# LectNotebook11

February 10, 2021

## 1 Lecture 11

-Introduction: »
-Module functions: »
-Creating a module: »
–Working in CoCalc: »
–Working on your own computer: »
-Using the module: »
-Accessing modules in other folders: »
–Working in CoCalc: »
–Working on your own computer: »
-Keyboard input to Python: »

### 1.1 Some organisational matters

Welcome back. I hope your exams went well!

This semester, Phys105 will run for the first five weeks. We will introduce some new material in the first two weeks, then use the remainig three weeks to complete any outstanding Notebooks.

The Computer Classes are timetabled as follows:

| Day | Time |
|---|---|
| Tuesday | 09:00…11:00 |
| Tuesday | 11:00…13:00 |
| Wednesday | 09:00…11:00 |

To get into your class, go to the Phys105 Canvas page and click on the relevant link (as we did last semester). Don't use the link in your timetable!

If anyone who has got previous computing experience wants to do a project in the last five weeks of Phys105, rather than the Notebooks in the computer classes, please mail me after this lecture! (A possible project would be the simulation of the behaviour of the lift in the Oliver Lodge building and the optimisation of its rest position - but I am open to suggestions.)

### 1.2 Introduction

This week, we will first see how we can turn the functions we have written in a Jupyter Notebook into a module that can be loaded and used in the same way as the Numpy, Matplotlib or other

libraries. We will try this using CoCalc, but also describe how it can be done on a PC. (To avoid confusion, the parts of this Notebook that don't refer to using CoCalc will be in blue text.) We will then look at how we can provide input to Python programs from the keyboard.

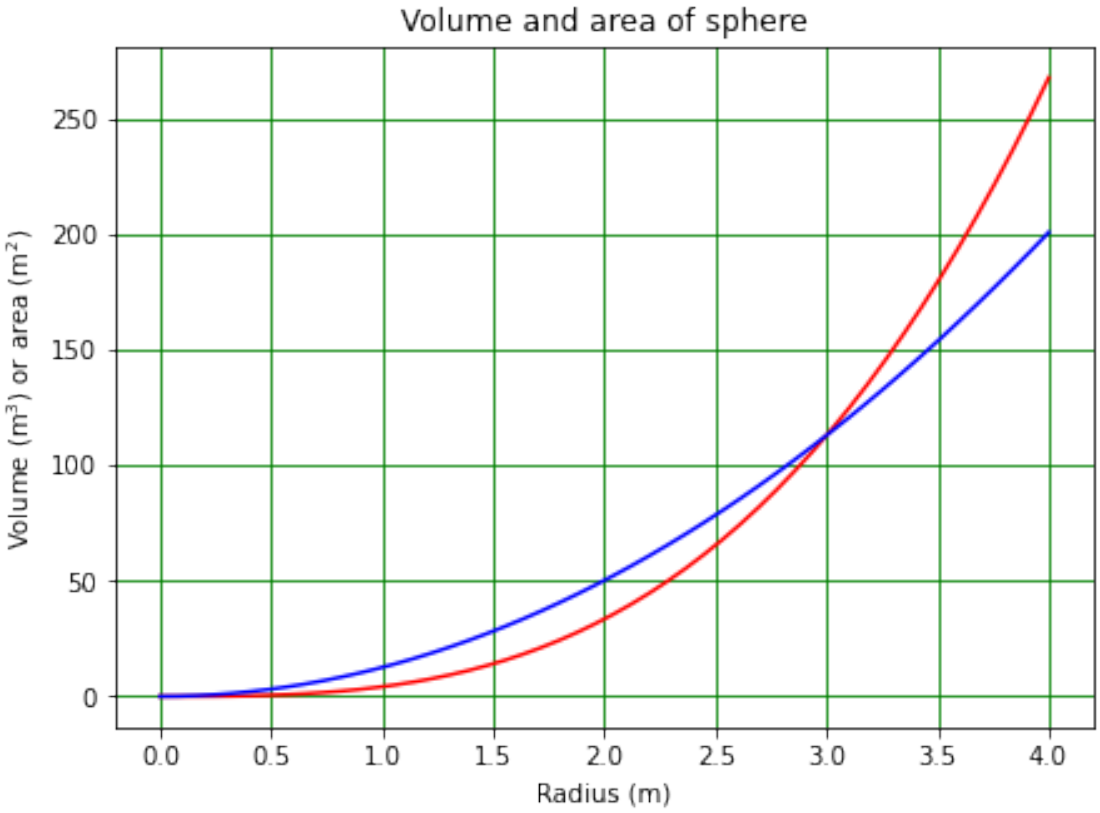
### 1.3 Module functions

Let's start by writing a few functions that we can use to create a module.

