Leo Irakliotis' courses

Leo Irakliotis

Jan 30, 2020

COURSES

This is a comprehensive site with material and information from the courses I teach.

CHAPTER

ONE

COMP 163 DISCRETE STRUCTURES

1.1 Course organization and logistics

1.1.1 Course outline

The course will cover the following topics.

- Set theory
- Logic
- Proofs
- Number theory
- Trees and graphs
- Algorithms
- Counting and probabilities
- Combinatorial circuits
- Finite state machines
- Languages and automata

1.1.2 Ground rules

Be proactive

In my experience, the single most common cause of poor performance in any course is when students are not proactive. When you are proactive you can recognize and address any issues with your courses. And learning to be proactive is one of the most valuable skills you can aquire in college.

Here's how you can be proactive, so you can succeed in this coure.

- Review homework (or other take-home tests) within 24 hours of assignment. Identify questions that may be challenging. Seek tutoring to work over these questions. The Department of Computer Science offers tutorting services almost on a daily basis.
- A grade of less than 50% in any assignment, is a warning sign. Compare your solution to the published solution. Try to identify the differences. Speak with a tutor if the differences are not clear.
- Come to class and take notes. Form or join a study group comprising fellow students.

- If, for any reason, you need to be absent from class, let me know in advance. I don't need to know the reason, but I need to know about the duration of your absense.

Textbook

There is no required textbook for this course. Reading material will be posted online as needed.

Grading scheme

Course performance is determined using in-class assignments, weekly homework, a take-home midterm, and a take-home final. Each assessment component carries a 25% weight. Grading scheme aside, the objective of assessments in this course is to ensure **that students are learning.** In this context students make mistakes, understand them, and do not repeat them.

Deadlines

Late submissions will not be accepted. Students are responsible for the timely return of homework and exam assignments. This responsibility requires students to notify the instructor in advance if they anticipate that they may not return an assigment by its stated deadline.

Bona fide emergencies are, of course, exempt from this rule. Students dealing with an emergency should take care of it first and notify the instructor when time and obligations permit.

Homework, midterm, and final assignments are due one week after assigned. These assignments should be returned via Sakai.

In-class assignments

For in-class assignments, students will have 20-30 minutes to complete them at the end of class session. For consistency, **in-class assignments will be held midweek,** on Wednesday or Thursday. There will be about 10 in-class assignments. Students will not be penalized for missing up to 3 of these assignments.

Exam dates

The final exam will be a take-home assignment. It will become available at 4:15 PM on 4/20/20 and due at 4:15 PM on Monday 4/27/20 (subject to change).

The midterm exam will be a take-home assignment. It will become available on Sakai at 4:15 PM on 2/21/20 and due at 4:15 PM on Friday 4/28/20 (subject to change).

Student hours - Spring 2020

Student hours (some call them "office hours") are Monday and Wednesday, 10:30 AM to 12:30 PM and 2 PM to 3:30 PM. Walk-ins are welcome but students who make an appointment take precedence. My office is in **Doyle 207**. I am also available via Zoom and Google Hangouts (video conferencing).

Academic integrity

Students are expected and encouraged to work together. It is also expected that students will explore the vastness of the internet to discover information, knowledge, and solutions to problems. I consider all that to be part of your learning experience. It is up to you, however, to demonstrate your learning.

In practical terms this means that you should be able to describe in your own words how a piece of code works or how you arrived to a mathematical derivation, etc. Failure to do so will affect your course performance and, ultimately, your grade. It may also raise concerns related to academic integrity. *Verbatim* use of material obtained from others is considered a violation of the university's policies for academic integrity.

Professionalism

In additional to the technical content of the course, there is a professional element to it. The professional element of the course is meant to cultivate your **soft skills.** These skills are highly sought after by employers. Soft skills include communication skills, neatness, punctuality, dependability, ability to work in teams, problem solving skills, etc.

In the context of this course, professionalis and soft skills are as follows.

- Be clear in your communications. When sending an email, make sure that it has an opening, an objective, and a closing.
- The opening is a greeting line. "Hey" is never an approrpriate opening in professional communincations.
- The objective is the main part of your email. Be precise and concise. For example, instead of writing "may I ask a question", just ask the question.
- The closing is a statement as to what you would like the other party to do in response to your message.
- Be respectful of others' time. Here're two examples:
- Come to meetings (class, student (aka office) hours, study groups, tutoring, etc) **prepared.** If the meeting is about reviewing your code, make sure that your laptop is sufficiently charged, the computer is turned on, and the code is already loaded into some editor. Do not spend 5 minutes in the meeting finding a power outlet, turning on and booting the computer, and firing up an editor.
- When in group meeting (e.g., class, study group, etc) keep your questions relevent to the objective of the meeting.
- When making an appointment and you realize that you cannot attend the meeting, notify the other party as soon as possible.
- Be organized and neat.
- When turning in homework or other assignments, make sure that your writing is legible and the overall appearance reflects quality and care. Content is always the most critical aspect of your work; but appearance counts too.
- When submitting code, make sure the files are properly named. For example `homework.java` may make sense on your side of the shop, but from the grader's perspective it's not very helpful. Instead, give your file a unique name that identifies it back to you, e.g., `JaneDoe-homework-2.java`.

Zoom

You need to download and install Zoom to your mobile device or laptop. Zoom is the University's preferred tool for videoconferencing. In case of inclement weather or travel delays that prevent us from meeting on campus, we'll use Zoom for an online class session. You can also use Zoom to meet with me during Student Hours, if one of us is not on campus.

1.1.3 Formal stuff

- The university's official Academic Calendar
- Every effort is made in this course to use open source, freely available material. However materials from the course cannot be shared outside the course without the instructor's *written permission.*
- **Duty to report.** Faculty and staff of the University have a mandated responsibility to report any incidents of gender-based misconduct that they are made aware of, even if it happened in the past. Gender-based misconduct includes discrimination based on actual or perceived sex, sexual orientation, gender expression or identity, or pregnancy or parenting status; dating and domestic violence; sexual misconduct (including sexual assault, sexual harassment, and sexual exploitation); and stalking.

