Overview:

In this activity students will explore inequalities and the number line while writing code to navigate a set of challenges. Students will apply their knowledge of compound inequalities to write programs for Rover to demonstrate the inequalities on the number line. This activity is appropriate for students familiar with graphing inequalities on a number line diagram.

Goals:

Students will:

- use their knowledge of compound inequalities and graphs of compound inequalities to create and edit TI-Nspire CXII Python programs that include several commonly used Rover and calculator commands.

Python Syntax Reference:

Statement	Example	Behavior
import module_name as name_space	`import ti_rover as rv`	Required for all TI Rover Python programs. Imports the ti_rover module into the Python program. The module provides the methods for controlling the Rover. Sets the current position of the Rover as the origin and the heading as 0 degrees measured from the x-axis.
import module_name as name_space	`import ti_plotlib as plt`	Required for the **text_at()** function. Imports the ti_plotlib module into the Python program. The module provides the methods for displaying text and data plots. The import ti_plotlib as plt statement is available from the TI Plotlib menu.
from module_name import *	`from time import *`	Imports all the functions in the **time** module for use in the program. It is necessary to import the time module to use the **sleep()** function. The from time import * statement is available from the More Modules>Time menu.
from module_name import *	`from ti_system import *`	Imports all the functions in the **ti_system** module for use in the program. It is necessary to import the ti_system module to use the **while get_key() !=** "esc" statement. The from ti_system import * statement is available from the More Modules>TI System menu.
rv.forward(distance)	`rv.forward(10)`	Rover drives 10 units forward. The default unit is 10 cm.
rv.backward(distance)	`rv.backward(10)`	Rover drives 10 units backward. The default unit is 10 cm.
rv.forward(distance,"unit",speed,"rate")	`rv.forward(5,"units",1.5,"units/s")`	Rover drives 5 units forward at 1.5 units per second. The default unit is 10 cm. The default speed is 2 units pers second.
rv.stop()	`rv.stop()`	Rover stops. This function is executed as soon as Rover receives it.
rv.color_rgb(red,green,blue)	`rv.color_rgb(255,0,0)`	Turns the color LED on with the color red. Values for the red, green and blue LED components are between 0 (off) and 255 (full power). The rv.color_rgb() function is available from the Rover Outputs menu.

sleep(seconds)	`sleep(3)`	The calculator will wait 3 seconds before moving to the next line in the program. sleep() is available from the TI Rover Commands menu.
var=rv.waypoint_x()	`x=rv.waypoint_x()`	Stores the current x-coordinate position in units of the TI-Rover into the variable *x*. The default unit is 10 cm. rv.waypoint_x() is available from the TI Rover Path menu (look toward the bottom of the menu).
text_at(row,"text","align")	`text_at(3,"x position = "+str(x),"left")`	The **text_at()** function displays a text string on a specified row with an alignment of left, center or right. When variable **x** has a value of 7.6, the following is displayed on row 3, aligned to the left: x position = 7.6 Note: The **str()** function converts a numeric value to a string. The **+** operator is used to join two strings. **str()** is available from the Built-ins> Type menu.
get_key()	`key_pressed=get_key()`	get_key() is a function that returns a string with the value associated with the last key pressed while a program is running. The value of the escape key is "esc". In the example, pressing the escape key updates the variable **key_pressed** to "esc".

while get_key() != "esc":
block
	`while get_key() != "esc":`
	`x=rv.waypoint_x()`
	`text_at(3,"x position ="+str(x),"left")`
		Defines a while loop that will continue until the escape key is pressed.
		While loops repeat the statements in the block if the condition at the top of the loop is true. In the example, looping continues until the escape key is pressed. Not pressing a key or pressing any key but escape means that get_key() will return a value that is not equal to "esc". The loop condition is true and looping continues. If the escape key is pressed, get_key() returns "esc". The condition will evaluate as "esc" not equal to "esc", which is false. A false result means that the loop statements are not repeated. Program execution skips to the statement just after the loop. Note: The block starts with a **colon** and includes the indented lines that follow. while get_key() != "esc": is available from the TI Hub > Commands menu.

