TCSS 142 – Introduction to Programming

Autumn 2014
Day 16

Day 16 Overview

- Typical list operations
- File reading and writing

List parameter question

- Write a function `swap` that accepts a list of integers and two indexes and swaps the elements at those indexes.

```
a1 = [12, 34, 56]
swap(a1, 1, 2)
print(a1)

# [12, 56, 34]
```

List parameter answers

```
# Swaps two values

def swap(a, i, j):
    temp = a[i]
    a[i] = a[j]
    a[j] = temp
```

List return question

- Write a function `merge` that accepts two lists of integers and returns a new list containing all elements of the first list followed by all elements of the second list.

```
a1 = [12, 34, 56]
a2 = [7, 8, 9, 10]

a3 = merge(a1, a2)
print(a3)

# [12, 34, 56, 7, 8, 9, 10]
```

List return answer

```
# Returns a new list containing all elements of a1
# followed by all elements of a2.

def merge(a1, a2):

  newList = []

  for el in a1:
        newList.append(el)

  for el in a2:
        newList.append(el)

   return newList
```

Passing single list elements

```
arr = [3, 4]

swap(arr[0], arr[1])

# swaps nothing
def swap(first, second):
      temp = first
      first = second
      second = temp
```

Exercise

Consider the following method, mystery:

```
def mystery(data, x, y):
        data[data[x]] = data[y]
        data[y] = x
```

What are the values of the list elements after the following code executes?

```
numbers = [3, 7, 1, 0, 25, 4, 18, -1, 5]
mystery(numbers, 3, 1)
mystery(numbers, 5, 6)
mystery(numbers, 8, 4)
```

List Histogram Question

- Given a file of integer exam scores, such as:

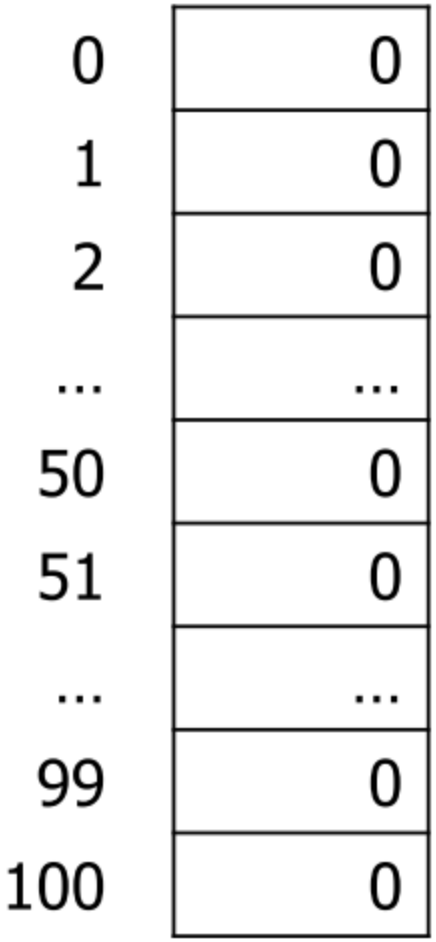
```
82
66
79
63
83
```

Write a program that will print a histogram of stars indicating the number of students who earned each unique exam score.

```
85: *****
86: **********
87: ***
88: *
91: ****
```

Lists for Tallying

- Use score as an index
- Use list location as a tally



List Histogram Answer

```
gradesTally = [0] * 101
fileobject = open("scores.txt", "r")

for line in fileobject:
    i = int(line)
    gradesTally[i] += 1

for x in range(101):
    print(x, end = ':')
    for z in range (gradesTally[x]):
        print('*', end = '')
    print()
fileobject.close()
```

Exercise

- Count the number of times each letter appears in a file using a list
  - Use characters as indexes (size?)
  - Lowercase vs uppercase
  - Value of A, a, Z, z
  - Filename `charcounter.py`

Problem

Read in student information data consisting of each student's name and test scores. Print the student's name along with his test average. At the bottom of the file, print the statistics that list the lowest average, along with the name, the highest average, along with the name, and all the students who scored in the upper 10% of the class.

Answer: Parallel Lists

DEFINITION

Parallel lists are 2 or more lists that have the same index range, and whose elements contain related information, possibly of different data types.

