TI-Innovator Rover Explorations

The TI-Innovator Rover can be used to learn simple and complex ideas in computer programming. This suite of activities contains 8 activities. Activity 6 is the only activity that requires completing the previous activity. While they do not need to be completed in succession, each subsequent activity is more complex than the previous.

Objectives:

Programming Objectives:

- Use variables to store information.
- Use selection statements to make decisions.
- Use iteration to repeat code.
- Use functions to modularize code.
- Use lists to store related data.
- Use libraries and library functions.

Key AP Computer Science Principles Standards:

- Represent a value with a variable (AAP-1.A)
- Represent a list or string using a variable (AAP-1.C)
- Write conditional Statements (AAP-2.H)
- Write nested conditionals (AAP-2.I)
- Select appropriate libraries or existing code segments to use in creating new programs (AAP-3 D)
- Write iteration statements (AAP-2.K)
- Write expressions that use list indexing and list procedures. (AAP-2.N)
- Write iteration statements to traverse a list. (AAP-2.O)
- Write statements to call procedures (AAP-3.A)
- Develop procedural abstractions to manage complexity in a program by writing procedures. (AAP-3 C)
- For generating random values, write expressions to generate possible values. (AAP-3 E)

This document contains Activities 3 and 4 of the 8 total TI-Innovator Rover activities.

Activity 3: Quiz Race Version 1

Students will create string, integer and float variables.
Students will use selection statements to make decisions.
Students use a while loop to determine when to stop repeating code.
Students will use .lower() to control logic errors.
Students will use fabs() to control roundoff errors.

Activity 4: Regular Polygons

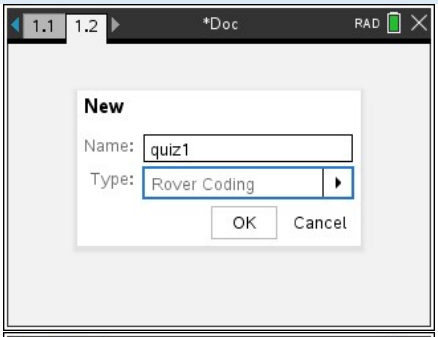
Students will use selection statements to create regular polygons.

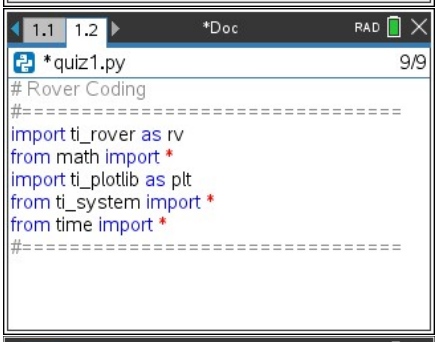
Activity 3: Quiz Race Version 1

Students will create string, integer and float variables.
Students will use selection statements to make decisions.
Students will use the .lower() function to control logic errors.
Students will use fabs() to control roundoff errors.

1. Create a new program named “quiz1”.

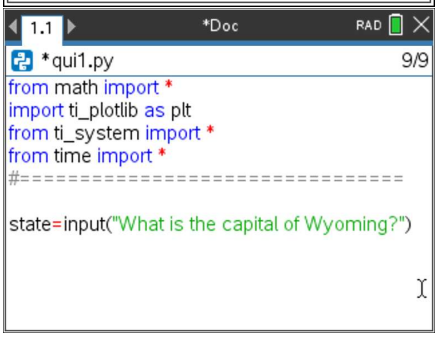
   Choose Rover Coding for the default type.





2. The first question in your quiz game will ask the user to enter the capitol for a given U.S. state. For example, you could ask, “What is the capital of Wyoming?”, or “What is the capital of North Dakota?”. Store the user’s answer in a variable named state.

   By default, your answer will stored as a ***string*** variable. That means the computer will treat the user’s answers as a string of characters.

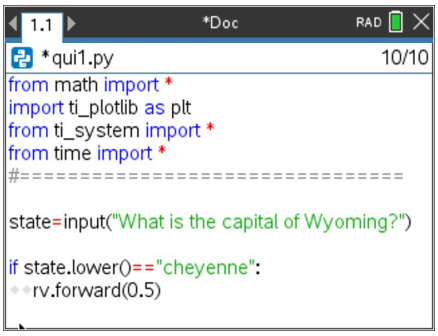


3. If the user answers the question correctly, the TI-Rover should move forward 0.5 meters.

   When comparing string variables, Python is case sensitive.
   That means, if you write the code:

```
if state==”Cheyenne”:
    rv.forward(0.5)
```

   the TI-Rover will move forward only if the user uses a capital C and all other letters in lower case.



