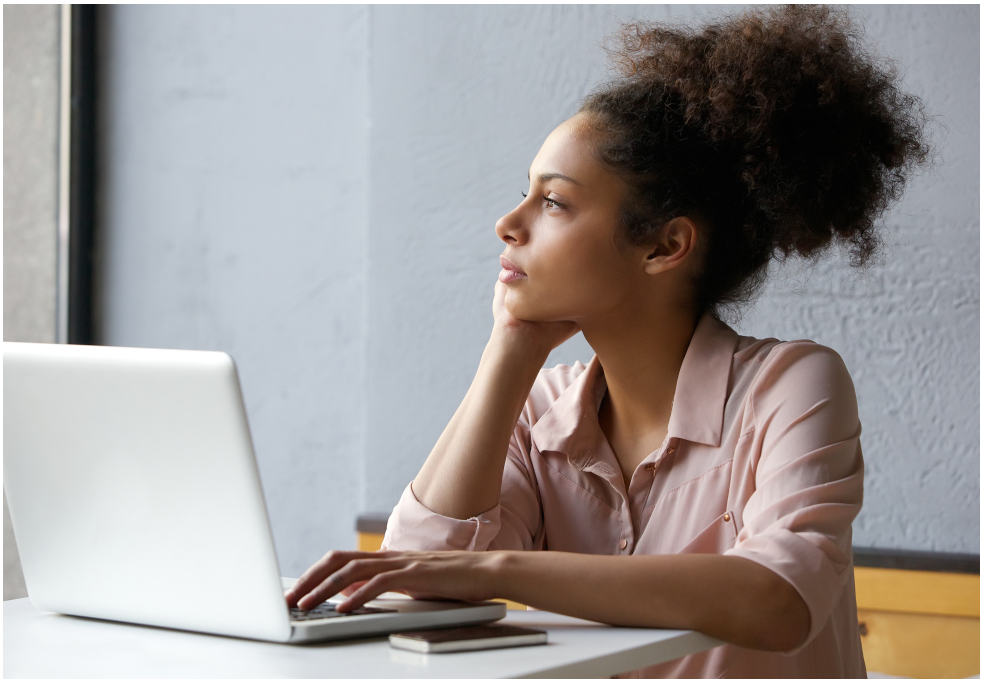
Identifying a Problem to Solve

by Sophia

WHAT'S COVERED

In this lesson, we will learn how to get started in defining a problem to solve with coding. Specifically, this lesson covers:

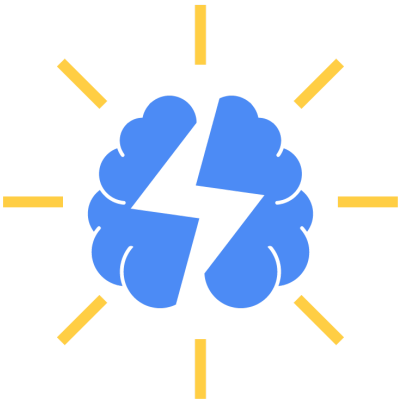
1. Getting Started With an Idea



For this lesson, we want to work towards defining a problem or a project that we can build a program around. Throughout this class, we've been given code to work through. Although we have gathered some skills together, it's a good idea to work on some projects for yourself. You should challenge yourself to exercise

everything that you've learned and create a project. This will not only help develop your coding ability but also the skills that you need to break down projects into workable solutions. This will also allow you to have improved conversations with others about what you've built.

BRAINSTORM



Getting started can be a challenge, as a good program should perform a task that makes life easier for an individual. This could be a task for yourself, someone you know, or for a larger audience. If you think about the things that you do on a regular basis, is there some way that you could automate a part of that with a program? It's worth taking the time to write down all of your ideas. Even if an idea seems ridiculous or impossible to do, you could think about what ways it could be improved upon. Another way to brainstorm ideas is to look at other programs that you may use. Are there things that they do that could be improved upon, or do they have areas that are lacking? Could they be done more effectively?

THINK ABOUT IT

It's best to try and think of a simple project as it's easier to start simple and then get complex with added features. If you start with an overly complex project, you may not have a working project at the end. In preparation for this type of development, you'll be working on documenting the features and functionality that you intend to achieve with the project. The more details about what the program should do that we can provide early on, the easier it will be to start to create the underlying code that helps to support the program behind the scenes.

BIG IDEA

Let's think back to our first unit where we defined what a program consists of. We mentioned that we needed to have input, processing, and output for a program. The input is the elements that we are taking in from the user or other means; other "means" would be a file or files for data the program will pull from. The processing includes all of the calculations based on the input. The output is the result of the processing and is presented to the user.