1.2 Reading material

The *recommended* textbook for the course is *Discrete Mathematics* by Richard Johnsonbaugh. The book is currently in its 8th edition. If you want the book, I suggest that you buy a 6th or 7th edition; they are usually cheaper and just as good.

1.3 Leo's notes

1.3.1 Mathematical Induction

Mathematical induction – often, just induction – is one of the simplest and yet most challenging concepts to master. It is also one of the most rewarding proof techniques. Induction allows us to prove a property for natural numbers, for which we have a hunch but we need solid proof. For example, we believe that the sum of the first n numbers is $n(n+1)/2$. In fact, every time we try for a different n, our hunch turns out to be true:

$$n = 3,\ \ 1+2+3 = 6 = \frac{3(3+1)}{2} = \frac{12}{2}$$

$$n = 4,\ \ 1+2+3+4 = 10 = \frac{4(4+1)}{2} = \frac{20}{2}$$

$$n = 5,\ \ 1+2+3+4+5 = 15 = \frac{5(5+1)}{2} = \frac{30}{2}$$

There is a story that when Gauss was 10 years old, he derived the sum formula

$$1+2+\ldots+n = \frac{n(n+1)}{2}$$

with a simple observation. When he was asked by a teacher to calculate the sum for $n = 10$, he noticed that

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 =$$

$$(1 + 10) + (2 + 9) + (3 + 8) + (4 + 7) + (5 + 6) =$$

$$11 + 11 + 11 + 11 + 11 =$$

$$5 \cdot 11 = \frac{10}{2} \cdot (10 + 1) =$$

$$\frac{n}{2} \cdot (n + 1)$$

But deriving a formula is not the same as proving it. We can claim that the formula $1 + 2 + \ldots + n = n(n + 1)/2$ works for $n = 3$ or 4 or 5 or even 10, but that doesn't mean that it works for every natural number. We need a more rigorous tool to prove this property. The rigor, in this case, comes from the axiomatic foundation of natural numbers.

In 1889, Giuseppe Peano, an Italian mathematician, put together a few fundamental statements about natural numbers. These statements, known as Peano's Axioms, are the building blocks of the natural numbers set $\mathbb{N}$. An axiom is a statement that we accept as true. Peano stated nine axioms. The last one is the rigorous tool we need to prove $1 + 2 + \ldots + n = n(n + 1)/2$. It is worth, however, to review all five axioms first.

Before stating his axioms, Peano defined a *successor function,* i.e., a function that takes a natural number as input and returns the next successive natural number:

$$S(n) = n + 1$$

Using the successor function, we can write:

$$\begin{aligned} 1 &= S(0) \\ 2 &= S(S(0)) \\ 3 &= S(S(S(0))) \\ &\vdots \end{aligned}$$

Peano's Axioms

1. 0 is a natural number.
2. For every natural number x, its successor $S(x)$ is also a natural number.
3. For two natural numbers x, y, if $x = y \Leftrightarrow S(x) = S(y)$.
4. There is no natural number x such that $S(x) = 0$.
5. If a set K contains 0, and if for every natural number n being in K implies that $S(n)$ is also in K, then K contains every natural number.

The last axiom is the basis for mathematical induction. Let's demonstrate it with the summation formula $1 + 2 + \ldots + n = n(n + 1)/2$. We want to show that the formula works for every natural number. So far, we have shown that it works for $n = 3, 4, 5$ and 10.

We start by defining a set K to contain the natural numbers that we have shown to work with the formula: $K = \{3, 4, 5, 10\}$. We can show that 0 also belongs to K, because $0 = 0 \cdot (0 + 1)/2$. Thus, set K fulfills the first condition of the induction axiom, i.e., $0 \in K$.

The next step is to *assume* that a natural number n is in K and then *prove* that $n+1$ is also in K. If we do, then we prove (based on Peano's fifth axiom) that all natural numbers are in set K; or in other words $K = \mathbb{N}$. And why is this important? Because K is the set with the numbers for which the sum formula works. And if $K = \mathbb{N}$, then the formula works for every natural number, which is the proof we wanted!

Let's assume then that $n \in K$, therefore:

$$1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

Next we examine if and how we can show that $n+1$ is in K. If — and for now, that's a big if — this is true, if $n+1$ is indeed in K, we expect that

$$1 + 2 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}$$

Our objective now is to start from the left side of the expression above and see if we can arrive to the right side. When we observe the left side:

$$1 + 2 + \cdots + n + (n+1)$$

we notice that it contains the expression from the case for n. And since we assumed that the case is true, we can substitute $1 + 2 + \ldots + n$ with $\frac{n(n+1)}{2}$. The substitution give us:

$$1 + 2 + \cdots + n + (n+1) =$$

$$\frac{n(n+1)}{2} + (n+1) =$$

$$\frac{n(n+1) + 2(n+1)}{2} =$$

$$\frac{(n+1)(n+2)}{2}$$

And so, we have shown that

$$1 + 2 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}$$

and therefore $n+1 \in K$. Or, in other words, every natural number belongs to K and thus $K = \mathbb{N}$.

The "price of admission" to set K is for a natural number n to satisfy the formula $1 + 2 + \ldots + n = n(n+1)/2$. And since we showed that all natural numbers are in K, we have proved that the formula works for any natural number.

A template for induction proofs

Induction problems ask us to prove a property of the form

$$L(n) = R(n)$$

In the example above, $L(n) = 1 + 2 + \ldots + n$ and $R(n) = n(n+1)/2$.

Given a problem in the form $L(n) = R(n)$, the induction proof can be done in three steps.

Step 1

Show that $L(n) = R(n)$ holds for $n = 1$. (Yes, Peano says to show for $n = 0$ but 1 is just as good).

Step 2

Assume that $L(n) = R(n)$ holds for $n = k$, i.e., $L(k) = R(k)$ is true.

