Unit Circle Cruncher

In this lesson, you will create two Unit Circle games using the Python Editor.

Your first game will randomly select a degree value from a list. It will randomly display either a cosine or sine question using that degree. The program will request the exact value on the unit circle for that cosine or sine value. If the answer entered is correct, game play will continue.

The second game is a modified version of the first game. Instead of randomly selecting a degree value from the list, the program will select a radian value from a list.

Objectives:

Programming Objectives:

- Use the input function and a variable to collect and store data from a user
- Use the randint() function to generate random integers.
- Use a while loop to repeat code
- Use a list to store values

Game 1: Math Objectives

- Evaluate sine and cosine values from the Unit Circle using degrees

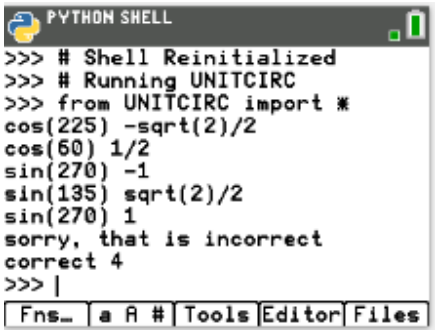
Game 2: Math Objectives

- Evaluate sine and cosine values from the Unit Circle using radians

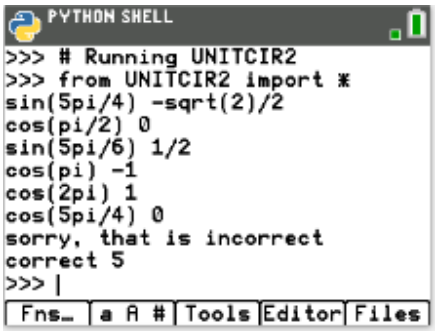
Math Course Connections: Algebra 2 or Pre-Calculus

In this project, you will create two Unit Circle Guessing games. The first version will request cosine and sine values in degree mode. The second version will request cosine and sine values in radian mode.

Version 1

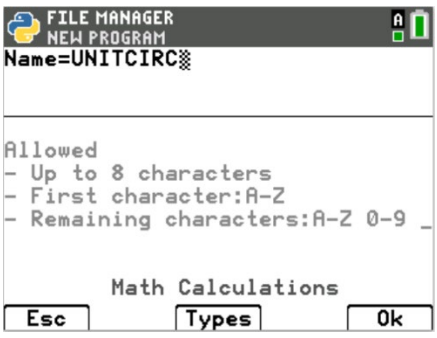


Version 2



1. Create a new python program named UNITCIRC

   Choose Math Calculations for the type.



2. You will need one more library, the random library.

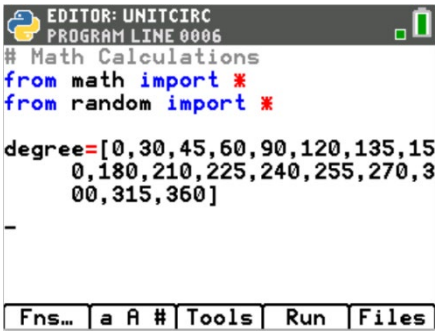
   The random library contains the randint() function, which you will use to generate random integer coordinates.

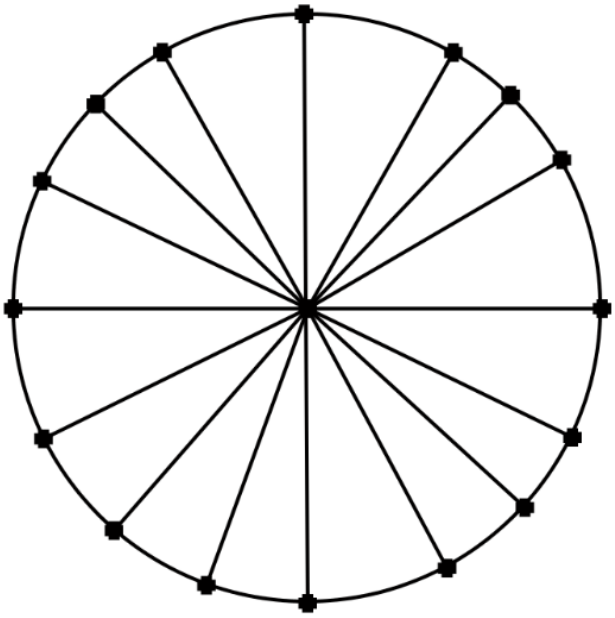
   Place your cursor on the line below the **import math**

   Fns → Modul → Random → from random import *

3. You will create a list of unit circle values.

   Label all the degrees on the unit circle.





   Add the values to a list named degrees.

   degree = [0, 30, 45, 60,……..