Python Syntax Reference (continued):

Statement	Example	Behavior

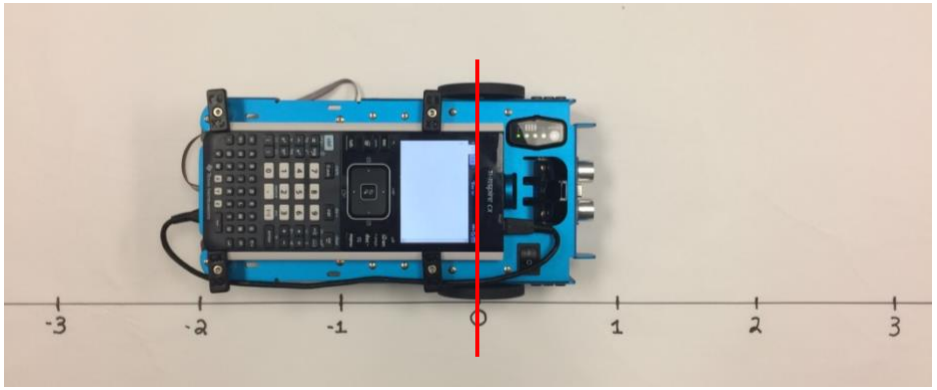
<Boolean expression>
value 1 operator value 2
	`2+3==6 (result is false)`
	
	`x+4>=y (if x=1 and y=3, the result is true)`
	
	`"enter"!="esc" (result is true)`
		Boolean expressions evaluate to either true or false. The examples show some of the relational operators available from the Built-ins Ops menu.
		Note: == is the Python operator to check equality. >= is the Python operator to check whether the value to the left is greater than or equal to the value on the right. != is the Python operator to check inequality.
		Boolean operators are available from the Built-in Ops menu or from the menu brought up by pressing ctrl [=] on the TI-Nspire keyboard.

if <Boolean expression>: block	`if 0<x<2:` `  rv.color.rgb(255,0,0)`	Checks to determine if the value of variable **x** is between 0 and 2. If the statement is "true" then the statements in the **if** block are executed. Otherwise, the block is skipped. In the example, when the value for the variable x is between 0 and 2, the calculator will send a command to the TI-Innovator to set the color rgb LED to be red.
if <Boolean expression> and <Boolean expression>: block	`If x>=2 and x<4:` `  rv.color.rgb(0,255,0)`	If both expressions are true the **and** function is "true", then the block is executed. Otherwise, the **and** function returns false, and the block is skipped. In the example, when the value for x is greater than or equal to 2 and less than 4, the calculator will send a command to the TI-Innovator to set the color rgb LED to be green.

For more on programming Rover with TI-Nspire CXII follow the links to the TI Rover Menu Map: TI-Nspire™ Python Programming > Python Menu Map > TI Rover Menu

Setup:

Students may work in groups of two or three. An example number line layout is below. Use this for the position of Rover. Rover's position corresponds to the center of the axle as shown by the red segment below. 1 Rover unit is 10 cm. Mark with 10 cm units to match with Rover units, from -10 units to +10 units.



Materials:

- TI-Nspire™ CXII graphing calculator
- Calculator unit-to-unit cable (USB mini A to USB mini B cable)
- TI-Innovator™ Hub
- TI-Innovator™ Rover
- Number line for position. Use a 2 meter course, from -10 units to +10 units. 1 Rover unit is 10 cm.
- Student "Challenge" cards have been provided and can be printed/ cut into individual task cards, and distributed to students (optional)

Student Activity:

Sit in small groups with your calculator and supplies for this activity. Practice the guidance modeled by your teacher.

Teacher Notes:

Review and introduce the calculator, Hub, and Rover commands needed for this activity. In preparation for the coding on this activity, refer to Unit 1 Skill Builder 1 of the 10 minutes of code activities and Unit 4 Skill Builder 1 of the 10 minutes of code with TI-Innovator activities.

- Start a new program
- Attach Rover

Challenge 1: Use the rv.color_rgb() function to explore using the color LED. Try to find RGB values for the primary and secondary colors.

e.g. rv.color_rgb(255,155,0) will make yellow.

Teacher Guidance during Challenge 1:

- Students may have different values for mixing colors. In this activity we will primarily use the red, green and blue components individually but this challenge allows them a chance to explore settings for other colors.

- Example Program: **color.py**

```
import ti_rover as rv
rv.color_rgb(255,155,0)
```

Challenge 2: Use the text_at() function to display your name at several locations on the screen.

Teacher Guidance during Challenge 2:

- Use the text_at function from the TI Rover>Commands menu. text_at() takes at least three inputs (also known as “arguments”). The first input is the row number to display on. The second input is the text string, value, variable, etc. to display. The third input is the alignment of the text: left, center or right.

