Junior Coding for Good, Badge 1 requirements:

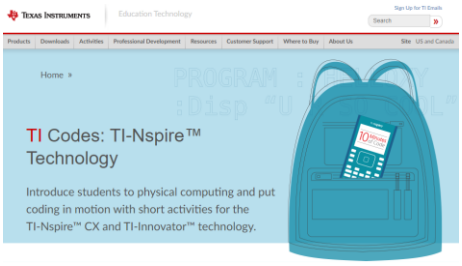
1. Create algorithms for a computer that follow a sequence.
2. Use loops to improve your algorithm.
3. Keep your code interesting with conditionals.
4. Create your own set of commands that use conditionals.
5. Learn about women in computer science.

Be sure to review the "Junior Coding for Good" guide for the badge requirements and badge steps provided by the Girl Scouts and your leader. It also includes relevant vocabulary and interesting background information to spark your interest in coding. This lesson will allow you to earn **Badge 1: Coding Basics** using your TI-Nspire™ CX family graphing calculator.

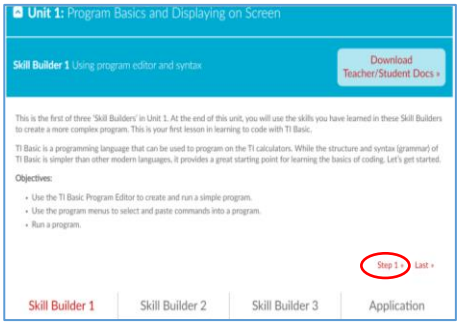
Introduction

The programming language **BASIC** (which stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) was developed in the 1960s as an easy system for teaching computer programming. TI-Basic is similar to other flavors of BASIC, but you should select the programming words and commands from the TI-Nspire™ Program Editor's menus as you learn in the **TI Codes** lessons.

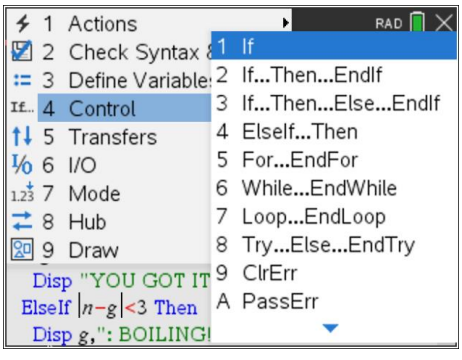
1. For an introduction to programming on the TI-Nspire™ CX family graphing calculator, see the TI Codes lessons at **education.ti.com > Activities > TI Codes > TI-Basic**. Units 1 through 4 should be enough to get started including coverage of algorithms and conditionals as the badge requirements state (1–3 above). After completing those lessons, you will then have a basic understanding and can return to this lesson for the **Junior Coding, Badge 1** project. Go to TI Codes (for TI-Nspire™ technology).



How to navigate TI Codes: Be sure to notice that each unit has three Skill Builders (SB) and one Application activity, as you can see here. You will navigate through each Skill Builder by clicking on the "Step 1" in the right bottom corner, which will take you through the content. This screenshot shows you are currently in Skill Builder 1 > Step 1. After you complete all the steps in SB 1, you will then move to SB 2, and so on, until you complete all of Unit 1. You will then move on to Units 2, 3, 4 … and do the same thing.

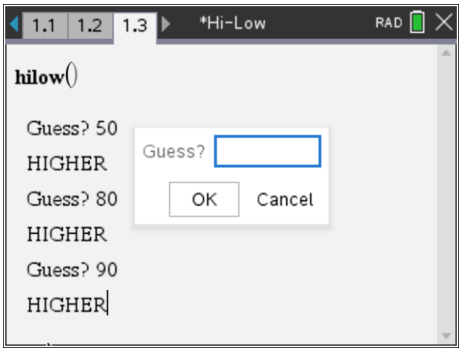


2. After you have completed the TI Codes Units 1–4, you know about:
   - The programming process (planning, coding, testing and debugging)
   - Using the TI-Nspire™ Program Editor and its menus
   - Running a program
   - Some important pieces of TI-Basic code: Input (Request), Disp, variables and assignment statements, **If…Then…Else** structures and loops like **For** and **While** (some of these statements are shown to the right)



