and even allows students to double major or minor pretty easily.
    - Software Engineering
    - Computing Systems
    - Human Centered Computing

        - Networks and Security
        - Artificial Intelligence
        - Computer Science Education.
    - Minors:
        - Minor in Computer Science - choose your own adventure minor
        - Minor in Machine Learning - popular with stats/math, and engineering
        - Minor in Bioinformatics - Biology + Computer Science

Operating Systems

- You use them daily
    - Most common OS in the world?
    - Android
        - Written in Java w/ Kotlin
- They control
    - Resources
    - Hardware Interaction
    - Devices
    - Running applications, memory, etc
    - Files!



- Definition:
    - Absolute Path - Path from the 'root' directory.
        - /local/usr/bin/myfile.txt

    - C:/users/lionheart/file.csv
- Relative Path - A path based on the current location
    - myfile.txt
    - somedirectory/myfile.txt
    - Also:
        - `..` - special character that means - go "up/back" one level
        - ../myfile.txt (my file is located above the current directory)
        - `.` - special character to say current directory
        - ./myfile.txt is the same as typing myfile.txt

For this class, everything we will do is:

- Relative Directory paths
- In Zybooks, we won't use directories at all, just the file names!
- In lecture, I will often use directories to help keep my files sorted

But how do we read files in Python? Let's look!

In [ ]:
```
f = open("data/avatar.txt") # notice we set the 'open file' to a variable!
content = f.read() # this dumps all the contents - as is - into a giant String!
f.close()
print(content)
```

```
Last Air Bender
Book 1: Water
Book 2: Earth
Book 3: Fire
Legend of Korra
Book 1: Air
Book 2: Spirits
Book 3: Change
Book 4: Balance
```

In [ ]:
```
with open("data/avatar.txt") as f:  # more common. auto closes the file, stores file i
    contents = f.readlines() ## This loads each line individually into a  **list**
print(contents)
```

```
['Last Air Bender\n', 'Book 1: Water\n', 'Book 2: Earth\n', 'Book 3: Fire\n', 'Legend
of Korra\n', 'Book 1: Air\n', 'Book 2: Spirits\n', 'Book 3: Change\n', 'Book 4: Balan
ce\n']
```

In Class Activity:

This activity is more about practicing loops!

1. Execute the code above, and that will read a file into the `contents` variable as a list!
2. Loop through (for loop!) the contents of the file.
    A. Print out the lines of the file, one on each line.
        - Hint: .strip() removes the '\n', so `line.strip()` if you named your for loop variable line
    B. Bonus problem: Build a second list for "Legend of Korra" books!

- Hint: You will want a boolean variable to help you keep track of where you are in the list
- Don't worry if you don't get it! This will just help with lab later

In [ ]:
```
##  start your loop here, remember format of the for loop is "for Variable in List:, s
addIt = False
korra = []
for line in contents:
    line = line.strip()
    if addIt:
        korra.append(line)
    if line == "Legend of Korra":
        addIt = True

print(korra)
```

```
['Book 1: Air', 'Book 2: Spirits', 'Book 3: Change', 'Book 4: Balance']
```

Comma Separated Value (CSV) Files

- A common data format is Comma Separated Values
- Think about Excel / Spreadsheets
  - The underlining design is:
  - flat data
  - row / column format
  - separated by commas

In fact, on most operating systems, if you open a .CSV file, it will open in excel or other spreadsheet application.

Region	Animal	Counted
Forest	Fox	3
Plains	Fox	5
Desert	Fox	2

Is represented as:

```
csv
Region,Animal,Counted
Forest,Fox,3
Plains,Fox,5
Desert,Fox,2
```

The first row is often called a 'header' row, but completely optional.

From a programming point of view, it is just Strings given meaning by the pattern.

> **Pro Discussion:**
> CSV and JSON formatted files are some of the most common formats to store

> data in for data analysis.
>
> Data is often flawed and needs fixed before running scripts on it!

In Python

- Python has special utilities to handle CSV files!

In [ ]:
```
import csv ## this is called an import statement, allows access to the CSV  library

with open("data/mobs.csv") as csvfile:
    mobs = csv.reader(csvfile)
    for mob in mobs:
        print(mob) #list including the header! (just a line)
```

```
['name', 'hp', 'ac', 'initiative']
['zombie', '22', '8', '8']
['zombie', '22', '8', '10']
['skeleton', '13', '13', '12']
['skeleton', '13', '12', '20']
['kobold', '5', '12', '15']
```

In Class Activity

Write code to read mobs.csv into a list structure, instead of just printing it.

Hint: copy the code above, and modify it

Hint: You should run the code first as is. Often the 'best' first step is just to print out what you know.

In [ ]:
```
values = []
with open("data/mobs.csv") as csvfile:
    mobs = csv.reader(csvfile)
    for mob in mobs:
        values.append(mob) #list including the header! (just a line)

print(values)
```

```
[['name', 'hp', 'ac', 'initiative'], ['zombie', '22', '8', '8'], ['zombie', '22',
'8', '10'], ['skeleton', '13', '13', '12'], ['skeleton', '13', '12', '20'], ['kobol
d', '5', '12', '15']]
```

You will notice the output of the above activity is a 'list of lists'. Now let's write a function that 'filters' that list!

In [ ]:
```
def filter_by_name(lst, name):
    rtn = []
    for item in lst:
        if item[0] == name:
            rtn.append(item)
    return rtn

zombies = filter_by_name("zombie", values)
skeleton = filter_by_name("skeleton", values)
```

```
print(zombies)
print(skeleton)
```

```
[['zombie', '22', '8', '8'], ['zombie', '22', '8', '10']]
[['skeleton', '13', '13', '12'], ['skeleton', '13', '12', '20']]
```

Overview

The code seems harder, but the goal is to keep it simple! We are layering ideas on top of each other, but the fundamentals are still true - divide-conquer-glue.

Also - remember to practice! There are a bunch of practice labs in zyBooks!