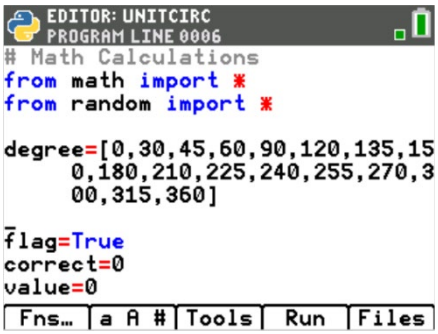
Teacher Tip: The order of the degree list is not important. Make sure students give only the numeric version of the degree. The degree symbol will cause an error.

4. The program will include a while loop that repeats while the questions are answered correctly. A variable named **flag** will store the value True or False to repeat or exit the loop.

   The variable **correct** will keep track of the number of questions answered correctly.

   The variable **value** will store the value of the correct answer.

   Add the lines:
   flag = True
   correct = 0
   value = 0



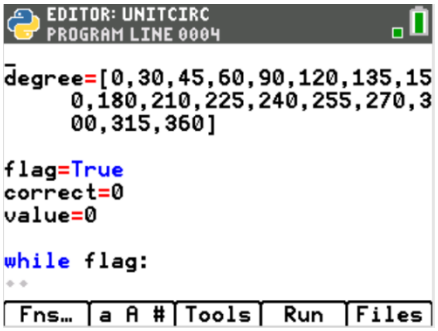
Tech Tip The [a A #] menu accessed through the [window] might make it easier to type words.

5. The program will repeatedly generate a new question, let the user enter an answer and check the validity. The while loop will allow this code to repeatedly execute while the value is true.

   Fns> Ctl> while

   while *BooleanExpr*:
   ◊◊block

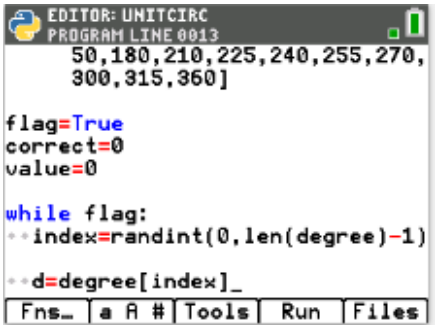
   The variable flag goes in the *BooleanExpr* placeholder. The rest of the code will go inside the indented block



6. The items in the degree list are referenced using an index number. For example, degree[0] equals 0 because the number zero is in the first location of the list. The value of degree[1] is 30 because the number 30 is in the 1st index. Since the first index is at index 0, the last index of the list is one less than the length of the list.

   To randomly select an item from the list you will:
   *generate an index value from 0 to one less than the length of the list

   *use the index to select the degree from the list.



The randint() function can be accessed using
Fns> Modul> Random> randint

Add the lines:

```
index = randint( 0, len(degree)-1 )
d = degree[index]
```

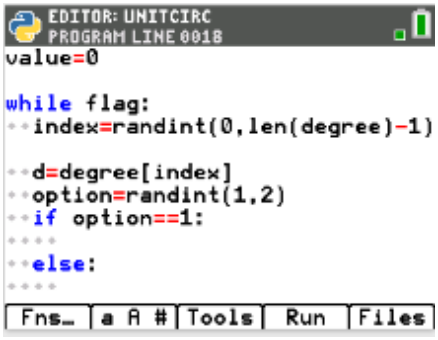
7. Your program will either ask for a cosine value or a sine value.

   Use the randint() function to generate either a 1 or a 2. Store the value in a variable named **option**.

8. If the option is a 1, the code will display a cosine question. Otherwise, it will display a sine question.

   Insert an if else statement.
   Fns> Ctl> if else

   In the if's BooleanExpr, check to see if the option is equal to 1.



9. The program will ask a question such as: cos(30°) and store the player's answer in the variable ans.

   To ask the question, you join the string "cos(" with the variable d and the string "°)"

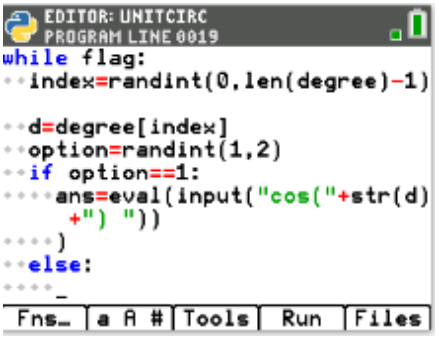
   Because "cos(" is a string and d is a variable, you will combine the two by transforming d to string using str(d).
   Fns → Type → Str()

   The function eval() will be used to evaluate the value entered in the input() box.
   Fns → I/O → eval

   Fns → I/O → input

   Add the line:

```
ans = eval(input("cos(" + str(d) + ")" ) )
```



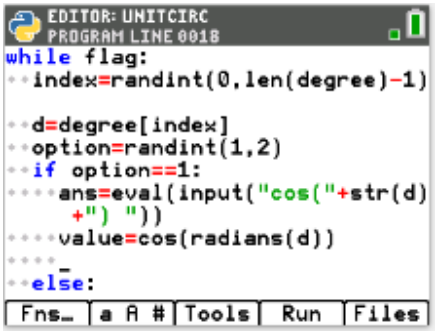
To store the correct answer, add the line:

```
value=cos(radians(d))
```

This will store the evaluated cosine value in the variable value. You must put the radians() function around the d value to evaluate the function in degree mode.

