# CMSI 185-04
## Computer Programming

*http://myweb.lmu.edu/dondi/fall2016/cmsi185*

**Fall 2016**—Doolan 219, TR 2:55–4:10pm, 3 semester hours
Office Hours: TR 1:45–2:45pm, T 4:15–6:30pm, W 1–2:30pm,
R 5:30–6:30pm, or by appointment

John David N. Dionisio, PhD
email: dondi@lmu.edu
Doolan 106; (310) 338-5782

## Objectives and Outcomes

This course introduces you to the art and craft of *computer programming*, and in so doing seeks to also expose you to the discipline of computer science and the mindset of computational thinking. Long after you finish this course, my hope is that you will be able to:

1. **State what the field of computing is, and why it is important.**
2. **Craft well-structured, working programs.**
3. **Discern how some ways of programming are better than others.**
4. **Appreciate the value of design and testing in programming.**

In addition to the course-specific content, you are also expected to:

5. **Follow disciplinary best practices throughout the course.**

## Prerequisites/Prior Background

No prior technical nor programming background is expected in this course.

## Materials and Texts

- John David N. Dionisio and Ray Toal. *Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers*, Jones & Bartlett Learning, 2011.
- Assorted handouts, articles, and sample code to be distributed throughout the semester.

In addition, do not hesitate to look for further information regarding the concepts, techniques, tools, and paradigms that we will discuss.

## Course Work and Grading

Your final grade will be based on the percentage of the points you get for the following deliverables against the total number of possible points:

| | |
|---|---|
| GitHub and YouTube account setup | 40 points |
| Startup/freestyle programming | 100 |
| Programming data | 100 |
| Programming statements | 100 |
| Programming functions | 100 |
| One. Big. Program. | 100 |
| **Total** | 540 points |

Percentages ≥ 90% get an A– or better; ≥ 80% get a B– or better; ≥ 70% get a C– or better. I may nudge grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course.

### Term Portfolio

Your accumulated work for the semester comprises the *term portfolio*—the final, definitive artifact that demonstrates the course's outcomes. It is how you show that you have, indeed, accomplished the objectives of this course.

Demonstrate outcomes *1* and *2* to maximize the points for the startup/freestyle programming assignment; demonstrate outcomes *2–5* to maximize the points for all of the other program sets. The program sets are also cumulative; for example, the "Programming statements" assignment will assume that you have mastered the activities in "Programming data." And "One. Big. Program." kind of says it all, don't you think?

### Making Progress *Before* the Due Date

An assignment's number is its due date in *mmdd* format, and it is always due by midnight. Point values are based on the state of your assignments at that moment.

None of the assignments can be completed (well) overnight; they should be the result of steady progress from the moment they are assigned to the date they are due. "One and done" submissions will negatively affect the final score.

From the "Programming data" assignment onward, the benefit of steady progress will be even more concrete: you will be able to "pre-submit" your work early and have it undergo automated QA (quality assurance) that identifies areas of improvement. Eliminate these QA flags and make sure your programs work correctly "out of the box" by the due date to maximize your points.

### Version Control

Version control is an indispensable part of today's computer science landscape in industry, the academe, and the open source community. We will gently introduce you to version control in this course: once you get the hang of it, you'll wonder how anyone can program without it.

### Workload Expectations

In line with LMU's *Credit Hour Policy*, the workload expectation for this course is that for every one (1) hour of classroom instruction (50 scheduled minutes), you will complete at least two (2) hours of out-of-class work each week. This is a 3-unit course with 3 hours of instruction per week, so you are expected to complete 3 × 2 = 6 hours of weekly work outside of class.

### Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to notify me about this and keep up with the course. The last day to add or drop a class without a grade of W is September 2. The withdrawal or credit/no-credit deadline is November 4.

### Academic Honesty

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit to expulsion. It is never permissible to turn in work that has been copied from another student or copied from a source (including the Internet) without properly acknowledging the source. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in:

*http://academics.lmu.edu/honesty*

### Special Accommodations

Students with special needs who require reasonable modifications or special assistance in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (ADHD, autism spectrum, learning, physical, or psychiatric) needing academic accommodations should contact DSS (Daum 224, x84216) as early in the semester as possible. All requests and discussions will remain confidential. Please visit *http://www.lmu.edu/dss* for additional information.

## Topics and Important Dates

Specifics may change as the course progresses. University dates (italicized) are less likely to change.

| | |
|---|---|
| **August/ September** | Introduction to the field of computing (Chapter 1) |
| *September 2* | *Last day to add or drop a class without a grade of W* |
| | Starting to program; JavaScript specifics; data (Chapters 2 & 3) |
| **October** | Statements (Chapter 4) |
| **November** | Functions (Chapter 5) |
| *November 4* | *Withdraw/credit/no-credit deadline* |
| *November 23–25* | *Thanksgiving; no class* |
| **December** | Other topics *(time permitting)* |
| *December 16* | *One. Big. Program. assignment due* |

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it's too long to provide in writing).

### Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision; students are responsible for any changes or modifications distributed in class or posted to the course web site.

## Course Outcomes

| | | |
|---|---|---|
| **1** | **State what the field of computing is, and why it is important.** | |
| | *For this objective, you are expected to show some knowledge of the field, including but not limited to key concepts, terms, personalities, and events. You should know not only what or who these are but also why they matter. Mainstream news and creative work involving computing can be put in context.* | You will work toward this objective by reading/watching assigned material and participating in class discussions. You will show that you have accomplished this by writing answers to assigned questions. |
| **2** | **Craft well-structured, working programs.** | |
| | *This objective expects you to write code, and write it well.* | You learn to program by…programming. You will be told what you are doing well and what you aren't doing well. Keep doing what you're doing well, and change what you aren't doing well, and you will be fine. |
| **3** | **Discern how some ways of programming are better than others.** | |
| | *As programs become more sophisticated, you will find that different approaches will accomplish the same thing. However, some approaches will be better than others.* | Class discussion and feedback on your programs will guide you to this objective. Pay attention when you are told that you could have done something in another way. |
| **4** | **Appreciate the value of design and testing in programming.** | |
| | *Programming, when not done right, becomes error-prone busy work. Good design and effective testing will keep things in line. Robust testing gives you confidence that new work will not break prior accomplishments.* | When you are told to structure something in a certain way, do it, and make sure that you understand why this choice is being made (*Hint:* Ask questions if you don't get it.). |
| **5** | **Follow disciplinary best practices throughout the course.** | |
| *5a* | *Write code that is easily understood by programmers other than yourself.* | Donald Knuth (Who is he? Look him up.) aptly states that programming is not only telling a computer how to do something, but also telling a person how they would tell a computer to do something. |
| *5b* | *Meet all designated deadlines.* | |