Importing Modules

Python comes with an extensive library of built-in modules that make it easy to accomplish everyday tasks. With just a few lines of code, you can do anything from generating random numbers and drawing graphics to sending emails and accessing websites.

Content Learning Objectives

After completing this activity, students should be able to:

- Use the `random` module to generate random float and integer sequences.
- Explain the purpose of the common line `if __name__ == "__main__"`.
- Summarize several built-in modules, including `random` and `turtle`.

Process Skill Goals

During the activity, students should make progress toward:

- Navigating the Python standard library documentation. (Information Processing)

Facilitation Notes

This activity addresses misconceptions about the `import` statement and shows how to write "longer" programs using multiple source files. Model 1 uses the `random` module as an example: someone else has already written code to generate random numbers, and you can import this code into your own program.

During Model 2, students learn how to write their own modules that can be imported. To save time, be sure to provide students with the source files at the beginning of the activity. When reporting out, spend time discussing the Python idiom `if __name__ == "__main__"`.

Model 3 introduces the `turtle` module, but it's more about navigating the Python library documentation. Before students answer the questions, you might show them how to bring up the documentation page (https://docs.python.org/3/library/turtle.html). Have each team report out on the last question and share with the class what they found in the standard library.

Model 1 Random Numbers

You can generate a sequence of numbers using the Python `random` module. A mathematical function is used to produce the sequence based on a ***seed*** value. (If no seed is given, the current system time is used.) The sequence is more accurately described as ***pseudorandom***, since its output is inherently predictable.

Python code	Shell output
`import randint`	ImportError
`import random`	
`randint(1, 10)`	NameError
`random.randint(1, 10)`	integer in range [1..10]
`from random import randint`	
`randint(1, 10)`	integer in range [1..10]
`seed(100)`	NameError
`random.seed(100)`	
`random.random()`	0.1456692551041303 – decimal in range [0..1)
`random.random()`	0.45492700451402135 – decimal in range [0..1)
`random.seed(100)`	
`random.random()`	0.1456692551041303 – decimal in range [0..1)
`random.random()`	0.45492700451402135 decimal in range [0..1)

Questions (20 min) Start time: ________

1. What is the name of the module that must be imported before generating a random number?

random

2. Based on Model 1, what are the names of three functions defined in the `random` module?

randint(), random(), seed()

3. Identify the syntax of the statement to import:

a) a module import module

b) a function from module import function

4. Identify the syntax of a function call assuming:

a) the module was imported module name "." function name

b) the function was imported function name

5. How could you eliminate the need for typing the word “random” twice (in a function call) to generate a random number?

from random import random

6. Compare the shell output of your team with at least one other team. Describe the similarities and differences observed.

Different random numbers initially and then same sequence of numbers after a seed is set – but unique for each different seed argument.

7. What is the effect on the random numbers generated after calling the `seed` method?

same sequence of random numbers generated

8. Describe one reason to set the same seed each time a program is run, and one reason to not use the `seed` method.

Use: debugging and testing. Don’t use: when it’s supposed to be random.

9. Run `random.random()` multiple times. Based on the results, describe:

a) the range of numbers returned by the `random` function between [0..1) exclusive

b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?) uniform

10. Run `random.randint(1, 10)` multiple times. Based on the results, describe:

a) the range of numbers returned by the `randint` function between [0..10] inclusive

b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?) uniform

Model 2 Multiple Modules

Create a new file `move.py`, and enter the code:

```python
import random

def angle():
    number = random.randint(-90, 90)
    return number

print("in move: __name__ ==", __name__)
print("will always execute: angle ==", angle())

if __name__ == "__main__":
    print("only if True: angle ==", angle())
```

Run `move.py`, and record the output below. (numbers will vary)

`Output Line 1`	`in move: __name__ == __main__`
`Output Line 2`	`will always execute: angle == 68`
`Output Line 3`	`only if True: angle == -39`

Create a new file `stop.py` (in the same directory), and enter the code:

```python
import move

print("in stop: __name__ ==", __name__)
print("from module: angle ==", move.angle())
```

Run `stop.py`, and record the output below. Draw an arrow from each line of output to its corresponding print statement in the code.

`Output Line 1`	`in move: __name__ == move`	arrow to Line 7 of `move.py`
`Output Line 2`	`will always execute: angle == 74`	arrow to Line 8 of `move.py`
`Output Line 3`	`in stop: __name__ == __main__`	arrow to Line 3 of `stop.py`
`Output Line 4`	`from module: angle == -11`	arrow to Line 4 of `stop.py`

Questions (15 min) Start time: ________

11. Upon execution of `move.py`:
    a) what is the value of the variable `__name__`? `__main__`

    b) does the output correspond solely to the print statements contained in this file? yes

12. Upon execution of `stop.py`:
    a) what is the value of the variable `__name__` from the print statement in `move` `move`

    b) what is the value of the variable `__name__` from the print statement in `stop` `__main__`

    c) does the output correspond solely to the print statements contained in this file? no

13. What was the reason to include the `import move` statement in `stop.py`?

To use the angle function defined in `move.py`.

14. Based on the output of `stop.py`, describe what happens (as a side effect) when another module is imported.

All the code in the imported file, including top-level print statements, is executed.

15. What line in `move.py` did not print when `stop.py` was executed? Why?

The print "only if True" statement inside `if __name__ == "__main__"` was False because `__name__ == "move"` that time.

16. In order for the output of `stop.py` to correspond solely to the print statements contained in `stop.py`, what modifications need to be made to `move.py`?

Move all the print statements inside `if __name__ == "__main__"`.

17. Describe what code in general to include inside `if __name__ == "__main__"`, and why.

Code that you don't want to be executed when the module is imported.

Model 3 Turtle Graphics

The `turtle` module can be used to create graphics. Create a new file `draw.py` (in the same directory), and enter the following code. Run the program and see what happens.

```python
import move
import turtle

def randomwalk(steps):
    turtle.shape("turtle")
    turtle.color("green")
    for i in range(steps):
        turtle.left(move.angle())
        turtle.forward(10)
    turtle.bye()

if __name__ == "__main__":
    randomwalk(100)
```

Questions (10 min) **Start time:** ________

18. For each outcome, describe the type of edit necessary to `draw.py` and `move.py`:

a) a blue turtle change the argument of `turtle.color()` to "blue"

b) a longer simulation change the argument of `randomwalk()` to a number greater than 100

c) a smaller range of angles (e.g., -45 to 45) that define the direction of the turtle

change the argument(s) of `random.ranint()` in the `angle` function defined in `move.py`

d) a random range of integers (e.g., 10 to 20) that define the length of a turtle move

import random and change argument of `turtle.forward()` to `random.ranint(10,20)`

19. Describe the type of edit necessary to produce the same outcome in Question #18d if the argument of `forward` is `move.length()` instead of 10:

Add a function named `length` in the file `move.py`.

20. Go to https://docs.python.org and click the modules link in the upper right corner. Find at least two built-in modules that interest you, and summarize what functions they provide.

See also https://github.com/vinta/awesome-python.