[1]:
```
import numpy as np
#
def circleParams(r):
    '''
    Given the radius of a circle, this function returns its area and␣
 ↪circumference.
    '''
    A = np.pi*r**2
    c = 2*np.pi*r
    return A, c
#
def rectangleParams(h, w):
    '''
    Given the height and width of a rectangle, this function returns its area␣
 ↪and perimeter.
    '''
    A = h*w
    p = 2*(h + w)
    return A, p
#
def sphereParams(r):
    '''
    Given the radius of a sphere, this function returns its volume and its␣
 ↪surface area.
    '''
    V = 4/3*np.pi*r**3
    A = 4*np.pi*r**2
    return V, A
#
def rectPrismParams(h, w, d):
    '''
    Given the height, width and depth of a rectangular prism, this function␣
 ↪returns its volume,
    surface area and total side length.
    '''
    V = h*w*d
    A = 2*(h*w + w*d + h*d)
    s = 4*(h + w + d)
    return V, A, s
```

As we have seen many times, we can use these functions in the Notebook in which they are defined. For example, here is a plot of the volume, $V$, and area, $A$, of a sphere as a function of its radius, $r$.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#
nArr = 50
rBot = 0.0
rTop = 4.0
rArr = np.linspace(rBot, rTop, nArr)
Varr, Aarr = sphereParams(rArr)
#
plt.figure(figsize = (7, 5))
plt.title("Volume and area of sphere")
plt.ylabel("Volume (m$^3$) or area (m$^2$)")
plt.xlabel("Radius (m)")
plt.plot(rArr, Varr, linestyle = '-', color = 'r')
plt.plot(rArr, Aarr, linestyle = '-', color = 'b')
plt.grid(color = 'green')
plt.show()
```



What we now want to do is to see how we can use these functions in a different Jupyter Notebook, without copying their definitions into that Notebook.

## 1.4 Creating a module

We'll first look at how to create a module when we are working in CoCalc.

### 1.4.1 Working in CoCalc

In order to create a module containing routines from this Notebook, go through the following steps:

1. Click on the *File* menu (in the Notebook, not the CoCalc Files menu!), then, in *Download as…* select *Executable script (.txt).*
2. Wait until the process is complete.
3. Close the *Download* box.
4. Close the Phys105-Week09-Student.ipynb Notebook.
5. You will now have a file called Phys105-Week09-Student.py in the Phys105-Week09 directory. Rename this file by selecting it (tick the box next to it) and choosing the *rename* option. Call it Phys105W09.py.
6. Open Phys105W09.py (by clicking on it in CoCalc). You will see that it is just a copy of this Notebook written as Python code, i.e. all the Markdown cells have been turned into Python comments by sticking a "#" in front of them.
