Libraries

Overview

- So far, all of our Python code has taken input from the terminal, and produced text output on the terminal.
- Is that all we can do with Python?
- No!
- We can use *modules* to enhance the capabilities of our programs.

Modules

- A module is a collection of functions that we can use to do more powerful things with our Python programs.
- These functions are either pre-written Python functions that we can use without knowing how they work, or are using a more powerful language behind the scenes to produce the effect we want.
- It is possible to write your own modules, if you want to share some useful functions you have written.

Using a Module

- To use a module, we first need to import it. This is done by writing, at the top of your program:
    - `import` *`modulename`*
- Once the module is imported, we can use functions that it contains using the following syntax:
    - *`modulename.functionname`*`(`*`param1, param2, etc...`*`)`
- Example:
    - The `math` module contains various advanced mathematical functions.
    - To calculate the sine of a value, we could write:

        -
          ```
          import math
          print(math.sin(0.5))
          ```
- Exercise:
    - Look at the documentation for the `math` module here: https://docs.python.org/3/library/math.html
    - Write a program that takes an angle in degrees, converts it to radians, and outputs its sine, cosine, and tangent.

    -
      ```
      import math
      angle = float(input("Enter an angle in degrees: "))
      radangle = math.radians(angle)
      print(math.sin(radangle))
      print(math.cos(radangle))
      print(math.tan(radangle))
      ```
- Note that each function had to be called as part of the module.
    - `math.sin(radangle)`
- We can import specific functions from a module directly into our program's *namespace* (i.e. list of names of variables / functions we have defined) using the following syntax:
    - `from` *`modulename`* `import` *`functionname`*

- We can then use those functions directly, but only those functions we have imported.

    -
      ```
      from math import sin, cos, tan, radians
      angle = float(input("Enter an angle in degrees: "))
      radangle = radians(angle)
      print(sin(radangle))
      print(cos(radangle))
      print(tan(radangle))
      ```
- If we wanted to, we could bring everything into the namespace…

    -
      ```
      from math import *
      ```
- ...but this could bring in things you weren't expecting...

    -
      ```
      from math import *

      a = 1
      b = 2
      c = 3
      d = 4

      print(a)
      print(b)
      print(c)
      print(d)
      print(e)
      ```
    - `e` is defined in `math`!

<u>Writing Your Own Modules</u>

- To create your own module, write the functions and variables that make up your module into a program and save it as a regular Python (.py) script.
- You can then import the module into another program as normal, using `import` *`filename`*.
    - Note that the .py extension is not part of the module name!
- Example:
    - my_module.py:

        -
          ```
          def my_cool_function():
              print("Hello from the cool function!")
          ```
    - my_program.py:

        -
          ```
          import my_module
          my_module.my_cool_function()
          ```

<u>Exercise: Fun with Modules</u>

- Modules exist for all kinds of things!
- `turtle` is a fun introduction to programming, where you can move a turtle around the screen using various commands.
- To play with it, enter into the Python *interpreter*:

    -
      ```
      import turtle
      turtle.home()
      ```

    - The first line imports turtle, and the second will cause the graphical window to appear.
- Take a look at the turtle documentation at https://docs.python.org/3/library/turtle.html, and try making the turtle do things by entering function calls into the interpreter!
- Once you are comfortable with this, try writing your own full programs that make the turtle do interesting things.
    - The window will close when your program exits, so you will probably want to have an infinite loop of your turtle doing things forever so it stays open.
    - As an example, this program will make the turtle trace out an octagon:

        -
          ```
          import turtle

          while True:
              turtle.forward(50)
              turtle.right(45)
          ```
    - Important: don't call your program *turtle.py*, or your program won't work!
        - If you do this, the `import turtle` line will try to import your own *turtle.py* file, rather than the `turtle` module!

Summary

- We can write our own functions, to make modular, reusable code.
- We can use functions provided by other developers in the form of modules.
- Modules let us write advanced programs, without needing to reinvent the wheel ourselves!

Next lesson: Classes and Object Oriented Programming