Step 3

Start with $L(k + 1)$ and show that it equals $R(k + 1)$. You will notice that the expression $L(k + 1)$ always contains $L(k)$. And since you just assumed that $L(k) = R(k)$ in Step 2, you can write

$$\begin{aligned} L(k+1) &= L(k) + (n+1) \\ &= R(k) + (n+1) \end{aligned}$$

From here, the problem is reduced to simple (usually) derivations to show that $R(k) + (n + 1) = R(k + 1)$. Showing that, completes the proof.

Examples

The summation formula

Prove that

$$1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

We worked this formula above to demonstrate the inductive method. We repeat the proof here in a more formal manner.

Step 1

For this step, first we need to identify the expressions $L(n)$ and $R(n)$. In this case, it's fairly easy:

$$L(n) = 1 + 2 + \ldots + n$$

$$R(n) = \frac{n(n+1)}{2}$$

Next we have to test if $L(n) = R(n)$ for a specific value of n. Peano said to use $n = 0$ but we prefer to test for $n = 1$.

$$L(1) = 1$$

$$R(1) = \frac{1(1+1)}{2} = 1$$

So, $L(1) = R(1)$.

Step 2

This is a simple and quick step. We only have to state that for an arbritrary $n = k$ the expression $L(k) = R(k)$ is true; in other words:

$$1 + 2 + \ldots + k = \frac{k(k+1)}{2}$$

Step 3

Here we use the next number up from the previous step. In Step 2 we looked at $n = k$. The next number up is $k + 1$. The question we must answer in this step now is if $L(k + 1)$ equals $R(k + 1)$.

$$L(k+1) = 1 + 2 + \ldots + k + (k+1)$$

Notice that $L(k + 1) = L(k) + k + 1$. Step 2 empowers us to replace $L(k)$ with $R(k)$, therefore

$$L(k+1) = \underbrace{1 + 2 + \ldots + k}_{=L(k)=R(k)} + (k+1)$$

$$= R(k) + (k+1)$$

$$= \frac{k(k+1)}{2} + (k+1)$$

$$= \frac{(k+1)(k+2)}{2}$$

$$= R(k+1)$$

We just proved that given $L(k) = R(k)$ the equation holds for the successor of k as well, $L(k + 1) = R(k + 1)$, and thefore the original expression

$$1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

is true for every natural number.

The stamps problems

Show that any postage amount greater than \$0.17 can be formed with \$0.03 and \$0.10 stamps. For example, \$0.22 worth of postage comprises one \$0.10 stamp and four \$0.03 stamps.

Step 1

Problems like this one do not have a readily available expression of the form $L(n) = R(n)$. We have to derive the formula by restating the problem. Basically, the problem says that any number greater than 17 can be written as the sum of multiples of 3 and multiples of 10:

$$n = 3a + 10b$$
$$n \geq 18$$

Here $L(n) = n$ but what about $R(n)$? The right side of the question does not contain n. Or does it? We can always write $3b + 10b = 3a + 10b + 0n$, and this will be our $R(n)$. So the problem is restated as follows:

$$\underbrace{n}_{L(n)} = \underbrace{3a + 10b + 0n}_{R(n)}$$

Now that we have established $L(n)$ and $R(n)$ we have to test that they hold true. We cannot test for $n = 1$ because the problem states that the property applies only to numbers $n \geq 18$. So we should try for $L(18) = R(18)$, in other words, can we write 18 as a sum of multiples of 3 and 10? Indeed we can:

$$18 = 3a + 10b$$
$$\text{when } a = 6, b = 0$$

Step 2

This is the easiest step of the proof. We assume that the property holds for an arbitrary integer k, i.e., $L(k) = R(k)$.

Step 3

Can we now prove that the property works for $S(k)$, by working our way from $L(k + 1)$ to $R(k + 1)$. Just remember that for this problem, R simply means a sum of multiples of 3 and 10.

$$\begin{aligned} L(k+1) &= k + 1 \quad (\text{and from step 2, } k = 3a + 10b) \\ &= 3a + 10b + 1 \quad (\text{rewrite } 1 = 10 - 9) \\ &= 3a + 10b + 10 - 9 \\ &= 3(a - 3) + 10(b + 1) \quad (\text{substituting } a' = a - 3,\ b' = b + 1) \\ &= 3a' + 10b' \end{aligned}$$

What the expression above says is that $k+1$ is also a sum of multiples of 3 and 10, and therefore $L(k+1) = R(k+1)$.

CHAPTER

TWO

COMP 170 INTRODUCTION TO JAVA PROGRAMMING

2.1 Course organization and logistics

2.1.1 Course outline

The course will cover the following topics.

- Variables and Types
- Control Structures
- Expressions and Evaluation
- Functions
- Problem Solving
- Object Oriented Program Structure
- Console and Text Input and Output
- Debugging, Testing, and Program Structure
- Arrays, Data Organizations and Use

The *tentative* weekly schedule for the course is:

1. Course Review; Required Preparation; Tools Introduction; Java Intro, Hello World
2. Arithmetic; Variables; Into to Types: String Class; Precedence; Intro to Console I/O
3. Control Structures; Decision Making; Boolean Logic
4. Control Structures: Loops (while, do, for, for each)
5. Exam ONE; Introduction to Problem Solving: Pseudo Code; Playing Computer
6. Functions, formal parameters; argument value passing; return values
7. Break, time approximate, also for making up topics
8. Stepwise refinement; top down design; problem solving strategies; Assertions
9. Objects and Classes: attributes and methods, public and private, constructors, overloading
10. Arrays and data organization; their use in classes; Simple Text File I/O; Brief UML
11. How to Build a Custom App of Your Own; Problem Decomposition; More UML
12. Consulting on Project Work; OO Interfaces; Reenforce OO design and programming
13. Consulting on Project Work; More Class / Object Examples; Assertions and Testing

The course is an introduction to computer programming, covering basic concepts using a popular object-oriented (OO) programming language. Its structure is based around basic questions such as: what is an algorithm? How does one write, debug, run ("execute"), and test an effective computer program? How does one convert an algorithm into a computer program? How does one judge a program? What does "object-oriented" mean?

