The Computing Challenge 2023

Lab Sheet 2: Simple Quiz

This worksheet assumes you have completed **Lab Sheet 1** and are familiar with the basic workings of Thunkable. **Remember that your assignment requires you to document either yesterdays or today's lab sheet in your portfolio.**

This worksheet is an INDIVIDUAL task NOT a team task. All team members are expected to complete this work as it is crucial to your understanding of fundamental programming concepts and the use of Thunkable.

Hopefully, you have familiarised yourself with the basic workings of Thunkable and are eager for something a little more challenging. Today we are going to create a simple multiple-choice quiz using some of the techniques covered in this morning's lecture. We are stepping things up quite a bit now, so if you get stuck, that's normal, don't be afraid to ask your tutor for help, it's what we are here for.

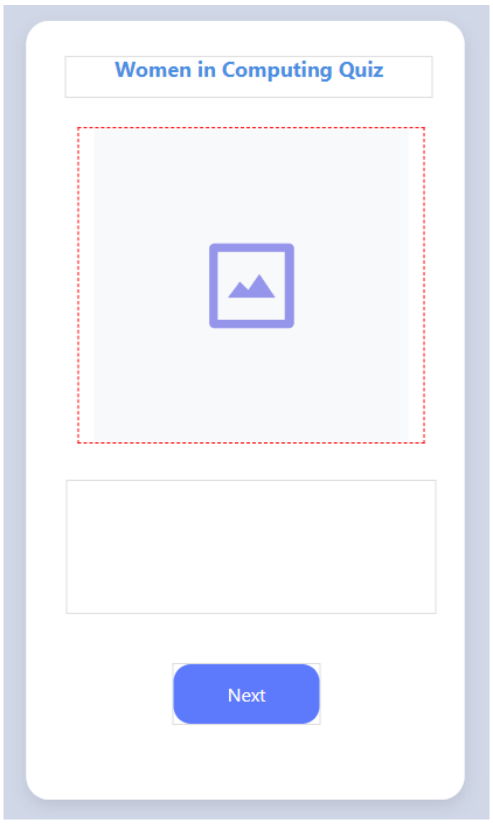
Part 1: The Questions

Login to Thunkable and create a new project. Name the new project something like **CO1111 My Quiz**.

Firstly, we'll need to import some pictures to accompany each question in our quiz. Download the **lab2quizmedia.zip** file from Blackboard and extract the files onto your computer. Click the **Assets** tab and upload the extracted files into Thunkable.

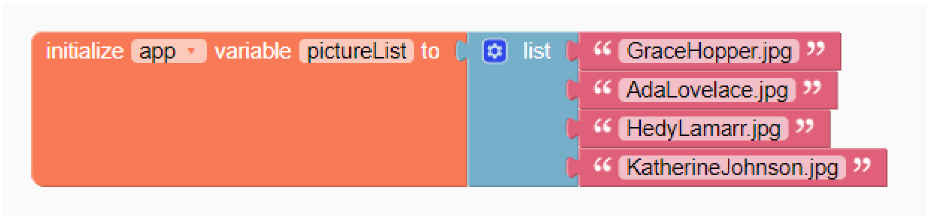
Now lets create the interface. Click on the **Design** tab and add a **label**, an **image,** another **label,** and a **button** component to **Screen1.** The first label will be our title, the image will be where we show a picture related to the question, the second label will be our question, and the button will be to navigate to the next question. Name these components something like **TitleLabel, QuestionImage, QuestionLabel** and **NextButton**.

The screen should look something like the one below:



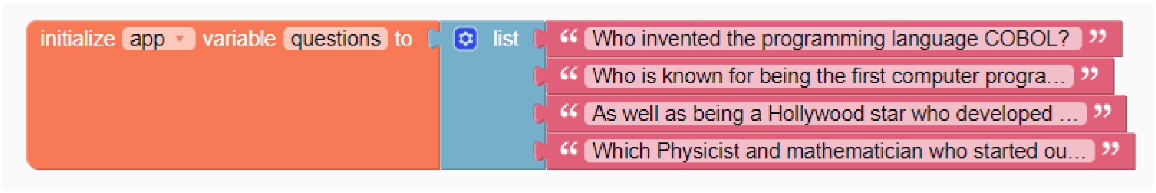
Now let's add some **Blocks**!

