Contents

1 A Python primer

This assignment aims to guide you through your first steps in Python. First you will need a compiler. (If you're using an electronic version of this document, the following addresses will be clickable.)

There are many good Python compilers to choose from. If you would like to have one installed on your computer, I can recommend Spyder: https://code.google.com/p/spyderlib/. Otherwise, use an online one like the one at Codecademy: http://labs.codecademy.com/ or http://www.compileonline.com/execute_python_online.php. Codecademy also has an excellent interactive track of introductory exercises:

http://www.codecademy.com/en/tracks/python

If you do the Codecademy exercises, they will give you a well rounded introduction, and it will certainly pay off later. This primer is focused on what you need for this week's homework.

1.1 Basics

Python is a very intuitive programming language, it's incredibly useful, and in particular it's great for mathematics. In most compliers you will have an editor on the left, here you input the code, and a console on the right, where your code is executed. If you want something to be displayed in the console, use the **print** statement. For example, if you type:

print 7

into the editor and run the program (by clicking "compile" or the green run icon in Spyder,) the number 7 will appear in the console.

You can do math directly in **print**, for example:

```
print 1+1
print 10*2
print 10/2
```

will return 2, 20 and 5 respectively. If you want to raise a number to a power, for example 10^2, write 10**2.

```
print 10**2
```

will return 100.

Programming starts when you declare variables, the building blocks of your construction. If you input:

```
X=0
```

the computer will understand "there is a variable X and it's equal to the integer 0," and store an entry for variable X in its memory. If you input:

```
X=0
print X
```

and run the code, the number 0 will appear in the console.

```
X=0
X=X+1
print X
```

Will display "1". It declares X to be equal to 0, and then adds 1 to X. The line "X=X+1" declares a new value for X in terms of its old value. This would be very much not OK in math, but here equality is a command, not a statement of fact.

Question: What happens if you try to print 10/4? Or 10/7? The answer is wrong. Why do you think this might be? Try to figure out how to get the precise answer, or find out on the web.

1.2 Lists

An important variable type is a list. Use square brackets for those. For example, A=[a,b,c] is a list. Computers start counting at 0 rather than 1, so if you want to access the first entry you have to ask for the 0th one, if you want the second entry ask for the 1st and so on.

```
A=[a,b,c]
print A[0]
```

will return a. Sometimes we want to start with an empty list, and add entries to it. **append** attaches an extra entry at the end of the list, like so:

```
A=[]
A.append(0)
A.append(1)
A.append(0)
print A
```

This will return [0,1,0]. The statement "A.append(0)" means "append 0 to list A." We can combine what we know about lists so far like so:

```
A=[]
A.append(0)
A.append(1)
A.append(0)
A[2]=2
print A
```

After appending A[2] in line 4, I redefined it in line 5. The computer will use the latest value. An important instance of a list is "range". range(6) is the list where the entries are the first 6 integers.

```
print range(6)
```

will display [0,1,2,3,4,5].

1.3 Randomness

We're talking about probability, so we'll need some way for the compiler to simulate chance. This will require adding an extra line at the beginning of your code:

```
import random
```

to access the special package that can handle randomness. A basic command of that package is randrange. You need to specify an integer, and it will give you one of the integers that appear in the list *range* for that integer, with equal probability. For example:

```
import random
print random.randrange(2)
```

returns either 0 or 1. And:

```
import random
print random.randrange(6)
```

returns 0,1,2,3,4 or 5.

1.4 "for" loop

We will want to automate processes with many repetitions, so we don't need to type the same set of instructions many times. For that, most programming languages use *loops*. We're going to need a *for* loop to make our life easier.

```
for i in range(10):
    print i
```

means "for each entry i in list range(10), execute the instruction print i." The indent an the colon are important. Most compilers insert them automatically, but the code will not work without them.

The loop implicitly declares i each time. So behind the scenes, the loop does:

i=0
print i
i=1
print i
i=2
...
i=9
print i

Question: How can you use this loop to simulate many coin tosses?

2 Exercises

Write Python code that does all the following in order, and send it to ewa.j.infeld.gr@dartmouth.edu. Feel free to add a note telling me if you have programmed before, and how much programming experience you have (it will not affect your grade).

2.1 Basics

Declare a variable X equal to 10.
Print the exact result of dividing 10 by 4.

2.2 Lists

Create a list A.
Append 5 entries to A, all equal to 0.
Print A in its current form.
Add 1 to middle and last entry each.
Print A in its current form.

2.3 Randomness

Write code that simulates rolling two six-sided dice and adding the resulting numbers.
Print the result.

2.4 "for" loop

Create a list B of 100 entries, each entry a number 1,2,3,4,5 or 6 drawn at random.
Print B.