This is a programming intensive course. Additional time and effort on reading, preparing, and programming will be required outside of class time.

Textbook

The textbook for this course is *Java, An Introduction to Problem Solving and Programming,* 8th edition, by Walter Savitch.

Computer Equipment

You will need access to a desktop or laptop computer with a simple text editor and a Java compiler. BlueJ is a convenient tool, called an Integrated Development Environment (IDE), that combines text editing and the Java Virtual Machine. As the course progresses we will migrate to IntelliJ, a more sophisticated and comprehensive IDE.

2.1.2 Ground rules

Be proactive

In my experience, the single most common cause of poor performance in any course is when students are not proactive. When you are proactive you can recognize and address any issues with your courses. And learning to be proactive is one of the most valuable skills you can aquire in college.

Here's how you can be proactive, so you can succeed in this coure.

- Review homework (or other take-home tests) within 24 hours of assignment. Identify questions that may be challenging. Seek tutoring to work over these questions. The Department of Computer Science offers tutorting services almost on a daily basis.
- A grade of less than 50% in any assignment, is a warning sign. Compare your solution to the published solution. Try to identify the differences. Speak with a tutor if the differences are not clear.
- Come to class and take notes. Form or join a study group comprising fellow students.
- If, for any reason, you need to be absent from class, let me know in advance. I don't need to know the reason, but I need to know about the duration of your absense.

Grading scheme

Course performance is determined using in-class assignments, weekly homework, a take-home midterm, and a take-home final. Each assessment component carries a 25% weight. Grading scheme aside, the objective of assessments in this course is to ensure **that students are learning.** In this context students make mistakes, understand them, and do not repeat them.

Deadlines

Late submissions will not be accepted. Students are responsible for the timely return of homework and exam assignments. This responsibility requires students to notify the instructor in advance if they anticipate that they may not return an assigment by its stated deadline.

Bona fide emergencies are, of course, exempt from this rule. Students dealing with an emergency should take care of it first and notify the instructor when time and obligations permit.

Homework, midterm, and final assignments are due one week after assigned. These assignments should be returned via Sakai.

In-class assignments

For in-class assignments, students will have 20-30 minutes to complete them at the end of class session. For consistency, **in-class assignments will be held midweek,** on Wednesday or Thursday. There will be about 10 in-class assignments. Students will not be penalized for missing up to 3 of these assignments.

Exam dates

The final exam will be a take-home assignment. It will become available on Sakai at #:## PM on 4/##/20 and due at #:## PM on Monday 4/##/20 (subject to change).

The midterm exam will be a take-home assignment. It will become available on Sakai at #:## PM on 4/##/20 and due at #:## PM on Monday 4/##/20 (subject to change).

Student hours - Spring 2020

Student hours (some call them "office hours") are Monday and Wednesday, 10:30 AM to 12:30 PM and 2 PM to 3:30 PM. Walk-ins are welcome but students who make an appointment take precedence. My office is in **Doyle 207**. I am also available via Zoom and Google Hangouts (video conferencing).

Academic integrity

Students are expected and encouraged to work together. It is also expected that students will explore the vastness of the internet to discover information, knowledge, and solutions to problems. I consider all that to be part of your learning experience. It is up to you, however, to demonstrate your learning.

In practical terms this means that you should be able to describe in your own words how a piece of code works or how you arrived to a mathematical derivation, etc. Failure to do so will affect your course performance and, ultimately, your grade. It may also raise concerns related to academic integrity. *Verbatim* use of material obtained from others is considered a violation of the university's policies for academic integrity.

Professionalism

In additional to the technical content of the course, there is a professional element to it. The professional element of the course is meant to cultivate your **soft skills.** These skills are highly sought after by employers. Soft skills include communication skills, neatness, punctuality, dependability, ability to work in teams, problem solving skills, etc.

In the context of this course, professionalis and soft skills are as follows.

- Be clear in your communications. When sending an email, make sure that it has an opening, an objective, and a closing.
- The opening is a greeting line. "Hey" is never an approrpriate opening in professional communincations.
- The objective is the main part of your email. Be precise and concise. For example, instead of writing "may I ask a question", just ask the question.
- The closing is a statement as to what you would like the other party to do in response to your message.
- Be respectful of others' time. Here're two examples:
- Come to meetings (class, student (aka office) hours, study groups, tutoring, etc) **prepared.** If the meeting is about reviewing your code, make sure that your laptop is sufficiently charged, the computer is turned on, and the code is already loaded into some editor. Do not spend 5 minutes in the meeting finding a power outlet, turning on and booting the computer, and firing up an editor.
- When in group meeting (e.g., class, study group, etc) keep your questions relevent to the objective of the meeting.
- When making an appointment and you realize that you cannot attend the meeting, notify the other party as soon as possible.
- Be organized and neat.
- When turning in homework or other assignments, make sure that your writing is legible and the overall appearance reflects quality and care. Content is always the most critical aspect of your work; but appearance counts too.
- When submitting code, make sure the files are properly named. For example `homework.java` may make sense on your side of the shop, but from the grader's perspective it's not very helpful. Instead, give your file a unique name that identifies it back to you, e.g., `JaneDoe-homework-2.java`.

Zoom

You need to download and install Zoom to your mobile device or laptop. Zoom is the University's preferred tool for videoconferencing. In case of inclement weather or travel delays that prevent us from meeting on campus, we'll use Zoom for an online class session. You can also use Zoom to meet with me during Student Hours, if one of us is not on campus.

2.1.3 Formal stuff

- The university's official Academic Calendar
- Every effort is made in this course to use open source, freely available material. However materials from the course cannot be shared outside the course without the instructor's *written permission.*
- **Duty to report.** Faculty and staff of the University have a mandated responsibility to report any incidents of gender-based misconduct that they are made aware of, even if it happened in the past. Gender-based misconduct includes discrimination based on actual or perceived sex, sexual orientation, gender expression or identity, or pregnancy or parenting status; dating and domestic violence; sexual misconduct (including sexual assault, sexual harassment, and sexual exploitation); and stalking.

