Get your hands dirty with Python!

Here are a few problems to solve that will introduce you to the world of python and some neat tricks it holds for you.

You may want to keep the reference close-by: https://docs.python.org/2/library/index.html
To keep code consistent across everyone - PEP8 Style Guide

Lets get started then →

Q1. Get your basics right - Implement selection sort algorithm in python. The function accepts a list in the input and returns a sorted list.
E.g.

```
Input f1([5,416,54,21,6135,15,741]) should
Return [5, 15, 21, 54, 416, 741, 6135]
```

Q2. Dictionary, what?

Write a program that returns the file type from a file name. The type of the file is determined from the extension. Initially, a list of values of the form "extension,type"(e.g. xls,spreadsheet; png,image) will be input.

The program takes input in the following form:

1. Input extension and type values in the form of a string having the following format:
    a. "extension1,type1;extension2,type2;extension3,type3"
    b. E.g. If we needed to input xls → spreadsheet, xlsx → spreadsheet, jpg → image our string would be something like: "xls,spreadsheet;xlsx,spreadsheet;jpg,image"
2. Input a list of filename.extension. E.g. an input list could be something like ["abc.html", "xyz.xls", "text.csv", "123"]

The program should return a dict of filename: type pairs
E.g.

```
f("xls,spreadsheet;xlsx,spreadsheet;jpg,image", ["abc.jpg",
"xyz.xls", "text.csv", "123"]) should return
{
    "abc.jpg": "image",
    "xyz.xls": "spreadsheet",
    "Text.csv": "unknown",
    "123": "unknown"
}
```

Q3. Column Sorting, yay!

Given a list of dicts, write a program to sort the list according to a key given in input.
E.g.

```
Input f([
    {"fruit": "orange", "color": "orange"},
    {"fruit": "apple", "color": "red"},
    {"fruit": "banana", "color": "yellow"},
    {"fruit": "blueberry", "color": "blue"}
], "fruit")
Should Output
[
    {"fruit": "apple", "color": "red"},
    {"fruit": "banana", "color": "yellow"},
    {"fruit": "blueberry", "color": "blue"},
    {"fruit": "orange", "color": "orange"}
]
```

AND

```
Input f([
    {"fruit": "orange", "color": "orange"},
    {"fruit": "apple", "color": "red"},
    {"fruit": "banana", "color": "yellow"},
    {"fruit": "blueberry", "color": "blue"}
], "color")
Should Output
[
    {"fruit": "blueberry", "color": "blue"},
    {"fruit": "orange", "color": "orange"},
    {"fruit": "apple", "color": "red"},
    {"fruit": "banana", "color": "yellow"}
]
```

Q4. The power of one line -
Given a dictionary, switch position of key and values in the dict, i.e., value becomes the key and key becomes value. The function's body shouldn't have more than one statement.

```
f({
   "key1": "value1",
   "key2": "value2",
   "key3": "value3",
   "key4": "value4",
   "key5": "value5"
}) should return

{
   "value1": "key1",
   "value2": "key2",
   "value3": "key3",
   "value4": "key4",
   "value5": "key5"
}
```

Q5. Common, Not Common
Given 2 lists in input. Write a program to return the elements, which are common to both lists(set intersection) and those which are not common(set symmetric difference) between the lists.

```
Input:
Mainstream = ["One Punch Man","Attack On Titan","One Piece","Sword Art Online","Bleach","Dragon Ball Z","One Piece"]
must_watch = ["Full Metal Alchemist","Code Geass","Death Note","Stein's Gate","The Devil is a Part Timer!","One Piece","Attack On Titan"]

f(mainstream, must_watch) should return:

["One Piece", "Attack On Titan"], ["Dragon Ball Z", "Death Note", "One Punch Man", "Stein's Gate", "The Devil is a Part Timer!", "Sword Art Online","Full Metal Alchemist","'Bleach", "Code Geass"]
```

Q6. Every other sub-list

Given a list and 2 indices as input, return the sub-list enclosed within these 2 indices. It should contain every second element.
E.g.

```
Input f([2,3,5,7,11,13,17,19,23,29,31,37,41], 2, 9)
Return [5, 11, 17, 23]
```

Q7. Calculate the factorial of a number using lambda function.

Q8. Some neat tricks up her sleeve:
Looking at the below code, write down the final values of A0, A1, ...An

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)
A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i:i*i for i in A1}
A6 = [[i,i*i] for i in A1]
A7 = reduce(lambda x,y: x+y, [10,23, -45, 33])
A8 = map(lambda x: x*2, [1,2,3,4])
A9 = filter(lambda x: len(x) >3, [“I” , “want”, “to”, “learn”, “python”])
```

Q9.
Write a func that takes 3 args:
from_date - string representing a date in the form of 'yy-mm-dd'
to_date - string representing a date in the form of 'yy-mm-dd'
difference - int
Returns True if from_date and to_date are less than difference days away from each other, else returns False.

Q10. Of date and days
Write a func that takes 2 args:
date - string representing a date in the form of 'yy-mm-dd'
n - integer

Returns the string representation of date n days before 'date'
E.g. f('16-12-10', 11) should return '16-11-29'

Q11. Something fishy there -
Find output of following:

```
def f(x,l=[]):
    for i in range(x):
        l.append(i*i)
    print(l)

f(2)
f(3,[3,2,1])
f(3)
```