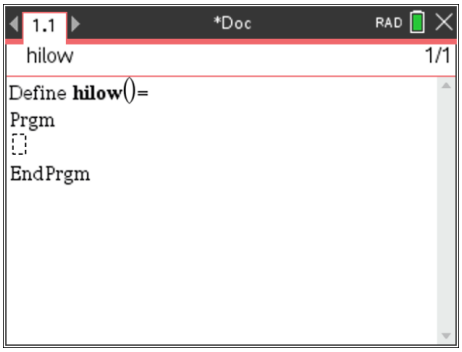
3. Your Badge 1 project will be a **Number Guessing Game**. The calculator will select a random number between one and 100 and the player must guess the number. And:
   - The game continues until the player guesses the number
   - If the guess is wrong the computer will tell whether to guess HIGHER or LOWER to get closer to the number
   - After the player guesses the number, the computer will report the number of guesses that the player took

   A sample run of the program is shown to the right. The player entered the numbers 50, 80 and 90. The program responded with "HIGHER" three times, so we kept guessing higher.



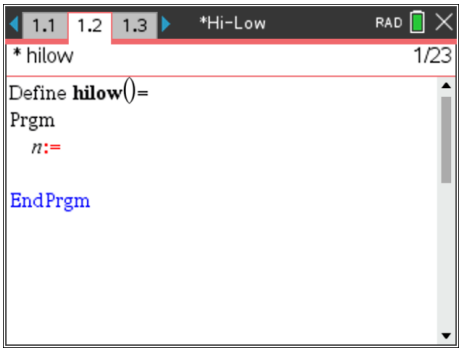
4. Our plan is:
   - Create some variables
   - Use a **While** loop to keep guessing as long as we have not guessed the number
   - In the loop
     - Count the guesses
     - **Input** a guess
     - Check to see whether the guess is too high or too low
   - After the loop
     - Display a congratulations message
     - Display the number of guesses

5. Start a new program. From the Home screen, select New Document Select Program Editor from the menu, name the program **HILOW** and press **[enter]**. You are now using the Program Editor. As you learned in the TI Codes lessons, this Editor has its own programming menu.



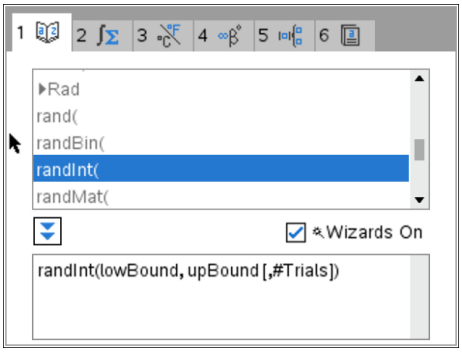
6. Start your program by setting up three variables: The computer's random number (**n**), the player's guess (**g**) and the guess-counter ©.

   Type the variable **n** and the **:=** symbol found at **ctrl**-**[math templates]** to the right of the **9** key.



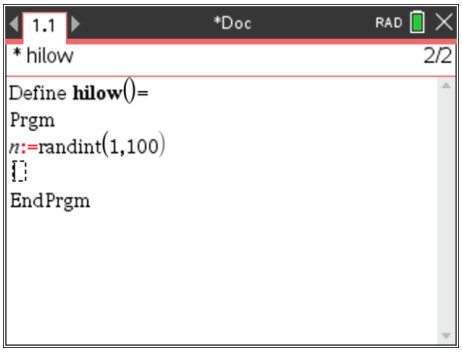
7. Now enter the **randInt()** function that produces a random integer.

   You can simply type **randInt()** or find it in the **[catalog]** (the key with the book on it), **tab 1.** Press the letter R and then scroll (down-arrow) to the **randInt()** function and press **[enter]**.



