48-782 Design Computation I

Fall Semester 2018 Mini A1 • 6 units • Tuesdays + Thursdays 3.00–4:20 pm (CFA 317)

Open to upper-year undergraduate and graduate students

Prerequisite: At least Junior standing.

Instructors: Ramesh Krishnamurti • ramesh@cmu.edu

Pedro Veloso • pveloso@andrew.cmu.edu

Office Hours: Tuesdays and Thursdays after class

Syllabus

Course description

This is a fast paced intense half-semester technical introduction to the fundamentals of object-oriented programming targeted at design students with an emphasis on producing clear, robust, and reasonably effective code.

We will cover a large subset of the Python programming language including a standard graphics library and programming paradigms. In order to reinforce the programming topics we will also introduce the generation and visualization of geometric form with graphics, animation and rule-based systems.

The course consists of lectures, computer cluster instruction and assignments.

This course assumes no prior programming experience.

This course serves as a prerequisite for 48-784 Design Computation II and 48-724 Scripting and Parametric Design.

Learning outcomes

Upon the successful completion of this course, students will be able to:

- write simple programs in Python to implement solutions to basic computing problems in design
- use sequential, conditional, and loop statements where appropriate in solving a programming task
- understand and use Python strings, lists, tuples, and dictionaries
- understand objects and classes
- understand recursive functions
- write simple event-driven graphics programs
- write simple form generators

References

There is no textbook but the following are useful references.

- Timothy A Budd. Exploring Python. McGraw-Hill. 2009.
- Mark Lutz. Learning Python, 5th Edition. O'Reilly. 2013.
- Marina von Steinkirch (bt3) An introduction to Python & Algorithms, 2nd Edition, 2013.
- Michael T Goodrich, Roberto Tomassia, Michael H Goldwasser, *Data Structures and Algorithms in Python*, Wiley 2013.

Other books that are useful:

- Kenneth Lambert, *Fundamentals of Python: From First Programs through Data Structures*, 2010.
- Allen B Downey, Think Complexity, Version 2.0.10, Green Tea Press, 2016.
- Jason Brownlee, *Clever Algorithms*: *Nature-Inspired Programming Recipes*, 2011.

Online resources

- Python >> general support for programming in Python
- Lynda >> great video tutorials on all things digital. Sign in through CMU for free access

Software platform

Repl, Sublime, …

Repl

All course material will be on Repl.

Canvas

Some additional course material may be placed on Canvas. We will be using Canvas for after class discussion. The system is highly catered to getting you help fast and efficiently from classmates and the teaching team.

Course Requirements

Assignments and quizzes

Grading

Grades are based on the **six best** weekly assignments and two quizzes. Students will receive feedback on each assignment and quiz. Each week students are given a mix of **in-class** and **out-of-class** exercises to complete. These exercises constitute the weekly assignment.

The breakdown for overall grade is as follows **Assignments 90%** (6× 15%) **Quiz 10%**

Grades are based on the following scale:

A: 90% and over **B:** 80-89% **C:** 70-79% **D:** 60-69% **R: <** 60%

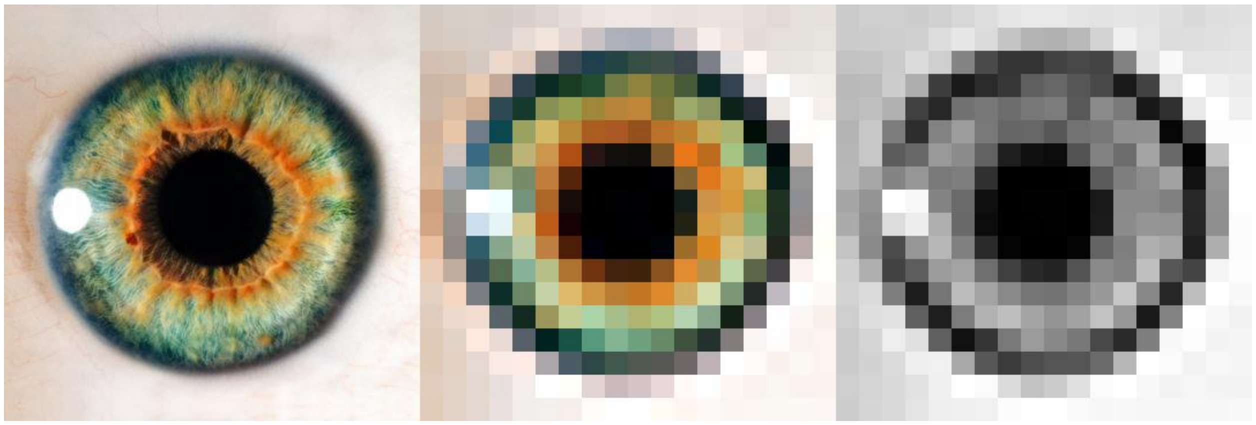
Students are not graded on a curve, however, we will consider the degree of difficulty experienced in the assignments in determining the final scores.