2.2 Reading material

The textbook for the course is *Java: An Introduction to Problem Solving and Programming* by Walter Savitch.

2.3 Leo's notes

2.3.1 Weekly Reviews - Spring 2020

Review of Week 1 and Week 2

During the first two weeks of the course we talked about the fundamentals of programming languages, how they interact with the computer hardware at a conceptual level, and how the steps we take to perform various tasks in every day life (such as cooking, for example) as similar to the contents of a computer program.

Our first program in Java

The typical first program in any language is a greeting to the world. In Java, this greeting is as follows.

```
public class helloWord {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Our first *real* program

Next, we wrote a program to covert kilograms to pounds, assuming a conversion factor of 2.2. In developing our approach to this program we essentially saw how the main operation involves three numbers:

$$p = c \cdot k$$

where c is the conversion factor, k is the weight in kilos, and p is the weight in pounds. We operationalize this equation in Java by assigning more meaningful names to our variables, and using the the operator $\star$ to perform the multiplication.

```
public class kilosPounds {
  public static void main(String[] args) {
    public static final double conversionFactor=2.2;
        double weightKilos = 10;
        double weightPounds;
        weightPounds = conversionFactor*weightKilos;
        System.out.println(weightKilos + "kilograms are " + weightPounds + " pounds");
  }
}
```

The program above also gave us an opportunity to discuss data types, and the difference between `int` variables and `double` variables.

Next, we introduced a technique to input data directly from the keyboard, allowing us to compile a program once and execute it multiple times:

```java
import java.util.Scanner;
public class kilosPounds {
    void main(String[] args) {
    public static final double conversionFactor=2.2;
        double weightKilos;
        double weightPounds;
        Scanner keyboard = new Scanner(System.in);
        weightKilos = keyboard.nextDouble();
        weightPounds = conversionFactor*weightKilos;
        System.out.println(weightKilos + "kilograms are " + weightPounds + " pounds");
  }
}
```

And finally we discussed about the ability to convert weights in either direction, i.e., from kilograms to pounds and from pounds to kilograms, by making our code a bit more flexible.

```java
import java.util.Scanner;
public class weightConversion {
  public static void main(String[] args) {

      final double K2P=2.2;
      double weightInput, weightOutput;
      char weightSystem;
      String unitsIn, unitsOut;

      Scanner keyboard = new Scanner(System.in);
      weightInput = keyboard.nextDouble();
      weightSystem = keyboard.next().charAt(0);

      if (weightSystem == 'K') {
          // convert from Kilos to pounds
          weightOutput = K2P * weightInput;
          unitsIn = " kilograms ";
          unitsOut = " pounds";
      } else {
          // convert from Pounds to kilos
          weightOutput = (1/K2P) * weightInput;
          unitsIn = " pounds ";
          unitsOut = " kilograms";
      }
      System.out.println(weightInput + unitsIn + "are " + weightOutput + unitsOut);
  }
}
```

In this context, we introduced Strings and emphasized their difference from `char` variables. That's why in line 12 above, we ask the Scanner object to read a String variable (via the `next()` method) and then we apply the `charAt(0)` method to obtain the first letter of the string, for our processing.,

Review of Week 3

In week 3 we looked at String variables and their relation to `char` primitive types. We also looked that the basic `if-then-else` control structure.

A not-so-simple look into `char` and Strings

One problem we saw in class has to do with the comparison between a *char* primitive and a *String* variable. Specifically, we saw that

```
char letter1 = 'a';
String letter2 = "a";
```

results into two variables that although they *appear* to hold the same information, i.e., the letter **a,** they are two different entities. And therefore, the comparison between the two of them is not possible:

```
if ( letter1 == letter2 ) ...
```

If we tried to compile a statement like the one above, we will get an error message. Because one entity is an object; the other is a primitive variable. The example we used in class is trying to compare two apples: on our left hand we have an apple. On our right hand we have a box that contains another apple. Both fruits are identical. But until we open the box, we are simply comparing an apple to a box. In this analogy, *String* is the boxed apple; *char* is the plain one. To make the entities comparable, in Java, we have to "deference" the String object, i.e., we must make it look like a primitive. This is accomplished with the *.equals()* method:

```
if ( letter2.equals(letter2) ) ...
```

If only things were that straightforward! Consider the following program (which you can copy and paste and compile and run to test it):

```
public class equalStrings {
  public static void main(String[] args) {

      // Part 1

      String s1 = "Hello";
      String s2 = "Hello";

      if ( s1 == s2 ) {
          System.out.println("The strings s1 and s2 are equal");
      } else {
          System.out.println("The strings s1 and s2 are NOT equal");
      }

      // Part 2

      String t1 = new String("Hello");
      String t2 = new String("Hello");

      if ( t1==t2 ) {
          System.out.println("The strings t1 and t2 are equal");
      } else {
          System.out.println("The strings t1 and t2 are NOT equal.");
      }
  }
}
```

In the code above, strings *s1* and *s2* are equal but *t1* and *t2* are not, even though both pairs contain the same data (the word "Hello"). What's happening here?

Java is smart enough to realize, as it is compiling the code above, that the value "Hello" is used at least twice: once for *String s1* and once more for *String s2*. So it makes these two variables reference the same memory location where a single copy of "Hello" is stored.

And what about variables *t1* and *t2*? There variables are created as **new** strings. This is forcing the compiler to allocate separate memory space for these two copies of "Hello".

The end result is that `s1==s2` evaluates to true because both entities point to the same memory location, but `t1==t2` evaluates to false because these entities point to different memory locations. In other words, the == operator between strings compares the entities' locations in memory, not their content. To compare the content between two entities that occupy different memory locations, as is the case with *t1* and *t2*, we must apply the *.equals()* method: `t1.equals(t2)` will evaluate to true.

Notice that the method `.equals()` works between entities of similar structure and complexity – in the example above, `t1` and `t2` are *string objects*, and as such have the similar complexity and structure that allows them to be compared to each other.

Assigning *s1* and *s2* to the same memory location is called *string interning*. String interning is beyond the scope of COMP 170. Its description however, can be found in the Java Language Specification, section 3.10.5 and in the documentation for the java.lang package.

Currency converter

It is the mismatch between `char` and `String` that prevents our currency conversion program to work correctly.

