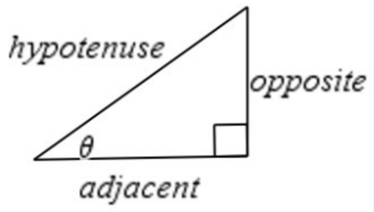
GETTIN' TRIG-Y WITH PYTHON: Trigonometry Ratios

In this lesson, you will write a program that generates a random sine, cosine or tangent ratio. It will ask the user to enter a different trigonometric ratio based on the given triangle information. It will evaluate the user's input. If the user's input is incorrect, give the correct ratio.



$$\sin(\theta) = \frac{\text{opposite}}{\text{hypotenuse}}$$

$$\cos(\theta) = \frac{\text{adjacent}}{\text{hypotenuse}}$$

$$\tan(\theta) = \frac{\text{opposite}}{\text{adjacent}}$$

Objectives:

Programming Objectives:

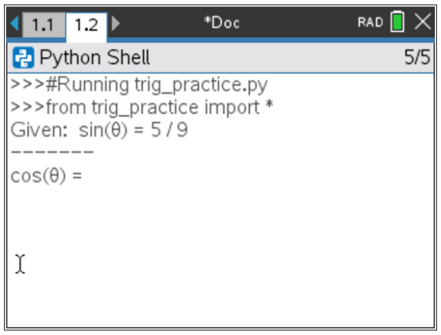
- Use the input function and a variable to collect and store data from a user
- Use the **randint** function to generate triangle side lengths
- Use the **randint** function to determine sine, cosine, or tangent prompts
- Use **if..elif..else** statements to make decisions

Math Objectives:

- Use the Pythagorean Theorem to find missing triangle side lengths
- Use right triangle trigonometry ratios to solve problems

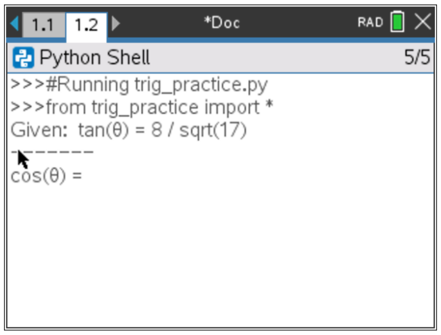
For this project, you will write a program that generates a random sine, cosine or tangent ratio. It will ask the user to enter a different trigonometric ratio based on the given triangle information. It will evaluate the user's input. If the user's input is incorrect, give the correct ratio.

Problem 1 Prompt



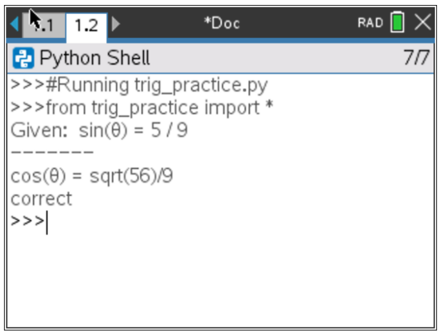
```
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  sin(θ) = 5 / 9
-------
cos(θ) =
```

Problem 2 Prompt



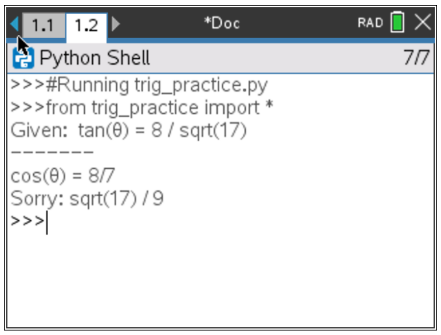
```
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  tan(θ) = 8 / sqrt(17)
-------
cos(θ) =
```

Correnct answer was given.



```
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  sin(θ) = 5 / 9
-------
cos(θ) = sqrt(56)/9
correct
>>>
```

Incorrect answer entered, gives the correct answer.