Example Assignments

The exercises of the course assignments will focus on the topics of programming, on solving problems and on generating visual compositions. There will be basic exercises focusing on one of these, while a weekly challenge will contain exercises integrating the different subjects. The following two exercises are examples of the kind of integrating tasks required in the assignments.

Example 1: Strings. loops and conditionals

Create the function **printImage(image, width, height)**. You will receive a string image containing a a grayscale representation of an image. Each character in image is a digit from 0 to 9, representing a value in gray scale (0 is black and 9 is white). You will use the following ascii characters to represent the colors: " .:-=+*#%@", from black to white (remember that the background of the terminal is black). The image is represented as an uni-dimensional sequence of characters, but it has 2 dimensions, width and height. Your print will have height + 2 lines, because you should add an empty line before and after the image (do not forget this!). In each line you will have to organize a sequence of width characters separated by " " (the image does not come with these spaces between characters).



For example, we reduced the resolution of the following image and converted it to grayscale (image on the right). Your function should be generate the following result:

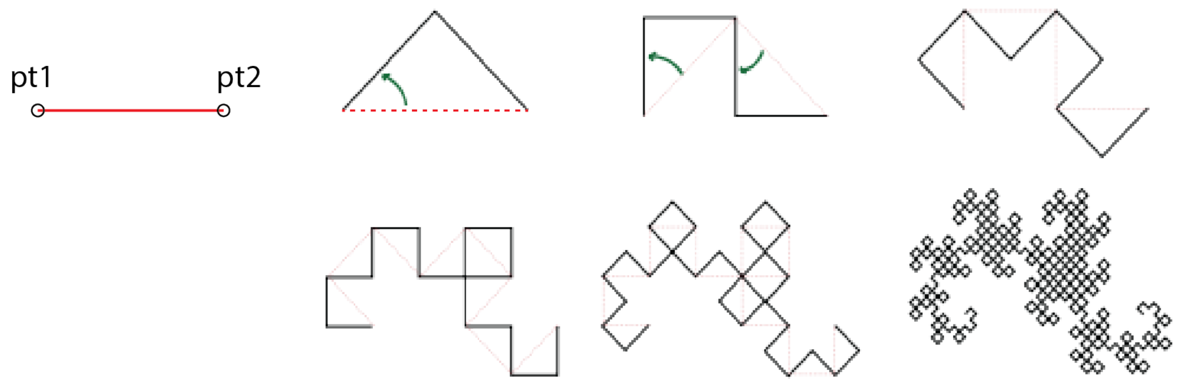
```
image =
"7787888888888888887777887766778888887777753222335788887776222244432379997763255455
45422899774256544455661399773466410036554179764555200004666169768855100001665259757
86510000266526975356520000467627986345653224666438988524666556675279988842566666652
5999888731245443258999888888632333479999988888998778999999998888999999999999999"
width = 18
height = 18 #but the resulting lines will have 35 characters with the interstitial
spaces
print(printImage(image, width, height))

# # % # % % % % % % % % % % % % % %
# # # # % % # # * * # # % % % % % %
# # # # # + - : : : - - + # % % % %
# # # * : : : : = = = - : - # @ @ @
# # * - : + + = + + = + = : : % @ @
# # = : + * + = = = + + * * . - @ @
# # - = * * = .     - * + + = . # @
# * = + + + :         = * * * . * @
# * % % + + .         . * * + : + @
# + # % * + .         : * * + : * @
# + - + * + :         = * # * : # @
% * - = + * + - : : = * * * = - % @
% % + : = * * * + + * * # + : # @ @
% % % = : + * * * * * * + : + @ @ @
% % % # - . : = + = = - : + % @ @ @
% % % % % * - : - - - = # @ @ @ @ @
% % % % % @ @ % # # % @ @ @ @ @ @ @
% % % % @ @ @ @ @ @ @ @ @ @ @ @ @ @

#the empty line above is a result of the print function
```