```
import java.util.Scanner;
public class currencyConverter {

    public static final double EURUSD = 1.2; // from EUR to USD

    public static void main(String[] args) {

        double amountToConvert, amountConverted;
        char fromThisCurrency;

        Scanner keyboard = new Scanner(System.in);
        amountToConvert = keyboard.nextDouble();
        fromThisCurrency = keyboard.next().charAt(0);

        if ( "D".equals(fromThisCurrency) ) {
            amountConverted = amountToConvert*(1/EURUSD);
        } else {
            amountConverted = amountToConvert*EURUSD;
        }

        if ( "D".equals(fromThisCurrency) ) {
            System.out.println ("Executing USD to EUR conversion");
            System.out.println ( "$ "+ amountToConvert + " equal to "
                + "E " + amountConverted);
        } else {
            System.out.println ("Executing EUR to USD conversion");
            System.out.println ( "E "+ amountToConvert + " equal to "
                + "$ " + amountConverted);
        }
```

(continues on next page)

(continued from previous page)

```
30          }
31  }
```

CHAPTER

THREE

COMP 180 DATA ANALYSIS FOR THE SCIENCES

3.1 Course organization and logistics

3.1.1 Course outline

The course will cover the following topics.

- Overview of analytics and visualization: narrating compeling stories with facts and data.
- The `Anaconda` navigator and `Jupyter` notebooks.
- Basic data analysis
- Data fit
- The *pandas* toolbox, pairplots, and visualization with *matplotlib*.
- Predictive analytics and elementary machine learning
- The Wisconsin Breast Cancer data set
- Surviving the HMS Titanic
- Telling the gender from self-narrative in online dating
- Natural language processing
- Document analysis: associating words and authors
- Sentiment analysis: controversial tweets
- Geospatial analytics
- The *bokeh* toolbox
- Chicago's food inspections
- Chicago police data

The course is an introduction to data science and visualization techniques, using Python. Students are expected to have a solid understanding of the basic properties of functions, with emphasis on polynomial functions, including linear and quadratic functions; Computing real and complex roots of polynomials; rational functions, absolute value functions, and inverse functions; and solutions of systems of equations. This is accomplished by having completed MATH 117 at LUC, or a similar course elsewhere.

As part of the course we'll cover matrix algebra, probabilities and statistics. Therefore, familiarity with the mathematical concepts in the paragraph above is essential. **There will be a series of in-class assignments the first week of the term, to assess that you have the prerequisite background to succeed in the course.**

This is an inherently programming-based course. We will use Python to conduct data analytics experiments and visualization. You don't have to know the language coming into the course, but you are expected to know *some* programming that will allow you to learn Python as you go along.

Textbook

There is no required textbook for this course. Reading material will be posted online as needed.

Computer Equipment

You will also need access to a computer (laptop or desktop) where you can install the Anaconda navigator. The computers in the classroom, as well as those in the *Information Commons* are equipped with a plain version of Anaconda that is sufficient for the early part of the course; however as the course progresses we'll need more advanced tools that are not available on the classroom's computers.

3.1.2 Ground rules

Be proactive

In my experience, the single most common cause of poor performance in any course is when students are not proactive. When you are proactive you can recognize and address any issues with your courses. And learning to be proactive is one of the most valuable skills you can aquire in college.

Here's how you can be proactive, so you can succeed in this coure.

- Review homework (or other take-home tests) within 24 hours of assignment. Identify questions that may be challenging. Seek tutoring to work over these questions. The Department of Computer Science offers tutorting services almost on a daily basis.
- A grade of less than 50% in any assignment, is a warning sign. Compare your solution to the published solution. Try to identify the differences. Speak with a tutor if the differences are not clear.
- Come to class and take notes. Form or join a study group comprising fellow students.
- If, for any reason, you need to be absent from class, let me know in advance. I don't need to know the reason, but I need to know about the duration of your absense.

Grading scheme

Course performance is determined using weekly homework, a take-home midterm, and a take-home final. Each assessment component carries a 33.33% weight. Grading scheme aside, the objective of assessments in this course is to ensure **that students are learning.** In this context students make mistakes, understand them, and do not repeat them.

Deadlines

Late submissions will not be accepted. Students are responsible for the timely return of homework and exam assignments. This responsibility requires students to notify the instructor in advance if they anticipate that they may not return an assigment by its stated deadline.

Bona fide emergencies are, of course, exempt from this rule. Students dealing with an emergency should take care of it first and notify the instructor when time and obligations permit.

Homework, midterm, and final assignments are due one week after assigned. These assignments should be returned via Sakai.

Exam dates

The final exam will be a take-home assignment. It will become available on Sakai at #:## PM on 4/##/20 and due at #:## PM on Monday 4/##/20 (subject to change).

The midterm exam will be a take-home assignment. It will become available on Sakai at 1 PM on 2/21/20 and due at 1 PM on Friday 2/28/20 (subject to change).

Student hours - Spring 2020

Student hours (some call them "office hours") are Monday and Wednesday, 10:30 AM to 12:30 PM and 2 PM to 3:30 PM. Walk-ins are welcome but students who make an appointment take precedence. My office is in **Doyle 207**. I am also available via Zoom and Google Hangouts (video conferencing).

Academic integrity

Students are expected and encouraged to work together. It is also expected that students will explore the vastness of the internet to discover information, knowledge, and solutions to problems. I consider all that to be part of your learning experience. It is up to you, however, to demonstrate your learning.

In practical terms this means that you should be able to describe in your own words how a piece of code works or how you arrived to a mathematical derivation, etc. Failure to do so will affect your course performance and, ultimately, your grade. It may also raise concerns related to academic integrity. *Verbatim* use of material obtained from others is considered a violation of the university's policies for academic integrity.