7. Tidy up the file by deleting everything except the code for the functions *circleParams*, *rectangleParams*, *sphereParams* and *rectPrismParams*. Do not delete the line that reads `import numpy as np`!
8. Close Phys105W09.py.
9. Open the Notebook (Phys105-Week09-Student.ipynb) again.

### 1.4.2 Working on your own computer

In order to create a module containing routines from this Notebook, click on the *File* menu, then on *Download as* and select *Python*. (If you are running Jupyter Lab rather than Jupyter Notebook, you need to use *File*, *Export Notebook As…* and then *Export Notebook to Executable Script*.) Depending on the security settings on your browser, you may get a warning about the file that is created, saying that it can damage your computer. You can ignore this and click *Keep* or *Save*. In your default download location (usually your Downloads folder) you will then have a file called Phys105-Week09-Student.py.

Move Phys105-Week09-Student.py into your working directory (the directory or folder which contains this Notebook) and rename it. Call it Phys105W09.py. Open Phys105W09.py by clicking on it in your Jupyter Notebook browser. You will see that it is just a copy of this Notebook written as Python code, i.e. all the Markdown cells have been turned into Python comments by sticking a "#" in front of them. Tidy up the file by deleting the superfluous comment lines - leave the ones that are useful! - and other material that isn't part of the functions *circleParams*, *rectangleParams*, *sphereParams* and *rectPrismParams*. Do not delete the line that reads `import numpy as np`!

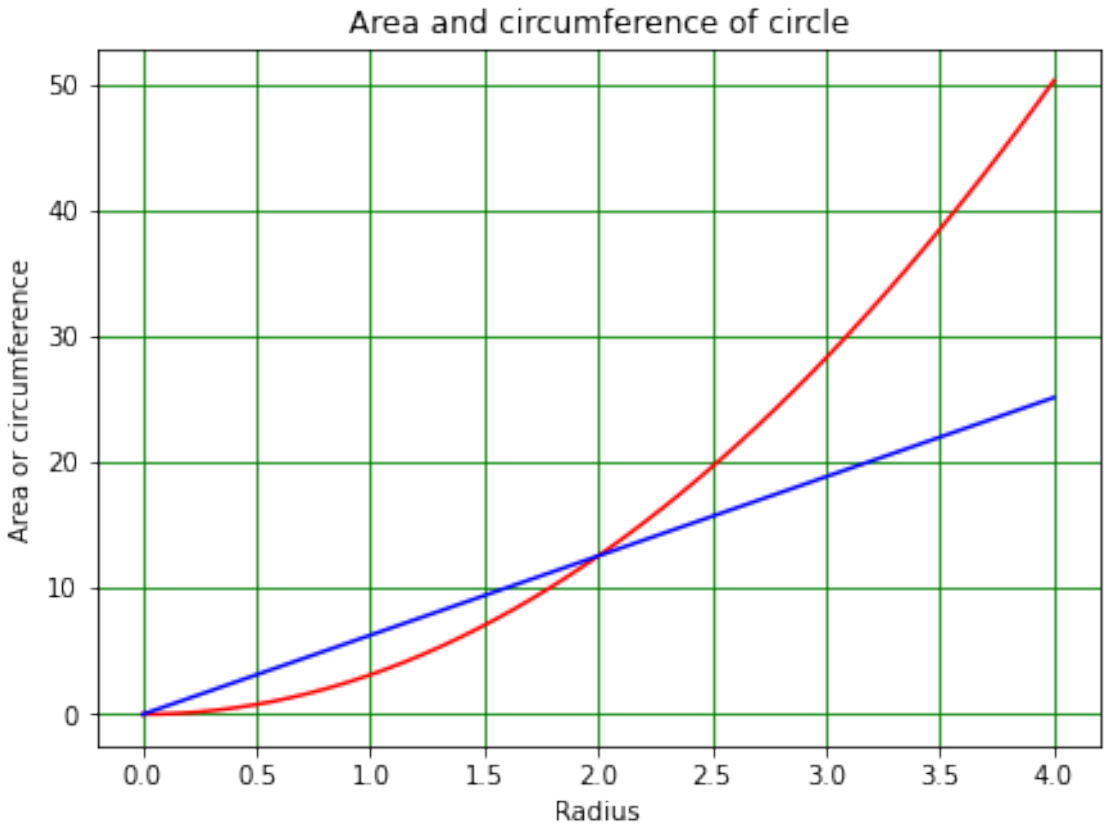
## 1.5 Using the module

You can now use all the functions in Phys105W09.py by importing it as a module, as shown in the following example. (The reason you had to rename the file is that hyphens are not allowed

in module names in Python, so you wouldn't be able to import the file if it was called Phys105-Week09-Student.py.) Note, your file name should have the extension .py, but you don't include this in the `import` statement.

Notice that, after doing `import Phys105W09 as ph` below, we have called the routine `ph.circleParams` (with a `ph.` in front of the name to indicate it comes from the Phys105W09 module, cf. using `np.cos` to use the cosine function from the Numpy library). The version of *circleParams* below is therefore that from the *Phys105W09* module, not the one defined in this Notebook!