Create a variable called ***pictureList*** and plug a **list** block into it. The list will come with a list of 3 numbers as default, delete them. We want to make a list of our 4 pictures, so click on the little blue cog and add one more list item. Now add 4 blank text blocks to these empty list item slots. Each one should be named **exactly** as each of the picture filenames. Your blocks should now look like this:



```
initialize app variable pictureList to  list  " GraceHopper.jpg "
                                              " AdaLovelace.jpg "
                                              " HedyLamarr.jpg "
                                              " KatherineJohnson.jpg "
```

You should also create another list that will hold the questions. Build the blocks shown below:



```
initialize app variable questions to  list  " Who invented the programming language COBOL? "
                                            " Who is known for being the first computer progra... "
                                            " As well as being a Hollywood star who developed ... "
                                            " Which Physicist and mathematician who started ou... "
```

Make sure that the list placing for the question and the corresponding answer picture is the same in each list. Here are the questions and answers.

Q1. Who invented the programming language COBOL? *Answer: Grace Hopper*.

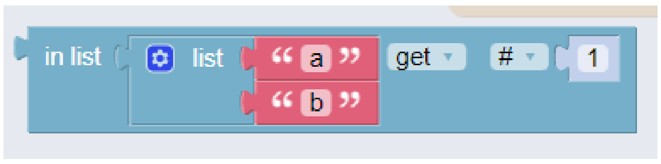
Q2. Who is known for being the first computer programmer? *Answer: Ada Lovelace*.

Q3. As well as being a Hollywood star who developed a radio guidance system for Allied torpedoes during WW2? *Answer: Hedy Lamarr*.

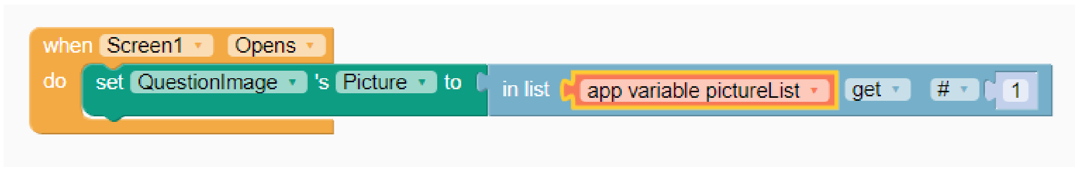
Q4. Which Physicist and mathematician who started out as a 'Human Computer' played a key role in the success of the Project Mercury and Apollo 2 mission? *Answer: Katherine Johnson*.

Feel free to make up your own questions and answers.

The first thing we want to happen when our app starts is to show the first question. So, go grab a **when Screen1.opens** event block. Within this block put a **set QuestionImage's Picture to** block. Plug in an **in list get** block:

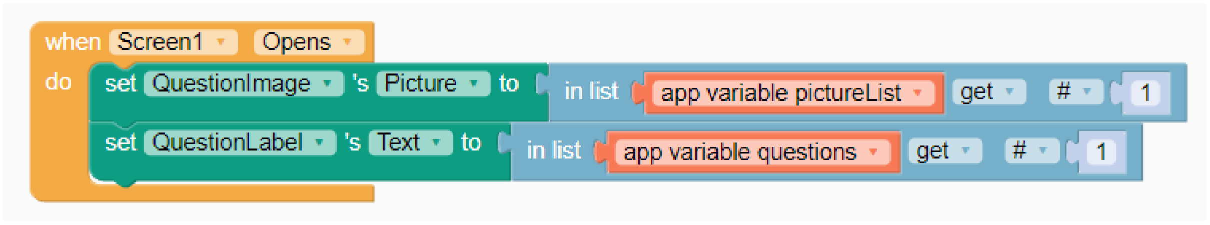


Remove the sub-list within this block, and place a copy of the pictureList variable. Leave the index number to 1 to make sure it selects the first picture in the list. The blocks should now look like this:

