12 Maths Functions

Aim: *To investigate some of Python's Maths functions.*

The built-in maths (or in programming terms, *math*) functions are similar to those available on a scientific calculator. Examples include Sin, Cos, Tan and square root. There are also functions which let you round off numbers in different ways.

- To find the sine of 2, you would use the code: `math.sin(2)`
- To find the sine of the variable *intAge*, you would use the code: `math.sin(intAge)`
- To round the number 1.56434 down to two decimal places, use the code: `math.round(1.56434, 2)`

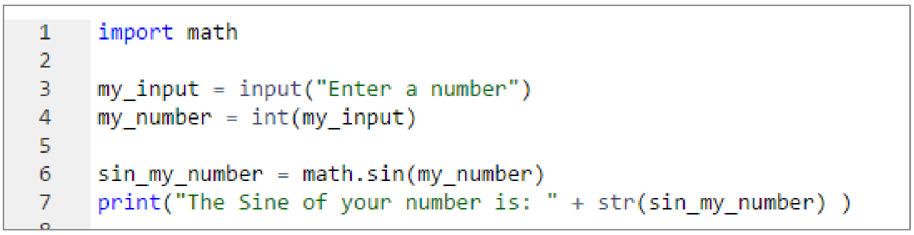
To use the maths functions, you need to import the math module at the start of your program using the code *import math*.

Note: *Unlike strings, quotes are not needed when dealing with numbers. Variables never need quotes.*

Strings	*Quotes*	`my_var = "rose":`	`print("rose")`
Numbers	*No quotes*	`my_var = 20:`	`print(20)`
Variables	*No quotes*	`my_var = your_var:`	`print(my_var)`

Task 1 – Maths Functions in Python

a. Set up the code below and save as '***12.1 Sin Cos***'. Test your program with several different integers.

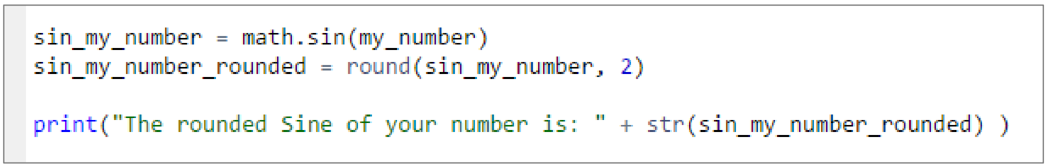


```
import math

my_input = input("Enter a number")
my_number = int(my_input)

sin_my_number = math.sin(my_number)
print("The Sine of your number is: " + str(sin_my_number) )
```

Note: *You might find that you don't get the outputs that you expect. The functions are working with radians rather than degrees. Don't worry about this for now.*

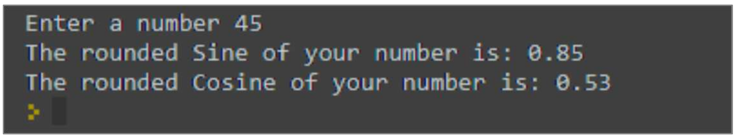
b. Add to your code so that it also displays the cosine of the number entered.

c. We want to round off the outputs to two decimal places using the *round* function. We will do this by introducing further variables and adding some extra steps. The code below shows how you might do this for the sin function. Complete this for both the Sin and Cos functions.



```
sin_my_number = math.sin(my_number)
sin_my_number_rounded = round(sin_my_number, 2)

print("The rounded Sine of your number is: " + str(sin_my_number_rounded) )
```

Your output should look something like that shown below. Save your work.



```
Enter a number 45
The rounded Sine of your number is: 0.85
The rounded Cosine of your number is: 0.53
```

Maths Functions (page 2)

Task 2 – Efficient Coding

Although the code we used in the last task works well, it is possible to make things work a little more efficiently. For example, we have used 6 different variables where we could easily have used less. Not only is this good practice, it will help you better understand the code written by other programmers. For example, rather than this code:

```
sin_my_number = math.sin(my_number)
sin_my_number_rounded = round(sin_my_number, 2)
```

we could use:

```
sin_my_number_rounded = round(math.sin(my_number), 2)
```

This uses less variables and less lines of code.

In a copy of your program saved as '***12.2 Efficient***' make your code more efficient by getting rid of any references to the variables *sin_my_number* and *cos_my_number*.