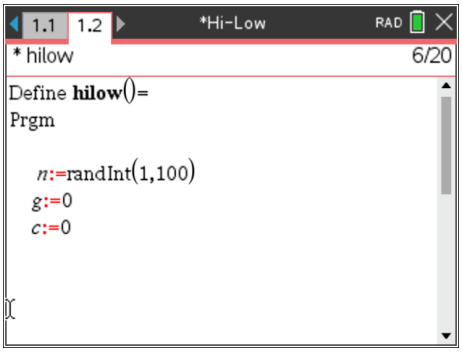
8. Complete the rest of the statement by adding the lower and upper values inside the parentheses:

   n := randInt(1, 100)



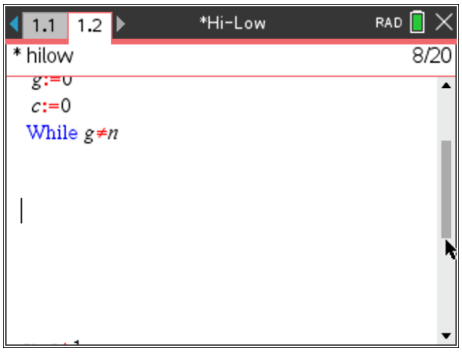
9. The other two variables, **g** and **c** are given the value 0. Remember to use **:=** (assignment) and the correct letters.

   Your program may contain blank lines and extra spaces (as pictured). This has no effect on the program.



10. Now make a **While…EndWhile** loop to continue entering guesses until the guess equals the computer's number (as long as **g** *does not equal* **n**).

    See the next step for more information on the **While** statement.

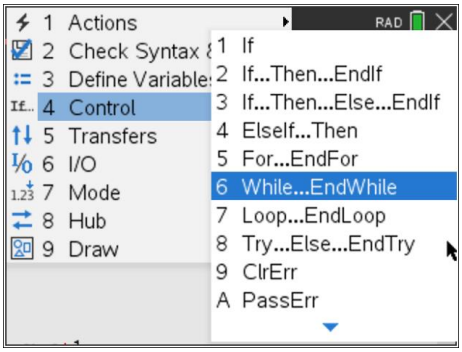


11. **While…Endwhile** is found on **[menu] > Control**, and you will type the condition after **While.**

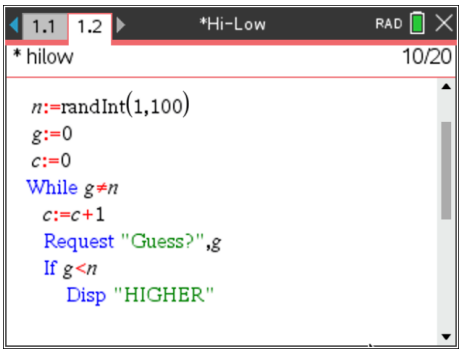
    The "does not equal" symbol is on the **[ctrl] [=]** menu with the all the other relational operators.

    While g ≠ n

    EndWhile



12. Inside the **While** loop (*we have indented the block for clarity*):
    a. Add 1 to the guess-counter variable **c.**
    b. Write a **Request** statement to accept the player's guess (**g**). **Request** is found on **[menu] I/O > Request. After the word Request** type the word **"Guess?"** in quotes, then a comma, then the letter g. **"Guess?"** will appear in green automatically.
        i. **"Guess?"** is the *prompt* **, g** is the variable.
    c. Now add two **If** statements:
        i. If **g** is *less than* (<) the number (**n**) – display "HIGHER"
        ii. If **g** is *greater than* (>) the number (**n**) – display "LOWER"
    d. You can simply type the word **If** or get it from **[menu] > Control.**
    e. Be sure to write *two* **If**… statements even though *only one is shown here*.