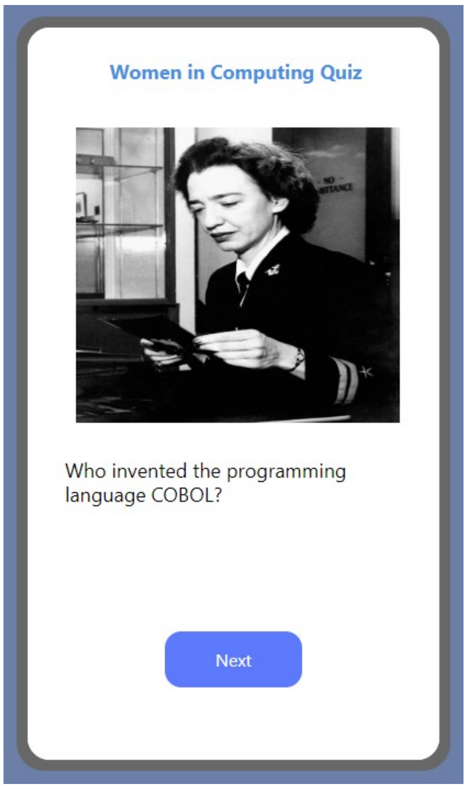

when Screen1 Opens
do set QuestionImage 's Picture to
in list app variable pictureList get # 1

What is happening here? Basically, when the app starts it will set the Question Image to the first one in the pictureList variable, which in our case is GraceHopper.jpg. Test it to see if that does indeed happen. If not you have probably incorrectly typed the filenames in the variable list (Hint: it is case sensitive).

Now let's add the question text:


when Screen1 Opens
do set QuestionImage 's Picture to
in list app variable pictureList get # 1
set QuestionLabel 's Text to
in list app variable questions get # 1

Test it again, it should look something like this:

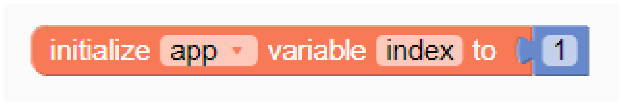


Women in Computing Quiz

Who invented the programming language COBOL?

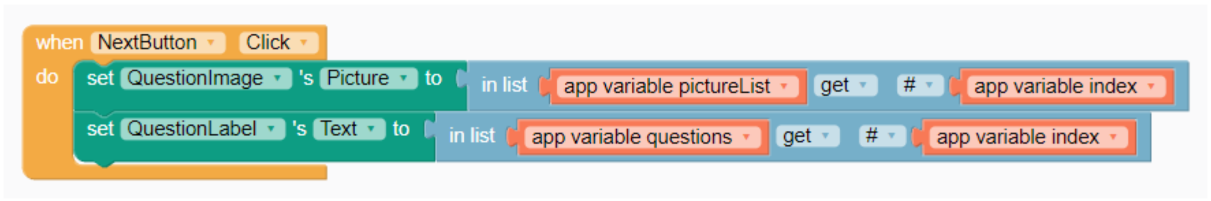
At the moment our next button does not work, as we haven't yet programmed an event for it. Let's do that now. When we click the button we want it to show the next question, and then the next, until we reach the end of our question list. To add this functionality we will need to know whereabouts we are in the list at all times, so that we know which list item to show next and to know when we reach the end of the list.

Create a variable called ***index***. This variable will hold a number which corresponds to the question number we are up to. Initialise it to '1':



```
initialize app variable index to 1
```

Now drag out a **when NextButton.click** event. Build the blocks for this event as shown below:



```
when NextButton Click
do  set QuestionImage 's Picture to  in list app variable pictureList get # app variable index
    set QuestionLabel 's Text to  in list app variable questions get # app variable index
```