Professionalism

In additional to the technical content of the course, there is a professional element to it. The professional element of the course is meant to cultivate your **soft skills.** These skills are highly sought after by employers. Soft skills include communication skills, neatness, punctuality, dependability, ability to work in teams, problem solving skills, etc.

In the context of this course, professionalis and soft skills are as follows.

- Be clear in your communications. When sending an email, make sure that it has an opening, an objective, and a closing.
- The opening is a greeting line. "Hey" is never an approrpriate opening in professional communincations.
- The objective is the main part of your email. Be precise and concise. For example, instead of writing "may I ask a question", just ask the question.
- The closing is a statement as to what you would like the other party to do in response to your message.
- Be respectful of others' time. Here're two examples:

- Come to meetings (class, student (aka office) hours, study groups, tutoring, etc) **prepared.** If the meeting is about reviewing your code, make sure that your laptop is sufficiently charged, the computer is turned on, and the code is already loaded into some editor. Do not spend 5 minutes in the meeting finding a power outlet, turning on and booting the computer, and firing up an editor.
- When in group meeting (e.g., class, study group, etc) keep your questions relevent to the objective of the meeting.
- When making an appointment and you realize that you cannot attend the meeting, notify the other party as soon as possible.
- Be organized and neat.
- When turning in homework or other assignments, make sure that your writing is legible and the overall appearance reflects quality and care. Content is always the most critical aspect of your work; but appearance counts too.
- When submitting code, make sure the files are properly named. For example `homework.java` may make sense on your side of the shop, but from the grader's perspective it's not very helpful. Instead, give your file a unique name that identifies it back to you, e.g., `JaneDoe-homework-2.java`.

Zoom

You need to download and install Zoom to your mobile device or laptop. Zoom is the University's preferred tool for videoconferencing. In case of inclement weather or travel delays that prevent us from meeting on campus, we'll use Zoom for an online class session. You can also use Zoom to meet with me during Student Hours, if one of us is not on campus.

3.1.3 Formal stuff

- The university's official Academic Calendar
- Every effort is made in this course to use open source, freely available material. However materials from the course cannot be shared outside the course without the instructor's *written permission.*
- **Duty to report.** Faculty and staff of the University have a mandated responsibility to report any incidents of gender-based misconduct that they are made aware of, even if it happened in the past. Gender-based misconduct includes discrimination based on actual or perceived sex, sexual orientation, gender expression or identity, or pregnancy or parenting status; dating and domestic violence; sexual misconduct (including sexual assault, sexual harassment, and sexual exploitation); and stalking.

3.2 Reading material

The *recommended* textbook for the course is *Discrete Mathematics* by Richard Johnsonbaugh. The book is currently in its 8th edition. If you want the book, I suggest that you buy a 6th or 7th edition; they are usually cheaper and just as good.

3.3 Leo's notes

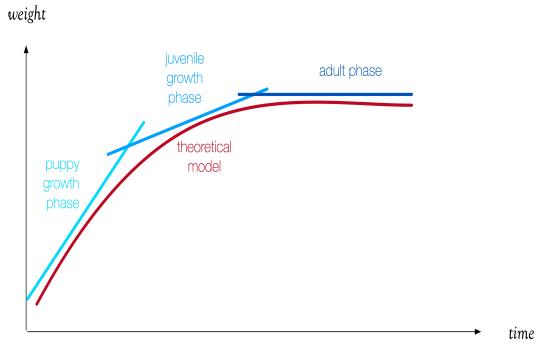
3.3.1 Weekly Reviews - Spring 2020

Review of Week 1 and Week 2

We are off to a good start. During the first two weeks of the course we reviewed some basic concepts in data analysis. We begun with a simple data analysis example to discuss data fit models, and then we moved on to reviewing the capabilities and utilities of the Anaconda tools. In the process you had a chance to assess your math skills for the course, learn how spreadsheets can sometimes provide insight into data analysis, and to begin using Jupyter Notebooks for data analysis.

The plan for the third week of the course to focus on the visualization utilities that are available for Python. This is based on the assumption that everyone has access to Anaconda and Jupyter book. This can be done in three ways: direct installation of the Anaconda distribution on your own computer; use of a virtual machine accessible via a web browser; or use of campus computers available at the Information Commons and in Crown 103/105 when the rooms are available.

Simple data fit models



We started with a simple example, looking at the weight measurements during the first 500 days of a puppy's life.

These measurements document a growth from 6 lbs to a near-adult weight of 41 lbs. In the early weeks of the dogs life, the growth is fast: she is gaining about a pound a week. Then the growth slows down and eventually levels off at 41 lbs.

The three distinct phases of growth can be approximated with linear models, like so:

$$w(t) = \begin{cases} 5t + w_b, \text{ while a puppy} \\ 0.3t + w_j, \text{ while a juvenile} \\ 0t + w_a, \text{ while an adult} \end{cases}$$

where $w(t)$ is the dog's weight at time t, w_b is the weight at birth, w_j is the starting juvenile weight, and w_a is the adult weight.

This three-stage model is problematic for at least two reasons. First, it requires clear boundaries between juvenile and adult phases. And second, it assumes abrupt changes in the growth pattern, which is not what happens in reality.

Three models are better than none, but one model is always better. A smooth continuous model often provides a better narrative. We found that such a model exists in the form

$$w(t) = w_a - \frac{(w_a - w_b)}{e^{ct}}$$

where c is a time-scaling coefficient, and e is, of course, the basis of natural logarithms.

But even this smooth model is problematic. Models help up predict the future – at least in the short term. This model presupposes knowledge of the future, i.e., it requires a value for the dog's adult weight. It would take a bit more work to derive a model that, given the puppy's weight at birth, will predict its adult weight.

Data analysis tools

We started analyzing the puppy's weight data using a spreadsheet. Spreadsheets are not ideal tools for data analysis but sometimes they work remarkably well. When dealing with small data sets, a spreadsheet can provide some insight that will inform further analysis. Spreadsheets, however, were not meant as data analysis tools. Often we need to analyze very large and complex data sets that may contain textual, categorical, and numerical, requiring advanced computations. For this kind of analysis it is best to use specialized tools (such as SPSS, R, etc) or general purpose programming languages that support data processing (such as Python, Java, etc).