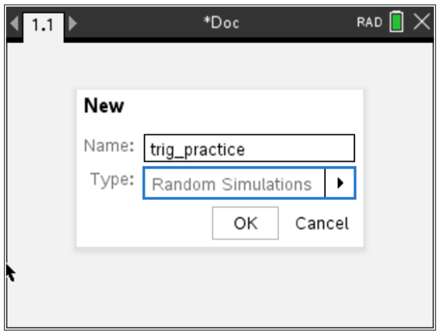


```
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  tan(θ) = 8 / sqrt(17)
-------
cos(θ) = 8/7
Sorry: sqrt(17) / 9
>>>
```

1. Start a new Python project. Name the project **trig_practice**.

   You are not allowed to use a space in the name of a project. Often, programmers will use an underscore to separate words variables and project names. They also use something called camel case, which does not include an underscore, instead it requires capitalizing the first letter of each word after the first word. In this case, trigPractice could be used to name the file.

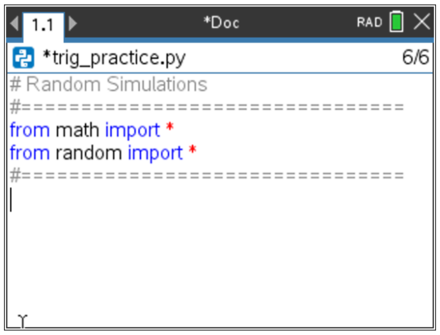
   This project will utilize the randint function to generate random integers. This requires the random library. Select **Random Simulations** as the type.



Name: trig_practice
Type: Random Simulations

2. Make sure the random library was imported.

   If you forgot to select **Random Sampling**, you can go to
   Menu → Math → from math import*
   Menu → Random → from random import*



```
# Random Simulations
#================================
from math import *
from random import *
#================================
```

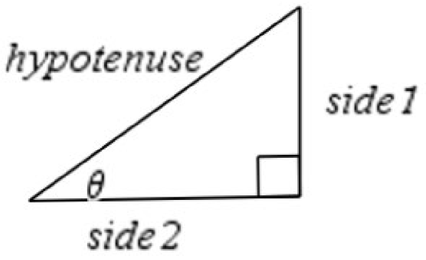
   The comments, denoted by the # appear in light gray. These lines of code are optional. They do not affect how the program executes. Instead, they are notes to you, the programmer.

   Comments are used to describe sections of code. This can be useful when debugging or modifying a program later. They help a programmer remember the purpose for different sections of code. These comments tell you the coder; this library is used for data sharing.

3. The project will generate two integer sides for a right triangle.
   It will calculate the missing side.

   What is the name of the formula than can be used to find the missing side of a right triangle?

   What is the formula that can be used to find a missing side of a triangle given two sides?



4. Using the Pythagorean theorem, $(\text{side}_1)^2 + (\text{side}_2)^2 = \text{hypotenuse}^2$, you will create random right triangle side lengths.

   To mix up the measurements, 1/3 of the time the sides will be integers, 1/3 of the time side1 and the hypotenuse will be integers and the remaining time side2 and the hypotenuse will be integers.

   You will use an if..elif..else statement to make find the three options.

   ```
   If randint(1,3) == 1:
       #generate two integer side lengths
       #calculate the hypotenuse

   elif randint(1,2) == 1:
       #generate opposite side and hypotenuse integers
       #calculate the adjacent side

   else:
       #generate adjacent side and hypotenuse integers
      #calculate the opposite side
   ```

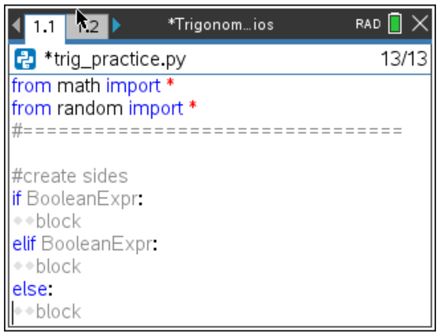
5. Add a comment to document this section of code.

   ctrl → t

   #create sides

   Add the if statement for the first option

   Menu> Built-ins> Control> If..elif..else



1.1 1.2 *Trigonom…ios RAD

*trig_practice.py 13/13

```
from math import *
from random import *
#===============================

#create sides
if BooleanExpr:
••block
elif BooleanExpr:
••block
else:
••block
```

6. The function randint(1,3) will generate numbers 1,2 and 3.
   The function randint(1,2) will generate numbers 1 and 2.