[3]:
```
import numpy as np
import matplotlib.pyplot as plt
import Phys105W09 as ph
#
nArr = 50
rBot = 0.0
rTop = 4.0
rArr = np.linspace(rBot, rTop, nArr)
Aarr, cArr = ph.circleParams(rArr)
#
plt.figure(figsize = (7, 5))
plt.title("Area and circumference of circle")
plt.ylabel("Area or circumference")
plt.xlabel("Radius")
plt.plot(rArr, Aarr, linestyle = '-', color = 'r')
plt.plot(rArr, cArr, linestyle = '-', color = 'b')
plt.grid(color = 'green')
plt.show()
```




We have been careful to include the statement `import numpy as np` at the top of *Phys105W09.py*. This statement is executed when the module is first loaded, so even if we use the functions in *Phys105W09.py* from a program which doesn't import numpy, they will work OK.

## 1.6 Accessing modules in other folders

What we have done so far only allows us to use functions from a module in the directory in which we are working. We can also get at modules in other directories.

### 1.6.1 Working in CoCalc

In order to do this, follow the procedure below:

1. Close this Notebook.
2. Use the menu to duplicate Phys105W09.py and call the new file Phys105W09new.py.
3. Make a new folder (using the CoCalc `+New` menu) called Phys105lib. This will appear in your Phys105-Week09 folder.
4. Using the CoCalc menu, move Phys105W09new.py into Phys105lib.

Now you can restart the Phys105-Week09-Student.ipynb Notebook and run the code cell below.

### 1.6.2 Working on your own computer

In order to try this, make a copy of *Phys105W09.py* and call it *Phys105W09new.py*. Make a new folder *Phys105lib* in your working directory. (You can do this using *File Explorer* on Windows, *Finder* on a Mac or the command *mkdir* on a Linux system.) Move the file *Phys105W09new* into the folder *Phys105lib*. Now try to run the cell below.

[5]:
```
# <!-- Student -->
#
import numpy as np
import matplotlib.pyplot as plt
import Phys105W09new as phnew
#
nArr = 50
rBot = 0.0
rTop = 4.0
rArr = np.linspace(rBot, rTop, nArr)
Aarr, cArr = phnew.circleParams(rArr)
#
plt.figure(figsize = (7, 5))
plt.title("Area and circumference of circle")
plt.ylabel("Area or circumference")
plt.xlabel("Radius")
plt.plot(rArr, Aarr, linestyle = '-', color = 'r')
plt.plot(rArr, cArr, linestyle = '-', color = 'b')
plt.grid(color = 'green')
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-5-ea0025ae351d> in <module>
      3 import numpy as np
      4 import matplotlib.pyplot as plt
----> 5 import Phys105W09new as phnew
      6 #
      7 nArr = 50

