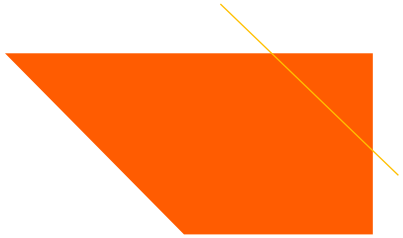
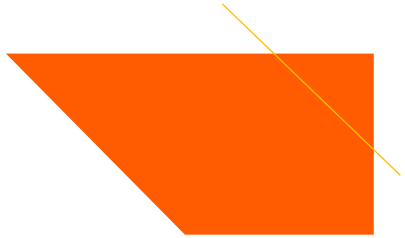
Introduction to Problem Solving in Python

COSI 10A



Class objectives

- File Processing (6.2-6.3)
  - Token/Line-based processing



File Processing

Review: File paths

- **Absolute path**: specifies a drive or a top "/" folder

  `C:/Documents/smith/hw6/input/data.csv`

- **Relative path**: does not specify any top-level folder

  `names.dat`

  `input/kinglear.txt`

- Assumed to be relative to the current directory: `file = open("data/readme.txt")`
- If our program is in `H:/hw6`, `open` will look for `H:/hw6/data/readme.txt`

Review: Reading a file

- The function `close` tells Python that your program is done using the file
  - The program will still run if you don't call `close`
  - Forgetting to close files can lead to bugs or lost of data if you are writing to a file

- The `with` statement is used to open a file and later close it automatically

```
with open("poem.txt") as file: # open file
     filetext = file.read()
     print(filetext)
     print(len(filetext))
```

- Recommended way to read and write files

Review: Reading a file

poem.txt

Roses are red,
Violets are blue.
All my base
Are belong to you.

file.readlines()

```
['Roses are red,\n', 'Violets are blue.\n', 'All my base\n', 'Are belong to you.']
```

file.readline()

```
'Roses are red,\n'
```

Review: Line-Based File Processing

- Line-based processing is the practice of processing input line by line

Syntax:

```
for line in file:
        statement
        statement
        …
        statement
```

Reading a file line by line

Example:

```
with open("poem.txt") as file:
        line_count = 0
        for line in file:
                print("next line:", line.rstrip())
                line_count += 1
        print("Line count:", line_count)
```

```
next line: Roses are red,
next line: Violets are blue.
next line: All my base
next line: Are belong to you.
Line count: 4
```

Review: Reading a file

poem.txt

Roses are red,
Violets are blue.
All my base
Are belong to you.

```
Roses are red,\nViolets are blue.\nAll my base\nAre belong to you.
```

input cursor

Each iteration of the `for` loop causes the cursor to move forward (consuming input)

```
Roses are red,\nViolets are blue.\nAll my base\nAre belong to you.
```

input cursor

First iteration

```
Roses are red,\nViolets are blue.\nAll my base\nAre belong to you.
```

input cursor

When the input cursor reaches the end of the file, the `for` loop stops

Trying to loop over a file twice

```
with open("poem.txt") as file:
      print("First time:")
      for line in file:
            print(line.rstrip())
      print()

      print("Second time:")
      for line in file:
            print(line.rstrip())
      print()
```

```
First time:
Roses are red,
Violets are blue.
All my base
Are belong to you.


Second time:
```

- A file object maintains an internal position or cursor as it reads input

Trying to loop over a file twice

- Two ways to fix the problem:

```
with open("poem.txt") as file:
      print("First time:")
      for line in file:
            print(line.rstrip())
      print()

with open("poem.txt") as file:
      print("Second time:")
      for line in file:
            print(line.rstrip())
      print()
```

```
with open("poem.txt") as file:
      print("First time:")
      for line in file:
            print(line.rstrip())
      print()

      file.seek(0)
      print("Second time:")
      for line in file:
            print(line.rstrip())
      print()
```

- `file.seek(0)` rewind the cursor to start

Weather question

```
16.2
23.5
19.1
7.4
22.8
18.5
-1.8
14.9
```

- We have a file `weather.txt`:

- Write a program that prints the change in temperature between each pair of neighboring days

```
16.2 to 23.5, change = 7.3
23.5 to 19.1, change = -4.4
19.1 to 7.4, change = -11.7
7.4 to 22.8, change = 15.4
22.8 to 18.5, change = -4.3
18.5 to -1.8, change = -20.3
-1.8 to 14.9, change = 16.7
```