Fns → Modul → Math → Trig → cos()

Fns → Modul → Math → Trig → radians()

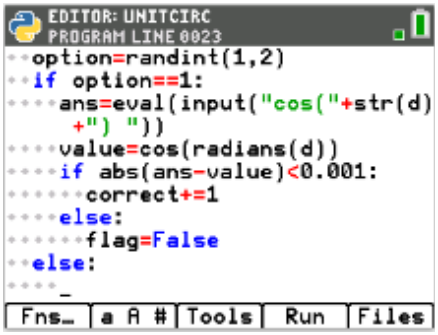


10. Now you need to compare the player's answer stored in **ans** with the actual answer stored in value. If the answers match, add 1 to the variable correct. If the answers don't match, change the flag to False so the loop will exit.

    What do you think this if statement will look like?

11. You might have written:

```
if ans == value:
    correct += 1
else:
    flag = False
```



While fundamentally correct, this will often cause a floating-point error to occur. For example, python will sometimes evaluate cos(60°) to 0.499999999 or 0.500000001.

When this error occurs, the code will evaluate to False when it is actually True. To control for possible floating-point errors, write the if statement in the form:

```
if abs(ans – value) < 0.001:
    correct += 1
else:
    flag = False
```

12. Add the six lines of code for the sine function.

```
if option == 1:
  ans=eval(input("cos("+str(d)+"°) "))
  value=cos(radians(d))
  if abs(ans-value)<0.001:
     correct +=1
  else:
     flag=False
else:
     #add the lines of code for the sine function here
```

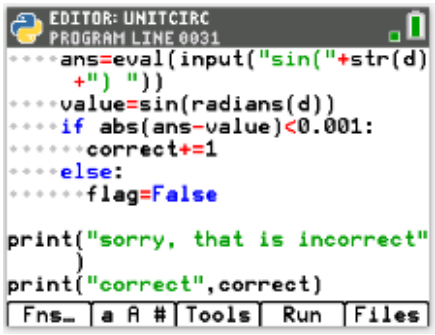
Tech Tip The menu [Tools] accessed by the button [zoom] will let you copy and paste lines. It might be helpful to copy, paste, and modify some lines.

13. Once the user answer the question incorrectly, exit the loop.

    Use code to print a message to let the user know the answer was incorrect.

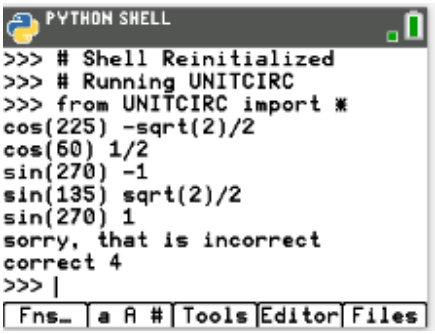
    Display the number of correct answers.

    print(“sorry, that is incorrect”)
    print(“correct:”,correct)



14. Play your game several times.
    Can you answer at least 10 questions in a row before the game terminates?

    The screen to the right shows one demo run of the game. The game had 4 correct answers before the user forgot to put the negative (-) in front of the 1.