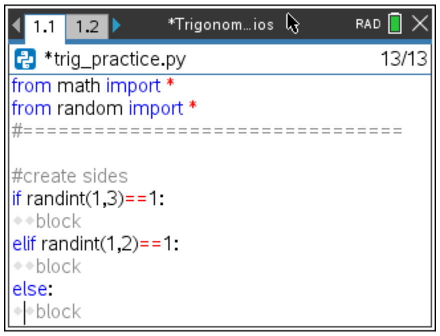
   Inside the first if statement, type

   randint(1,3) == 1

   Inside the elif statement type

   randint(1,2) == 1

   Menu> Random> randint



1.1 1.2 *Trigonom…ios RAD

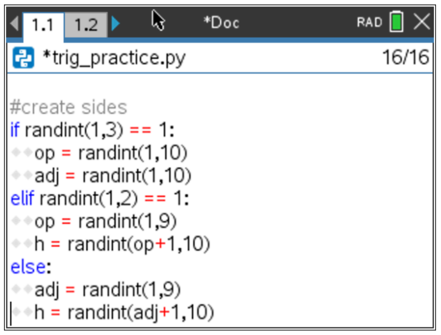
*trig_practice.py 13/13

```
from math import *
from random import *
#===============================

#create sides
if randint(1,3)==1:
••block
elif randint(1,2)==1:
••block
else:
••block
```

7. Use **randint** to generate integer side lengths from 1 to 10.
   In the if statement generate the opposite (op) and adjacent (adj).

   In the elif statement generate the hypotenuse (h) and opposite (op).
   The hypotenuse needs to be larger than the opposite side.
   Therefore, start the hypotenuse options one more than op.

   In the else statement generate the hypotenuse (h) and the adjacent (adj).
   The hypotenuse needs to be larger than the adjacent side.
   Therefore, start the hypotenuse options one more than adj.

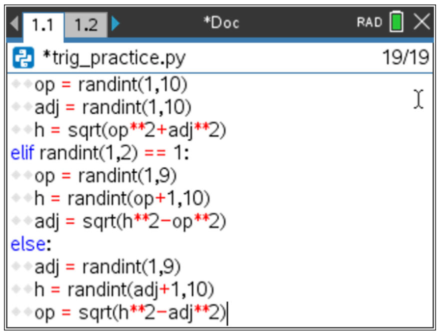


```
#create sides
if randint(1,3) == 1:
  op = randint(1,10)
  adj = randint(1,10)
elif randint(1,2) == 1:
  op = randint(1,9)
  h = randint(op+1,10)
else:
  adj = randint(1,9)
  h = randint(adj+1,10)
```

8. Use the Pythagorean theorem,
   $(\text{opposite})^2 + (\text{side}_2)^2 = \text{hypotenuse}^2$
   to find the missing side length for each part of the if..elif..else.

   To take the square root, type sqrt() or Menu → Math → Sqrt

   To square each side you can multiple the side by itself, such as
   adj*adj
   or use **2 to square such as
   adj**2



```
  op = randint(1,10)
  adj = randint(1,10)
  h = sqrt(op**2+adj**2)
elif randint(1,2) == 1:
  op = randint(1,9)
  h = randint(op+1,10)
  adj = sqrt(h**2−op**2)
else:
  adj = randint(1,9)
  h = randint(adj+1,10)
  op = sqrt(h**2−adj**2)
```

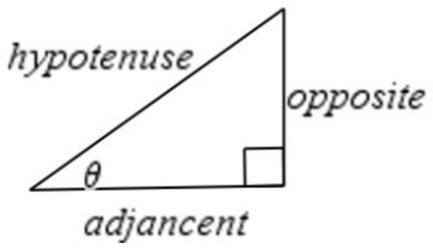
9. Double check your equations.

   Did you use:

   h = sqrt(op**2 + adj**2)

   op = sqrt(h**2 – adj**2)

   adj = sqrt(h**2 – op**2)



10. Right now the values for the side lengths are stored as integers.
    For display purposes we will need a string copy.