   If you use the .lower() function on the state, you can avoid a ***logic error.*** A logic error occurs when the program runs unlike it was intended.

   Add the line

```
    if state.lower()==”cheyenne”:
```

```
rv.forward(0.5)
```

Because the answer, Cheyenne, is string, it has to be in quotes. Notice when you put quotes around “cheyenne”, the calculator changes the text green to show it is string.

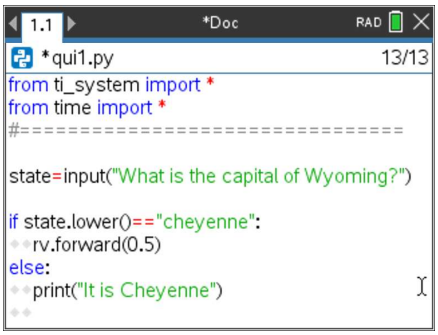
4. Run your program several times.

   Does it move forward only when you enter the correct answer?

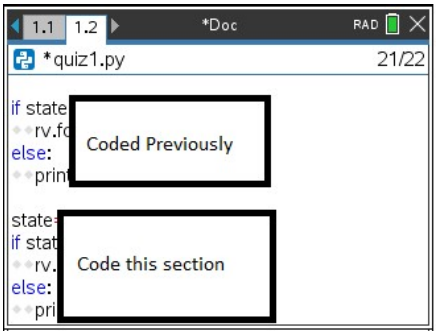
   Does it work if you enter your answer with and out without capital letters?

5. Shouldn’t your program tell the user the correct answer if the answer was incorrect?

   Add an else statement that tells the user the correct answer if the wrong answer was entered.



6. Add another question that requires string input as an answer.
   If the user is correct, move forward 0.5, otherwise give the correct answer.

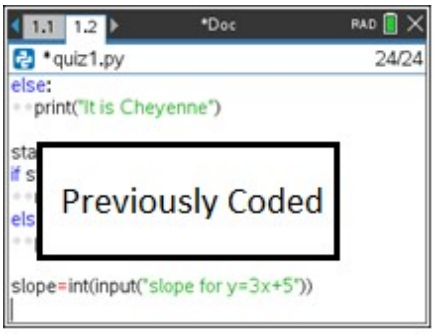


7. In this step, you’ll ask a question that requires an ***integer*** answer.

   Ask the user for the slope of the line in the equation y=3x+5.

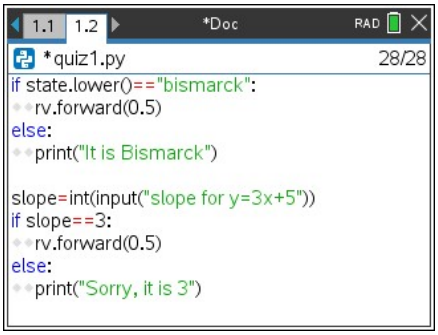
   Because the answer is an ***integer*** you need to put int() around the input(), otherwise Python will treat the answer like a string.

   slope=int(input(“slope for y=3x+5”))



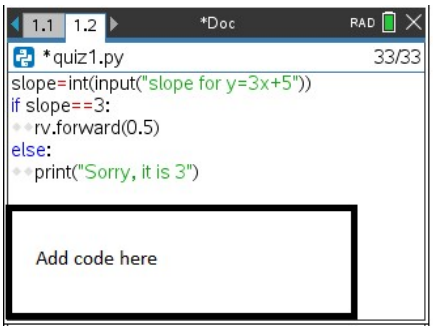
8. Because the answer 3 is not a string, you will not put quotes around it like you did the state questions before.