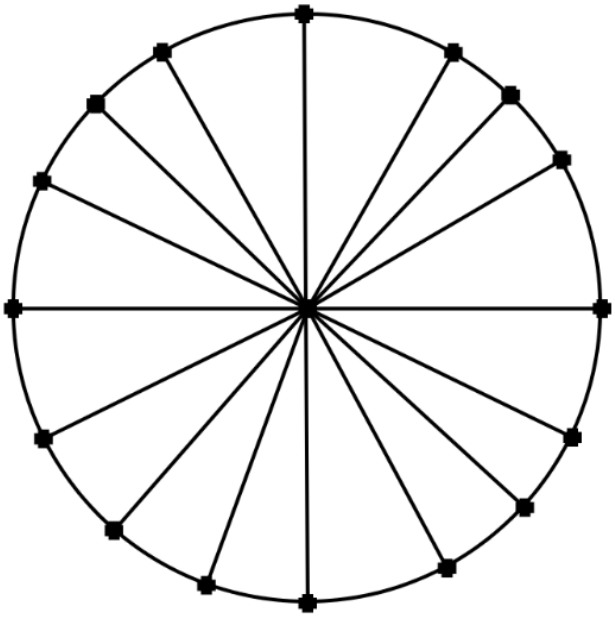


Teacher Tip:

```
# Math Calculations
from math import *
from random import *

degree=[0,30,45,60,90,120,135,150,180,210,225,240,255,270,300,315,360]

flag=True
correct=0
value=0

while flag:
 index=randint(0,len(degree)-1)
 d=degree[index]
 option=randint(1,2)
 if option==1:
    ans=eval(input("cos("+str(d)+") "))
    value=cos(radians(d))
    if abs(ans-value)<0.001:
       correct+=1
    else:
       flag=False
 else:
    ans=eval(input("sin("+str(d)+") "))
    value=sin(radians(d))
    if abs(ans-value)<0.001:
       correct+=1
    else:
       flag=False

print("sorry, that is incorrect")
print("correct",correct)
```

15. Let's make a second version to display the questions in radian mode.

    Label all the radian measurements on the unit circle below.



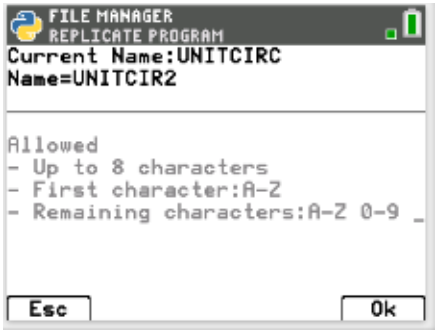
16. From the editor, select Files.

    Move the arrow keys to move the cursor in front of the UNITCIRC program.

    Select Manage

    Replicate Program

    Name the new program UNITCIR2.

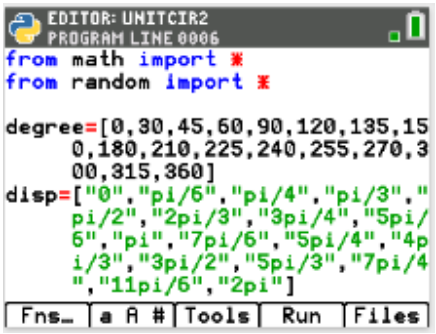


17. Below the line:

    degree=[0,30,45,60,90,120,135,150,180,210…..

    add your list of radian values

    disp=["0","π/6","π/4","π/3","π/2","2π/3",…



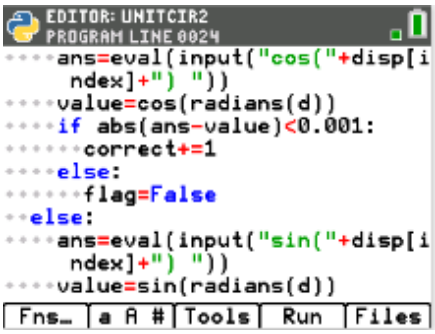
18. There are only two more lines that have to be modified. Which two lines do you think need modified?

19. The display line needs to display the corresponding radian instead of the degree. Change the ans = line to the following:

    ans=eval(input("cos("+disp[index]+") "))

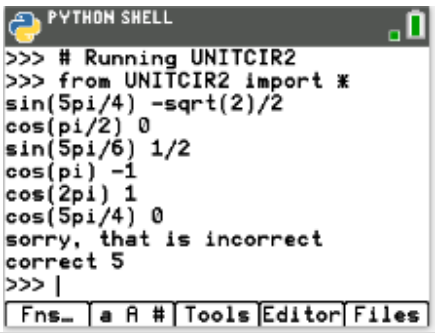
    and

    ans=eval(input("sin("+disp[index]+") "))



20. Can you get 10 in a row correct in radian mode?

    The example on the right answered 5 correctly before the incorrect value was entered.



Teacher Tip:

```
# Math Calculations
from math import *
from random import *

degree=[0,30,45,60,90,120,135,150,180,210,225,240,255,270,300,315,360]
disp=["0","pi/6","pi/4","pi/3","pi/2","2pi/3","3pi/4","5pi/6","pi","7pi/6","5pi/4","4pi/3","3pi/2","5pi/3","7pi/4","11pi/6"
,"2pi"]
```

```
flag=True
correct=0
value=0

while flag:
 index=randint(0,len(degree)-1)
 d=degree[index]
 option=randint(1,2)
 if option==1:
    ans=eval(input("cos("+disp[index]+") "))
    value=cos(radians(d))
    if abs(ans-value)<0.001:
       correct+=1
    else:
       flag=False
 else:
    ans=eval(input("sin("+disp[index]+") "))
    value=sin(radians(d))
    if abs(ans-value)<0.001:
       correct+=1
    else:
       flag=False

print("sorry, that is incorrect")
print("correct",correct)
```