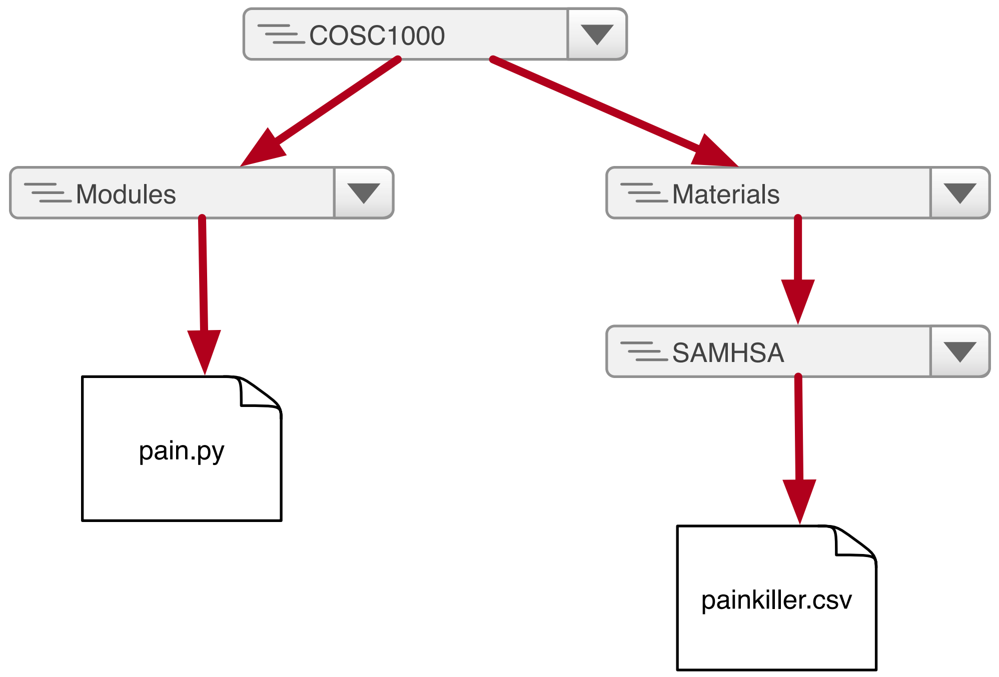
Python Review

Thomas Schwarz, SJ

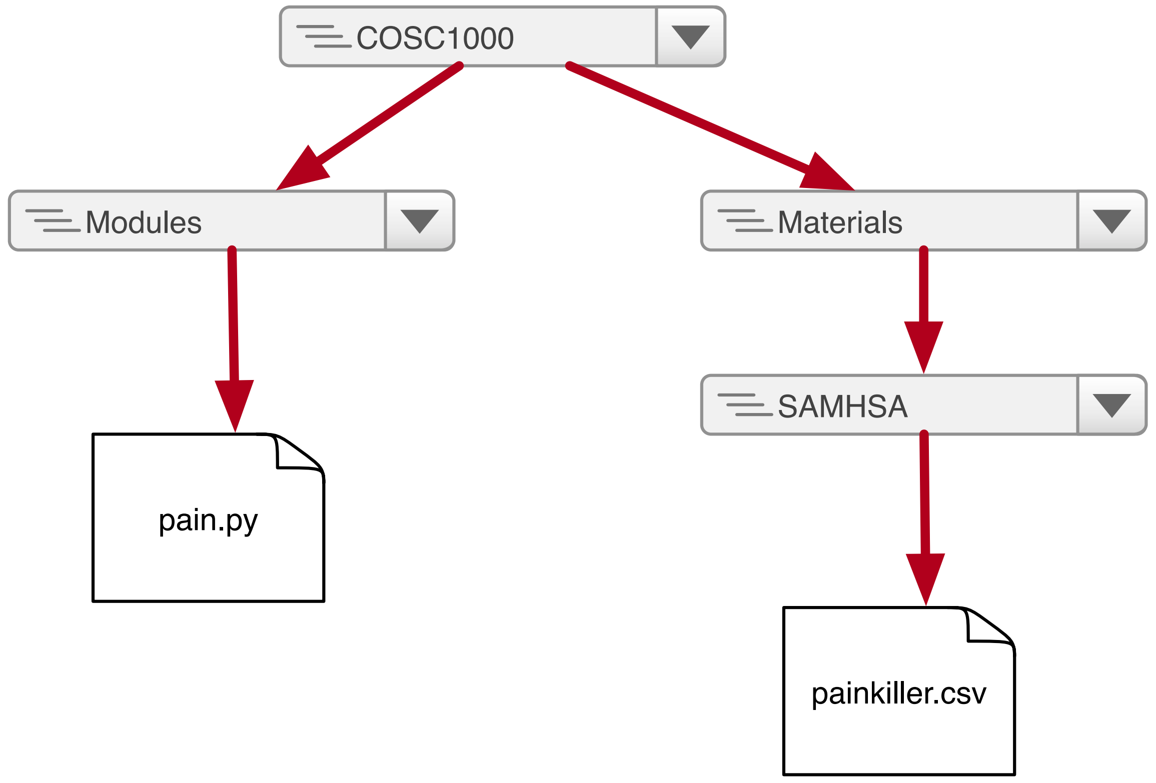
Opening a file

- To open a file:
  - We need its location
    - Absolute path
    - Relative path
      - Example:
        - Go up to "COSC1000"
        - Go to "Materials"
        - Go to SAMHSA



