PASADENA CITY COLLEGE

CIS 112

Intro to Programming Using Python

Module 6 Part 1

Agenda for the Day!

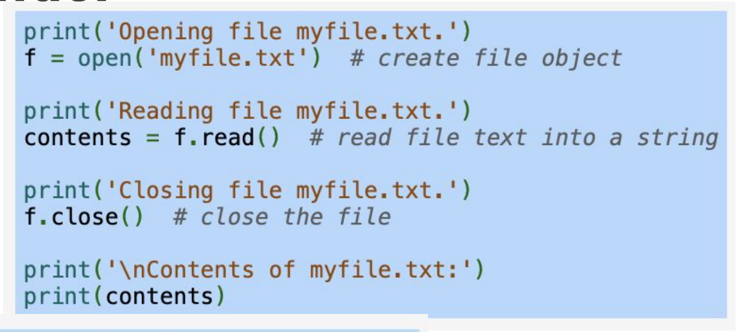
- processing files:
    - sqlite3,
    - xml,
    - csv,
    - logging,
    - configparser;

File Management

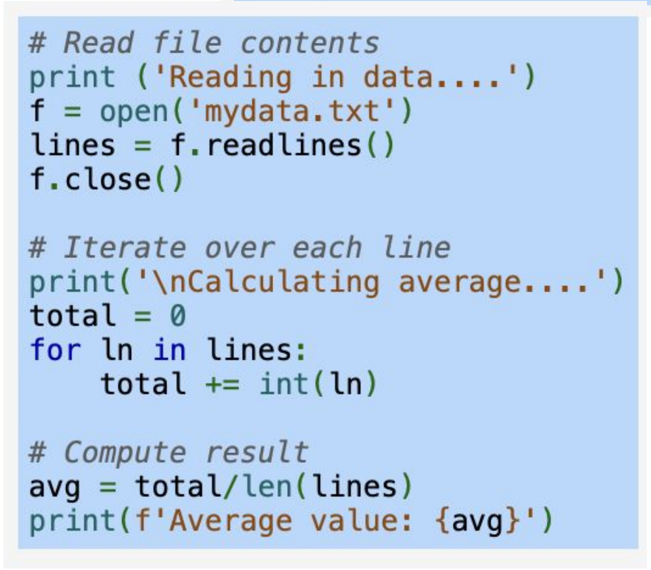
Trying to get the faucet sensor to work in the bathroom like:

It’s time to open our hearts and minds!

- How do we bring content into a python file?
  - Up till now: input statements
  - Introducing **open()**
    - Allows us to import a **txt** file into a data object in our program
      - Only txt! Stay tuned…
  - From there we can then manipulate the data by converting into a python datatype:
    - **read()** will convert the file into a string
    - **readlines()** will convert the file into a list delineated by line (each line is its own list entry
  - When we’re done, we close the file with the **close()** function



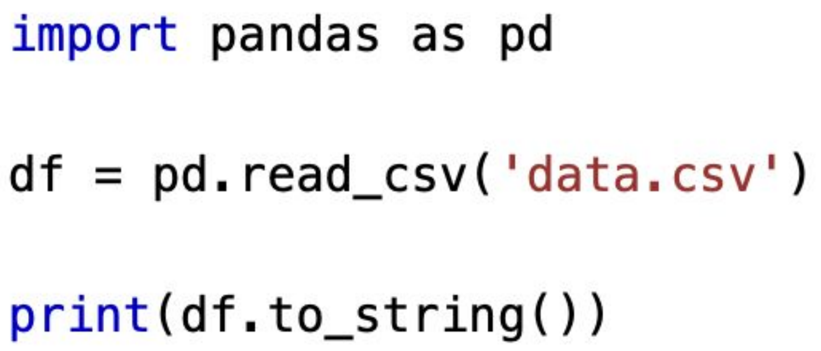
```python
print('Opening file myfile.txt.')
f = open('myfile.txt')  # create file object

print('Reading file myfile.txt.')
contents = f.read()  # read file text into a string

print('Closing file myfile.txt.')
f.close()  # close the file

print('\nContents of myfile.txt:')
print(contents)
```



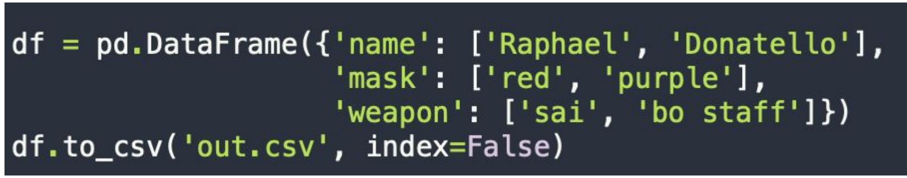
```python
# Read file contents
print ('Reading in data....')
f = open('mydata.txt')
lines = f.readlines()
f.close()

# Iterate over each line
print('\nCalculating average....')
total = 0
for ln in lines:
    total += int(ln)

# Compute result
avg = total/len(lines)
print(f'Average value: {avg}')
```

What about other datatypes?

