EMAT10007 – Introduction to Computer Programming

Exercises – Week 6. Importing Python Files

6.1 Packages and Modules

Essential Questions

Exercise 1 - Importing Python Modules

```
geometry/
|
|---- __init__.py
|---- main.py
|---- volumes.py
```

1. Create the file system shown above. Leave all .py files empty to begin with.
2. In file volumes.py, write two functions, `sphere` and `cube`. Each function should take one input argument and return the volume of the shape in the name of the function.
3. Edit the content of main.py so that when it is run, it prints the volume of a sphere of radius 3 m.
4. Is there a way to achieve the same operation, using shorter code in main.py?.
5. Add another file, areas.py within the 'geometry' folder. In areas.py, write two functions, `sphere` and `cube`. Each function should take one input argument and return the surface area of the shape in the name of the function.
6. Edit the content of main.py so that a variable `A = 5 m` is created and the area and volume of a sphere of radius `A` and a cube of side length `A = 5 m` are printed.

Exercise 2 - Importing Python Modules from different sources

```
building/
|
|---- __init__.py
|---- main.py
|---- funcs.py
|---- house.py
```

1. Create the file system shown above. Leave all .py files empty to begin with.
2. In file house.py, create three variables `floor` = the floor number of the building as an integer, `width` = the width of the building on this floor in metres as a float, `length` = the length of the building on this floor in metres as a float, and assign them numerical values.
3. In file funcs.py, create a function `ceil` that returns the area of the ceiling using the height and width.

4. Import all contents of house.py, and roof.py to main.py using `*`.
5. In file main.py, print the following when called, replacing `<x>` and `<y>` with the area of the value of variable `floor` and the area calculated respectively:

```
The area of the ceiling on floor <x> is <y> m2
```

6. In file main.py, calculate the height of the roof using `width` as shown in Figure 1, where $\theta = \frac{\pi}{6}$ radians. Import all contents if the Python package `math` to use trigonometric functions e.g. `tan`.
7. Look at the functions and variables in the `math` package (https://docs.python.org/3/library/math.html) What potential namespace issues could arise in this program? Are any errors generated due to namespace issues when running your program? If so, how can you prevent them from happening?
8. Instead of computing the roof height within the main file, create a function `roof_height` in funcs.py that returns this value, and call the function in main.py. Notice how this effects use of functions and variables from `math`.
   If imported objects (e.g. `math.pi`) are used to define a function, for example `roof_height`, then the module that the object belongs to (e.g. `math`) must be imported in the file where the function is defined (e.g. funcs.py) rather than where it is called (e.g. main.py).

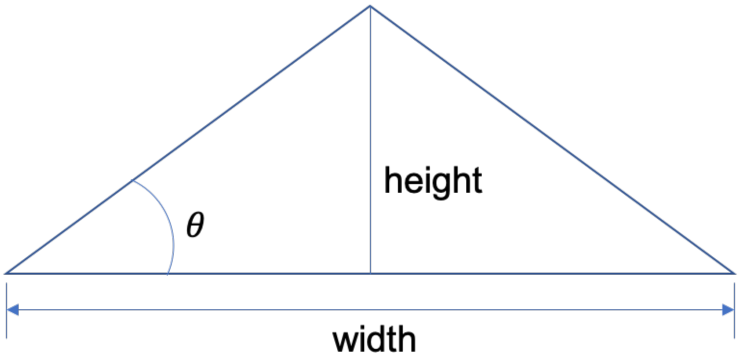

Figure 1: Roof dimensions

Advanced Questions

(A) Create your own Python module (.py file). For example, you could store some useful equations that you have learnt/used recently in another unit as Python functions. Or you could simply take some of the functions you wrote in Week 4 and practise assembling them as a Python module and importing them.

(B) Practise different ways of calling variables, functions and classes from within main.py such as changing the namespace and importing individual functions and/or variables.

6.2 Importing from other locations

Essential Questions

```
cubes/
|
|---- __init__.py
|---- cube.py
|---- shapes/
        |
        |---- main.py
        |---- spheres/
                |
                |---- __init__.py
                |---- sphere.py
```

Create the file system shown above. Leave all .py files empty to begin with.

Exercise 3 - Importing from downstream locations

1. In file sphere.py, create a class `Sphere` that takes one input argument, the radius.
2. Give the class two methods that return the surface area and volume of the sphere respectively.
3. Edit main.py to print the surface area of a sphere of radius 5 m.
   Hint: There are a number of ways to deal with the need for constants e.g. π
   The most reliable way to ensure a constant value is used is to import it from:
   - an external package e.g. `math` for well known constants like π
   - a module created by you to store for program-specific constants - we'll try this out in the next question ...
4. Create a sub-directory within 'spheres' and a file within it, dimensions.py.
5. Create a variable within dimensions.py, `radius` and give it a numerical value.
6. Import the entire contents of dimensions.py to main.py and use `radius` as the input argument to generate a `Sphere` object.

Exercise 4 - Importing from upstream locations

1. In file cube.py, create a class `Cube` that takes one input argument, the side length with a default value of 1 m.
2. Give the class two methods that return the surface area and volume of the cube respectively.
3. Edit main.py to print the surface area of a cube with side length equal to the value of `radius` in dimensions.py

Exercise 5 - Importing from upstream and downstream locations

1. Create a new .py file upstream of main.py
2. Create a function in the file.
3. Call the function from within main.py
4. Create a new file downstream of main.py
5. Add some variables to the file
6. Print the variables within main.py

Advanced Questions

(A) Python files can be run as a module (imported file) or script (run as program, not imported). When the Python interpreter reads a Python file it sets some variables, then executes the code in the file. One of these variables is `__name__`.
`__name__` takes the value `__main__` if the file is run as a script, and the value is the filename if the file is run as a module. The format shown below is widely used to allow a python file to be run as either a module or a script:

```
if __name__ == "__main__":
   print("Executed when file run directly")
else:
   print("Executed when file imported")
```

Edit the file you created in Exercise 5.1 so that the function it contains is run (e.g. to test the function works) if the file is run directly, but not if the file is called as a module.

(B) So far we have used **relative imports** to import .py files in other directories to a python programme. The path to the file we want to import is given *relative* to the current directory. We can alternatively use **absolute imports** where the path is given relative to the computers home directory.
For example, to import sphere.py to main.py using relative imports, we can edit the Python path `sys.path.append('../../')`

```
Desktop/
|
|---- __init__.py
|---- sphere.py
|---- my_folder/
        |
        |---- my_sub_folder/
                |
                |---- main.py
```

To edit the Python path using absolute imports instead, we can use:

Windows

```
 sys.path.append('C:\Desktop\my_folder\my_sub_folder')
```

Mac, Linux

```
sys.path.append('/Users/Hemma/Desktop/my_folder/my_sub_folder')
```

Note, you must change `Hemma` to your personal user name.

Notice, the slashes used to separate the sub-directories lean a different way on Windows than on Mac and Linux.

Change the arguments to `sys.path.append` in Exercises 4 and 5 so that the files are imported using the absolute file path.

(C) Create another sub-directory within the 'shapes' sub-directory and create your own python package. Create Python file(s) within the directory and practise calling them from within main.py