EXAMPLE

studentNames = [ ]

testAverage = [ ]

Answer: Parallel Arrays

15

While Controlled File Reading

```
rainfile = open("rainfall.txt","r")

aline = rainfile.readline()          # priming read

while aline != "":                   # test
  values = aline.split()
  print(values[0], " had ",values[1]," inches of rain.")
  aline = rainfile.readline()        # update read

rainfile.close()
```

Reading Web Pages

- A Web page is simply a type of file readable by a Web browser.

- We will look at a typical Web page file saved in HTML format (hypertext markup language)

```
<html>
 <head> info about the file </head>
 <body> info to be displayed in a browser</body>
</html>
```

Reading Web Pages

- URL – Uniform Resource Locator

  (protocol: // server / folders / document), e.g.

  http://www.tacoma.uw.edu/institute-technology/institute-technology

  http://faculty.washington.edu/monikaso/index.html

- Python provides `urlib` package and `urlib.request` module to get online data
  - once the URL is open, you can read the webpage as if you were reading data from a plain textfile using `read` and `readline` (returning `bytes` object, not strings)

Reading Web Pages

- Download `readWebpage.py`, examine, and run

- Suppose we want to:
  - count the number of lines in the heading of a Web page (<head></head>)
  - count the number of lines in the body of a Web page (<body></body>)
  - let's write a program `webcounter.py` to implement it

Algorithm

1. Read the lines until you reach <head> (do nothing with them)

2. Read the lines until you reach </head> and count them

3. Read the lines until you reach <body> (do nothing with them)

4. Read the lines until you reach </body> and count them

Internet as a Source of Data

- A lot of data nowadays resides in `csv` files.
  - comma separated values
  - just a text file that organizes records into rows and delimits fields with commas instead of blanks
  - download `amazonstock.csv` and open with `Notepad`
  - `Excel` provides support for `csv` files as well – open in `Excel`

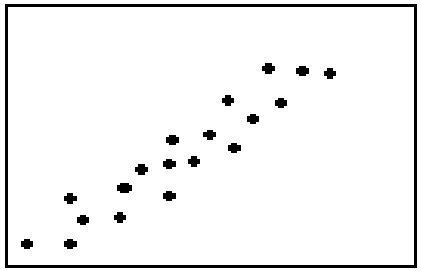
- Consider http://ichart.finance.yahoo.com/table.csv?s=AAPL&d=9&e=23&f=2014
  - ? Indicates parameters coming in key-value pairs
  - s gives us the stock name in this case Apple
  - d, e, f, stand for the date

Correlation

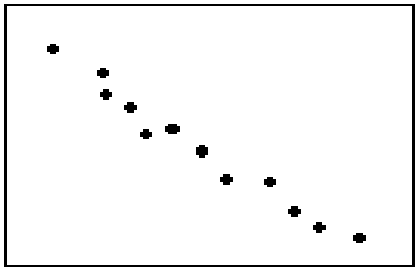
- We will look at the correlation between Apple and Amazon stock data

- Correlation measures the strength and direction of the relationship between two variables (but don't assume causal relationship)
  - 1.0 means positive correlation (both vars get larger)
  - 0.0 no correlation
  - -1.0 negative correlation (as one gets larger, the other var gets smaller)