Opening a file

- Go up: ../
- Go down: Give folder name



```
with open('../Materials/SAMHSA/painkiller.csv') as input_file:

with open('..\\Materials\\SAMSHA\\painkiller.csv') as input_file
```

Repeating Statements

- use a for loop
  - Typical use: for i in range(5):
    - Here: need to skip over 8 input lines
  - To read a line, can use readline( )
    - This returns a line, but if we do not store it, then we read, but not keep the line, effectively skipping over it

Repeating Statements

- In practice:
  - Use "_" as a loop variable name if we do not use it

```
for _ in range(8):
    input_file.readline()
```

Repeating Statements

- To read all (remaining) lines:
    - Use the for _name_ in input_file_handler_name construct

```
for line in input_file:
    print(line)
```

Processing Strings

- Python has many methods for Strings
- .strip('chars') removes all characters in the string 'chars' from beginning and end
    - If we do not give 'chars', then we strip all white characters:
        - space
        - tabs
        - new lines
    - .strip( ) returns the cleansed string

Processing Strings

```
with open('../Materials/SAMHSA/painkiller.csv') as input_file:
    for _ in range(8):
        input_file.readline()
    for line in input_file:
        line = line.strip( )
        print(line)
```

Processing Strings

- Another way to process strings is to split them apart
    - In a .csv file, the fields are separated by commata
    - .split method returns a list of sub-strings
- In this file, we are interested in the name of the entity, the 12-17 estimate, the 18-25 estimate, and the 26-up estimate

```
for line in input_file:
   line = line.strip( )
   contents = line.split(',')
   print(contents[1], contents[5], contents[8], contents[11])
```

Processing Strings

- To make strings to numbers, we use the int( ) and float( ) methods
  - This assumes that the string can be made into a number
  - In our case, we need to get rid of the '%' as in '0.60%'
    - Possibilities:
      - `float(contents[8][:-1])`
      - `float(contents[8][0:4])`
      - `float(contents[8].rstrip('%')`

Processing Strings

```
for line in input_file:
    line = line.strip( )
    contents = line.split(',')
    age12_17 = float(contents[5].rstrip('%'))
    age18_25 = float(contents[8].rstrip('%'))
    age26_ = float(contents[11].rstrip('%'))
```

Processing Files

- Let's make a bogus statistics
  - Is republican or democratic government more likely to create a dependence on pain killers
- Use a list of trifecta: government, house, and senate belong to a single party:

```
repub = ['Alaska', 'Montana', 'Idaho', 'Wyoming', 'North Dakota', 'South Dakota',
         'Utah', 'Arizona', 'Oklahoma', 'Texas', 'Iowa', 'Missouri', 'Arkansas',
         'Mississippi', 'Alabama', 'Florida', 'Georgia', 'South Carolina',
         'Indiana', 'West Virginia', 'Ohio', 'New Hampshire', 'Tennessee']
democ = ['Washington', 'Hawaii', 'California', 'Nevada', 'Colorado',
         'New Mexico', 'Illinois', 'Virginia', 'Delaware', 'New Jersey',
         'New York', 'Connecticut', 'Rhode Island', 'Maine',
         'Oregon','District of Columbia']
```

Conditional Statements

- Done with if - elif - else
- Condition needs to evaluate to Boolean
    - For a string: can use `string in list_of_strings`

```
if contents[1]  in repub :
    block1
elif contents[1] in democ:
    block2
```

Conditional Statements

- Let's calculate the average of the three data depending on governing party
- We keep the results in arrays

```
rep = [0,0,0]
dem = [0,0,0]
for line in input_file:
…
    if contents[1]  in repub :
        rep[0] += age12_17
        rep[1] += age18_25
        rep[2] += age26_
    elif contents[1] in democ:
        dem[0] += age12_17
        dem[1] += age18_25
        dem[2] += age26_
```

Processing Files

- For each list, we then divide by the number of states in the trifecta list

```
for i in range(3):
        rep[i] = rep[i]/len(repub)
        dem[i] = dem[i]/len(democ)
```

- And then print out the results

```
print(rep)
print(dem)
```

Processing Files

- Result: Republican government leads to more pain killer addictions

```
[0.55, 1.1900000000000002, 0.73]
[0.5250000000000001, 1.02, 0.6425000000000001]
```

```
with open('../Materials/SAMHSA/painkiller.csv') as input_file:
    for _ in range(8):
        input_file.readline()
    rep = [0,0,0]
    dem = [0,0,0]
    for line in input_file:
        line = line.strip( )
        contents = line.split(',')
        age12_17 = float(contents[5].rstrip('%'))
        age18_25 = float(contents[8].rstrip('%'))
        age26_ = float(contents[11].rstrip('%'))
        if contents[1]  in repub :
            rep[0] += age12_17
            rep[1] += age18_25
            rep[2] += age26_
        elif contents[1] in democ:
            dem[0] += age12_17
            dem[1] += age18_25
            dem[2] += age26_
    for i in range(3):
        rep[i] = rep[i]/len(repub)
        dem[i] = dem[i]/len(democ)
    print(rep)
    print(dem)
```

Repeating Statements