Python is a good choice for data analysis for several reasons. First, it's relatively easy to learn. Second, it requires less overhead than other languages (for example, Java). Third, it is a language with *phenomenal* community support. And fourth, it is used to develop and maintain significant collections of utility programs to analyze and visualize data – we call these collections, libraries. It's those libraries that make Python a good choice for this course.

Anaconda is a distribution of Python that comes complete with various development environments. One of them is **Jupyter Notebooks** that will be the main tool for this course. This tool allows us to create interactive, web-based computational environments using Python for data processing and Markdown (a markup language) for documentation.

In exploring the possibilities of Python, we looked at three specific libraries that we will use extensively:

- *pandas*, is a collection of data analysis tools based on dataframes and other datastructures. Read mode about: pandas, dataframes, more about dataframes (a different perspective), data structures, and the full pandas documentation.
- *numpy*, is a collection of numerical methods and the most popular library of numerical techniques for Python. Read more about: numpy.
- *matplotlib*, is a popular collection of plotting utilities in Python, including an interface (*pyplot*) that resembles visualization utilities in MATLAB for those familiar with that system. Read more about: matplotlit, and review this interesting tutorial on *matplotlib*.

CHAPTER

FOUR

COMP 271 DATA STRUCTURES

4.1 Course organization and logistics

4.1.1 Course outline

The course will cover the following topics.

- Set theory
- Logic
- Proofs
- Number theory
- Trees and graphs
- Algorithms
- Counting and probabilities
- Combinatorial circuits
- Finite state machines
- Languages and automata

4.1.2 Ground rules

Be proactive

In my experience, the single most common cause of poor performance in any course is when students are not proactive. When you are proactive you can recognize and address any issues with your courses. And learning to be proactive is one of the most valuable skills you can aquire in college.

Here's how you can be proactive, so you can succeed in this coure.

- Review homework (or other take-home tests) within 24 hours of assignment. Identify questions that may be challenging. Seek tutoring to work over these questions. The Department of Computer Science offers tutorting services almost on a daily basis.
- A grade of less than 50% in any assignment, is a warning sign. Compare your solution to the published solution. Try to identify the differences. Speak with a tutor if the differences are not clear.
- Come to class and take notes. Form or join a study group comprising fellow students.

- If, for any reason, you need to be absent from class, let me know in advance. I don't need to know the reason, but I need to know about the duration of your absense.

Grading scheme

Course performance is determined using in-class assignments, weekly homework, a take-home midterm, and a take-home final. Each assessment component carries a 25% weight. Grading scheme aside, the objective of assessments in this course is to ensure **that students are learning.** In this context students make mistakes, understand them, and do not repeat them.

Deadlines

Late submissions will not be accepted. Students are responsible for the timely return of homework and exam assignments. This responsibility requires students to notify the instructor in advance if they anticipate that they may not return an assigment by its stated deadline.

Bona fide emergencies are, of course, exempt from this rule. Students dealing with an emergency should take care of it first and notify the instructor when time and obligations permit.

Homework, midterm, and final assignments are due one week after assigned. These assignments should be returned via Sakai.

For in-class assignments, students will have 20-30 minutes to complete them at the end of class session. For consistency, **in-class assignments will be held on Wednesday.** There will be about 6 in-class assignments. Students will not be penalized for missing up to 2 of these assignments.

Exam dates

The final exam will be a take-home assignment. It will become available on Sakai at 4:15 PM on 4/20/20 and due at 4:15 PM on Monday 4/27/20 (subject to change).

The midterm exam will be a take-home assignment. It will become available on Sakai at 4:15 PM on 2/21/20 and due at 4:15 PM on Friday 4/28/20 (subject to change).

Student hours

Student hours (some call them "office hours") are Monday and Wednesday, 10:30 AM to 12:30 PM and 2 PM to 3:30 PM. Walk-ins are welcome but students who make an appointment take precedence. My office is in Doyle 207. I am also available via Zoom and Google Hangouts (video conferencing).

Academic integrity

Students are expected and encouraged to work together. It is also expected that students will explore the vastness of the internet to discover information, knowledge, and solutions to problems. I consider all that to be part of your learning experience. It is up to you, however, to demonstrate your learning.

In practical terms this means that you should be able to describe in your own words how a piece of code works or how you arrived to a mathematical derivation, etc. Failure to do so will affect your course performance and, ultimately, your grade. It may also raise concerns related to academic integrity. * Verbatim use of material obtained from others is considered a violation of the university's policies for academic integrit

4.1.3 Formal stuff

- The university's official Academic Calendar
- Every effort is made in this course to use open source, freely available material. However materials from the course cannot be shared outside the course without the instructor's *written permission.*
- **Duty to report.** Faculty and staff of the University have a mandated responsibility to report any incidents of gender-based misconduct that they are made aware of, even if it happened in the past. Gender-based misconduct includes discrimination based on actual or perceived sex, sexual orientation, gender expression or identity, or pregnancy or parenting status; dating and domestic violence; sexual misconduct (including sexual assault, sexual harassment, and sexual exploitation); and stalking.

4.2 Reading material

The *recommended* textbook for the course is *Discrete Mathematics* by Richard Johnsonbaugh. The book is currently in its 8th edition. If you want the book, I suggest that you buy a 6th or 7th edition; they are usually cheaper and just as good.

4.3 Leo's notes