CSc 372 — Comparative Programming Languages
Spring 2026 (McCann)

https://cs.arizona.edu/classes/cs372/spring26/

Project: Learn a New (to You!) Programming Language

Due Dates:

Part 1:	April 8th, 2026, at the beginning of class
Part 2:	April 20th, 2026, at the beginning of class
Part 3:	May 4th, 2026, at the beginning of class

Overview: This class exposed you to (probably new–to–you) languages from the three major language paradigms that aren't strictly imperative (namely, object–oriented, functional, and logic) to broaden your language horizon, but also to help you to learn the new languages you will encounter in the upcoming decades of your professional lives. This assignment is meant to support that latter goal.

Assignment: This assignment is in three parts. Part 1 is meant to help you get started in a timely fashion, Part 2 ensures that you're making progress learning your language, and Part 3 lets you be creative.

Part 1: You have five weeks total to work on this project, but just nine days to make two key choices: Your team composition and the language you'll study.

1. Team Composition: You may work on this project alone or with one other student in the class. Teams of two will have some additional work to complete (see the various "Teams of Two" sections throughout this handout). Of course, both partners must agree to work together, and both partners will receive the same score on the assignment. Note that the amount of work we're asking of two people is less than double what we're asking from those working alone. Take that into account when making your team/no–team decision.

   Should poor partner chemistry or any other factor(s) force the team to be dissolved by mutual consent before the Part 2 due date, both members are to (a) notify the TAs and (b) continue on the project entirely separately from that point. After the Part 2 due date, permission of the instructor is required to dissolve a team.

2. Language Selection: You may choose any programming language that you wish, subject to these conditions:

   (a) The language is NOT: A machine or assembly language, an OS shell (e.g., bash, ksh, . . .), Java, C, C++, Ruby, Haskell, Prolog, Python, any language very similar to any of those, nor any language that isn't really a general–purpose language (e.g., HTML and SQL), nor any language you already know in more than trivial detail. We will ask you to 'try again' if we feel your choice is unsuitable.

   Examples: Processing is not an acceptable choice because it is basically Java. Similarly, C# and Objective C are also unacceptable, being similar to C++. We can't read minds, so we don't have a good way to know that you already know a language, meaning that the last language restriction (no prior knowledge) is enforced through the honor system.

   (b) The language has a translator (compiler or interpreter) that is accessible to everyone. Note that this doesn't mean it has to be free, just accessible, but we expect you'll opt for free. You're welcome to change to another compiler/interpreter of your language later, but you need to have at least one identified by Part 1's due date.

   See the "Want to Learn More?" section of this handout, below, for some resources that may help you select a language.

(Continued...)

When you have your solo–or–team choice and language decision made, tell the TAs the following info (see the Hand In section for how):

1. Your name (and your teammate's name, if you have a partner).
2. The language you've chosen to study.
3. The URL of the site from which you downloaded the language translator you've identified.
4. Your reason(s) for selecting that language. (Why? Because we're curious!)

Part 2: By the second due date, the following are to be submitted:

1. Language Study: Create a *publicly accessible* document (web page, Google Doc, or the like) containing at minimum the following sections:

   (a) **History and Current Status** — consisting of a brief history of the language (who created it and why, and brief descriptions of major additions to language features over the years), its current version, where to get a translator for it, and anything else you think is interesting about it.

   (b) **Paradigm** — what kind of language is it (OO / Functional / Logic / Other), and how you know.

   (c) **Typing System** — is the language strongly or weakly typed, are type declarations required, can a programmer create new types, and are functions first-class objects?

   (d) **Control Structures** — how can control flow be managed? (That is, what are the language's selection and repetition options?)

   (e) **Semantics** — briefly explain how your language is scoped (static or dynamic?), which kinds of constants are supported, how storage is allocated (which combination of static, stack–dynamic, and/or heap–dynamic?), and how 'garbage' is managed.

   (f) **Desirable Language Characteristics** — in Topic 2, we covered four categories of language characteristics that are generally thought of as 'desirable': (i) Efficiency, (ii) Regularity, (iii) Security/Reliability, and (iv) Extensibility. (We're intentionally skipping readability/writability.) Choose **one** of those four, and discuss features of your language that support (or limit!) it.

   Teams of Two: You are to add all of these to the above:

   (f) **Desirable Language Characteristics** — You must discuss **all four** categories listed in part (e).

   (g) **Support for Data Abstractions** — Which major abstractions are provided, and can a programmer create new ones (and if so, how)?

