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]

Answer:

```
def selectionSort(list):

   n=len(list)

   for i in range(n):

      min=i

      for j in range(i+1, n):

         if list[j]<list[min]:

            min=j

      temp=list[i]

      list[i]=list[min]

      list[min]=temp

   return list

inputList= input()

list= [int(x) for x in inputList[1:-1].split(",")]

selectionSort(list)
```

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"

}

Answer:

```
def fileType(value1,value2):
    res ={}
    res2={}
    pairs =value1.split(';')
    for i in pairs:
        ext,fileName = i.split(',')
        res1[ext] = fileName
    for j in value2:
        ext=j.split('.')[-1]
        fileName=res1.get(ext, 'unknown')
        res2[j]=fileName
    return res2

input1 = input()
input2 = input().split()
output = fileType(input1,input2)
print(output)
```

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.

```
def listSort(lst, key):
    sortedlist = sorted(lst, key=lambda x: x.get(key, ""))
    return sortedlist
```

Answer:

```
lst= [{"fruit":"orange", "color": "orange"},
        {"fruit": "apple", "color": "red"},
        {"fruit":"banana", "color": "yellow"},
        {"fruit":"blueberry", "color": "blue"}]


print(listSort(lst,"color"))
print(listSort(lst,"fruit"))
```

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"

}

Answer:

```
def switchKeyValue(dict):
    res={}
    for key,value in dict.items():
        res[value]=res.get(value,[])+[key]
    return res;


dict={
"key1": "value1", "key2": "value2", "key3": "value3", "key4": "value4", "key5": "value5"
}
print(switchKeyValue(dict))
```

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"]

Answer:

```
def compareList(l1, l2):
   common = list(set(l1) & set(l2))
   notcommon= list(set(l1) ^ set(l2))
   return common, notcommon

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"]

common, notcommon = compareList(mainstream, must_watch)
print(common)
print(notcommon)
```

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]

Answer:

```
def subList(list,s,e):
   return lst[s:e+1:2]

list = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]
s=2
e=9
sublist=subList(list,s, e)
print(sublist)
```

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

```
factorial = lambda n:n and n * factorial(n - 1) or 1

n=int(input())
print(factorial(n))
```

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"])
```

Answers:

A0= {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

A1=range(0, 10)

A2=[]

A3= [1, 2, 3, 4, 5]

A4= [1, 2, 3, 4, 5]

A5= {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6= [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

A7= 21

A8= [2, 4, 6, 8]

A9= ['want', '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.

Answer:

```
from datetime import datetime

def dateDifference(from_date, to_date, difference):
    fromdate=datetime.strptime(from_date, '%y-%m-%d')
    todate=datetime.strptime(to_date, '%y-%m-%d')
    diff=todate-fromdate
    res=diff.days
    if res<difference:
        return True
    else:
        return False

result=dateDifference('23-05-10','23-05-28',40)
print(result)
```

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'

Answer:

```
from datetime import datetime,timedelta

def beforeDate(date,n):
    dateobj=datetime.strptime(date,'%y-%m-%d')
    diff=timedelta(days=n)
    newdate=dateobj-diff
    resdate=newdate.strftime('%y-%m-%d')
    return resdate

result=beforeDate('16-12-10',11)
print(result)
```

Q11. Something fishy there - Find output of following:

```
def f(x,l=[]):

for i in range(x): l.append(i*i)

print(l)
```

Answer:

```
f(2)

output  :   [0, 1]

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

output   :   [3, 2, 1, 0, 1, 4]

f(3)

output    :   [0, 1, 0, 1, 4]
```