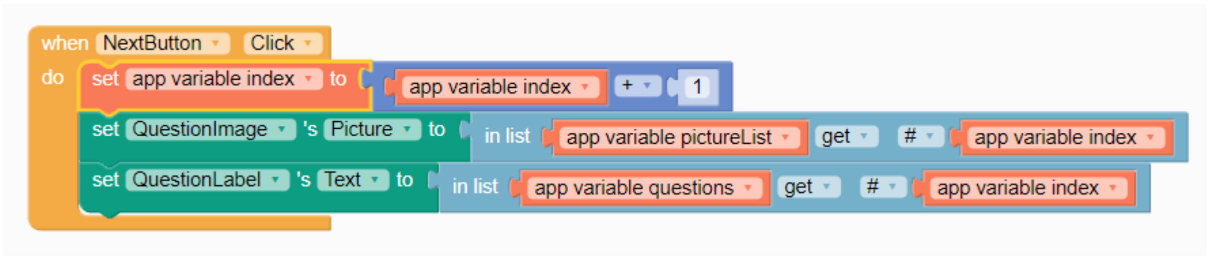
What are we doing in these blocks? Have a go at explaining it to one of your team mates.

Test your app to see that everything works correctly. Hopefully you will see that you now get the 2nd question when you click the next button. You will also notice that if you click the next button again you don't get the 3rd question. **Why is this?**

Answer:

Every time we click the next button we are getting the image and question at ***index* +1** (1 + 1 = 2) in the lists. In order to work our way through the lists we need to increment the value of ***index*** by **1** each time we click the next button.

Change the **NextButton click** event block so that it looks like this:

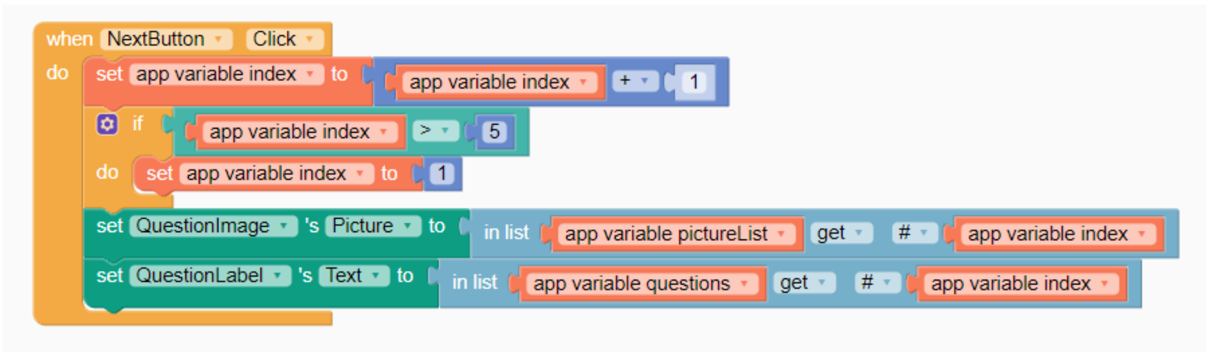


```
when NextButton Click
do  set app variable index to  app variable index + 1
    set QuestionImage 's Picture to  in list app variable pictureList get # app variable index
    set QuestionLabel 's Text to  in list app variable questions get # app variable index
```

Now test your app. If all is working correctly you should be able to cycle through the pictures by clicking the next button.

We still have a problem though. Our app doesn’t know when it reaches the end of the question list. The value of the index variable is allowed to increment to 5, but we only have 4 questions. So, it drops of the end of the list and throws an error. Let’s fix this now.

Get an **if** block from the ‘Control section and place it underneath the block where we increment the index variable. In the **if** part add an **equality** block (‘Logic’ section) and check if the variable ***index*** is equal to ‘**5**’. Your blocks should now look like this:



```
when NextButton Click
do  set app variable index to  app variable index + 1
    if  app variable index > 5
    do  set app variable index to 1
    set QuestionImage 's Picture to  in list app variable pictureList get # app variable index
    set QuestionLabel 's Text to  in list app variable questions get # app variable index
```