Example 2: Recursion and vectors

Implement the recursive function **dragon(line, n, isLeft = True)**. It will develop a general version of the Heighway dragon Fractal. In the Heighway Dragon, you start from a base line, then replace each segment by 2 segments with a right angle and with a rotation of 45° alternatively to the left and to the right. DO NOT USE LOOPS.



```
print(dragon([[0, 0], [100, 0]], 1))
>[[[0, 0], [50.0, -50.0]], [[50.0, -50.0], [100, 0]]]

print(dragon([[0, 0], [100, 0]], 2))
>[[[0, 0], [0.0, -50.0]], [[0.0, -50.0], [50.0, -50.0]], [[50.0, -50.0], [50.0,
0.0]], [[50.0, 0.0], [100, 0]]]

print(dragon([[0, 0], [100, 0]], 3))
[[[0, 0], [-25.0, -25.0]], [[-25.0, -25.0], [0.0, -50.0]], [[0.0, -50.0], [25.0, -
25.0]], [[25.0, -25.0], [50.0, -50.0]], [[50.0, -50.0], [75.0, -25.0]], [[75.0, -
25.0], [50.0, 0.0]], [[50.0, 0.0], [75.0, 25.0]], [[75.0, 25.0], [100, 0]]]
```

Prerequisite

None. At least Junior standing. No previous coding or math course is required, however, it is preferable for students to be familiar with at least introductory level college math and affinity with logic.

Policies

All university academic and student policies as set out in http://www.cmu.edu/graduate/policies/ and https://www.cmu.edu/policies/student-and-student-life/index.html apply to this course.

Specifically:

- You are expected to be on time at all lecture and lab sessions.
- Please backup your work in the cloud. We cannot accept hardware failure as a valid excuse.
- You may not copy code without citation. Copying code without citation is plagiarism.
- Late work may result in a reduced grade.
- Email should only be used for crucial queries and concerns. Please direct software related questions to Pedro during office/lab sessions.

Accommodations for students with disabilities

If you have a disability and are registered with the Office of Disability Resources, I encourage you to use their online system to notify me of your accommodations and discuss your needs with me as early in the semester as possible. I will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, I encourage you to contact them at access@andrew.cmu.edu.

Student well-being and support

Carnegie Mellon University is deeply committed to creating a healthy and safe campus community including one that is free from all forms of sexual and relationship violence. To that end, University Health Services, the Office of Community Standards & Integrity, and the Office of Title IX Initiatives have partnered to expand their educational efforts for graduate students in this domain. There is an educational opportunity for all graduate students at Carnegie Mellon that reflects its commitment to sexual assault and relationship violence prevention as well as to your overall safety:

Haven Plus for Graduate Students. For more information follow the link:
https://shib.everfi.net/login/default.aspx?id=CarnegieMellonHavenPlus

Additionally, it is important to take care of yourself and try as best as possible to reduce, preferably avoid, stress. Do your best to maintain a healthy lifestyle by eating well, exercising, getting sufficient sleep and taking some time to relax. All of us benefit from support during times of struggle. This will help you achieve your goals and cope with stress. There are many helpful resources available to all students on campus. Asking for support sooner rather than later is more often helpful.

If you or anyone you know is experiencing academic stress, difficult life events, or feelings like anxiety or depression, we strongly encourage you to seek support. Counseling and Psychological Services (CaPS) is here to help: call 412-268-2922 and visit their website at http://www.cmu.edu/counseling/.

