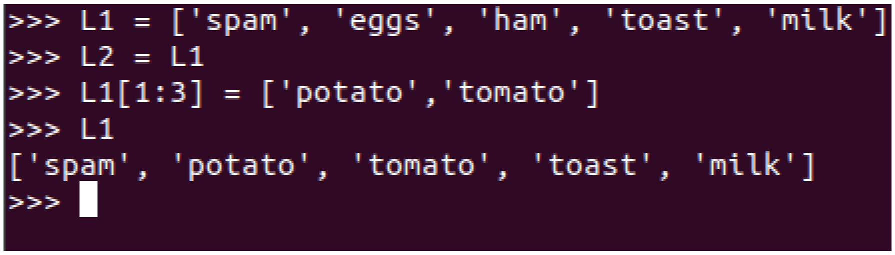
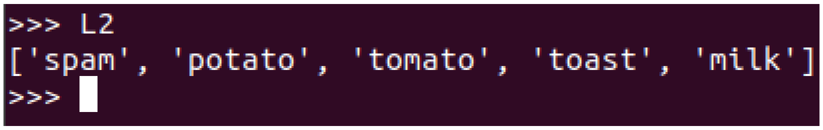
Lab 13
Dexter Kale – CS 3035 - 05

#3, I.



```
>>> L1 = ['spam', 'eggs', 'ham', 'toast', 'milk']
>>> L2 = L1
>>> L1[1:3] = ['potato','tomato']
>>> L1
['spam', 'potato', 'tomato', 'toast', 'milk']
>>> 
```

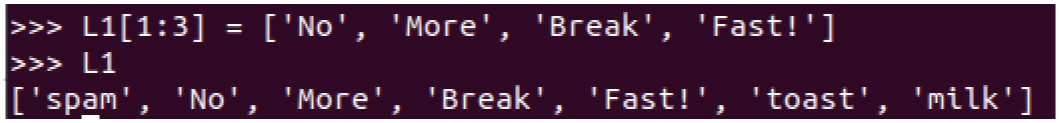
II.



```
>>> L2
['spam', 'potato', 'tomato', 'toast', 'milk']
>>> 
```

L2 is the same as L1.

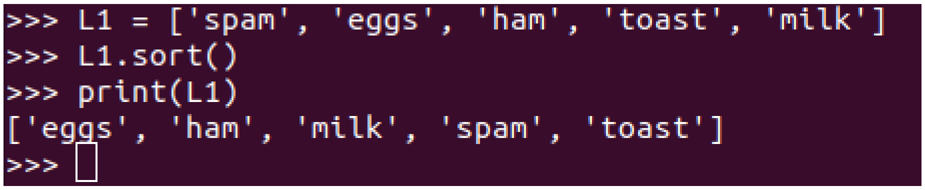
#4 I.



```
>>> L1[1:3] = ['No', 'More', 'Break', 'Fast!']
>>> L1
['spam', 'No', 'More', 'Break', 'Fast!', 'toast', 'milk']
```

L2 does carry over the new list.

#5 I.



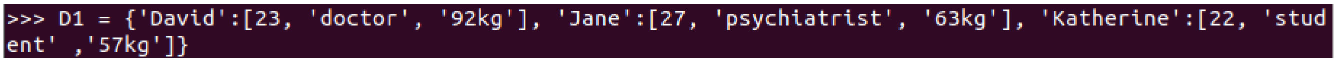
```
>>> L1 = ['spam', 'eggs', 'ham', 'toast', 'milk']
>>> L1.sort()
>>> print(L1)
['eggs', 'ham', 'milk', 'spam', 'toast']
>>> 
```

The original L1 list sorted.

L2 does also become sorted, thus confirming the mutability of lists.

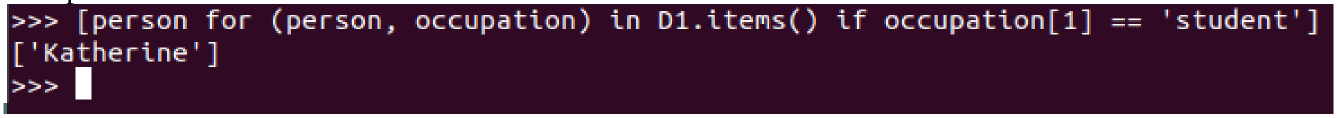
III. When switching out some words for capitalized words, we see that all capitalized words get sorted to the front.
It seems to be sorting the values by going by ASCII, since uppercase letters appear first, before lowercase letters.
Sort will not work with multiple data types.

Dictionary:
D1 = {‘David’:[23, ‘doctor’, ‘92kg’], ‘Jane’:[27, ‘psychiatrist’, ‘63kg’], ‘Katherine’:[22, ‘student’ ,‘57kg’]}



```
>>> D1 = {'David':[23, 'doctor', '92kg'], 'Jane':[27, 'psychiatrist', '63kg'], 'Katherine':[22, 'stud
ent' ,'57kg']}
```

III.
Comprehension statement



```
>>> [person for (person, occupation) in D1.items() if occupation[1] == 'student']
['Katherine']
>>> 
```

The difference in declaring local and global variables don’t matter too much, besides the fact that you can’t do things like use functions in global variables since they are declared outside main. They work the same likewise besides that.

Declaring the same variable locally and globally with the same name results in the local variable overwriting and overtaking the global variable in priority.

If two functions, one external, and one nested are declared with the same name, the nested function takes priority.

A static variable would be any variable that doesn’t change from object to object, which is kind of like global variables. Python technically does not have static variables, even though global variables operate similarly to static variables.