Studio: Developers as Decision Makers

A modified version of the assignment created by Evan Peck at Bucknell University

In this lab, we will create algorithms that determine housing for SFSU students.

When we say the word 'algorithm', we tend to ascribe agency to the computer. It is *deciding* things for us. But the reality is that there is no magic. There are software developers like you and me who design and create sets of rules that the computer carries out for us.

These algorithms are all around us, and they are *constantly* making decisions.

The decisions we make in code impact the lives of real people. For example, the **Silicon Valley Triage Tool** is an *algorithm* that identifies homeless people for whom giving them housing would cost the public less than keeping them homeless. So even as we learn the simple structures of code, we need to think about *how we can make **good** decisions?*

We are going to explore this idea in a more familiar context to you - **university housing allocation**. At SFSU, on-campus housing is in high demand but very limited. This has resulted in the university relying on a first-come first-serve policy: https://housing.sfsu.edu/apply. You might not think of it as one, but this method is **an algorithm.**

In this project, you will have the opportunity to design your *own* algorithm. We're also going to begin dabbling with a human-centered design process to make sure that **the decisions we make are never untethered from the people we impact.**

In this project, you'll practice...

- translating English rule-sets into code,
- soliciting text input from people,
- ***applying conditionals (if, else if, else) to make decisions with a program*,**
- using an accumulation variable to keep track of information in a program, and
- integrating basic human-centered design processes into your programs.

First of all: Is the first-come first-serve fair? Can we do better?

You have to start by thinking if the existing way of doing things is fair. Is the first-come first-serve approach to housing allocation fair? Notice how you have to think about what “fairness” involves.

Who may be at greater disadvantage with the current system?

Do you think there is a better way to allocate housing such that it is fairer?

The Decision-Maker: Who gets to choose their housing first?

Your job is to build an algorithm that helps determine the order in which students will get to select their housing. To simplify things, we’re going to use a *point system.*

- Students are awarded a number of points based on a variety of factors.
- Students with the most amount of points get first choice at housing.

This real approach is used by many universities. For example, consider the following *real point system* used by another college in the United States:

- *Current Freshman:* 1 point
- *Current Sophomore:* 2 points
- *Current Junior:* 3 points
- *Current Senior:* 4 points
- *23+ Years of Age:* 1 point
- *Full-Time, Off-Campus Program Credit (e.g. student teaching):* 1 point
- *Academic Probation:* -1 point
- *Possible Academic Suspension:* -2 points
- *On Disciplinary Probation at Any Point during the Academic Year:* -3 points

So, a junior (+3 points) who is 23 years old (+1 point) would have priority over a senior (+4 points) who is on academic probation (-1 point).

Overall goal: Create a program that assigns points to students in order to prioritize them in housing selection.