- ‘CSV’ or comma-separated-valued files are a typical mechanism for transmitting tabular data in a text format
- While there are multiple libraries to handle CSV, my go to is Pandas
- The Pandas dataframe is the workhorse for tabular data in python data sciences
    - Columns (or series as they’re known in Pandas) are essentially singularly-typed lists
        - They are indexable and mutable
        - Support a number of list-level operations (hence the requirement for uniform typing)
    - Columns are then stacked together to construct a table

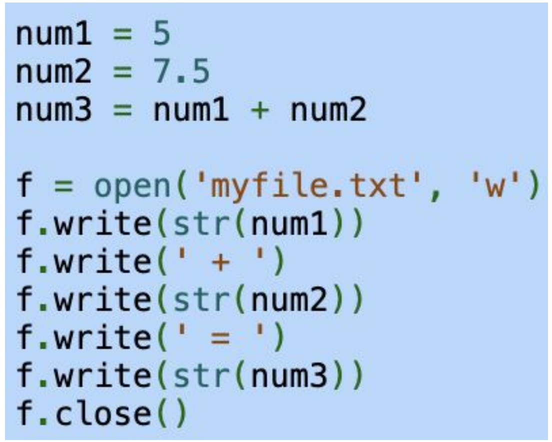


```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())
```



```python
df = pd.DataFrame({'name': ['Raphael', 'Donatello'],
                   'mask': ['red', 'purple'],
                   'weapon': ['sai', 'bo staff']})
df.to_csv('out.csv', index=False)
```

Modifying File Data



```
num1 = 5
num2 = 7.5
num3 = num1 + num2

f = open('myfile.txt', 'w')
f.write(str(num1))
f.write(' + ')
f.write(str(num2))
f.write(' = ')
f.write(str(num3))
f.close()
```

- When accessing a file you may want to control the permissions associated with manipulating the file.
- The **open()** function includes an input parameter that specifies permissions:
  - **'r'** limits your access to read-only
  - **'w'** will allow you to overwrite the existing file entirely
  - **'a'** will allow you to append new content without overwriting previous entries

The ‘With’ code block

keely flaherty @keelyflaherty · 9h

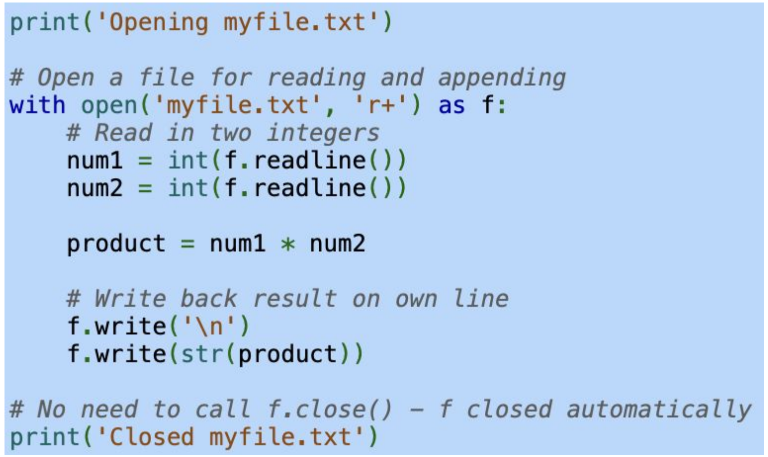
strongly relate to the honey cake’s needs

Honey Cake

By Joan Nathan

David Malosh for The New York Times. Food Stylist: Greg Lofts.

Time 1 ½ hours, plus at least 3 hours’ chilling and 25 hours’ resting



```
print('Opening myfile.txt')

# Open a file for reading and appending
with open('myfile.txt', 'r+') as f:
    # Read in two integers
    num1 = int(f.readline())
    num2 = int(f.readline())

    product = num1 * num2

    # Write back result on own line
    f.write('\n')
    f.write(str(product))

# No need to call f.close() - f closed automatically
print('Closed myfile.txt')
```

- Final code block of the course!
    - With allows up to take a file, open it, conduct operations and close automatically once completed.
    - Nice, concise method of file management!

Reading files

- As already noted, many important sources of data are stored externally and need to be read in and parsed for our programs
- One element we haven’t explored is parsing config files.
  - Remember YAML?

Writing Files

- What about writing?
- Turns out we have many useful use-cases
    - Program log textfiles can document activities and output results and errors
    - Certain usecases may want to produce summaries or receipts (e.g., transactions)
    - Certain programs may output program files (e.g., take in text and produce an html script)