



Girls' Programming Network

2017

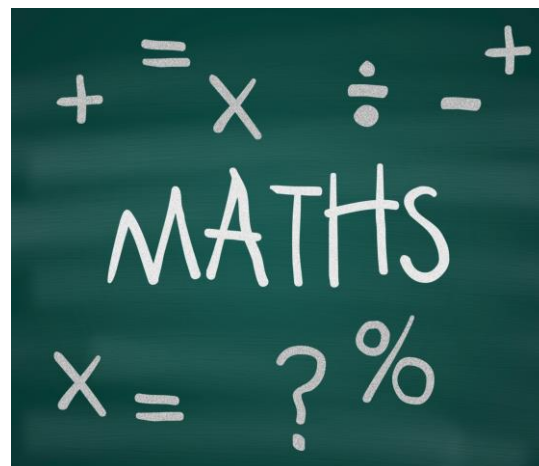
Sassy Security Chatbots!

More security!



So our secret information is pretty secure right now. But wait, what if a nosey younger sibling finds out our secret password!!

Let's add in a test to catch people out, we'll make an intelligence test that will be hard for little kids to crack!



Part 5: Intelligence Test Time

Let's add some extra security in case someone guesses our secret password! Adding an intelligence test will mean that nosey little brothers and sisters can't get in!

Goal: Add a maths question to make sure they are smart enough!!

Task 5.1: Ask the question

Step 1) We're going to put new code in between your password while loop and your password checking if statement. **Find that spot!**

Step 2) Ask the user a maths question that test if they can do addition

Something like "What is 5 plus 7: "

Hint: Use `input` to ask for their answer, assign the user's answer to a variable

Step 3) Create a variable called `real_answer`, store the real answer to your question in it

Step 4) Remember how input gives us back a string? We need to convert the user's answer to an integer (a number) so we can compare it to `real_answer`.

Hint: Remember how `int("6")` gives us the number 6 back? Store the answer in a new variable!

This could look like:

What is 5 plus 7: 13



Task 5.2: Check the answer!

We need to check the answer, then store whether they passed or not!

Step 1) Check if the users answer is equal to `real_answer` (*Hint: `if` statement*)

If they passed:

- Create variable called `test_result`, store the word “passed” in it
- print a nice message to the user like:
`What is 5 plus 7: 12`
`Well you got it this time!`

Step 2) If they didn't pass, (*Hint: Add an `else` statement to your `if` statement*)

- Create variable called `test_result`, store the word “failed” in it
- print a nice message to the user like
`What is 5 plus 7: 10`
`Looks like someone needs to practice arithmetic :P`

We are storing “passed” and “failed” in the variable `test_result` to decide what answer we want to give our user.

Task 5.3: Secure the secret information

Use the test result to help decide what secret info they get to see!

Step 1) Go to the `if-elif-else` statement at the end of your file that displays the different secret information

Step 2) Real secret info: We only want to display the real secret information if:
They entered the real secret password *and* their `test_result` is “passed”
(*Hint: your `if` statement*)

Step 3) Decoy info: We only want to display the decoy information if:
They entered the decoy password
(*Hint: your `elif` statement*)

Step 4) The *else* stays the same!



Part 6: Harder maths!!

So right now it asks the same maths question every time!! This means sneaky little kids could just keep guessing different numbers until they get the right answer!

Goal: Make it so it asks a new maths questions every time using random numbers!

Task 6.1: Choosing random numbers!!

Step 1) At the top of your file write `import random`
this gives us access to things that helps us chose things randomly

Step 2) We want to choose 2 numbers randomly.
If we want to choose a number from 1, 2 or 3 we can write:
`num1 = random.choice([1, 2, 3])`
The randomly chosen number is stored in `num1`

Add the code for `num1`, but make it choose from the numbers 1 to 10.

Step 3) Repeat it to get a second number `num2`

Want to see your random numbers? Try printing them out!

Task 6.2: A Better Question and Answer

Step 1) Update the question to use `num1` and `num2` instead of the two numbers you hard-coded in before.
Hint: Remember we can't add integers (numbers) to strings. `str(6)` which gives us back "6" as a string. We can use this to add the rest of the question to our numbers!

Step 2) Let's update the answer to match our random numbers! Use the variable `real_answer` that you created in Task 5.1, but instead calculate the answer using `num1` and `num2`.
Hint: We just need to add them together and store them in `real_answer`!

★Bonus 6.3: Bigger Numbers!★

Ten numbers just isn't enough!! We want more numbers for bigger harder questions, but don't want to spend a lot of time writing out lots of numbers that's boring!

We can use a python function called `range`.

Writing `range(1, 6)` is the equivalent of `[1, 2, 3, 4, 5]`

It only printed up to 5, but you said 6!! Yep that's right, it doesn't include the last number!

We can change our line of code:

```
num1 = random.choice([1, 2, 3])
```

Use `range` instead of `[1, 2, 3]` to ask maths questions with numbers 1 to 100