If the length is an integer, we will simply put str() around the value and save it under a different name.

For example, in the first if statement you found and stored the value of the adjacent side using the formula:

```
adj=randint(1,10)
```

To make a display string you will type

```
adj1=str(adj)
```

If the length is a number with a square root, it most likely is an irrational number. Instead of displaying the value as an evaluated decimal value, we will get a little fancy. We will put the words "sqrt(" in front of the non-evaluated number.

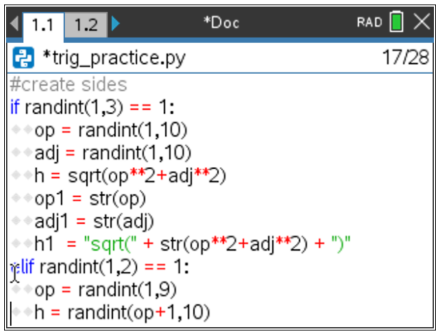
For example, in the first if statement you found and stored the value of the hypotenuse using the formula:

```
h=sqrt(op**2+adj*2)
```

To make a display value you will type:

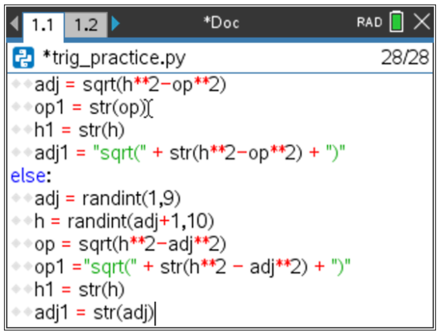
```
h1 = "sqrt(" + str(op**2+adj**2) + ")"
```

```
if randint(1,3)==1:
   op=randint(1,10)
   adj=randint(1,10)
   h=sqrt(op**2+adj*2)
   op1=str(op)
   adj1=str(adj)
   h1="sqr("+str(op**2+adj**2)+")"
elif randint(1,2)==1:
   h=randint(1,10)
   op=randint(1,10)
   adj=sqrt(h**2-op**2)
   h1=str(h)
   op1=str(op)
   adj1="sqr("+str(h**2-op**2)+")"
else:
   h=randint(1,10)
   adj=randint(1,10)
   op=sqrt(h**2-op**2)
   h1=str(h)
   adj1=str(adj)
   op1="sqr("+str(h**2-op**2)+")"
```



*trig_practice.py 17/28

```
#create sides
if randint(1,3) == 1:
  op = randint(1,10)
  adj = randint(1,10)
  h = sqrt(op**2+adj**2)
  op1 = str(op)
  adj1 = str(adj)
  h1  = "sqrt(" + str(op**2+adj**2) + ")"
elif randint(1,2) == 1:
  op = randint(1,9)
  h = randint(op+1,10)
```



*trig_practice.py 28/28

```
  adj = sqrt(h**2−op**2)
  op1 = str(op)
  h1 = str(h)
  adj1 = "sqrt(" + str(h**2−op**2) + ")"
else:
  adj = randint(1,9)
  h = randint(adj+1,10)
  op = sqrt(h**2−adj**2)
  op1 ="sqrt(" + str(h**2 − adj**2) + ")"
  h1 = str(h)
  adj1 = str(adj)
```

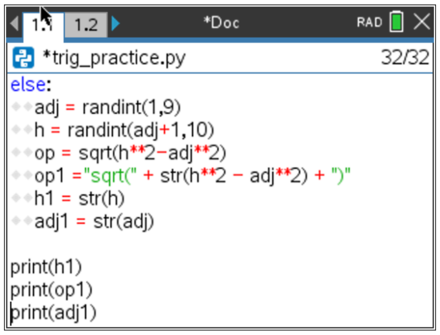
11. Let's check your code so far. In large projects such as this one, programmers execute the code often to check for errors.