Generally, as a programmer or a developer, you would be providing a service to an end-user rather than creating a program purely for yourself. One of the first steps, in this case, is to understand what the users are looking for and what they want the program to accomplish. For example, a DJ (Disc Jockey) may want a program that allows them to select all of the songs to be played in a specific order and output them into a list. A photographer may want to take a set of photos and identify the names of the individuals in each photo to send those individuals a list. Since we would be providing this service to these users, we need to get into the mindset of those individuals and consider what they expect.

THINK ABOUT IT

When we plan a program, we generally think about what the output is first. This is what the end-user is looking for. Once we know what the result should be, we can then go into planning the input and processing steps that we need to do to get to that end result.

2. Our Example Problem to Solve

Let's first start by identifying a potential problem. We have a friend that is going to Las Vegas and would like to know what the odds are in order to win a game of dice called casino craps. This is a game that involves some strategy, skill, and action. They would like some information about the odds after a certain number of games and would like to know how often they win or lose, how many times they had to roll for each, and what their winning percentage was.

THINK ABOUT IT

This doesn't sound like it's too bad of a request but there are always additional questions that we have to pose as this request isn't fully defined.

- Would they be interested in having a single sample game to see how lucky they would be?
- How would the game of casino craps be played?
- What would the rules be for the game?
- If they played multiple games, what information would they like to track?
- What information would they like to store?
- What kind of output would they want?
- Where would they like the information tracked?

With all of these questions, a developer cannot make the decisions of what should happen as they affect the rules of the game, the program, and the client's (or in this case our friend's) requests. If we don't know the answers to certain questions or steps, we can't simply make up the answer. Rather, it is important that we pose those questions to our friend. Here are some typical responses to our questions.

- Would they be interested in having a single sample game to see how lucky they would be?
  - Yes, I would like to have a quick way to see if I randomly played a game while going through the casino.
- How would the game of casino craps be played?
  - With the game of craps, the game is made with players putting bets on the first roll of the sum of two dice, also known as the come-out roll. In this round, the players must decide whether the dice will land on 7 or 11 (pass bet) or 2, 3, or 12 (don't pass bet). The round ends if the total dice value is 7 or 11 (known as a natural), or 2, 3, or 12 (craps). If the total of the dice value rolled is a 4, 5, 6, 8, 9, or 10, then that number becomes the point and starts the next stage of the game. The shooter then rolls the dice again and repeats until the shooter rolls a 7 or the point number, which ends the craps game. If a 7 is rolled, they "sevens out" and lose. If they rolled the initial point score, they win the round.

- What would the rules be for the game?
  - Player rolls two dice the first time.
    - If 2, 3, or 12 is rolled on the first time, the player loses.
    - If 7 or 11 is rolled on the first time, the player wins.
    - If any other number is rolled, that number becomes the point value or initial sum.
  - If the player has not won or lost, roll again.
    - If the sum of the roll of the two dice is equal to 7, the player loses.
    - If the sum of the roll of the two dice is equal to the initial sum, the player wins.
    - If the sum of the two dice is anything else, reroll again.
- If they played multiple games, what information would they need to track?
  - Ideally, I would like to be able to enter a number of games to play. In part, that would be based on my budget for the day. In knowing the number of games to play, I would like to know how many rolls on average it took to win and how many it took to lose. In addition, I'd like to know what my winning percentage would be.
- What information would they like to store?
  - On a daily basis for each set of games, I'd like the statistics about the games I played stored in a file. The file can just keep track of each time.
- What kind of output would they want?
  - I would like to see on screen the number of times the game was played, how many times out of those games I won, how many times I lost, the average number of rolls per win, the average number of rolls per loss, and the winning percentage.
- Where would they like the information tracked?
  - In this case, I would like the information being tracked to be placed in a file for every time I played the game.

As part of logging these questions and answers along with other details, it can help to place these details into documentation. The documentation may include some of the following:

- Forms (paper forms, online forms, surveys, etc., to get data for input)
- Reports (how output should look, raw data items, items that need to be calculated)
- Files (data files, .csv files, images like logos)
- Sample results (example runs of what the program is expected to do)
- Mockups (design of the application, input fields, workflow of the program)
- Existing tools (anything that the client/user currently uses to help workaround the program that they need)
- Any other item that comes from the client/user

Throughout this process, you'll be following the same type of format where you'll be documenting your progress for the touchstone using the template to help as a guide.