   Add the lines:
   ```
   if slope == 3:
     rv.forward(0.5)
   else:
     print(“Sorry, it is 3”)
   ```



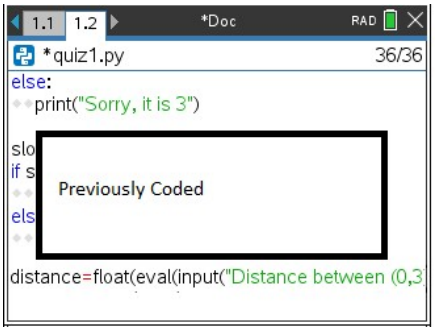
9. Execute your code. Make sure all three sections work correctly.

10. Add another question with an integer solution.



11. Did you know you can let the user enter a math expression such as sqrt(18) or sqrt(3**2+3**2)?

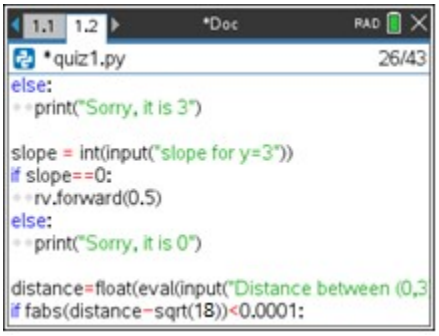
    if you type:
    distance = float(eval(input(“Distance between (0,3) and (3,0)”)))
    the computer will evaluate the input and store it as a ***float*** variable.



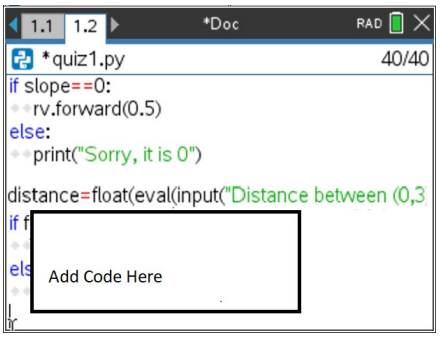
12. Using the eval() function might lead to a ***round-off error*** since float variables have limited precision. Since sqrt(18) mathematically is irrational, it continues forever when written as a decimal, it will be rounded when stored in a variable.

    To account for round-off errors, we will NOT write the code as
    ```
    if distance==sqrt(18):
    ```

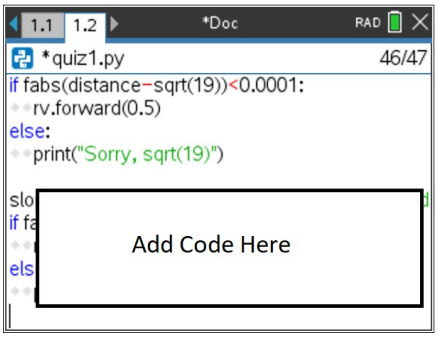
    Instead, we will write the code as:
    ```
    if fabs(distance-sqrt(18)) < 0.0001:
    ```
    This says, if the absolute value of the difference between the answer and √18 is very small, less than 0.0001, the user’s information was correct. The fabs function is located under the math menu.



13. Add the lines of code to move the rv forward 0.5 meters if the distance is correct, otherwise, print “Sorry, sqrt(18)”.



14. Add a question that asks the user to enter the slope between the points (7,10) and (8, 13). If the code is correct, move forward 0.5 meters. Otherwise print the correct answer.



15. Execute your code. Play several times to ensure it functions properly.

Teacher Notes:

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================

state=input("What is the capital of Wyoming?")

if state.lower()=="cheyenne":
  rv.forward(0.5)
else:
  print("It is Cheyenne")

state=input("What is the capital of North Dakota"):
if state.lower()=="bismarck":
  rv.forward(0.5)
else:
  print("It is Bismarck")
```

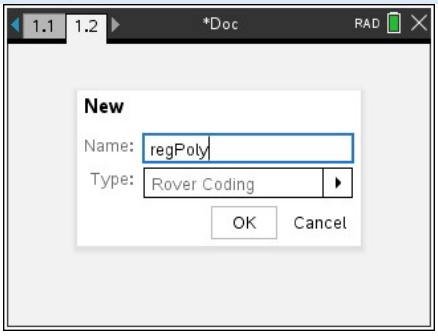
```
slope=int(input("slope for y=3x+5"))
if slope==3:
  rv.forward(0.5)
else:
  print("Sorry, it is 3")

slope = int(input("slope for y=3"))
if slope==0:
  rv.forward(0.5)
else:
  print("Sorry, it is 0")

