Identifying a Problem to Solve

by Sophia

WHAT'S COVERED

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

1. Getting Started With an Idea

For this lesson, you will work on defining a problem or a project that you can build a program around. Throughout this class, you have been given code to work through. Although you acquired some skills using this approach, it’s a good idea to work on some code for a program that you devise yourself. You should challenge yourself to exercise as much as you can what you’ve learned and create a project that reflects it. This will not only help develop your coding ability but also the skills that you need to break down projects into pieces that lead to 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. A good program should perform a task that makes life easier in some way. 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 on. 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 basic project with limited scope since it's easier to start simple and then add features one at a time. 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, provided early on, the easier it will be to start to write the underlying code.

BIG IDEA

Think back to the first unit where we defined what a program consists of. You noted that most programs need to have input, processing, and output to be useful. The input is the information that a program needs from the user or other sources, such as a file the program will pull data from. The processing includes all of the calculations or transformations performed on the input. The output is the result of the processing and may be presented to the user in the console or written to a file.

Generally, as a programmer or developer, you would be providing a piece of software to an end user, though in some cases, you may be the end user of the program that you create. One of the first steps 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 the selection of songs to be played in a specific order and produce a play list. A photographer may want to take a set of photos and keep track of the names of the individuals in each photo. It is important to get into the mindset of the customers and consider what they expect.

THINK ABOUT IT

When planning a program, you generally first think about what the result of running the program should be (usually what output the program should produce). This is what the end user is looking for. Once you know what the result should be, you can identify what input is needed and start planning what processing steps are needed to get to that end result.

2. An Example Problem to Solve

Let's start by identifying a potential problem to solve. Imagine a person who is going to Las Vegas and would like to learn more about what the odds are of winning a dice game like craps. This is a game that involves some strategy, skill and luck. This person would like to get some practice to have a better sense for the odds and what it's like to play the game.

THINK ABOUT IT

This doesn't sound like a bad place to start, but there are additional questions that need to be asked, since this request isn't a fully defined problem.

- Is the person interested in a program that plays a single sample game or multiple games?
- What are the rules for the game?
- What information should be tracked?
- What kind of output should be produced?

Typically, it is important to ask questions and get the customer to tell the story of what the program should do and how it should work.

EXAMPLE

Here are some typical questions and responses.

Question	Answer
Would they be interested in having a single sample game to see how the program works before playing multiple games?	Yes, I would like to see how the game works before deciding how many games to play.
	The game is played by rolling two standard six-sided dice. For each roll, the pips on the top sides of the

How is craps played?	dice are added up to get a total for the roll. If the first roll produces a sum of 7 or 11, the player wins the game on the first roll. If the sum of the dice on the first roll is 2, 3, or 12 (these values are called “craps”), the player loses the game on the first roll. If the total of the dice on the first roll is any of the remaining possible values (4, 5, 6, 8, 9, or 10), that value is called the “point” and the player continues rolling the dice until the point is rolled again or the roll produces a 7. If the player rolls the point again before rolling a 7, the player wins the game (no matter the number of rolls), but if the player rolls a 7, the game is lost.
What are the rules of 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 multiple games are played, what information would need to be tracked?	Ideally, I would like to be able to enter a number of games to play. In part, that would be based on my gambling 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 was at the end of the day.
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 a game is played.
What kind of output would they want?	I would like to see on screen the number of times the game was played, the total number of rolls, how many times out of those games I won, how many times I lost, and the percentage of wins.
Where would they like the information tracked?	In this case, I would like the statistics about the games 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.

↪ EXAMPLE
The documentation may include some of the following:

- Forms (paper forms, online forms, surveys, etc., to get data for input)
- Reports (how the output should look like, raw data items, items that need to be calculated)
- Files (data files, .csv files, etc.)
- 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.

BRAINSTORM

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. The First Journal Entry

For the initial steps, it's important to present the questions and answers that would help us eventually define a solution.

↪ EXAMPLE
Even if we have the right questions to ask, imagine if you had the following answers to our prior questions:

Question	Answer
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?	In a folder.

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

THINK ABOUT IT

At this point, we are ready to submit 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 submit for our journal entry is the inadequate answers to our questions that we just witnessed. For example, this would be a bad entry.

3a. 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."

REFLECT

This entry does not adequately define the problem or present questions and answers that we developed during our brainstorming session to help us refine the problem.

3b. 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."

REFLECT

A better entry would break down the requirements of the entry and add as much detail as possible to truly define the problem and expected solution based on answers to our questions. If you preview the Example Java Journal Submission document, you would 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 is based on the input.

THINK ABOUT IT

If you're trying to create 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. 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 will want to try and limit how far you go with this program to ensure that you will complete it.

Using the casino craps game that you 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 and bounding 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 Java Journal Submission document and add your journal entry for Part 1 to your Java Journal.

REFLECT

Craps and other casino games may not be your forte, but you don't have to be an expert on them to really learn from the demonstration program given. In fact, we experience a significant amount of learning from games as children, and it's great to carry this into adulthood. Focus on the process regardless of the context. At this stage, we are identifying and defining a problem. The old saying, "knowing that you have a problem is the first step to solving it," is somewhat applicable here. You have to know, and thus understand, what the problem is to solve, or for us, code it. The questions and answers about the craps game define what we need to do so that we truly understand what we need to code and bound the problem so that we can reach our goal in a reasonable amount of time.

SUMMARY

In this lesson, you explored how to **get started with an idea** for a program to build. If the program is for ourselves 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 that we need to do in the program. You considered an **example problem that you might want to solve**. A casino craps simulation will be used throughout Unit 4 as an example program. Moving on to **the first journal entry**, you first saw a **bad example of journal entry for Part 1**, followed by a **good example of journal entry for Part 1**, noting that it should be as comprehensive as possible to define and characterize the input, what the program will solve for, and expected output. Finally, you explored some **guided brainstorming** to examine some additional opportunities for ideas while thinking about your own program to build.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source py4e.com/html3/