Modules

Modules

A *module* is a collection of functions that we can use to do more powerful things with our Python programs.

Lots of modules exist, written by other developers.

You can use a module without needing to understand what it is doing behind the scenes!

You can also write your own modules.

Using a Module

Import a module:

```
import modulename
```

Use a module function:

```
modulename.functionname(param1, param2, etc...)
```

Example: `math` Module

The `math` module contains advanced mathematical functions.

```
import math
print(math.sin(0.5))
```

https://docs.python.org/3/library/math.html

Exercise: Using `math`

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.

Exercise: Using `math`

```
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))
```

Importing Parts of a Module

Note that each function is called as part of its module:

```
math.sin(radangle)
```

We can choose to import specific functions, directly into our namespace:

```
from modulename import functionname
```

Importing Parts of a Module

```
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))
```

Directly Importing an Entire Module

We can directly import an *entire* module into our namespace:

```
from math import *
```

...but this may have unintended results!

Directly Importing an Entire Module

```
from math import *

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

print(a)
print(b)
print(c)
print(d)
print(e)
```

Writing Your Own Modules

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!

Writing Your Own Modules

my_module.py:

```
def my_cool_function():
    print(“Hello from the cool function!”)
```

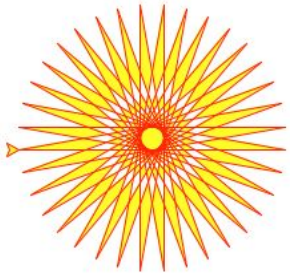
my_program.py:

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

Exercise: Fun with Modules

The `turtle` module is a fun introduction to programming and modules. Move a ‘turtle’ around the screen, tracing a path!

https://docs.python.org/3/library/turtle.html



First, in the interpreter:

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

Try out various commands!

Then, write a full program to make the turtle do something. You’ll want to use an infinite loop:

```
import turtle

while True:
    turtle.forward(50)
    turtle.right(45)
```

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!

How do we structure truly complex programs and data structures?