But wait! Don’t start yet. First…

Before you Code, Assess the Needs of Your Users

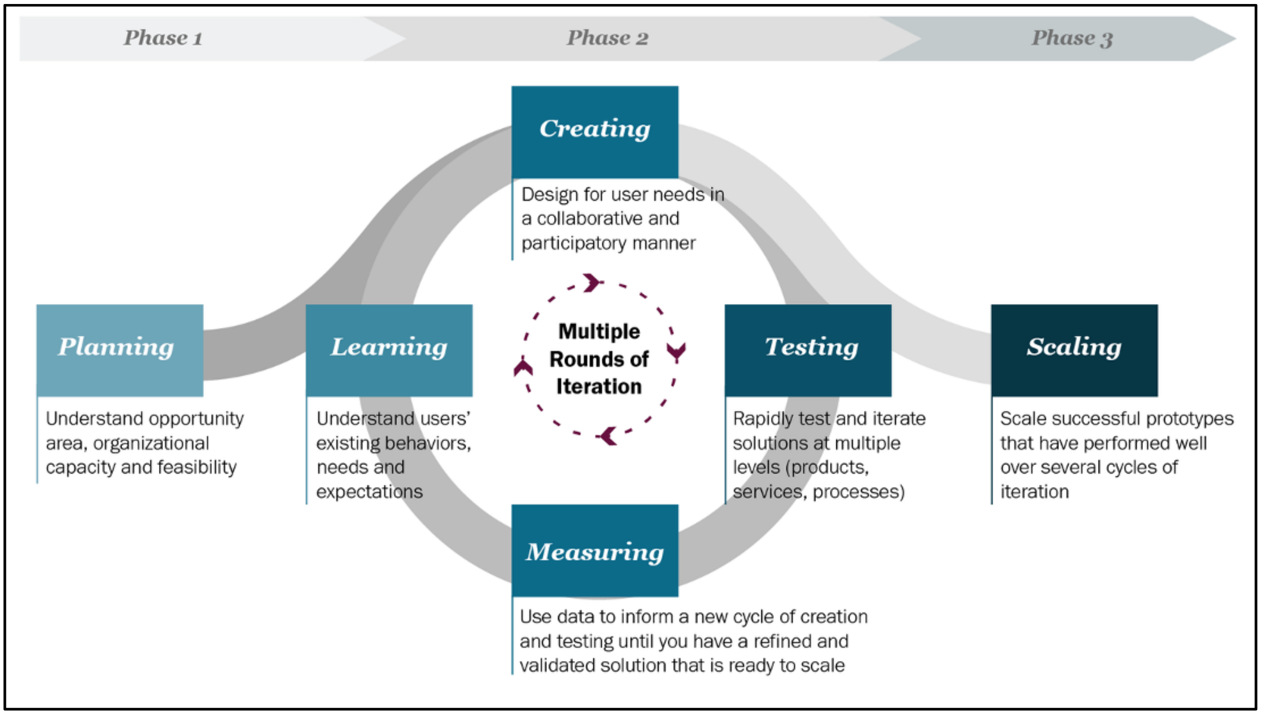
While the list above was *one* college's take, there are **many** more potential aspects to consider if you want to create a **fair** algorithm that takes into account the diverse needs of students.

You should not create a program that serves people without talking to people. Talk to other students in your class. Ask them about their needs. What other unique factors may be important in deciding who should choose housing first?

Group Work: Form a group of 3—4 students. Discuss the various factors to consider when deciding housing allocation. Create a bullet-point list of factors that came up in conversation. Give each one a numeric priority (-1 to 5 points).

- Factor 1:
- Factor 2:
- Factor 3:
- Factor 4:
- Factor 5:
- Factor 6:
- ....

Notice that this is the learning phase of the **human-centered design process** shown in the following figure.


Phase 1
Phase 2
Phase 3
Creating
Design for user needs in a collaborative and participatory manner
Planning
Understand opportunity area, organizational capacity and feasibility
Learning
Understand users' existing behaviors, needs and expectations
Multiple Rounds of Iteration
Testing
Rapidly test and iterate solutions at multiple levels (products, services, processes)
Scaling
Scale successful prototypes that have performed well over several cycles of iteration
Measuring
Use data to inform a new cycle of creation and testing until you have a refined and validated solution that is ready to scale

Talking to users, prototyping ideas, and then testing them with users, is part of what is known as the **human-centered design process**... a key process for developing useful and usable programs

Assessing the Needs of Our Users

So far, you have discussed the factors amongst yourself. It is not enough; other people may think differently about what is fair.

Using the list that your prepared in your group, interview other people. Try to interview people outside the CSC 101 class. Ask people what factors are important to them. Show the factors that you had identified and ask them of their priority.

Individual Work:

Results of the Interviews and Discussions: Factors that Matter

I interviewed _____ SFSU students about their personal factors.

(EDIT THE FOLLOWING LIST WITH OBSERVATIONS FROM YOUR INTERVIEWS. You can add or delete factors. Provide a number with the factors to indicate which factors that you think are the most important. For example, if you think class year is the most important, place a “1” by that factor. If you think two or more factors are equally important, place the same number by those factors. **)**

- Class year
    - Students closer to graduation should have preference
- Age
    - Older students should have preference
- Campus job responsibilities
    - Students who need to work part time should be able to choose places close to that work.
- Disability issues
    - Students with disabilities should be able to choose housing that accommodates those issues.
- Mental health issues

Design and Plan Your Point-Assigning Algorithm

Now it is time to translate our student needs into a concrete algorithm. Be careful and limit yourself to the most important factors. *You have a limited amount of time in this class!*

Our algorithm will:

1. Ask students questions (like *What class year are you?*).

2. Assign points based on their answers (like *4 points for senior*).
3. Accumulate their total points across all answers (like *You have 23 housing points*).