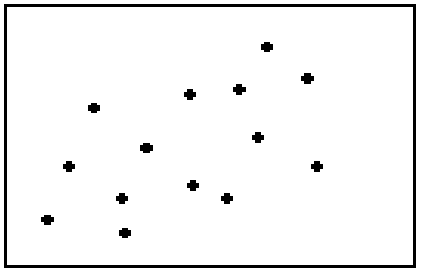
Degree of Correlation



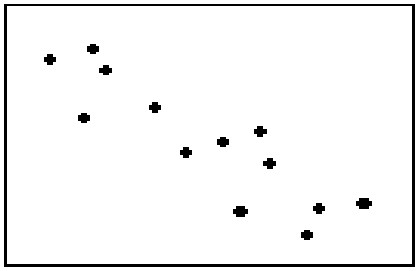
Strong Positive



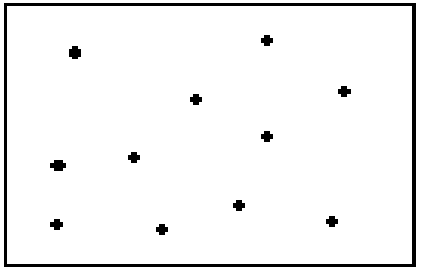
Strong Negative



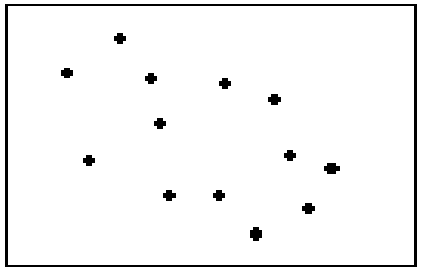
Weak Positive



Moderate Negative



None



Weak Negative

x axis values of one variable
y axis of the other variable

Correlation

- We will use the Pearson correlation coefficient to tell us the correlation between closing stock prices
  - we will use a function that calculates the correlation for us as a black box (like a library function)
  - we will feed two lists of values to it to get back the correlation factor
    - the lists need to be of the same length
  - download `amazonstock.csv` and `stocks.py` from `Canvas`

Preprocessing Data

- To make sure the data is compatible, we will extract only the closing prices for dates that both stocks have in common:
  - this will also ensure that lists are of the same length

- We have two files of uneven length
  - Approach 1: construct appropriate lists as you are reading from a file
  - Approach 2: read each file into a separate list and then adjust lists based on dates

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13
2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13
2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13

2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2
339.04	102.06

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13

2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2
339.04	102.06
312.99	94.72

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13
2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2
339.04	102.06
312.99	94.72

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13

2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2
339.04	102.06
312.99	94.72

Approach 1

Date	Open	High	Low	Close	Volume	Adj Close
2014-10-01	322.04	325.16	311.31	322.2	3565200	322.2
2014-09-02	339.98	349.38	317.64	322.44	3363900	322.44
2014-08-01	313.69	346.67	304.59	339.04	3045400	339.04
2014-07-01	325.86	364.85	311.86	312.99	4755300	312.99
2014-04-01	338.09	348.3	288	304.13	6779300	304.13

2014-08-01	94.9	102.9	93.28	102.5	46746200	102.06
2014-07-01	93.52	99.44	92.57	95.6	49637900	94.72
2014-06-02	633.96	651.26	89.65	92.93	59839500	92.07
2014-05-01	592	644.17	580.33	633	74996300	89.59
2014-04-01	537.76	599.43	511.33	590.09	82044000	83.06
2014-03-03	523.42	549	522.81	536.74	61552000	75.55
2014-02-03	502.61	551.19	499.3	526.24	82267500	74.07

list1	list2
339.04	102.06
312.99	94.72
304.13	83.06

Approach 1 Algorithm

1. Read one line from each file

2. While the reads succeeded:
   a. Compare dates
      - If date the same, append closing prices to relevant lists and read one line from each file again
      - If date1 > date2, read line from file1
      - If date1 < date2, read line from file2

3. Send both lists to correlation function

Comparing Dates

- Dates are read as strings – how are strings compared?

- Open the csv file in `Notepad` – how are dates formatted – would string comparison yield proper results?
  - If yes, use it
  - If not, convert a string to a date first

Putting It All Together

- Let's put all these ideas together to solve the problem

- Download the file `stockCorrelation1.py` from Canvas – it contains two functions that will be necessary for our processing: standard deviation of a list and correlation between two lists of the same length

Strings as Dates

- Python contains a module datetime that has a function date
  - datetime.date(year, month, day)
  - where year, month, day need to be integers

- We need to parse a string so that we can create a date out of it
  - split on '-'
  - convert each subcomponent to int and send to date function

Approach 2 Algorithm

1. Read data from file 1 into list 1

2. Read data from file 2 into list 2

3. Compare both lists position by position

    a. if dates the same, update both positions

    b. if date 1 > date 2, delete date 1 record from list 1, update list 1 position

    c. if date 1 < date 2, delete date 2 record from list 2, update list 2 position

Last Slide ☺

- Class ends at 17:10