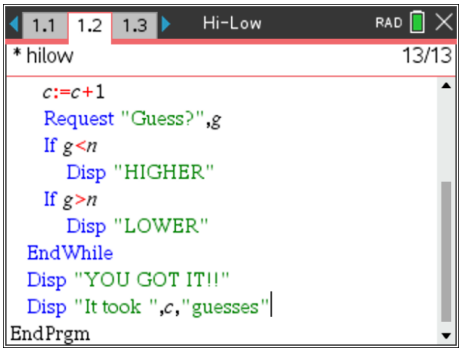


    Reminder: The **[ctrl] [=]** menu has the symbols **<**, **>**.

    Note 1: The indentations in the code are for clarity only. Indentations have no effect on the program and are optional, but useful for debugging.

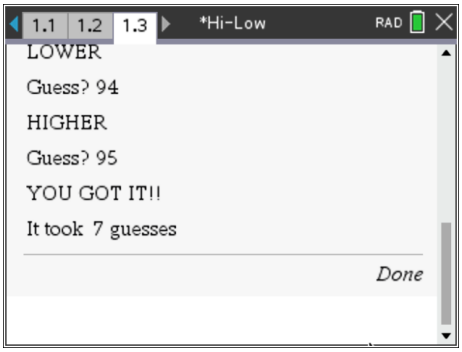
    Note 2: if you want UPPERCASE letters use the **[shift]** *key, or use* **[ctrl] [shift]** *for caps lock.*

13. After the **EndWhile** statement:
    - Display **"YOU GOT IT!"** and
    - Display the number of guesses (**c**) that the player took to guess the number

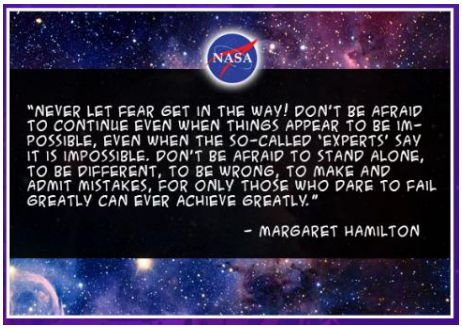


14. To run the program, press **ctrl+R** to paste the name of the program on a Calculator page, and press **[enter]** to begin running the program.

    When the program ends (see *Done*) you can return to the Program Editor page by pressing **[ctrl]**-**[leftarrow]** (previous page), and make changes as needed.



15. For the final part of the badge requirement, remember to research "women in computer science," such as Margaret Hamilton, who helped humans land on the moon. Use the internet for your searching.



Congratulations!

You have completed the requirements for earning your **Junior Coding for Good, Badge 1: Coding Basics**. Now that you have completed this requirement, you are challenged to *give service* by *sharing* what you have learned about coding with others. Refer to the *Coding for Good* Girl Scout guide for suggestions on how to do so.

What's next? (Optional extensions)

Ready to try some additional practices with your new skills?

1. Try these to edit your game from above:

- Before the game begins, let the player enter the upper number (100 in the lessons) to play with. **Input** T and use **randInt**(1,T) instead of **randInt**(1,100).

- ***"Trickster":*** Want to play a trick on the player? After each guess, have the program pick a new random number. Do you think the player will *ever* get the number?

- **"Warm or Cold"**: Guess the number from one to 100 again, but this time, the computer responds with "**Freezing-Cold-Warm-Hot-Boiling**" depending on how close the guess is to the answer. You can decide just what each of those words means in your code, but do not tell the player.

2. **Whitney Wolfe Herd** is an American entrepreneur. She is founder and CEO of Bumble, a popular social and dating app. Bumble launched in 2014 and now the company is now valued at more than *1 billion* dollars.
    a. How much is 1 billion (1,000,000,000)? How long is 1 billion seconds in days and years? Use the calculator. How long will it take to spend 1 billion dollars? Write a program to let the user enter the amount she spends in one day and display 1) how many days and 2) how many years it will take to spend 1 billion dollars.

3. Did you know you can create other "graphical" games on your calculator? Take a peek at the projects **"SNAKE**" or the "**MAZE**" in the Beyond Basics projects to try your hand at a few simple games.

   Warning: These are coded using TI-Basic for theTI-84 Plus family of graphing calculators, but the languages are very similar. The projects are more complex, but would be an excellent challenge!