~~**Constraints:** Since we are exercising our ability to write code in addition to solving problems, we are going to put a couple of constraints to make sure you get practice with the different ways in which conditionals can be used.~~

- ~~You must have at least 1 question that *only* appears if the previous question is answered a specific way.~~
    - ~~*For example:* if someone says they are a 4th year student, you may ask the question "Are you about to graduate?" If they say yes, they receive no points. *ONLY* a 4th year student would receive this question.~~
    - ~~***Hint:*** Use nested if statements.~~

Your goal: Create a bullet-point list that describes the factors you are considering, and how you are mapping those factors to point values (positive or negative).

Our Algorithm in English

(EDIT THE FOLLOWING LIST TO REFLECT *YOUR* HOUSING PRIORITIES. Include at least one nested if and one equation.**)**

For our algorithm, we are going to prioritize class year and age. Remember to include a nested conditional and an equation.

- Question 1: *What year are you? (1, 2, 3, 4):*
    - 1: +1 pts
    - 2: +2 pts
    - 3: +3 pts
    - 4: +4 pts
- Question 2: *How old are you?*
    - if greater than 22, then + 1pt
    - if 22 or less, then no points

Before you Code, Make Sure it Works: User Testing

How will you know if your program serves people well? At a minimum, you should have tests considering different groups of people who may be affected by your program.

- Were the results what you expect?
- Did you discover any cases which you haven't accounted for previously?

Your goal: Write **at least** 3-5 hypothetical *test* cases for your program.

Test Cases

(EDIT THE FOLLOWING LIST TO REFLECT *YOUR* PROPOSED POINT VALUES. Add at least 2 test cases of your own.**)**

- A 25 year old senior who is on academic probation should output 4 points
- A 22 year old junior who is student teaching should output 4 points
- A 20 year old sophomore on disciplinary probation should output -1 points

---

Write Code that Automates your Decision-Making Process

Now it's time to translate your algorithm into code.

Your goal: Implement *the algorithm you designed* in IntelliJ.

- Step 1: Create a new class called HousingDecisionMaker

- Step 2: Implement your algorithm inside the main method

- Step 3: Is it correct? You should check your code with the test cases you outlined above. Input the values that you have identified in the test cases above and evaluate if your program works well. **You must ensure that your code is correct.**

- Step 4: Submit a zip file with the Java file and this completed document.

During your creation, keep a couple of things in mind:

- Use comments to describe what was happening in the program.
- Choose variable names that clearly describe that data that they hold.
- Use spacing to group similar code.

Your Code Works... but is it fair? (Individual)

You should never deploy real code without checking your assumptions.

Your test cases tested your *technical* assumptions, but not *your social* assumptions.

1. Find classmates either inside or outside of the lab.
2. Run your code with them
3. Get feedback on what worked and what didn't?

In particular, you should reflect on...

1. **Which students are most likely to benefit from your algorithm?**
2. **Which students are most likely to be forgotten by your algorithm?**

Finally add your reflections to this document.

Your Reflection: Tradeoffs

(REPLACE THE FOLLOWING REFLECTION WITH YOUR REFLECTIONS.)

While the ordering of class years makes sense, my algorithm should probably do more service to the needs of non-traditional students. For example, you could be a 31-year-old sophomore... but have no more points than any junior. Of course, that person should be able to live off campus!

In the future, I would revise this to weigh student age heavier. I would also rethink the point allotment to classes. 3rd years should have priority over 1st years... but should they really have 3x as much? (This is what the points suggest).

If you're interested...

- **Optional reading:** *What Happens When An Algorithm Cuts Your Health Care By Colin Lecher*
- *We created poverty. Algorithms won't make that go away. By Virginia Eubanks*