INDEX

Using the list `lst` below, `result`'s value would be: `[('pretzels', 6), ('cookies', 5), ('chips', 3), ('carrots', 6)]`

[(snacks , num + 1)]

Note that the numbers in the list of tuples reflect one more than the original order for each snack.

Write your code below and be sure result's value is the answer.

```
     0            1          2          3
snacks = ['pretzels', 'cookies', 'chips', 'carrots']
numbers = [5, 4, 2, 5]
```

you want the corresponding positions

result = [(snacks [index] , numbers [index] + 1) for index in range (len(snacks))]

LAMBDA

NETID: ---------

Part E (8 pts) (8 minutes)

Write the function named **howManyInRange** that has one parameter named **datalist**, which is a list of lists in the format described earlier.

We repeat the format of parameter datalist, which is a list of lists. Each inner list is information about one food item as 1) a string representing the food item 2) a float representing the price of the food item and 3) a list of names of people who like the food item.

This function returns a **sorted** list of tuples of pairs of numbers, where the first number is an integer, say N, and the second number is the number of food items that were liked that cost at least N and less than N+1. A food item is counted as many times as it is liked. Only those tuples with second number greater than 0 are in the list. **The tuples are sorted on the second number in reverse order, with ties broken by the first number.**

For example, consider the datalist example given at the beginning of this problem. The call howManyInRange(datalist) would return the list: [(4, 12), (10, 10), (3, 5), (8, 4), (12, 4), (6, 3)]. In the first tuple (4, 12), the 4 shows there are food items that cost at least 4.0 and less than 5.0, and the 12 shows the food items are liked by 12 people. Note cornbread is $4.25 (liked by 4 people), field peas is $4.50 (liked by 4 people) and blackeyed peas is $4.80 (liked by 4 people). The tuple (4, 12) is listed first because 12 is the largest second number.

```
def howManyInRange(datalist):
    d = {}
    for alist in datalist:
        num = int(alist[1])
        if num not in d:
            d[num] = 0
        d[num] += len(alist[2])
    return sorted(d.items(), key=lambda x:x[1], reverse=True)
```

SORTED & DICT.

NETID: ---------

Part D (8 pts) (8 minutes)

Write the function named **mostLiked** that has one parameter named **datalist**, which a list of lists in the format described earlier.

We repeat the format of parameter datalist, which is a list of lists. Each inner list is information about one food item as 1) a string representing the food item 2) a float representing the price of the food item and 3) a list of names of people who like the food item.

This function returns a **sorted** list of unique names of people that liked the most food items in datalist.

sorted & set

For example, using the datalist on the first page of this problem, the call mostLiked(datalist) returns the list `['Jiao', 'Raj', 'Susan']`, as each of them liked five food items, which was the largest number of liked food items.

Complete the function below.

```
def mostLiked(datalist):
    d = dictPersonToItems(datalist)
    maxSize = max([len(v) for v in d.values()])
    maxPeople = [name for (name,itemlist) in d.items() if len(itemlist) == maxSize]
    return sorted(maxPeople)
```

This function returns a string that is the name of the player at the school **school** that scored the most points over all of this school's games. Assume there is only one player with the most points. We give several examples of calls to this function.

call	**returns**
`playerMostPoints(datalist, 'miami')`	'rtopp'
`playerMostPoints(datalist, 'duke')`	'bwalton'
`playerMostPoints(datalist, 'unc')`	'yxiao'

In writing this function, you MUST call the function **dictPlayerTotalPoints** from Part B in a meaningful way. Complete the function below.

```
def playerMostPoints(datalist, school):
    d= dictPlayer Total Points (datalist, school)   calls a helper func. to create a dictionary 'd' that maps player names to their total points scored for the given school.
    maxpts= max(d. values())   finds max value in the dict 'd' which represents the highest # of points scored by any player for the given school.
loops through each key-value pair in dictionary 'd' ← for (key, value) in d.items () :
        if value == maxpts :
            return key   returns player name
    return " no players "
```

MAX FUNCTION & KEY-VALUE PAIR

"Computers are good at following instructions, but not at reading your mind." - Donald Kuth

"Everybody should learn to program a computer, because it teaches you how to think." - Steve Jobs

DICTIONARY FUNCTIONS

d[key] returns value associated w/ key ; 'error' otherwise

d.get (key) returns value associated w/ key or 'None'

d.keys ( ) returns a list of the keys in dictionary

d.values ( ) returns a list of values in dict.