    Add three print lines to display the three display values.
    Make sure the print lines are **NOT INDENTED**.
    Menu → Built-ins → I/O → print

    print(h1)
    print(op1)
    print(adj1)

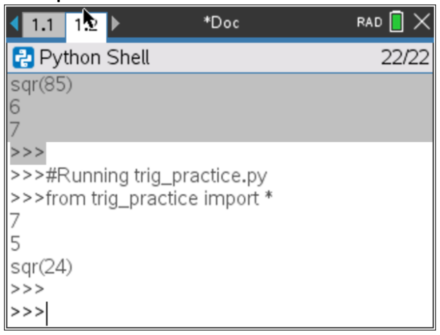
    Execute the program multiple times. Ctrl → r

    Each time only one value should have sqrt() displayed.



```
*trig_practice.py                                    32/32
else:
    adj = randint(1,9)
    h = randint(adj+1,10)
    op = sqrt(h**2-adj**2)
    op1 ="sqrt(" + str(h**2 - adj**2) + ")"
    h1 = str(h)
    adj1 = str(adj)

print(h1)
print(op1)
print(adj1)
```

Sample Runs:



```
Python Shell                                         22/22
sqr(85)
6
7
>>>
>>>#Running trig_practice.py
>>>from trig_practice import *
7
5
sqr(24)
>>>
>>>|
```

The first time h1 is a square root.

The second time adj1 is a square root.

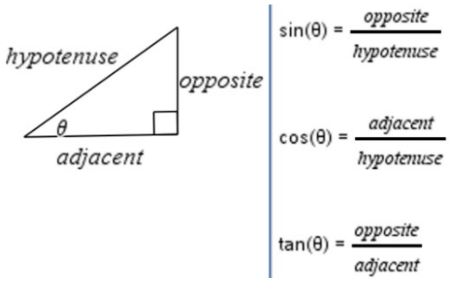
12. Remove the three print lines.

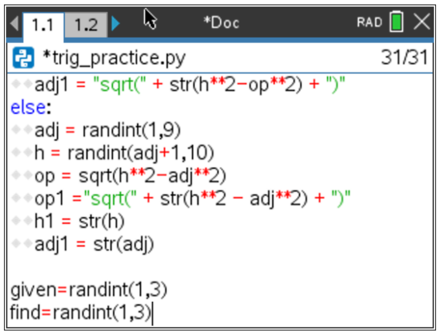
    This app should let you practice three different trig functions.

    For example, it might give
    the ratio for sin(θ) and ask the user to find the ratio for tan(θ)
    or
    the ratio for cos(θ) and ask the user to find the ratio for sin(θ)

    There are three choices for the given function and three for the function to find.

    given = randint(1,3)
    find = randint(1,3)


hypotenuse
opposite
θ
adjacent
sin(θ) = opposite / hypotenuse
cos(θ) = adjacent / hypotenuse
tan(θ) = opposite / adjacent



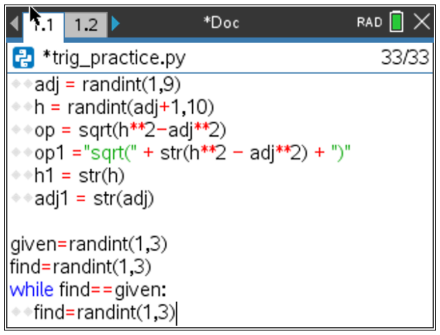
```
*trig_practice.py                                    31/31
    adj1 = "sqrt(" + str(h**2-op**2) + ")"
else:
    adj = randint(1,9)
    h = randint(adj+1,10)
    op = sqrt(h**2-adj**2)
    op1 ="sqrt(" + str(h**2 - adj**2) + ")"
    h1 = str(h)
    adj1 = str(adj)

given=randint(1,3)
find=randint(1,3)
```

13. It is highly likely (1/3) that the variable to find matches the given variable.

    To fix this, add a while loop that will continually generate the find until while the find matches the given.

```
while find == given:
    find=randint(1,3)
```

