LEVEL 1

Tix Finder

👇 *Click the green shaded bar below to reveal user stories for this app*

👧🏽👧🏽👦 Analyzing User Stories

When developing software, creators often rely on ***user stories***- descriptions and needs from the perspective of someone using the app-in-development. They often try to take in as many points of view as possible, so that the final product appeals to a wide variety of users. Throughout these app development levels, there will be several user stories that will be shared and which will be translated into tasks for you to accomplish. Here is an overarching user story for this app:

I love going to concerts and seeing my favorite bands, but ticket prices are so unpredictable. I would love an app that gives me a good estimate of how much I might pay to attend the show.

✅ Do This

1. The app to the left is a *completed* version of the one that you will develop in this lesson. Play around with the app to see how it works. Try entering different values for `artist`, which `month` you want to attend the concert, and `age` to see how that changes the output on the screen when you click the "Find My Ticket!" button.

With your partner, discuss the following:

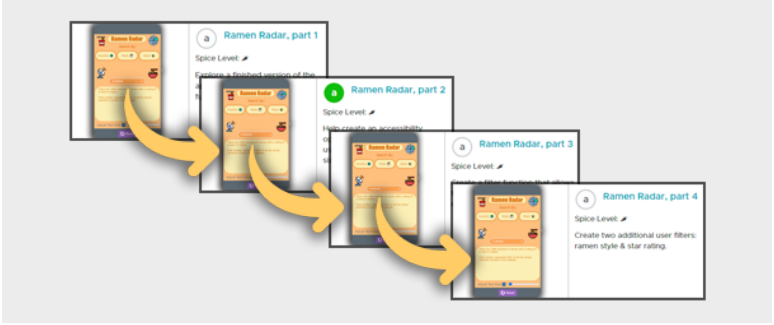
❓ What is the purpose of the app?

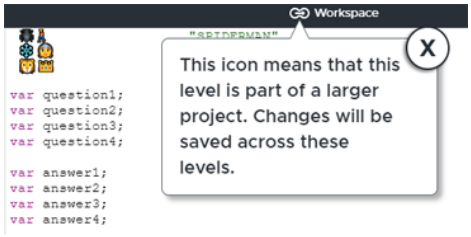
❓ Describe the functionality of the app. How do these design decisions fit the purpose? What are some inputs and outputs that aid in achieving this functionality?

👇 *Click the shaded accordion bar below to reveal helpful tips about how to navigate and complete these practice levels*

🔎 How do these Practice levels work?

There are seven practice levels that should be completed in order to complete the app. They are intended to be followed in order. Move from *Part 1* all the way to *Part 7*. Therefore, as you move from level to level, make sure that you select the Concert Tix Finder selection on the choice screen.





As you move through each of the levels, the code that you create, alter, or remove will travel with you!

That means that as you write code and implement cool features in your app, you will be building on the work that you have completed previously.

You will see a pop-up on the levels that will alert you about this shared code the first time that you visit the level.

LEVEL 2

Assigning Values to Variables From User Input

👩🏾👧🏽👦 Analyzing User Stories

I need to be able to use different inputs on the app screen to provide information for the app.
The milestone that you are accomplishing on this level is directly related to the User Story seen here. By completing the tasks below, we will have accomplished the milestone and provided a new feature for the users!

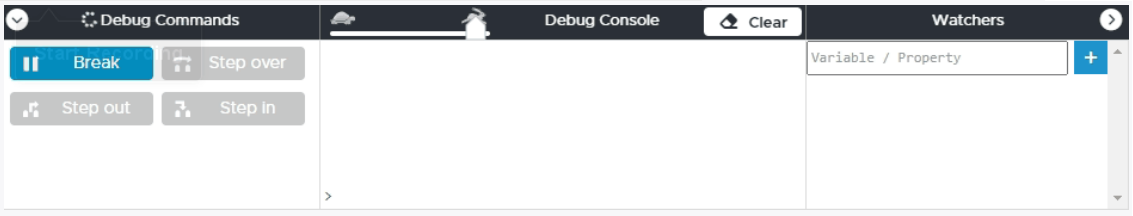
✅ Do This - Milestone 1 / 6

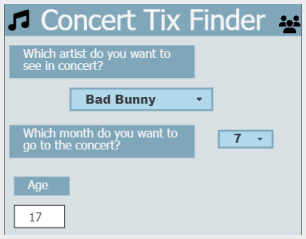
Tasks:

1. Look at all the elements on the app screen to get a sense of where the user inputs information.
2. Look through lines 1-7 to see the *all* the variables that will be used in this program and think about which ones will be directly updated by user input.
3. Get the needed user input values and assign them to the appropriate variables. 👉 Show me where 👈
    - Use `x =` and `getNumber()` to assign the user input for both `concertMonth` and `age`.
        - *Example*: `concertMonth = getNumber("monthDropdown");`
    - Use `x =` and `getText()` to assign the user input for `artist`.
        - *Example*: `artist = getText("artistDropdown");`

🎯 Test Your Code

Note: You can click the tests below to expand them and see more specific instructions

- Add the variables you assigned values to with your code to the "Watcher" area at the bottom of your workspace. You can expand this area by clicking `Show Debug Commands`
- 

- Click the "Find My Ticket!" button after using the dropdowns and text box on the app screen to assign values to the variables and those values should be reflected in the Watcher area.
    - For example, if you select these values:
- 

    - Your watcher area should look like this:
- 

LEVEL 3

Using Boolean Expressions

👧🏽👧🏽👦 Analyzing User Stories

I know that Taylor Swift charges more for a ticket than the other artists. Can the base price be adjusted to account for the difference in price?

✅ Do This - Milestone 2 / 6

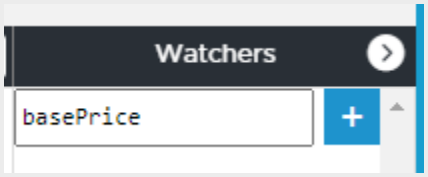
Note: You can click the instructions below to expand them and see more specific instructions

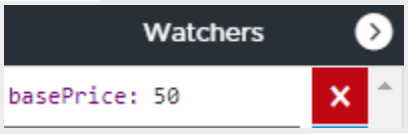
Tasks:

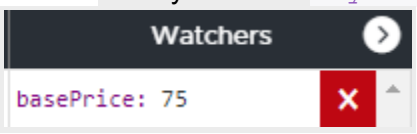
1. If the user selects **Taylor Swift** as the artist, set the `basePrice` to `75`. For all other artists, the `basePrice` begins at `50`.
    - Use == to replace the `true` value in the if-statement like this `artist == "Taylor Swift"` 👉 Show me where 👈
    - Assign the value of `75` to `basePrice` when the above boolean expression is true
    - In the else section, assign the value of `50` to `basePrice`

🎯 Test Your Code

Note: You can click the tests below to expand them and see more specific instructions

- Add the `basePrice` variable into the Watcher Pane at the bottom-right of the screen
- 

- Click the "Find My Ticket!" button after using the artist dropdown to test if the correct price appears if Taylor Swift is selected.
    - For example, if you select `Bad Bunny` from the artist dropdown, you should see:
- 

    - If you select `Taylor Swift` from the artist dropdown, you should see:
- 

LEVEL 4

Writing If-Else Statments

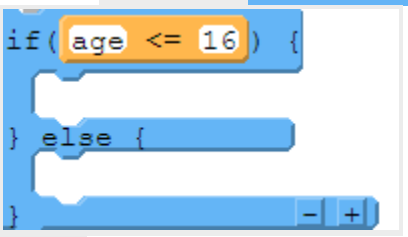
🧑👧👦 Analyzing User Stories

What kinds of discounts do venues offer when children are attending?

✅ Do This - Milestone 3 / 6

Note: You can click the instructions below to expand them and see more specific instructions

Tasks:

1. For users who have an `age` of 16 or less, update the `isChild` variable to `true`
    - Use an `if/else statement` and `<=` blocks to create an if-else statement like this:
2. 

3. or this:

None

```
if(age <= 16){

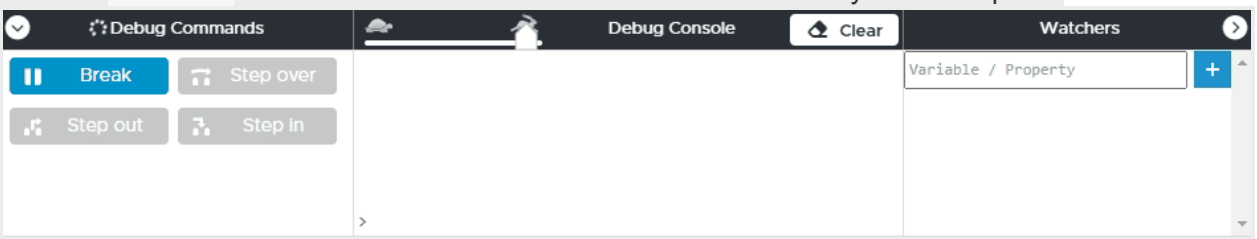
} else {

}
```