At this point, start with your formal idea that you plan to go through the process of designing and developing. If you don't have an idea, you can always ask those around you if they have a recommendation or a need for a

program. Once you have your idea, pose (and document) as many of the questions that you can think of to figure out how the program needs to work and include the answers to those questions. Use the examples given to help as a guide with your questions but each idea/problem will have different questions that you may need to ask.

---

3. Adding Our First Journal Entry

When it comes to these initial steps, it's important to present the questions and answers that would help us define a solution. Even if we have the right questions to ask, imagine if you had the following answers to our prior questions:

- Would they be interested in having a single sample game to see how lucky they would be?
  - Not sure.
- How would the game of casino craps be played?
  - Like the game found in the casino.
- What would the rules be for the game?
  - Roll dice and see who wins.
- If they played multiple games, what information would they need to track?
  - All information.
- What information would they like to store?
  - All information.
- What kind of output would they want?
  - The results of the game.
- Where would they like the information tracked?
  - The date.

In looking at the answers above, there's not much information that can be derived to determine how the program would actually work. If we do not get adequate responses, we should continue to ask questions until we get all of the necessary information that we need to start creating the algorithms for the pseudocode on the project. The more details that we can provide at this stage, the easier it will be to develop the program.

At this point, we are ready to add the first journal entry for the touchstone. Based on the questions we asked and the answers our friend provided, we have information that can address the requirements of our task, which was to state a problem that we are looking to solve.

What we would not want to add for our journal entry is the inadequate answers to the questions that we just witnessed. For example, this would be a bad entry.

Bad Example of Journal Entry for Part 1

EXAMPLE

“The problem to solve is to create a program that plays casino craps like one found in a casino. My input data would be to roll dice and see who wins. Our friend would like to track all game information and then store that information. Our output would be the results of the game.”

A better entry would break down the requirements of the entry and add as much detail as possible to truly define what the problem is and the expected solution based on answers to our questions, like this:

Good Example of Journal Entry for Part 1

EXAMPLE
“Casino craps is a dice game in which players bet on the outcome of the roll of a pair of dice. The problem to solve is to help a user better understand the odds of winning and losing at casino craps at an actual casino. I will create a simulation of the craps game to solve this problem. When executed, the program will first provide a sample game as an example for the user. Second, the program will ask the user to input the number of games to play. The program will play the requested number of games. After the last game is played, the program will output (display) the results of the games. The program will store a record of the simulations in an external log file. By playing the game multiple times and reviewing the statistics, users will get a better understanding of their chances of winning.”

If we preview the Example Python Journal Submission document, we see this was added as the entry to Part 1.

4. Guided Brainstorming

There are a lot of different, complex potential problems that you may have but it is important to try and limit the complexity of the program to something that you can create as your first application. This could be a program that stores email addresses for a newsletter. It could be a recipe program that stores a list of recipes, ingredients, and directions. It could be a program that does conversions of different measurements between each other. The possibilities are endless but what’s important is to try and fully define the initial problem, ask the right questions and provide the answers. Remember that the foundation of any program surrounds the input.

THINK ABOUT IT

For example, if it’s a recipe program, you should ask the question of what a program like that would do. There are so many different ways that you can have a recipe program work. For example, you may have existing recipes that are just being displayed from a file. You could even extend it further to have a menu that would prompt if the user would like to add a recipe or list the recipe based on a name that has been entered. You could even extend it further to have the program display images of the recipes as well. The options and selections are endless, but you want to try and limit how far you go with these programs.

Using the casino craps game that we’ve defined, we could have two different players interactively play the game and take in bets rather than just showing the wins and losses. Having an interactive game would be a lot more fun, especially if some animation of the dice being rolled would also be shown. However, having a more interactive program could also make it a lot more complex as well as take longer to fully test out. If we’ve done a good job in defining our program, it will make it easier to test and even expand on later.

TRY IT

Directions: Now would be a good time for you to define your problem. Think about a program you would like to build to solve that problem. Ask the questions to gather the requirements for this task. Review the example of a good entry for Part 1 in the Example Python Journal Submission document and add your journal entry for Part 1 to your Python Journal.

SUMMARY

In this lesson, we discussed how to **get started with an idea** for a program to build. If the program is for us or for someone else, there are questions we need to ask and answers we need to collect to make sure we are preparing for the correct input, processing, and output in the program. We then discussed an **example problem that we want to solve** for and will be using throughout Unit 4, which is a casino craps simulation. Then, we prepared for **adding our first journal entry** by reviewing a good entry for Part 1 of the journal and noted that it should be as comprehensive as possible to detail out the input, what the program will solve for, and the expected output. Finally, in the **Guided Brainstorming** section, we learned some additional opportunities for ideas while thinking about our own program to build.

Best of luck in your learning!