There are two types of difficulties you are likely to encounter when programming: inability to figure out the steps required to achieve your aim, and inability to figure out what code to write to perform a given task. Both will happen less frequently with practice.

It is critical, if you're getting stuck to actually force yourself to sit down and write out each of the 'small steps' that will be required to achieve your aim. Once the problem is broken down into bite-sized pieces, then you can focus on figuring out the syntax you need to achieve each of these small pieces one at a time.

In this practice work, I will give you some simple problems to solve which will be relevant to the homework assignment, but won't completely solve every aspect of it.

Practice #1: You have a list of integers, for example
x=[1,2,3,5,7,9,10,3,2,5,5,5,1,12,14,9,7]
For each number present in the list, count the number of copies of that number, then print a sorted list of the numbers and counts. Note that this is quite similar to a problem in homework 2.

Practice #2: Use Biopython to download the reference for pubmed id 11356062

Practice #3: You have these strings:
x="<type1><value>1234</value></type1><type2><value>5678</value></type2>"
y="<type3><value>1234</value></type3><type2><value>4567</value></type2>"
write a script to extract the value as an integer, but only the type2 value, in both cases.

Answers on the next page:

Practice #1:

```
x=[1,2,3,5,7,9,10,3,2,5,5,5,1,12,14,9,7]
keys=sorted(list(set(x)))
for i in keys: print i,x.count(i)
```

Practice #2:

```
from Bio import Entrez
Entrez.email=”your email”
handle=Entrez.efetch(db="pubmed",id="11356062",retmode="xml")
r=handle.read()
handle.close()
```

Practice #3:

```
x=”<type1><value>1234</value></type1><type2><value>5678</value></type2>”
y=”<type3><value>1234</value></type3><type2><value>4567</value></type2>”

sub=x[x.find(“<type2>”):x.find(“</type2>”)]
sub2=sub[sub.find(“<value>”)+7:sub.find(“</value>”)]
sub2=int(sub2)
```

similar for y. Two notes:

First, this would be an ideal opportunity to use a function. First, you could write a function that you would call with ‘x’ then again with ‘y’. You could also write a second function, to be called by the first function, to extract a substring between a start tag and and end tag.

Second point, this code, while functional, is not very robust. For example, if there were a space between the ‘/’ and ‘value’, which is permitted by XML, then the code would break. We will learn more about handling this when we discuss XML.

Warning, the above code has word-processing style double-quotes, which python will not be able to interpret. If you copy/paste this code, you’ll have to fix the quotes if you want it to work.