Consider reaching out to a friend, faculty or family member you trust for help getting connected to the support that can help. If you or someone you know is feeling suicidal or in danger of self-harm, call someone immediately, day or night:

CaPS: 412-268-2922

Re:solve Crisis Network: 888-796-8226

If the situation is life threatening, call the police:

On campus: CMU Police: 412-268-2323

Off campus: 911

If you have questions about this or your coursework, please let us know.

Course Schedule*

* Schedule subject to changes

Weeks	Lecture, computer lab and lab challenge	Assignment
1 Introduction and single data types 08/28/18	____ **introduction to course** ____ **single data types** • object in Python: identifier and assignments • core data types / object types (table in *Learning Python* pp.96) • number datatypes x binary representation • number datatypes: `int` and `float` ____ **binary numbers and operations** • `int`s as binaries • binary operators	W01 01 W01 02 CHALLENGE
8/30/18	____ **boolean** • single datatype: `boolean` • operators for single types • checking type: `type` and `isinstance` ____ **control flow: conditionals** • `if, elif, else` statements • conditional expressions ____ **office hours** strategies for solving W1 exercises	W01 03 W01 04
2 Control and passing 09/04/18	____ **functions** • information passing: procedural abstraction of functions (black-box) • structure: argument, body and `return` • scope: the source of problems (local, nonlocal and global variables) ____ **collection: strings** • strings • string methods and general operations ____ **office hours** reviewing exercises in W1	W02 01 W02 02
09/06/18	____ **strings +** • string methods and general operations ____**control flow: loops** • `for` loop * – index, iterator and collection loop • nested loops • `while` loop ____ **office hours** strategies for solving W2 exercises	W02 03 W02 04 CHALLENGE

3 **Lists** 09/11/18	____ **collection: lists** • list methods • indexing and slicing • traversing a list ____ **more on lists** ____ **office hours** reviewing exercises in W2	W03 01 W03 02
09/13/18	____ **tuples and nd-lists** • lists x tuples: immutability x mutability • packing / unpacking / simultaneous assignments • 2d lists • *n*-d lists ____ **office hours** strategies for solving W3 exercises	W03 03 CHALLENGE
4 **Graphics** 09/18/18	QUIZ 1 – Booleans, strings, conditionals, functions, lists ____**control flow: list comprehension** • functional programming • list comprehension • mapping, filtering … zipping ____ **graphics: (pygame)** explain template for W04 01b • draw circles in a grid • draw lines • draw patterns ____ **office hours** reviewing exercises in W3	W04 01
09/20/18	____ **grid layout** Do half of the assignment as part of the class. Leave the other half as exercise ____**randomness** • random functions • monte carlo method ____ **office hours** strategies for solving W4 exercises	W04 02 W04 03 CHALLENGE

5 **Visualizing Recursion** 09/25/18	____ **control flow: recursion** • russian dolls: idea of recursion • recursion vs loop • recursion in maths (fibonacci, list operations, …) • memoization • dynamic programming ____ **office hours** reviewing exercises in W4	W05 01
09/27/18	____ **graphics: (pygame)** • event-based animation **working with animation** ____ **office hours** strategies for solving W5 exercises	W05 02 (is a pygame template for animation) CHALLENGE
6 **Sets, Dictionaries and Functions** 10/02/18	____ **sets and dictionaries** • efficiency in collections: hash function • sets notation, methods and general operations • dictionaries ____ **office hours** reviewing exercises in W5	W06 01 W06 02
10/04/18	QUIZ 2 – Recursion, hash functions etc. ____**functions ++** • optional args, kwargs … • dealing with exceptions and errors • tests/assertions • lambda function • generator functions(?) ____ office **hours** strategies for solving W6 exercises	W06 03 CHALLENGE
7 **OOP** 10/09/18	____ **introduction to object-oriented programming** • modularity, abstraction and encapsulation • self, methods and attributes • interface ____ **office hours** reviewing exercises in W6	W07
10/11/18	____ **object-oriented programming: inheritance** ____ office **hours** strategies for solving W7 exercises	W07
8 10/16/18	**READING PERIOD** (No Course Meetings)	