Extension

Can you also remove the *my_number* variable from your code?

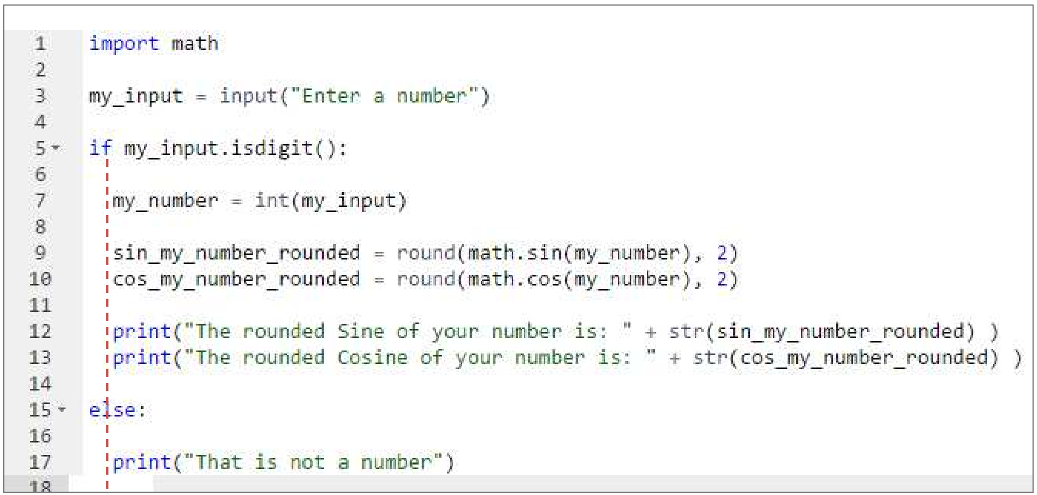
Task 3 – Validating Inputs

Run your program from either of the last tasks. What happens if you enter some letters into the console instead of a number? The error occurs when we try and use the *int* function on a piece of text. However, even if this problem was solved, you still can't find the sin or cosine of a string. We should add a check to make sure that the input only contains numbers.

Look at the program on the right. It uses an *if clause* and the *isdigit* method to make a decision about what to do.

Make a copy of your last program and name it '***12.3 Validate***'. Add the extra lines of code. Test the program with some numbers and letters. Does it work as you would hope?

Note for the future: *After copying and pasting code into a new session, you may have to recreate the indents at the start of each line. Use one tab for each indent here.*



```
1   import math
2
3   my_input = input("Enter a number")
4
5   if my_input.isdigit():
6
7       my_number = int(my_input)
8
9       sin_my_number_rounded = round(math.sin(my_number), 2)
10      cos_my_number_rounded = round(math.cos(my_number), 2)
11
12      print("The rounded Sine of your number is: " + str(sin_my_number_rounded) )
13      print("The rounded Cosine of your number is: " + str(cos_my_number_rounded) )
14
15  else:
16
17      print("That is not a number")
```

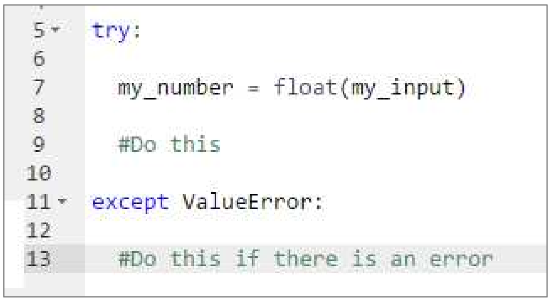
The indentation of lines within an *if* clause is important. Lines 7-13 must have exactly the same indentation (and you can't mix tabs and spaces). This is how Python knows what code to run. Lines 5 and 15 must also be the same i.e. no indent. Programming languages often use brackets or *End If* statements to separate a block of code. Python uses your indents.

Extension

Try entering a decimal number into the code. Decimal numbers do have sines and cosines, so this needs to work. We need a better solution.

In a copy of your program named '***12.3 Extension***', experiment with the *try…except* code shown. This will attempt to run the first block of code. If there is an error with the value, it will run the second block.

While you're at it, see if you can get this program to work with degrees rather than radians. Try the *math.radians()* function.



```
try:

    my_number = float(my_input)

    #Do this

except ValueError:

    #Do this if there is an error
```