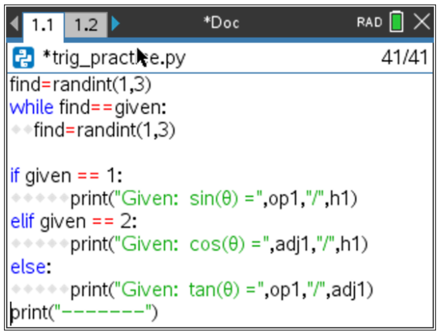
Menu → Built-ins → Control → while
Make sure to use == to check if they match.



```
1.1  1.2   *Doc   RAD
*trig_practice.py                 33/33
  adj = randint(1,9)
  h = randint(adj+1,10)
  op = sqrt(h**2-adj**2)
  op1 ="sqrt(" + str(h**2 - adj**2) + ")"
  h1 = str(h)
  adj1 = str(adj)

given=randint(1,3)
find=randint(1,3)
while find==given:
  find=randint(1,3)
```

14. If the given is a 1, display the ratio for sin(θ).
    If the given is a 2, display the ratio for cos(θ).
    If the given is a 3, display the ratio for tan(θ).
    Outside the if statement print "-----------"

```
if given == 1:
    print("Given:  sin(θ) =",op1,"/",h1)
elif given == 2:
    print("Given:  cos(θ) =",adj1,"/",h1)
else:
    print("Given:  tan(θ) =",op1,"/",adj1)
print("-------")
```

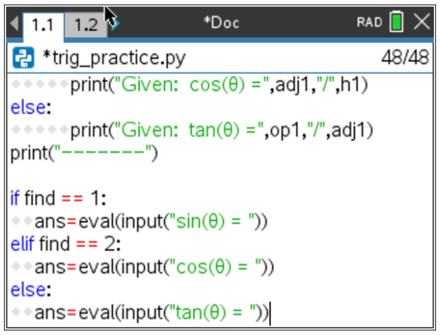


```
1.1  1.2   *Doc   RAD
*trig_practice.py                 41/41
find=randint(1,3)
while find==given:
  find=randint(1,3)

if given == 1:
    print("Given:  sin(θ) =",op1,"/",h1)
elif given == 2:
    print("Given:  cos(θ) =",adj1,"/",h1)
else:
    print("Given:  tan(θ) =",op1,"/",adj1)
print("-------")
```

15. A similar if..elif..else statement will print the ration to find, retrieve the user's result **AND** determine if the user is correct.

    To start let's display the question.

```
if find == 1:
   ans=eval(input("sin(θ) = "))
elif find == 2:
    ans=eval(input("cos(θ) = "))
else:
       ans=eval(input("tan(θ) = "))
```



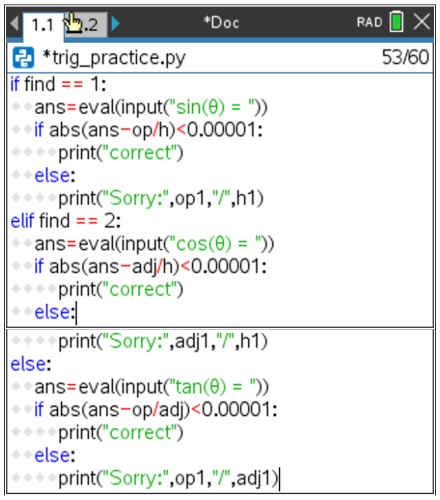
```
1.1  1.2   *Doc   RAD
*trig_practice.py                 48/48
    print("Given:  cos(θ) =",adj1,"/",h1)
else:
    print("Given:  tan(θ) =",op1,"/",adj1)
print("-------")

if find == 1:
  ans=eval(input("sin(θ) = "))
elif find == 2:
  ans=eval(input("cos(θ) = "))
else:
  ans=eval(input("tan(θ) = "))
```

16. Now, compare the entered answer to the correct answer.

    For example, if find = 1, compare the answer to sin(θ).
    We need to know if ans == op/h.