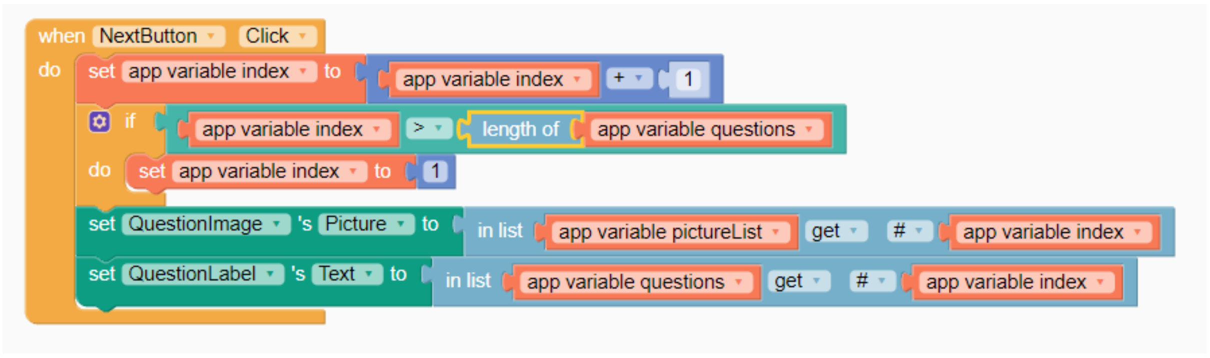
In the blocks above if ***index*** is equal to 5 it resets it to 1.

Test your app. This should now work much better. Instead of dropping off the end of the list the app loops back to the first question.

There is still a problem with this code though. See if you can figure out what it is before reading on.

Answer: The number **5** restricts our quiz to 4 questions. If we want to add more questions to our quiz, we would need to change the number 5 to another number. In programming this is not considered good practice. However, it is quite easy to improve this code.

Thunkable has a block called **length of** which checks the length of a list. We can change the **if** statement so that it checks if the value of the ***index*** variable is greater than the end of the question list:



```
when NextButton . Click
do  set app variable index to ( app variable index + 1 )
    if  app variable index > length of app variable questions
    do  set app variable index to 1
    set QuestionImage . 's Picture to  in list app variable pictureList get # app variable index
    set QuestionLabel . 's Text to  in list app variable questions get # app variable index
```

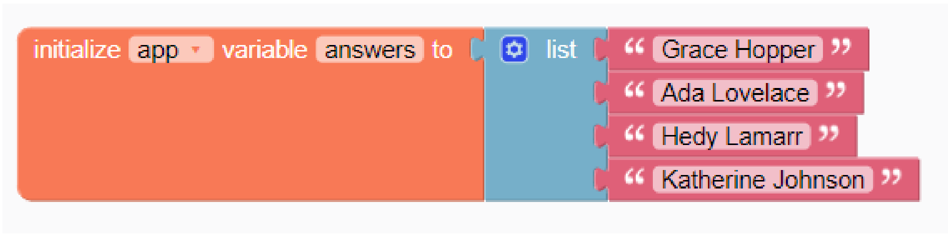
Now we can add as many questions as we like to our question list and our app will always know when it has reached the end of the list.

Extra Challenge: add a previous button so that you can navigate backwards and forwards through the questions.

Part 2: The Answers

So, that's our questions, now we need to allow the user to answer each question.

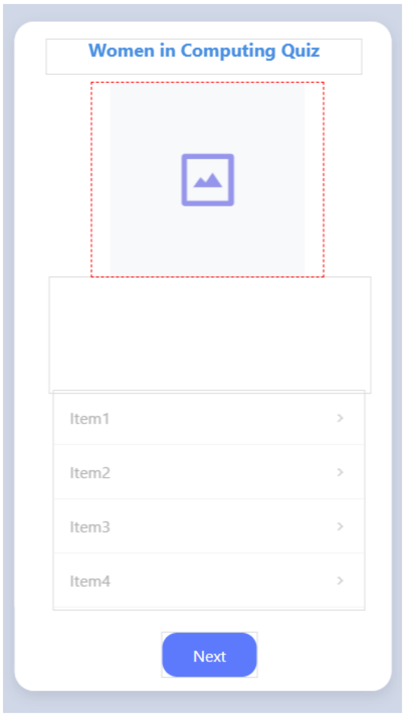
Let's start by creating a list variable to hold a list of the correct answers. Remember to make sure the answers are at the same position in the list as the corresponding question in the question list.