- Example program: **name.py**

```
import ti_rover as rv
import ti_plotlib as plt
plt.text_at(3,"Hello, my name is: ","left")
plt.text_at(4,"Rover","left")
```

Challenge 3: Have Rover drive 5 units forward. Use the rv.waypoint_x() function to read and display Rover's horizontal position when Rover is finished driving.

Teacher Guidance during Challenge 3:

- rv.waypoint_x() reads Rover's current x position in Rover units, relative to the origin. rv.waypoint_x() is found on the TI Rover path menu. In this program the position reading is stored to the variable *x*.
    - Note: When **import ti_rover as rv** is executed, along with importing Rover functions and establishing communication between the calculator and Rover, it also sets the current position to (0,0) and the heading to 0 degrees (pointing down the positive x-axis). For this activity we are ignoring the Rover's y position.
- It will be helpful for students to think about how long it will take Rover to drive forward 5 units and use a sleep() function to wait until Rover has stopped driving before reading Rover's position.
    - Note: Rover's default speed is 2 units per second. Students will need to make note of this in order to determine how long to wait.
    - Students may also notice a roughly 1 second delay between when the program is executed and Rover starts. Students should account for this time in their sleep() statement before reading the Rover's position.
- Have students note there may be some variability in the position that is measured.
- Example Program: **position.py**

```
import ti_rover as rv
from time import *
import ti_plotlib as plt
rv.forward(5)
sleep(3.5)
x=rv.waypoint_x()
plt.text_at(3,"position= "+str(x),"left")
```

- *Discussion Starters*
    - The following are suggested discussion starters to engage your students with the mathematics inherent in Challenge 3:
        - How long did you wait to read the position of Rover? Explain how you determined how long to wait before reading the position of Rover?
        - If you changed Rover's speed to 1.5 units per second, explain how this would change the amount of time you would need to wait before reading the distance Rover traveled to be sure it was finished driving. Write and solve an equation for the wait time.

Challenge 4: Use a While loop to turn on the LED red, then green, then blue each for 1 second until the escape key is pressed.

Teacher Guidance during Challenge 4:

- Loops are used to repeat a set of statements. A While loop repeats a set of statements as long as a specified condition is "true". The loop block starts with a colon and includes the indented lines that follow.
  - This while loop in will execute as long as the escape key is not pressed.
  - The get_key() function returns the name of the last pressed key as a text string. Pressing the escape key causes get_key() to return the string "esc". The While loop Boolean operator compares the value returned by the get_key() function against the text string "esc". **!=** is the not equal operator in Python.
  - If the get_key() value is not "esc" the statement is true and the While loop proceeds.
  - If the get_key() value is "esc", meaning that the escape key has been pressed, the statement is false and the program skips to the next statement after the end of the While loop. In this case the next statement is rv.color(0,0,0), which turns of the Rover RGB.
- **while get_key() != "esc":** is available from the TI Hub > Commands menu.
- It is necessary to import the ti_system module to use the **while get_key() != "esc"** statement. The from ti_system import * statement is available from the More Modules>TI System menu.

- Example Program: **colors.py**

```
import ti_rover as rv
from time import *
from ti_system import *

while get_key()!= "esc":
 rv.color_rgb(255,0,0)
 sleep(1)
 rv.color_rgb(0,255,0)
 sleep(1)
 rv.color_rgb(0,0,255)
 sleep(1)
rv.color_rgb(0,0,0)
```

Challenge 5: Have Rover drive 5 units forward. Predict the amount of time for Rover to reach 4 units and read Rover's position at that time. If the value returned is equal to 4 turn the LED green, if the value returned is less than 4 turn the LED red, and if the value is greater than 4, turn the LED blue.

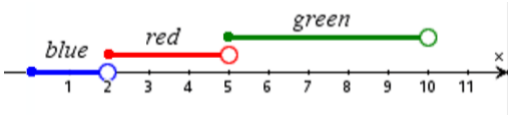
Teacher Guidance during Challenge 5:

- This challenge will require students to edit their program **position2** to add conditional statements that will control the RGB LED.
- To save time you can create a copy of the **position** program from Challenge 3 to build on.
  - Create the copy by going back to the edit page for position by using ctrl left arrow repeatedly until you arrive at the correct page. Press the Menu key followed by Actions>Create Copy. Name your program **position2**.
- The comparison operators, ==, <=, >=, etc. can be accessed from a menu by pressing /=.
  - Note: == is the Python comparison operator for equals.
- if blocks will only evaluate the contained statements if the condition given after if is "true". If the statement is "false", then the block is skipped.
  - Note: The if block starts with a colon and includes the indented lines that follow.
- Students will enter their predicted time to travel 4 units as the input to sleep() function. The program will read the value for the x coordinate immediately after the sleep functions completes.
- Note: Entering a value the time to drive exactly 4 units will be nearly impossible. This gives an opportunity for discussion about tolerance and being at an exact point. Since Rover measures its position relative to the start, and the code is looking for that position to be equal to 4, green will almost never appear.
- Example Program: **position2.py**

```
import ti_rover as rv
from time import *
import ti_plotlib as plt
rv.forward(5)
sleep(3.5)
x=rv.waypoint_x()
plt.text_at(3,"position= "+str(x),"left")
if x==4:
  rv.color_rgb(0,255,0)
if x<4:
  rv.color_rgb(255,0,0)
if x>4:
  rv.color_rgb(0,0,255)
```

- *Discussion Starters*
  - The following are suggested discussion starters to engage your students with the mathematics inherent in Challenge 5:
    - If your LED turns blue, what adjustment can you make to your program to make it turn green or red? Explain.
    - If Binu's LED is red after his drive and Jisha's LED is blue, what can you say about the time when Rover is at 4 units from zero? Explain.
    - Do you expect to see the LED turn green? Why or Why not?

Challenge 6: Have Rover drive on the number line between 0 and 10. While Rover is driving, read its position and control the LED so that the LED displays colors corresponding to the number line diagram below.



Teacher Guidance during Challenge 6:

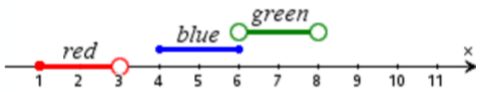
- For this challenge, students will benefit from going through the discussion starters before attempting to write the code.
- This challenge will require students putting Rover in motion using rv.forward() and rv.backward() functions and then using the While loop to monitor Rover's position while it drives.
  - The While loop should have a Boolean expression looking for a key press similar to challenge 4.
  - If students decide to drive to numbers less than zero or greater than 10 units, they will need to include If..Then.. conditions for those points as well. The example program does not address driving outside of the domain 0 to 10.
- Students may want to use compound inequalities of the form 5<x<10 for the If..Then.. conditions. This will work. However, students may benefit from using the "and" operator and considering the logic inherent in the statement 5<x<10. This is equivalent to 5<x and x<10. Both statements will work. For this solution note the "and" operator is shown.
- Students will need to remember to press the escape key to stop the program.
- Example Program: **inequal1.py**

```
import ti_rover as rv
import ti_plotlib as plt
from ti_system import *
x=0
rv.color_rgb(0,0,255)
rv.forward(10)
rv.backward(10)
plt.text_at(3,"press escape key to exit","left")
while get_key()!= "esc":
  x=rv.waypoint_x()
  if 0<=x and x<2:
    rv.color_rgb(0,0,255)
  if 2<=x and x<5:
    rv.color_rgb(255,0,0)
  if 5<=x and x<10:
    rv.color_rgb(0,255,0)
rv.color_rgb(0,0,0)
```

```
rv.stop()
```

- *Discussion Starters*
  - The following are suggested discussion starters to engage your students with the mathematics inherent in Challenge 6:
    - For what values of *x* will you turn on the RGB LED? Write an inequality to represent your answer.
    - Write three inequalities defining the boundaries for each color.
    - If the x position of Rover is 5, what color should the RGB LED be? Explain how you know.
    - Maria says that a command to turn off the LED's should be included for when x positon is greater than or equal to 10 units, explain why you agree or disagree with her statement.
    - For which values of x should the LED be off? Write your answer as an inequality statement or statements and then write the If..Then.. blocks that could be added to your code to accommodate driving outside of $0 \le x < 10$.
    - If Rover starts at x position 0, should the LED be on? If so, what color? Explain.

Challenge 7: Have Rover drive on the number line between 0 and 10. While Rover is driving, read its position and control the LED so that the LED displays colors corresponding to the number line diagram below.



Teacher Guidance during Challenge 7:

- For this challenge, students will benefit from going through the discussion starters before attempting to write the code.
- Note: Students could make use of the "or" operator for when the LED should be off. An alternative version of the program is listed after the discussion starters.