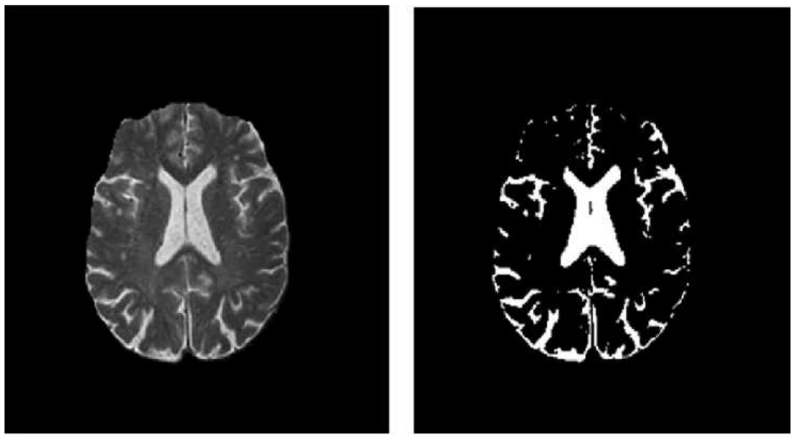
   (h) **Syntax** — Think about the syntax used by your language. What are the syntax choices that appeal to you? Are there any syntactic choices that you'd like to see changed (why, and for which reason)?

   (Why "publicly accessible?" Because we plan to make links to your page or Doc or whatever available to the rest of the class after the Part 2 due date, so that you can learn about other languages from the work of your classmates. Tell us the location of your web site or Google Doc or whatever by including it in the 'external' block comment at the top of your Part 2 Common Program submission (read on!).)

(Continued...)

2. Common Program: To help you learn the answers to some of the above questions, both individuals and teams are to write the following program using the language they chose. Of course, you are to write a complete, well–documented program.

   - Background: A key task in image processing is *segmentation*, which attempts to highlight features within an image. A basic segmentation approach is *thresholding*. The simplest type, *global thresholding*, produces a binary (black and white) image from a grayscale image by partitioning the pixels into two groups based on a single threshold value. This kind of thresholding is simple, but effective on images with clear intensity differences, such as those with foreground shapes on an indistinct background. Here's an example; the left image is the input[1]:

     A good segmentation depends on a good threshold value. The following algorithm will be used to determine that value:

     ```
     Let the initial value be the mean of 10 randomly-selected image pixels
     Repeat
         Partition the image pixels (those smaller than the value, and the rest)
         Compute the mean of the pixels in each partition
         Save the previous value and set the new to be the mean of the two means
     Until the difference of the new and previous values is less than 0.001,
     or 100 iterations have been performed, whichever is reached first
     ```

     Using the final computed threshold value, create a new segmented image in which the corresponding pixels from the input image that are less than the value are black and the rest are white.

     Now all we need is a grayscale image to segment. Fortunately, the PGM (Portable GrayMap) format is a very simple data structure the is ideal for this purpose. It's an ASCII file, stored with the `.pgm` file extension, with content following this structure:

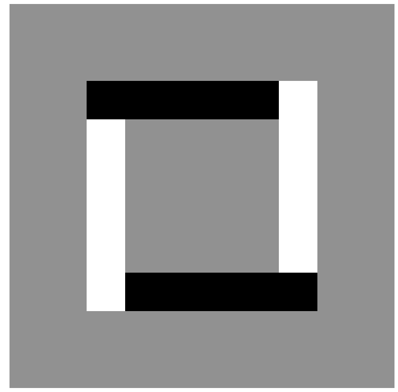
     – Line 0: The characters 'P' and '2'
     – Lines $1 - n$: Comment(s), each line beginning with '#'
     – Line $n + 1$: Two integers (the quantities of columns and rows of pixels) separated by a space
     – Line $n + 2$: The max (brightest) value of a pixel (the min is always 0)
     – Rest of the lines: The values of the pixels, in row–major order, separated by white space characters

     Here is a 10 x 10 example with a rendering. Note that the pixel values need not be arranged in 2D as shown; they could also be stored one pixel per line, all pixels on one line, or anything in–between – the pixel values only need be separated by white–space characters:

(Continued...)

[1]Source: https://www.researchgate.net/figure/Simple-thresholding-on-MRI-a-Sample-Image-b-Threshold-image-T-14-128_fig7_318952747

```
P2
# a square on a gray background
10 10
7
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
4 4 0 0 0 0 0 7 4 4
4 4 7 4 4 4 4 7 4 4
4 4 7 4 4 4 4 7 4 4
4 4 7 4 4 4 4 7 4 4
4 4 7 4 4 4 4 7 4 4
4 4 7 0 0 0 0 0 4 4
4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4
```



(highly magnified!)

Even more fortunately, there's a related Portable BitMap (PBM) format to represent strictly black and white images. The possible pixel values are 0 (white) and 1 (black). PBM is the same as the PGM, except that (a) the first line contains P1 instead of P2, (b) there is no line with the max pixel value, and (c) `.pbm` is the file extension.