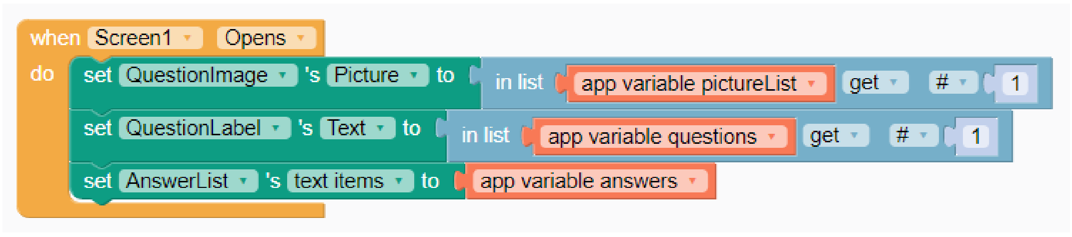
```
initialize app variable answers to  list  " Grace Hopper "
                                          " Ada Lovelace "
                                          " Hedy Lamarr "
                                          " Katherine Johnson "
```

To start with we are going to use this list as the four choices we show the user for each question. This would be a pretty rubbish quiz in real life, but for now we will use it to test the functionality of presenting the user with some answer options.

To show the answer options we are going to use a component called a **Simple List**. Add one of these to your interface and name it **AnswerList.** You will probably have to rearrange everything on your screen to fit it all on, but it should look something like this:



We now need to populate our **AnswerList** with the answers. Go to the Blocks window, and drag out the **set AnswerList's text items to** block**.** Plug it into the **when Screen1.opens** click event handler. We are going to set it to the list of answers in our ***answers*** list variable.

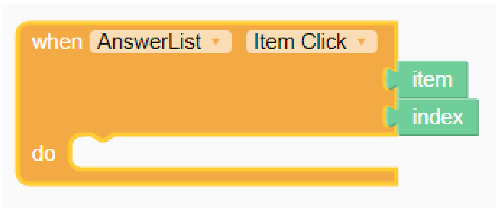


```
when Screen1 Opens
do  set QuestionImage 's Picture to  in list  app variable pictureList  get  #  1
    set QuestionLabel 's Text to  in list  app variable questions  get  #  1
    set AnswerList 's text items to  app variable answers
```

Test your app to check everything is working correctly. You should be able to see the 4 answers in the list.

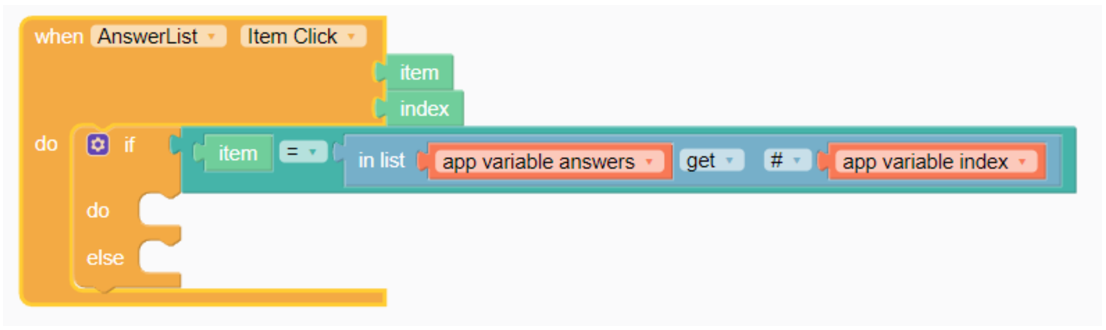
Next we need to check the users answer. This is simply a case of getting the users answer (the answer they chose in the Answer List) and comparing it with the correct list item in the variable list ***answers*.**

Start by dragging out an event handler for when the **AnswerList** is clicked. You will notice that it comes with two event parameters: ***item*** and ***index***. Item is the text that has been clicked and index is the index number of the item the user chose from the list.