ModuleNotFoundError: No module named 'Phys105W09new'
```

Python can't find the *Phys105W19new* module, because it only looks for it in the current working directory and in directories specified by a system variable called `path`. We can see which directories are in `path` using the `sys.path` command, after we have imported the `sys` module, as follows.

[6]:
```
# <!-- Student -->
#
import sys
#
```

```
print("Directories in path are:\n",sys.path)
```

```
Directories in path are:
 ['C:\\Users\\green\\OneDrive\\OneDocuments\\Liverpool\\Teaching\\Phys105-Comp01
-2020\\Phys105-Lectures2020\\Phys105-Lect11',
'C:\\Users\\green\\Anaconda3\\python38.zip',
'C:\\Users\\green\\Anaconda3\\DLLs', 'C:\\Users\\green\\Anaconda3\\lib',
'C:\\Users\\green\\Anaconda3', '', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages\\locket-0.2.1-py3.8.egg', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages\\win32', 'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\win32\\lib',
'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\Pythonwin',
'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\IPython\\extensions',
'C:\\Users\\green\\.ipython']
```

The `path` variable is set up when Anaconda is installed on your computer, or when you create your CoCalc ID. On your computer, exactly what you see will depend on your computer's operating system and where Anaconda was installed. On CoCalc, everyone should see the same `path`. The `path` entries will always have the structure *top_level/second_level/third_level*, and this is what you will see on a Macintosh or a Linux system. On a Windows computer, the forward slashes (/) will be replaced by back-slashes (\). These have to be represented by a double back-slash, as the first backslash is treated as an escape character (in both Python and Markdown). CoCalc runs on Linux, you will see forward slashes when using it.

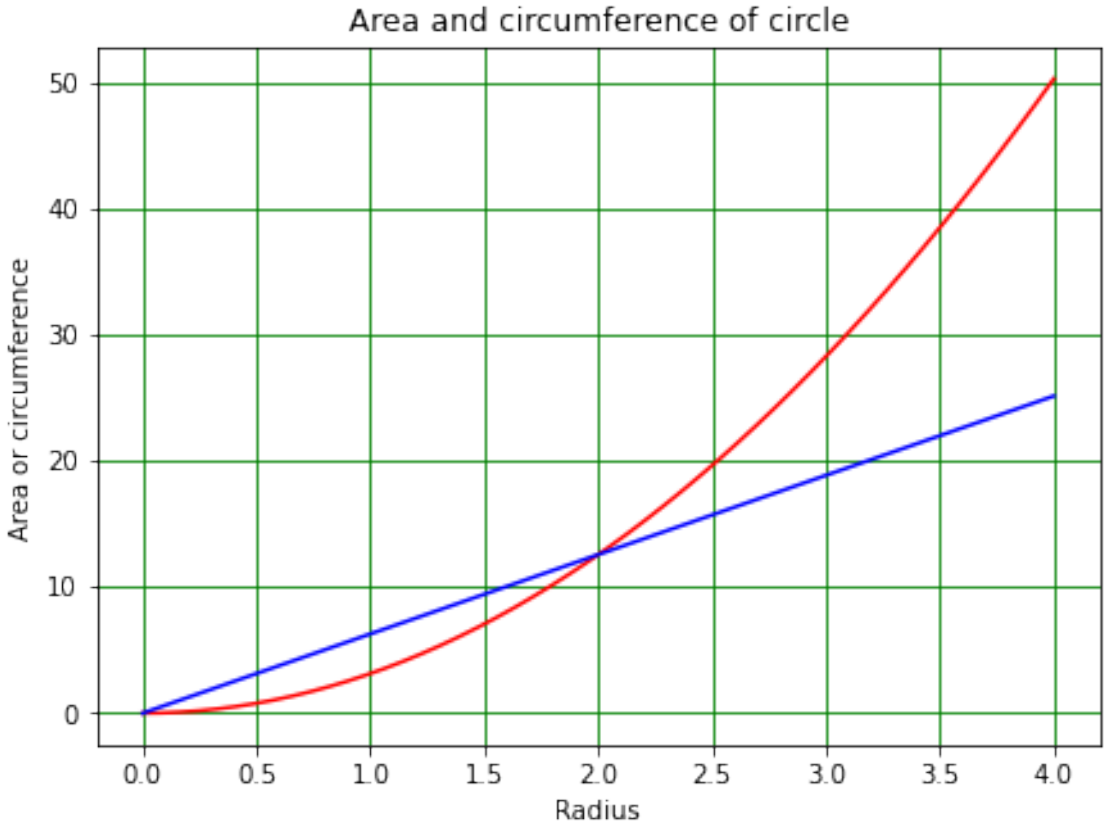
If we want to temporarily add a new directory to `path`, we can do it using `path.append` from the `sys` module as follows.

[7]:
```
# <!-- Student -->
#
sys.path.append('Phys105lib')
print("Directories in path are:\n",sys.path)
```