4. 👉 Show me where 👈
    - Assign the value of `true` to `isChild` when the above boolean expression is true
    - In the else section, assign the value of `false` to `isChild`

🎯 Test Your Code

Note: You can click the tests below to expand them and see more specific instructions

- Add the `isChild` variable to the "Watcher" area at the bottom of your workspace.
- 

- Click the "Find My Tickets!" button after using the `age` text box on the app screen and watch that the appropriate value is assigned to `isChild` in the Watcher area.
    - If you select a value of `16` or lower, you should see `isChild: true` in the Watcher
    - If you select a value higher than `16`, you should see `isChild: false` in the Watcher

LEVEL 5

If-Else If-Else Statments

👧🏽👧🏽👦 Analyzing User Stories

During certain peak months, concert tickets tend to go up considerably. This change in price should be reflected in the app.

✅ Do This - Milestone 4 / 6

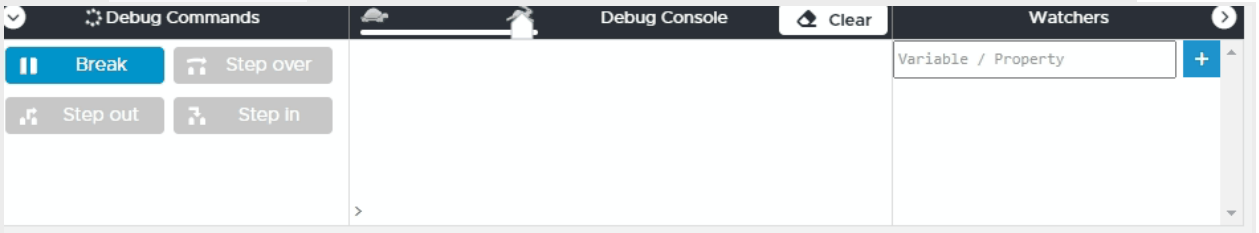
Note: You can click the instructions below to expand them and see more specific instructions

Tasks:

1. Set the value of `isPeakMonth` based on the specific months identified below:
    - The summer months (*June / July / August*) are prime concert times and are considered peak
        - To check a range of possibilites, we can use the `&&` operator to check more than one condition at a time.
        - Use `>=`, `<=`, and `&&` to replace the first `true` value in the if-else if-else statement like this: `concertMonth >= 6 && concertMonth <= 8`
        👉 Show me where 👈
    - Concerts taking place in January or December are also considered peak times
        - To check between two different possibilities we can use the `||` operator to check if either is true.
        - Use `==` and `||` to replace the second `true` value in the if-else if-else statement like this `concertMonth == 1 || concertMonth == 12`
        - Assign the value of `true` to `isPeakMonth` when the above boolean expression is true
    - In the else section, assign the value of `false` to `isPeakMonth`

🎯 Test Your Code

Note: You can click the tests below to expand them and see more specific instructions

- Add the `isPeakMonth` variable to the "Watcher" area at the bottom of your workspace.
- 

- Click the "Find My Tickets!" button after using the month dropdown on the app screen and watch the value that is assigned to `isPeakMonth` in the Watcher area.
    - If you select the values 1, 6, 7, 8, or 12, you should see `isPeakMonth: true` in the Watcher
    - If you select any other value, you should see `isPeakMonth: false` in the Watcher

LEVEL 6

If-Else If-Else Statments

👧👦 Analyzing User Stories

I want the app to accurately update the base price of my concert tickets based on the inputs that I provide.

✅ Do This - Milestone 5 / 6

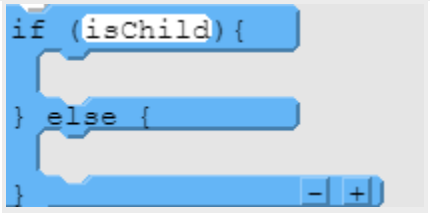
Tasks:

1. According to the table below, display the appropriate activities to the user depending on `isChild` and `isPeakMonth`.

Condition	Price Adjustment	Description
Child (Regardless of Month)	`basePrice * 0.5`	All children receive a discount, regardless of month.
Adult & Peak Month	`basePrice * 1.2`	Adults pay a surcharge for peak months.
Adult & Not Peak Month	`basePrice`	Adults pay the standard fare during off-peak months.

Here are the steps broken down a bit more. Use the Test Your Code section below as you go through each row of the table.