Weather answer

```
16.2
23.5
19.1
7.4
22.8
18.5
-1.8
14.9
```

```
# Displays changes in temperature from data in an input file.
```

```
# prints the change in temperature between each pair of neighboring days
def weather():
    with open("weather.txt") as file:
        lines = file.readlines()
        prev = float(lines[0])

        for i in range(1, len(lines)):
            next = float(lines[i])
            print(prev, "to", next,  ", change =", (next - prev))
            prev = next
```

Token-Based File Processing

- Token-based processing is the practice of processing input token by token
- Strings have a function called `split` that divides the string into tokens (split a string using what are called delimiters. The default delimiter is whitespace)

Syntax:

```
for name in file.read().split():
        statement
        statement
        …
        statement
```

Example:

```
with open("poem.txt") as file:
        for word in file.read().split():
                print(word)
```

Reading a file token by token

`poem.txt`

Roses are red,
Violets are blue.
All my base
Are belong to you.

```
Roses
are
red,
Violets
are
blue.
All
my
base
Are
belong
to
you.
```

Token-Based File Processing

- Write a program that reports the number of times, a given word occurs in the file.

Roses are red,
Violets are blue.
All my base
Are belong to you.

`poem.txt`

Token-Based File Processing

- Write a program that reports the number of times, a given word occurs in the file.

Roses are red,
Violets are blue.
All my base
Are belong to you.

`poem.txt`

```
target = input("Target word? ")
count = 0
with open("poem.txt") as file:
        for word in file.read().split():
                if word.lower() == target.lower():
                        count += 1
        print("The word", target, "occurs", count, "times")
```

Token-Based File Processing

- Assume our file consists of numbers. Write a program that adds up all the numbers in the file and reports the sum.

```
308.2 14.9 7.4
2.8 81

5.0
3.9 4.7 67.0 -15.4
```

numbers.txt

Token-Based File Processing

- Assume our file consists of numbers. Write a program that adds up all the numbers in the file and reports the sum.

```
308.2 14.9 7.4
2.8 81

5.0
3.9 4.7 67.0 -15.4
```

numbers.txt

```
sum = 0.0
with open("numbers.txt") as file:
        for n in file.read().split():
                sum += n
        print("Sum is:", sum)
```

You get an error

```
sum = 0.0
with open("numbers.txt") as file:
        for n in file.read().split():
                sum += float(n)
        print("Sum is:", round(sum, 1))
```

Gas prices question

- Write a program that reads a file `gasprices.txt`
  - Format: *Belgium $/gal US $/gal date …*
  - `8.20 3.81 3/21/11 8.08 3.84 3/28/11 ...`

- The program should print the average gas price over all data in the file for both countries:

```
Belgium average: 8.3
USA average: 3.9
```

Gas prices solution

```
def gas_average():
    with open("gasprices.txt") as file:
        belgium = 0
        usa = 0
        count = 0
        lines = file.read().split()

        for i in range(0, len(lines), 3):
          belgium += float(lines[i])
          usa += float(lines[i + 1])
          count += 1

        print("Belgium average:", (belgium / count), "$/gal")
        print("USA average:", (usa / count), "$/gal")
```

Handling Invalid Input

308.2 hello 14.9 7.4
2.8 81 how are you?

5.0 :-) oops
bad 3.9 4.7 67.0 yipes -15.4

`numbers.txt`

- Suppose we have the following file:
- Write a program that adds up all the numbers in the file and reports the sum

```
sum = 0.0
with open("numbers.txt") as file:
        for n in file.read().split():
                sum += float(n)
        print("Sum is:", round(sum, 1))
```

This program will crash. Why?

- We can try to convert each token into a float. If we are successful, add that number to our sum, otherwise print error message
- `try/except`

Handling Invalid Input

- `try/except`

numbers.txt

```
308.2 hello 14.9 7.4
2.8 81 how are you?

5.0 :-) oops
bad 3.9 4.7 67.0 yipes -15.4
```

```
sum = 0.0
    with open("numbers.txt") as file:
        for n in file.read().split():
            try:
                sum += float(n)
            except ValueError:
                print("Invalid number:", n)
    print("sum is:", round(sum, 1))
```

```
Invalid number: hello
Invalid number: how
Invalid number: are
Invalid number: you?
Invalid number: :-)
Invalid number: oops
Invalid number: bad
Invalid number: yipes
sum is: 479.5
```