```
Directories in path are:
 ['C:\\Users\\green\\OneDrive\\OneDocuments\\Liverpool\\Teaching\\Phys105-Comp01
-2020\\Phys105-Lectures2020\\Phys105-Lect11',
'C:\\Users\\green\\Anaconda3\\python38.zip',
'C:\\Users\\green\\Anaconda3\\DLLs', 'C:\\Users\\green\\Anaconda3\\lib',
'C:\\Users\\green\\Anaconda3', '', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages\\locket-0.2.1-py3.8.egg', 'C:\\Users\\green\\Anaconda3\\lib\\site-
packages\\win32', 'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\win32\\lib',
'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\Pythonwin',
'C:\\Users\\green\\Anaconda3\\lib\\site-packages\\IPython\\extensions',
'C:\\Users\\green\\.ipython', 'Phys105lib']
```

You will see that *Phys105lib* has now been added to *path*. Now `import Phys105W09new as phnew` will work.

[8]:
```
# <!-- Demo -->
#
import numpy as np
import matplotlib.pyplot as plt
import Phys105W09new as phnew
#
nArr = 50
rBot = 0.0
rTop = 4.0
rArr = np.linspace(rBot, rTop, nArr)
Aarr, cArr = phnew.circleParams(rArr)
#
plt.figure(figsize = (7, 5))
plt.title("Area and circumference of circle")
plt.ylabel("Area or circumference")
plt.xlabel("Radius")
plt.plot(rArr, Aarr, linestyle = '-', color = 'r')
plt.plot(rArr, cArr, linestyle = '-', color = 'b')
plt.grid(color = 'green')
plt.show()
```



The addition to `path` we have made above will allow us to use anything in the module *Phys105lib* if it is in our current working directory. If we want to be able to use routines from *Phys105lib* from *any* directory, we have to add the full description of its location to `path`. On CoCalc, this implies…

```
import sys
#
sys.path.append('/home/user/Phys105 Introduction to Computational Physics/ComputerClasses/Phys
```

On my computer, what is needed is…

```
import sys
#
sys.path.append('C:/Users/green/OneDrive/OneDocuments/Liverpool/Teaching/Phys105-Comp01-2020/Pl
```

In both cases, this is a bit of a mouthful. (You can work out what the full description of the location of *Phys105lib* should be on your computer by looking at the existing entries in your `path` variable.)

Notice that you can use forward slashes in the `sys.path.append` command even on a Windows system; Python changes these to the format that is relevant for your operating system. (I could also have used the double back-slash notation on my Windows machine, it's just a bit clumsier.)

There are (system dependent) ways of permanently adding folders like `mylib` to `path`, but getting this wrong can cause problems, so we will use the above method. The downside is that before using any of the routines in the library `mylib`, we have to include the statement:

```
import sys
sys.path.append('path to mylib')
```

The upside is that when we shut down our Jupyter Notebook, or restart the kernel, `path` returns to its original value and we don't influence how anything else on the computer works.

## 1.7 Keyboard input to Python

Python programs can read input from the keyboard. They do this with the function `input()`, as is shown in the following example.

[11]:
```
# <!-- Student -->
#
name = input("What's your name?")
print("Nice to meet you " + name + "!")
age = input("How old are you?")
print("So, you are already",age,"years old",name,"\b!")
```

```
What's your name? Tim

Nice to meet you Tim!

How old are you? 123

So, you are already 123 years old Tim!
```

We can check the type of the input as below:

[12]:
```
# <!-- Student -->
#
print("Type of name is",type(name))
print("Type of age is",type(age))
```

```
Type of name is <class 'str'>
Type of age is <class 'str'>
```

Everything (whether numbers or letters) is read as strings. This means the following code will not work (try it!):

[14]:
```
# <!-- Student -->
#
retirementAge = 67
#
name = input("What's your name?")
print("Nice to meet you " + name + "!")
age = input("How old are you?")
#
retireIn = retirementAge - age
if retireIn > 0:
    print("I guess you will retire in",retireIn,"years,",name,"\b.")
else:
    print("I guess you retired",-retireIn,"years ago,",name,"\b.")
```

```
What's your name? Tim

Nice to meet you Tim!

How old are you? 123
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-29728eb4f808> in <module>
      7 age = input("How old are you?")
      8 #
----> 9 retireIn = retirementAge - age
     10 if retireIn > 0:
     11     print("I guess you will retire in",retireIn,"years,",name,"\b.")

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

Can you see what is going wrong here? How could it be fixed? This is one of the exercises you will do in this week's Computer Classes!