2. For users who are children, multiply the `basePrice` by `0.5`
   - Use `if/else` to create an if-else statement like this:



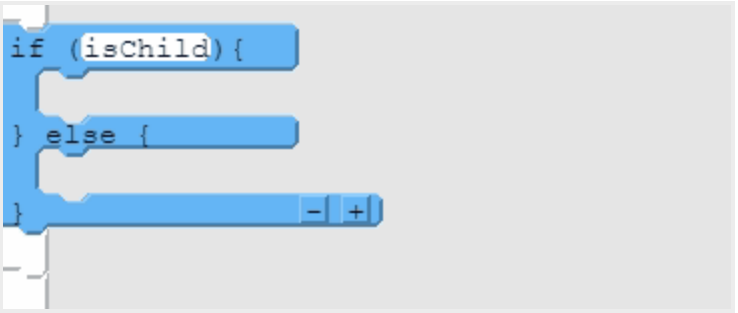
3. or this:

```
if (isChild){

} else {

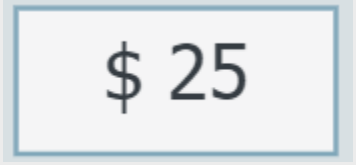
}
```

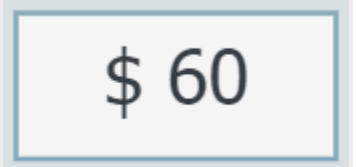
   - 👉 Show me where 👈
4. For users who are adults and want to attend a concert during a peak time, multiply the `basePrice` by `1.2`
   - Expand the if-else statement into an if-else if-else statement (see below)

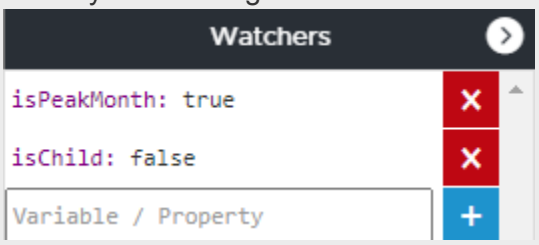
5. 

    - Use `&&` and `!` to create a boolean expression like this: `!isChild && isPeakMonth`
    - Assign the value of `basePrice * 1.2` to `finalPrice` when the above boolean expression is true
6. Assign all other users the value of `basePrice`
    - You can utilize the `else` part of the conditional statement and skip the boolean expression.
        - Assign the value of `basePrice` to `finalPrice`

🎯 Test Your Code

- Click the "Find My Tickets!" button after using the month dropdown and age text box to test if the correct value for `finalPrice` appears at the bottom of the app.
    - For example, if you select `1` for `concertMonth` and `16` for `age`, you should see:
- 

    - If you select `1` for `concertMonth` and `17` for `age`, you should see:
- 

- If you aren't getting the output on the app screen you expect, a good next step is to add variables to the "Watcher" area. You will most likely want to add `isPeakMonth` and `isChild` in addition to others you think might be involved in the bug.
- 

- Once you've added the appropriate variables to the Watcher, use the debugging skills you've practiced so far in this class to help you get your program working.

LEVEL 7

Make It Your Own

✅ Do This - Milestone 6 / 6

- Look through the possible ways you can continue to modify and extend this app. Then pick one or more to work on!
    - 🔧 Refactor Your Code
        - This program's logic can be written differently. Do you see a way that the same app functionality could be achieved with easier to follow code or with fewer lines of code overall?
            - Can you rewrite the conditional in Milestone 5 to include a nested conditional (an if-else contained within an if-else statement)?
            - You could choose to rewrite some of the conditional logic in a way that makes more sense to you.
            - You could try to refactor this program so that the variables `isPeakMonth` and `isChild` aren't needed anymore
    - 🎨 Enhance User Experience
        - Can you add in another use input that adds further complexity to ticket price? Add in a dropdown / textbox for such things as: Promo Codes, area of the country, number of tickets, etc.
        - Update your conditional logic to account for these new additions to your app!
    - ✨ Add New Features
        - Are there other things that would make the suggestions this app provides more useful?
        - Add a little bit of randomness to the ticket buying process by utilizing `randomNumber` if certain conditions are met.
        - The table in the last step laid out one way to organize how the price is affected by `isChild` and `isPeakMonth`. Do you have a better idea of how that table could be organized to give more useful suggestions?
            - Write out your own table and change the last part of your program code to match your version.

🧐 Made Changes That Broke Your Program?

- You can use the "Version History" button to backtrack to a previous version of your program.
- 