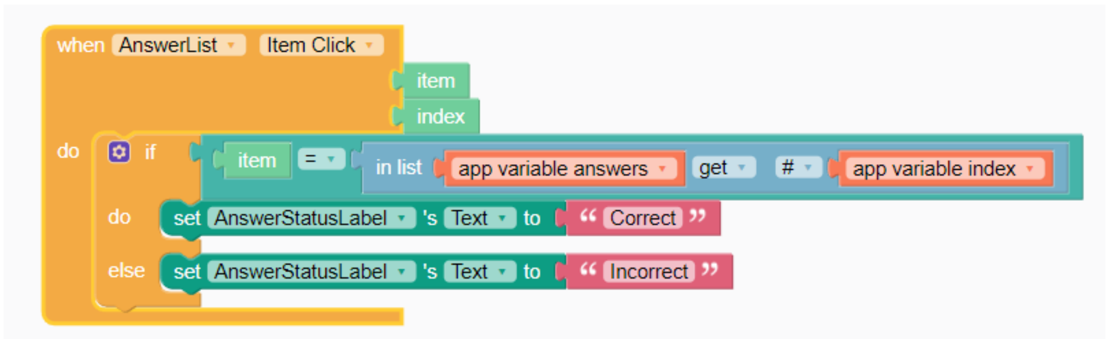


```
when AnswerList Item Click
                                item
                                index
do
```

Now, let's compare what the user chose with the correct answer. Add the following blocks:


when AnswerList Item Click
item
index
do if
item = in list app variable answers get # app variable index
do
else

We should now tell the user if they answered correctly or incorrectly. Add another **label** to your interface, and finish off the **if do el**se block as shown:


when AnswerList Item Click
item
index
do if
item = in list app variable answers get # app variable index
do set AnswerStatusLabel 's Text to "Correct"
else set AnswerStatusLabel 's Text to "Incorrect"

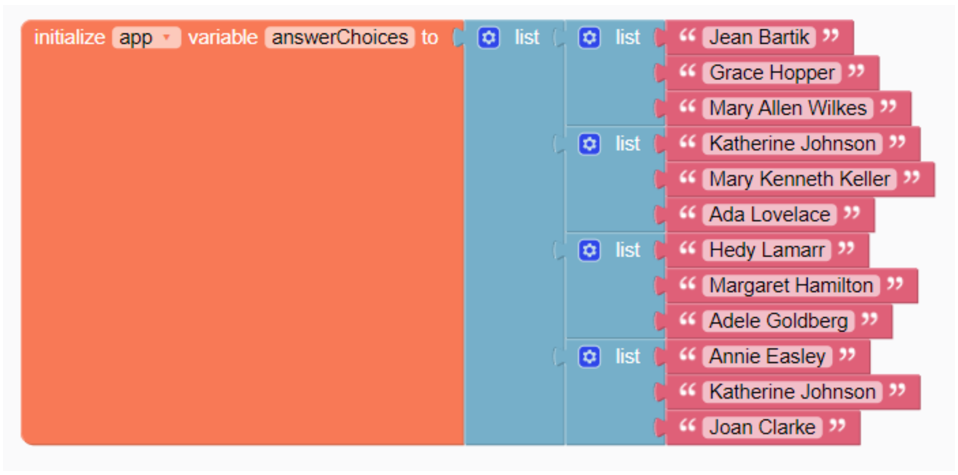
Test your app. If something has gone wrong check your blocks and if you get stuck ask your tutor for help.

So, there we have the very basic workings of a quiz. But, at the moment it is quite limited. Before moving on to the final section have a think about how we could improve it.

