University of Engineering and Technology Lahore
Department of Computer Science and Engineering

Programming Fundamentals February 20, 2017

A computer is like a violin. You can imagine a novice trying first a phonograph and then a violin. The latter, he says, sounds terrible. That is the argument we have heard from our humanists and most of our computer scientists. Computer programs are good, they say, for particular purposes, but they aren't flexible. Neither is a violin......, until you learn how to use it.
Marvin Minsky, *Why Programming Is a Good Medium for Expressing Poorly-Understood and Sloppily-Formulated Ideas"*

Instructors:
Irfan Chaudhary and Hina Khalid

Contact information:

Name	email	office hours
Irfan Chaudhary	irfanc@mit.edu	TR 12-1
Hina Khalid	hinakhalid@uet.edu.pk	MR 11-12

If you cannot make these times, feel free to set up appointments or walk-in to the office any time you want.

About the course:
Welcome to Programming Fundamentals! As advertised, this course covers the fundamentals of programming. We will be using Python 3 as our programming language. This course is all about using different programming styles to control complexity. Hence during the semester you will get introduced to various programming styles.

Textbook:
Free online textbook
"Composing Programs" by John DeNero. The website is `http://composingprograms.com`.
Parent textbook: "Structure and Interpretation of Computer Programs" by Abelson, Sussman, and Sussman. In Scheme but probably the only book on programming you ever need to read.

Website:
`http://inst.eecs.berkeley.edu/~cs61a/fa16/`

Course learning outcomes:
Students should be able to

- Use control structures, basic functions, and primitive data types to write simple programs (CLO 1)
- Apply the techniques of building abstractions with functions (CLO 2)
- Apply principles of data abstraction to programming (CLO 3)

- Design software based on the principles of OOP (CLO 4)
- Modify an interpreter for a simple language like Scheme(CLO 5)

Weekly schedule:

Week	Topic	CLO
1	Introduction. Expressions. Functions	1
2	Control. Higher order functions	1, 2
3	Recursion. Examples. Quiz 1 (till section 1.5)	2
4	Data abstraction, sequences.	3
5	Mutable data. OOP. Quiz 2(chapter 1)	3, 4
6	Implementing classes and objects. Abstractions.	4
7	Recursive objects	4
	MIDTERM	
8	Wrap up of data abstraction	4
9	Functional programming	5
10	Scheme, exceptions	5
11	Calculator. Quiz 3 (till section 3.3 + Scheme)	5
12	Interpreters	5
13	To be decided	
14	To be decided	
	FINAL	

Advice on surviving the course:

Most (or may be all) of you have been star students all your life. This course is probably the first time in your life that you will be intellectually challenged. The course is full of interesting and deep concepts. Do not hesitate to ask for help. Remember that the only stupid question is the one that never got asked. We have been teaching long enough to say with confidence that if you ask what you consider to be "the stupidest question", more than half the students in the class will not know the answer.

The most important part of the course is to do your homeworks/labs/projects. In fact we can say that doing the homeworks/labs/projects IS the course. We cannot overemphasize this point. Computer science cannot be learnt passively. Do not copy homeworks/labs/projects. If you copy, you are only cheating yourself. If you cheat yourself, you will not learn the material and not do well in exams. Do not come to us and tell us that " I have not done well in this exam" yet "I spent the last 4 days preparing for this exam" or "I know everything in the book" or "I learnt by heart all the solutions of all the homeworks/labs/projects you gave us" or "I attempted all the homeworks/labs/projects". The only thing that matters in the course is what fraction of the homeworks did you correctly do yourself. This is not an easy course and the homeworks/labs/projects will be challenging. When you encounter a difficult problem, do not give up. Talk to your friends or come and talk to us. Do not hesitate to come and seek help from the staff. After all this is what we are there for. If you do not copy and do not give up, you will do well in the course. More importantly you will learn amazing stuff and start enjoying solving difficult problems.

To summarize here are our survival tips. Please follow them.

- Do not lag behind. The pace of the course is such that if you fall behind, you will find it very difficult to catch up.
- Follow the algorithm:

```
While (homework and lab is not completed properly)
    Listen to online lecture
    Ask questions
    Read the book
    Ask questions
    Attempt the homework and labs
    Ask questions
```

- Whenever you do not understand anything, ask for help. Do not hesitate at all.
- Do the homeworks/labs/projects religiously. Start as soon as possible. Do not wait till a day before the due date to start doing the homework/lab/project. You will not finish it. While the assignments are not too hard, they will require some work.
- The most important thing: Get rid of this notion that you can study for this course two or three days before an exam. If you have not worked every week on the homeworks/labs/projects, we can guarantee you that you will not do well in this course. Year after year, students come in with the belief that they can learn the material a few days before an exam. It does not happen. It may have happened till your F.Sc. or your A-levels. But it will not happen any more.
- **Do the homeworks and labs yourself!**

Grading:

- Weekly homeworks - 10%.
- Quizzes - 20%.
- Midterm - 30%
- Final - 40%

You may feel that our advice on surviving the course is flawed, since only a maximum of 10% of the grade is based on the homework/labs/projects. It seems that you should be focusing on quizzes, midterm and the final. However, please understand that actually your entire grade is based on the homework/lab/project assignments. If you do not do your homework/lab/project assignments, you will not do well in your quizzes, midterm and final. Conversely if you do your homework assignments you will do well in the course.

Policy on collaboration:

Understand the difference between collaboration and cheating. Cheating is when you copy an assignment from a friend, collaboration is when you discuss a problem with a friend. When you write a solution, you should do it alone. When you write code, you should be alone in front of the computer. Collaborate but do not cheat! We will have a zero-tolerance policy towards cheating. If we find people cheating in a quiz, we are very much inclined to give them zeros on all quizzes. Similarly if you copy a homework/lab/project assignment, it will lead to a zero in this homework/lab and zeros on two other homeworks. All these rules apply to those who copy and to those who allow their work to be copied. More important than all of these retributions is the fact that you will miss out learning about some of the coolest material on the planet.

Lecture etiquette:

- If you do not understand something, raise your hand and ask a question.
- If you come in late, slip in without greeting your instructor or your fellow students.
- Make sure your mobile phones are switched off or are in silent mode.
- Do not whisper. If 100 students start to whisper it becomes noise. Any student found whispering will lose a point in the course.
- If your fellow student is asking a question, listen to it quietly.

Final word:

For many of you this is the first time away from home and it is also the first time you will get exposed to a new way of thinking and learning. This can be stressful. If you feel stressed out, come and talk to us. We are there to help. So, let us roll up our sleeves, and start having fun with computer science. Hopefully it will be a ride of a lifetime!

> *I think that it's extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun.*
> Alan J. Perlis