**Creating Modules**

A ***module*** is a package of functions. We create our own modules in the same way we create a python program. As an example, let’s create two simple functions called sum() and diff()

# The simpleMath module - just some simple math functions!

def sum(a,b):

return a+b

def diff(a,b):

return a-b

These two functions will be part of a module we will call “simpleMath”.

Save the file as "**simpleMath"** (in the WingIDE you don’t have to type the .py extension - just save as “simpleMath”. The IDE will automatically add the .py extension) . Now we have a module.

Now start with a fresh screen. Here we will import the module and make some function calls. Copy this code and save it as "**math test**".

import simpleMath

x = simpleMath.sum(3,4)

y = simpleMath.diff(3,4)

print (x)

print (y)

Try running it. It’s that simple!

The only thing you have to be careful about is the module you import must be in the same file folder as the program that calls it. In our example, the simpleMath module and the math test program were saved in the same directory.

Professionally-made modules are large and often a programmer only needs a small function in it. If that's the case, you can just take the function(s) you need:

from simpleMath import sum # just get the sum function

print (sum(3,4))

The nice thing is that you don't have to specify the module name in the function call. In other words, you don’t have to say “simpleMath.sum(3,4)”.

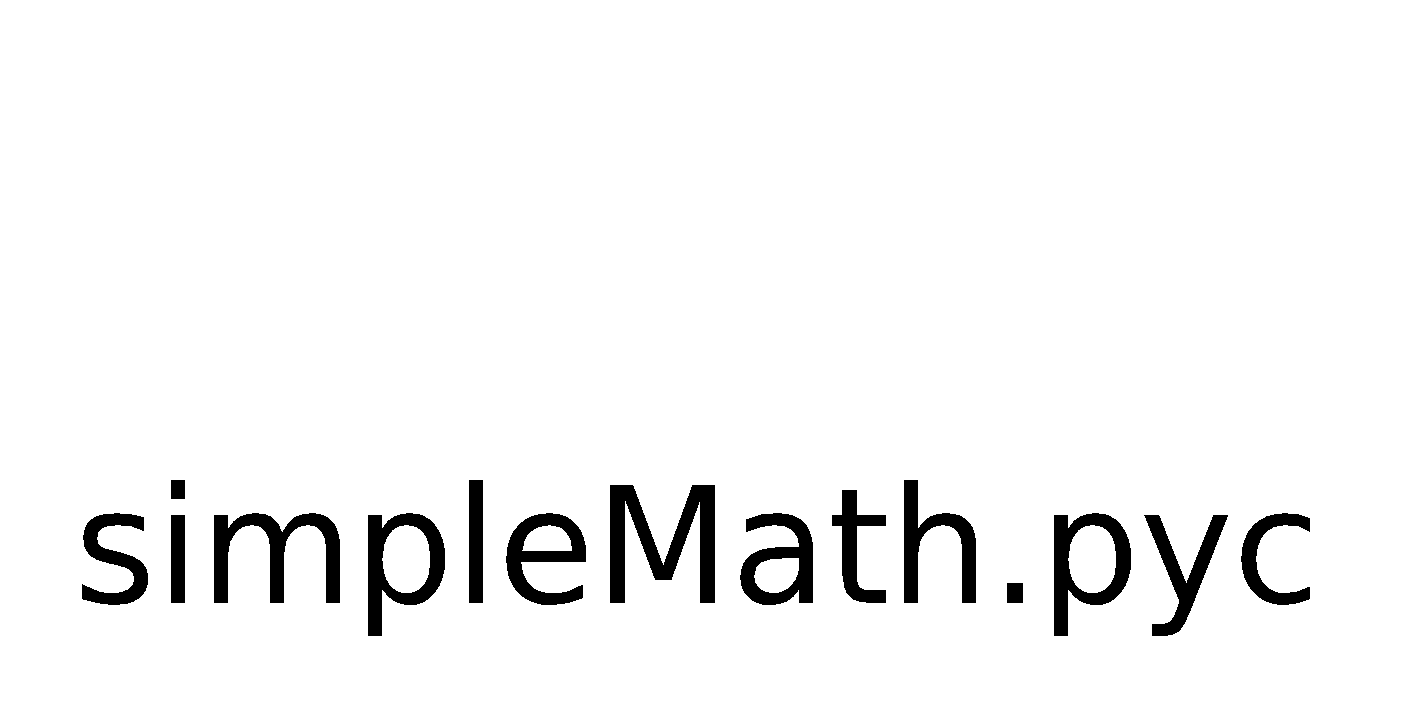
If the module is small enough, you can import everything from it:

from simpleMath import \* # Import all functions

print (sum(3,4))

print (diff(5,2))

When you save the module, a compiled version is created as well, with extension .pyc. If you look in your directory where you saved your simpleMath.py module, there should also be a second file with the same name.



This is the compiled version of the module - it has been compiled into byte code. ***Byte*** ***code*** is a streamlined version of the ***source*** ***code*** (your Python program) but isn’t full-on machine code. Machine code is machine specific, whereas byte code is universal, as is the Python source code. This means that the byte code can be read by any machine on any platform. In other words, you can give your simpleMath byte code and give it to another programmer on any platform (say on a Mac) and that programmer could use it.

**Keywords**: ***module, function, byte code, source code***