- Example Program: **inequal2.py**

```
import ti_rover as rv
import ti_plotlib as plt
from ti_system import *
x=0
rv.color_rgb(0,0,0)
rv.forward(10)
rv.backward(10)
plt.text_at(3,"press escape key to exit","left")
while get_key()!= "esc":
  x=rv.waypoint_x()
  if 0<=x and x<1:
    rv.color_rgb(0,0,0)
  if 1<=x and x<3:
    rv.color_rgb(255,0,0)
  if 3<=x and x<4:
    rv.color_rgb(0,0,0)
  if 4<=x and x<=6:
    rv.color_rgb(0,0,255)
  if 6<x and x<8:
    rv.color_rgb(0,255,0)
  if 8<=x:
    rv.color_rgb(0,0,0)
rv.color_rgb(0,0,0)
rv.stop()
```

- *Discussion Starters*
  - The following are suggested discussion starters to engage your students with the mathematics inherent in Challenge 7:
    - For what values of *x* will you turn on the RGB LED? Write your answer in inequality statements.
    - For what values of x will the LED be turned off? Write your answer in inequality statements.
    - Write inequalities defining the boundaries for each color.
    - If the x position of Rover is 6, what color will the RGB LED need to be? Explain how you know.
    - Maria says that a command to turn off the LED's should be included for when x positon is greater than or equal to 8 units, explain why you agree or disagree with her statement.
    - If Rover starts at x position 0, should the LED be on? If so, what color? Explain.

- Alternative example program using "or". This example combines all the times when the LED is off into one conditional statement. **inequal2or.py**

```
import ti_rover as rv
import ti_plotlib as plt
from ti_system import *
x=0
rv.color_rgb(0,0,0)
rv.forward(10)
rv.backward(10)
plt.text_at(3,"press escape key to exit","left")
while get_key()!= "esc":
  x=rv.waypoint_x()
  if 0<=x<1 or 3<=x<4 or 8<=x:
    rv.color_rgb(0,0,0)
  if 1<=x and x<3:
    rv.color_rgb(255,0,0)
  if 4<=x and x<=6:
```

```
    rv.color_rgb(0,0,255)
  if 6<x and x<8:
    rv.color_rgb(0,255,0)
rv.color_rgb(0,0,0)
rv.stop()
```

Challenge 8: Have Rover drive on the number line between -10 and 10. While Rover is driving, read its position and control the LED so that the LED displays colors corresponding to the description below.

- While Rover's position is less than or equal to zero, the LED is magenta.
- While Rover's position is greater than 0 and less than 2, the LED is off.
- While Rover's position is greater than or equal to 2 and less than or equal to 4, the LED is red.
- While Rover's position is greater than 4 and less than 5, the LED is blue.
- While Rover's position is greater than or equal to 5 and less than 10, the LED is green.
- While Rover's position is greater than or equal to 10, the LED is yellow.

Teacher Guidance during Challenge 8:

- For this challenge it will be helpful for students to draw and color a number line diagram representing the challenge before translating the challenge into the code.
- Students will need to use rv.backward() to make sure Rover's position can be less than zero.
- As in the previous challenges, this challenge will require students putting Rover in motion using rv.forward() and rv.backward() functions and then, while Rover is moving, students use the While loop to monitor Rover's position while it drives.

- Example Program: **verbal.py**

```
import ti_rover as rv
import ti_plotlib as plt
from ti_system import *
x=0
rv.color_rgb(255,0,255)
rv.backward(5)
rv.forward(16)
plt.text_at(3,"press escape key to exit","left")
while get_key()!= "esc":
  x=rv.waypoint_x()
  if x<=0:
    rv.color_rgb(255,0,255)
  if 0<x and x<2:
    rv.color_rgb(0,0,0)
  if 2<=x and x<=4:
    rv.color_rgb(255,0,0)
  if 4<x and x<5:
    rv.color_rgb(0,0,255)
  if 5<=x and x<10:
    rv.color_rgb(0,255,0)
  if 10<=x:
    rv.color_rgb(255,155,0)
rv.color_rgb(0,0,0)
```

```
rv.stop()
```

- *Discussion Starters*
  - The following are suggested discussion starters to engage your students with the mathematics inherent in Challenge 8:
    - For what values of *x* will you turn on the RGB LED? Write your answer in inequality statements.
    - For which values of *x* will you turn off the RGB LED? Write your answers in inequality statements.
    - Maggie made her Rover drive backward 10 units first, she says the LED should stay magenta until she drives forward 10 units and then it should turn off. Would you agree or disagree? Explain.