New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Introduction to Python
Homework Discussion, Session 5

5.1 Reading through student_db.txt, create a dictionary where the key is the student id (first field) and the value is the rest of the line for that id.

When loaded, the dict will look like this:

```
students = { 'jk43':  ['23 Marfield Lane', 'Plainview' ,'NY', '10023'],
           'axe99': ['315 W. 115th Street, Apt. 11B', 'New York', 'NY', '10027'],
           ...[etc]...
```

First, give yourself a chance to write an outline before referring to these "answers" (which I should stress are not the only way to arrange your code).

This assignment allows you to build up a dictionary from a file, then query the dictionary. Since the file is understood to have unique "keys" (i.e., ids), there is no need to check the dictionary before adding a key. We are simply converting the file into a dictionary so we can query it easily (i.e., by requesting a key, we are "querying" the dict for the data related to an id).

Basically, this boils down to loading the dict from the file, then reading a single key from the dict and splitting and formatting the string value.

Here is my list of steps:

1. Build the dict:
   a. Initialize an empty dict
   b. loop through the file and split each line into a list of string elements
   c. assign the first element (the id) to a new variable name
   d. slice the list of elements so that the first element (the id) is excluded
   e. set a key and value in the dict with the key being the id (the first field split from the line) and the value being the sliced list (which is basically the record of data associated with that id).
2. Report the number of records read from the file (use the dict itself, check its length - don't use a separate counter).
3. In a while True loop, ask the user for an id.
   a. If the user types q, break out of the loop
   b. if the user's id is not in the dict, print an error message and continue
   c. if the id is in the dict, use the elements from the associated list to print a formatted address

Pseudocode:

```
initialize an empty dict
open the file
read the file into a list of lines (with readlines())
slice the list of lines to exclude the 1st element (the header line)
loop through each line of the sliced list
    split the line into elements
    assign the first element to a variable name
    slice the elements to exlude the first element (the id)
    set a key (the 1st element, the id) and a value (the sliced list)
    in the dict
report (print) the number of records loaded (use len() on the dict)
begin a while True loop
    take user input for an id, q for quit
    if input id is equal to q, break out of the loop
    if input id is not found in the dict (use the in operator), report the error; continue
    assign the value for this key to a list variable
create a formatted string with elements from the list.  you can use triple-quoted strings to
create the string, or simply use the \n newline character to indicate a new line
    print the address
```

5.2 Reading through F-F_Research_Data_Factors_daily.txt, create a dictionary that sums all the Mkt-RF values for each year. Report the number of years found in the file (this comes directly from the size of the dict). Ask the user to enter the n number of results desired, as well as whether they want to see the 'highest' values or the 'lowest' values. Validate input: for number of results, reject a non-integer or an integer greater than the available number of years; for 'highest' or 'lowest', reject any value that is not one of those. Show the highest n years or lowest n years along with their Mkt-RF value sums.

Comparing this to the spec for 5.1, notice that I'm not asking for a 'q' value. Instead, errors cause the program to exit. This just happened to be what I decided to do; if you want to include the 'q' value option as part of your design, you are welcome to do so.

So here are the steps I outlined when I designed the overall flow:

* get the lines of the FF file without the header and footer
* compile the dict of years and summed Mkt-RF values
* get user input for number of results and 'highest' or 'lowest' values
* get a sorted list of keys based on dict value, and order according to 'highest' or 'botom'
* report the sorted list and the values associated with them

Again I should emphasize that this is the design that occurred / appealed to me; if you can, try to conceptualize your own solution and then compare to mine. Seeing two approaches and being able to compare and contrast can be instructive.

Pseudocode:

```
open the file and read into a list of lines
slice the lines to exclude the header and footer lines

initialize an empty dictionary

for each line in the file lines:
    slice out the year
    split out the mkt-rf value and convert to float
    if the year is not in the dicti:
        set the year key and value 0.0 in the dict
    add the mkt-rf float to the current value for the year in this dict

store the integer length of the dict (so we can see the max value allowed)

start a while True loop
    ask the user for the number of results desired
    ask the user for direction value "highest" or "lowest"
    if the number of results is not all digits,
        print an error message
        continue to top of while loop
    if the direction value is not "highest" or "lowest",
        print an error message
        continue to top of while loop
    convert the number of results to an integer
    if the number of results requested is greater than the dict length,
        print an error message
        continue to top of while loop
    break out of the loop (no errors detected)

set the "this_reverse" variable to True or False based on the value of the "direction" choice:
if "direction" == 'highest', set "this_reverse" to True.  if "direction" == 'lowest', set
"this_reverse" to False

sort the keys of the dict based on value (using the key= value dict.get (see slides)).  also
include the reverse=True or reverse=False depending on the value of the "reverse" variable.
```

Here is sample Python code of setting a value to True or False and using it in the reverse= argument to sorted():

```
direction = input('select "highest" or "lowest" results:  ')

if direction == "highest":
    this_reverse = True

else:
    this_reverse = False

sorted_keys = sorted(this_dict, reverse=this_reverse, key=this_dict.get)
```