distance=float(eval(input("Distance between (0,3) and (3,0)")))
if fabs(distance-sqrt(18))<0.0001:
  rv.forward(0.5)
else:
  print("Sorry, sqrt(18)")

slope=float(eval(input("Slope between (7,10) and (8,13)")))
if fabs(slope-1/3)<0.0001:
  rv.forward(0.5)
else:
  print("Sorry, 1/3")
```

Possible Extensions:
Have students write their own games. Give students a set number of questions. Require students provide some string, some integer and some float answers. If possible, have students swap code and play each other's games.

Activity 4: Regular Polygons
Students will use section statements to create regular polygons.

1. Create a new program named "regPoly".

   Choose Rover Coding for the default type.



2. In this project, the user will determine if the rover drives a triangle, square, pentagon or hexagon. The user will enter a number between 3 and 6. If the user enters a valid integer, the rover will draw the appropriate shape.

Otherwise, the rover will say “Invalid” and stay in the same spot.

How many degrees must you turn for each shape?
Complete the table below.

	Triangle	Square	Pentagon	Hexagon
Number Of Degrees				

3. Write a line of code that uses the input() function to ask the user for a number from 3 to 6.

   The number of sides will be a whole number. Therefore, use an *integer* variable to store the value. Store the entered value as an ***integer*** in a variable named **sides**.

4. Write a line of code that asks the user for the side length of the regular polygon.

   The length of the side can be an integer such as 1, or a rational number such as 0.75. Therefore, use a *float* variable to store the value. Store this value as a ***float*** named **length**.

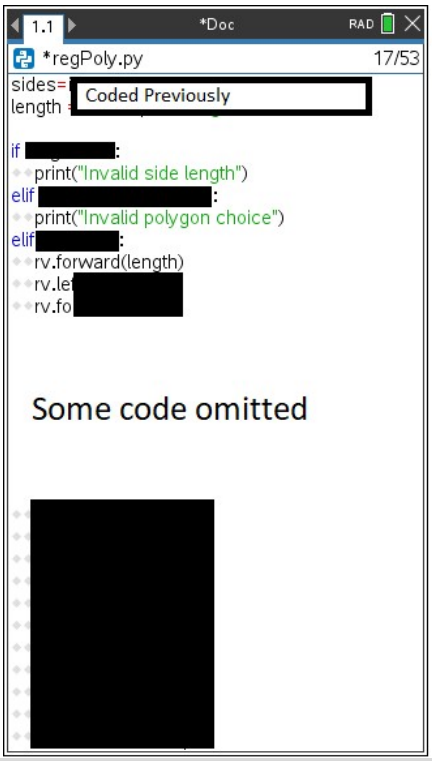
5. Now to write the if statement. The ***pseudocode*** for the if statement is:

```
if side length is less than or equal to zero:
    print(“Invalid side length”)
elif the user enters a number less than three or more than six:
    print(“Invalid polygon choice”)
elif side length equals three:
    draw a triangle
elif side length equals four:
    draw a square
elif side length equals five:
    draw a pentagon
else:
    draw a hexagon
```

6. Use the pseudocode from above to guide your code for the *selection* statement.

   Remember, in Python to write less than or equal to use <=
   To see if two values are equal, use ==.

   Make sure each section of the selection is indented two spaces.



Teacher Notes:

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
sides=int(input("Number of sides: "))
length = float(input("Length of sides: "))

if length <=0:
  print("Invalid side length")
elif sides<3 or sides > 6:
  print("Invalid polygon choice")
elif sides==3:
  rv.forward(length)
  rv.left(120)
  rv.forward(length)
  rv.left(120)
  rv.forward(length)
elif sides==4:
  rv.forward(length)
  rv.left(90)
  rv.forward(length)
  rv.left(90)
  rv.forward(length)
  rv.left(90)
```

```
  rv.forward(length)
elif sides==5:
  rv.forward(length)
  rv.left(72)
  rv.forward(length)
  rv.left(72)
  rv.forward(length)
  rv.left(72)
  rv.forward(length)
  rv.left(72)
  rv.forward(length)
else:
  rv.forward(length)
  rv.left(60)
  rv.forward(length)
  rv.left(60)
  rv.forward(length)
  rv.left(60)
  rv.forward(length)
  rv.left(60)
  rv.forward(length)
  rv.left(60)
  rv.forward(length)
```