d.items returns a list of tuples (k, v) pairs

d.update (dict) update dictionary with another dictionary 'dict'!

```
d = {'M': [8, 9], 'C':[7,12], 'Y':[4,10]}
ans = sorted(d.keys())   print only keys!
print(ans)
                         ['C', 'm', 'Y']
```

```
def match(word1, word2):
  ans = [   ]
                       position  length
  for  position index  in  range (len(words)) :
     if  word1 [index] == word 2 [index]:
        ans. append ( word1 [index] )
     else :
        ans. append ('_')
  return  ans.
```

INDEX & RANGE

OPERATORS

+ addition

- subtraction

* multiplication

/ division

// integer division

% integer remainder

+ str concatenation

* repeat (str)

== equal

!= not equal

SET FUNCTIONS

s | t union of both sets

s & t returns shared elem.

s - t values in s not in t

s ^ t values in both NOT shared

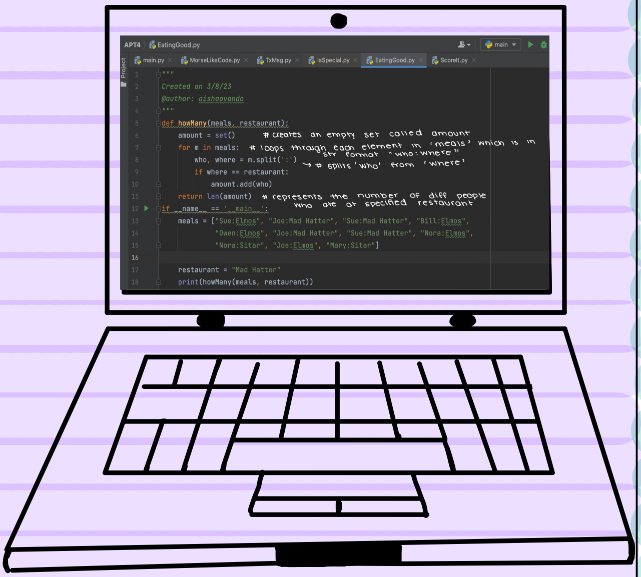
s.remove(item) removes or error if not there

s.update(lst) adds elem from lst to set

s.add(item) adds item to set

APT 4

EATINGGOOD.PY



```
"""
Created on 3/8/23
@author: aishaovando
"""
def howMany(meals, restaurant):
    amount = set()      # creates an empty set called amount
    for m in meals:  # loops through each element in 'meals' which is in str format "who:where"
        who, where = m.split(':')  → # splits 'who' from 'where'
        if where == restaurant:
            amount.add(who)
    return len(amount)  # represents the number of diff people who ate at specified restaurant
if __name__ == '__main__':
    meals = ["Sue:Elmos", "Joe:Mad Hatter", "Sue:Mad Hatter", "Bill:Elmos",
             "Owen:Elmos", "Joe:Mad Hatter", "Sue:Mad Hatter", "Nora:Elmos",
             "Nora:Sitar", "Joe:Elmos", "Mary:Sitar"]

    restaurant = "Mad Hatter"
    print(howMany(meals, restaurant))
```

~ Exam 1 ~

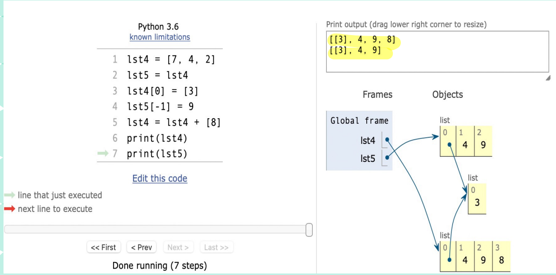
Spring 2023

PART D (3 pts)

```
lst4 = [7, 4, 2]
lst5 = lst4
lst4[0] = [3]
lst5[-1] = 9
lst4 = lst4 + [8]
print(lst4)
print(lst5)
```

Output:

★ Changes to lst[4] imply changes to lst[5] ONLY if an item already in the original list is altered. no changes are made to the copy if an item is simply added. ★



Python 3.6
known limitations

```
lst4 = [7, 4, 2]
lst5 = lst4
lst4[0] = [3]
lst5[-1] = 9
lst4 = lst4 + [8]
print(lst4)
print(lst5)
```

Edit this code

line that just executed
next line to execute

<< First   < Prev   Next >   Last >>

Done running (7 steps)

Print output (drag lower right corner to resize)

```
[[3], 4, 9, 8]
[[3], 4, 9]
```

COMPSCI 101