- Assignment: Write a program in your language that accepts the pathname of a PGM file on the command line, uses the pixel values from that file to perform global thresholding as described above, and outputs the resulting black and white image as a PBM file with the same name (but different extension).
- Turn In: Submit your program (containing the location of your web site or Google Doc or whatever in the program documentation) on behalf of yourself or your team (just one submission per team, please!) via the turnin program on lectura to the `cs372project` folder.
- Grading: The TAs will ask you to demo this Part 2 Common Program when you demo Part 3's program, and they will be looking at the source code to verify style and documentation.
- Want to Learn More?
  - Descriptions of global thresholding are easy to find. Wikipedia's page is more readable than most.

    https://en.wikipedia.org/wiki/Thresholding_(image_processing)
  - PGM and PBM file formats are part of the Netpbm package. Many image display and manipulation programs (e.g., Gimp) can read and write them.

    https://netpbm.sourceforge.net/

Part 3: After Part 2, you'll have a basic working knowledge of your language. Time for some fun!

1. Creative Program: You are to write a program from scratch in your language that ...

   (a) ... makes good use of the strengths of your language,

   (b) ... is interesting enough to make the TAs think, "That's interesting" while they're grading it, and

   (c) ... is complex enough that, if a technical job interviewer were to ask you, "Tell me about an interesting programming project that you worked on, and what you learned from it," you'd be proud to tell them about this program.

   Yes, the grading of this part will necessarily be subjective. We aren't expecting you to read our minds about what we think is 'interesting.' We're looking for your program to be (a) a well–crafted, well–documented software product, (b) more complex than the Part 2 Common Program, and (c) the result of a 'final–class–project' amount of thought and effort. In short, aim to impress us!

   <u>Teams of two</u>: You are expected to produce programs that are even more interesting and even more pride–inducing than those produced by most of the students working alone.

(Continued...)

2. Program Description: Add to your Part 2 document at least a two paragraph description of your program and what you believe is interesting about it. Please feel free to write a lot more!

   Teams of two: You are required to add visuals to your description, in the form of screenshots of the output or, better yet, a separate video that shows your program in action.

3. Demo: Schedule a live 15–20 minute demo with the TAs. For a class of this size, the TAs can't be expected to install all of the translators and test each program. Instead, you'll demo your programs for them. They will also ask you some questions about your programs or even the language itself. We'll have more information about the demos and how to sign up for a demo time well before Part 3 is due.

Data: Please see the program descriptions in Parts 2 and 3, above.

Output: Again, please see the program descriptions in Parts 2 and 3, above.

Hand In:

- *Part 1:* Everyone is to submit team and language info to the TAs via a Google Sheet to which the TAs will post a link soon.
- *Part 2:* You (or your partner) are required to submit your Common Program source code file(s), and a text file named README containing the location of your publicly accessible document, using the `turnin` facility on lectura. The submission folder is `cs372project`. Submit all files as-is, *without* packaging them into `.zip`, `.jar`, `.tar`, etc., files.
- *Part 3:* Submit your source code file(s) via turnin to the same `cs372project` submission folder.

Late days: Late days can be used on this assignment, but only on the third due date. How many a team has to use is determined as follows: Team members total their remaining late days, and divide by the number of members in the team (integer division), producing the number of late days the team has available, **to a max of two days late**. (Why a max of two? The TAs need to get grading done soon after the due date, you need time to study for your final exams, and the department has a policy that nothing can be due after the last day of classes.)

For example, a team whose two members have 2 and 3 late days remaining have $\lfloor \frac{2+3}{2} \rfloor = 2$ late days to use, if needed, and if demo slots are still available before the end of classes.

Want to Learn More?

- There are many things that could be considered when trying to select a language to study, including popularity (commonly–used languages have more resources available), paradigm, and personal preferences. Don't pick a language "just because;" look at several so that you (and your partner, if you choose to have one) make a sound choice.
- If you're looking for a language within a particular paradigm or category, this Wikipedia page will help:

  `https://en.wikipedia.org/wiki/List_of_programming_languages_by_type`
- There are many ways to determine a language's popularity. Here are two frequently–cited language popularity rankings that might give you some ideas for a language to study:
  - `Tiobe Index` — A monthly list of language popularity: `https://www.tiobe.com/tiobe-index/`
  - `PYPL Index` — Languages ranked by tutorial searches: `http://pypl.github.io/PYPL.html`

Additional Hints, Reminders, and Requirements:

- **Start Early!** You have a little more than a week to make your partner/no–partner and language decisions. Use that time to make good selections! The sooner you do, the sooner you can start on Part 2 (and then Part 3).
- What a partner but need help finding one? Try the "Search for Teammates!" pinned thread in Piazza!