    Because the answer is stored as a float (decimal), and op/h evaluates to a decimal, it is possible to have a round-off error when comparing. To fix this, we will check to see if the difference between the answer and op/h is less than 0.00001.

```
 if abs(ans – op/h) <0.00001:
     print("correct")
else:
     print("Sorry",op1,"/",h1)
```



```
1.1  1.2        *Doc          RAD
*trig_practice.py                53/60
if find == 1:
  ans=eval(input("sin(θ) = "))
  if abs(ans-op/h)<0.00001:
    print("correct")
  else:
    print("Sorry:",op1,"/",h1)
elif find == 2:
  ans=eval(input("cos(θ) = "))
  if abs(ans-adj/h)<0.00001:
    print("correct")
  else:
    print("Sorry:",adj1,"/",h1)
else:
  ans=eval(input("tan(θ) = "))
  if abs(ans-op/adj)<0.00001:
    print("correct")
  else:
    print("Sorry:",op1,"/",adj1)
```

17. Execute your program. (ctr →r)

    Sample 1

$$\tan(\theta) = \frac{8}{sqrt(17)} = \frac{opposite}{adjacent}$$

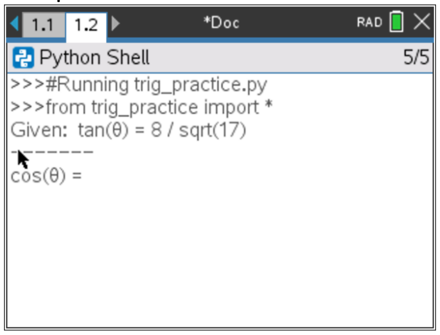
opposite^2 + adjacent^2 = hypotenuse^2

8^2 + sqrt(17)^2 = hypotenuse^2

hypotenuse = sqrt(81) = 9

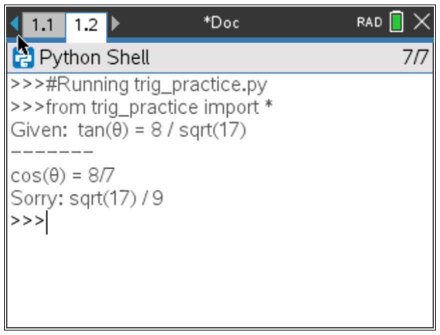
$$\cos(\theta) = \frac{adjacent}{hypotenuse} = \frac{sqrt(17)}{9}$$

Sample 1



```
1.1  1.2        *Doc          RAD
Python Shell                     5/5
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  tan(θ) = 8 / sqrt(17)
-------
cos(θ) =
```

Entered 8/7 incorrect answer



```
1.1  1.2        *Doc          RAD
Python Shell                     7/7
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  tan(θ) = 8 / sqrt(17)
-------
cos(θ) = 8/7
Sorry: sqrt(17) / 9
>>>
```

Sample 2

$\sin(\theta) = \frac{5}{9} = \frac{opposite}{hypotenuse}$

opposite^2 + adjacent^2 = hypotenuse^2

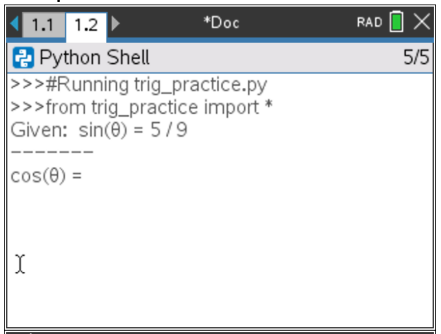
5^2 + adjacent^2 = 9^2

adjacent^2= 56

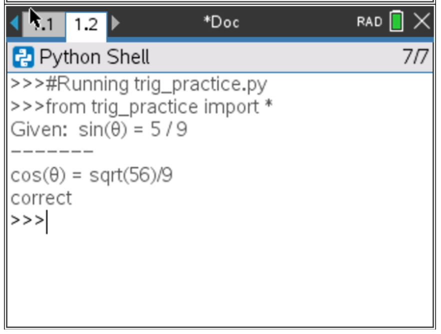
adjacent = sqrt(56)

$\cos(\theta) = \frac{adjacent}{hypotenuse} = \frac{sqrt(56)}{9}$

Sample 2



```
Python Shell                5/5
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  sin(θ) = 5 / 9
-------
cos(θ) =
```



```
Python Shell                7/7
>>>#Running trig_practice.py
>>>from trig_practice import *
Given:  sin(θ) = 5 / 9
-------
cos(θ) = sqrt(56)/9
correct
>>>
```

18. Now that you have a trigonemtry practice app, can you answer 5 questions in a row correctly?

Can you get 10 in a row correct?