Part 3: Improving Our Quiz

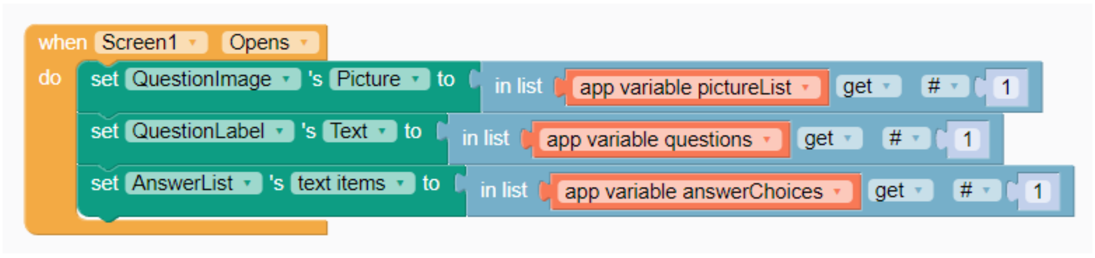
One of the problems with our quiz is that the same list of answers is shown for each of the questions. This isn't ideal, and a better idea would be to have a different choice of answers for each question. Let's give each question three different answer choices. To do this we will need to create a **list of lists**. If that sounds confusing, don't worry, it will all become clear soon.

Create a new variable to hold our list of lists. Name it something like ***answerChoices.*** Add a list block with four inputs (one for each of our four questions). Now add four sub-lists and slot them into each of the inputs and create the answer choices as shown:



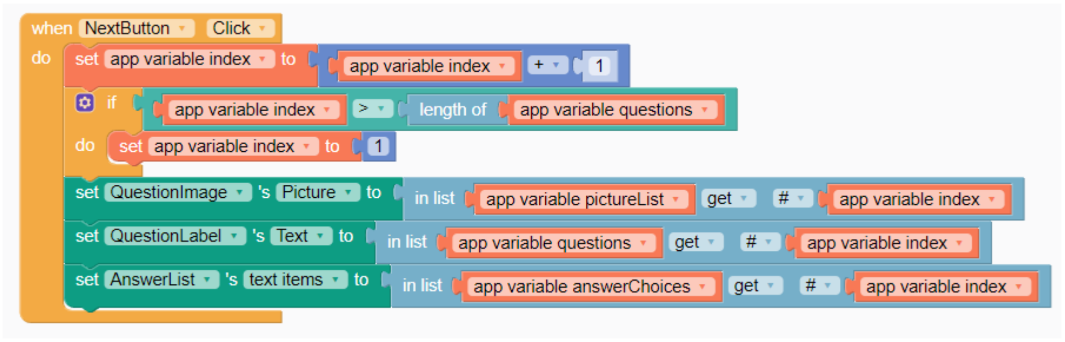
```
initialize app variable answerChoices to list
    list "Jean Bartik" "Grace Hopper" "Mary Allen Wilkes"
    list "Katherine Johnson" "Mary Kenneth Keller" "Ada Lovelace"
    list "Hedy Lamarr" "Margaret Hamilton" "Adele Goldberg"
    list "Annie Easley" "Katherine Johnson" "Joan Clarke"
```

Now we need to change our quiz so that these answer choices are shown to the user rather than the same four from the ***answer*** variable. When the app starts, we want to show the first set of answers in ***answerChoices***. Modify the **when Screen1.opens** block to do that now:



```
when Screen1 Opens
do  set QuestionImage 's Picture to in list app variable pictureList get # 1
    set QuestionLabel 's Text to in list app variable questions get # 1
    set AnswerList 's text items to in list app variable answerChoices get # 1
```

We also need to change the **when NextButton.click** event block:



```
when NextButton Click
do  set app variable index to app variable index + 1
    if app variable index > length of app variable questions
    do  set app variable index to 1
    set QuestionImage 's Picture to in list app variable pictureList get # app variable index
    set QuestionLabel 's Text to in list app variable questions get # app variable index
    set AnswerList 's text items to in list app variable answerChoices get # app variable index
```

And that is it.. now test your app!

Taking Things Further

There are many more improvements that could be made to this app. See if you can think of any and list them here. If you are feeling brave have a go at implementing them.

Taking Things Even Further (and maybe earn some points)

If you are really up for a challenge, think about how we could make the code more efficient – specifically relating to the lists. We currently use multiple separate lists, could this be improved upon? HINT: investigate the use of objects.