Lecture 6: Modules (and methods)

Course outline

- Part 1 Introduction to Computing and Programming (first 2 weeks):
  - Problem solving: Problem statement, algorithm design, programming, testing, debugging
  - Scalar data types: integers, floating point, Boolean, others (letters, colours)
  - Arithmetic, relational, and logical operators, and expressions
  - Data representation of integers, floating point, Boolean
  - Composite data structures: string, tuple, list, dictionary, array
  - Sample operations on string, tuple, list, dictionary, array
  - Algorithms (written in pseudo code) vs. programs
  - Variables and constants (literals): association of names with data objects
  - A language to write pseudo code
  - Programming languages: compiled vs. interpreted programming languages
  - Python as a programming language
  - Computer organization: processor, volatile and non-volatile memory, I/O

Course outline (may change a bit)

- Part 2 Algorithm design and Programming in Python (balance 11 weeks):
  - Arithmetic/Logical/Boolean expressions and their evaluations in Python
  - Input/output statements (pseudo code, and in Python)
  - Assignment statement (pseudo code, and in Python)
  - Conditional statements, with sample applications
  - Iterative statements, with sample applications
  - Function sub-programs, arguments and scope of variables
  - Recursion
  - Modules
  - Specific data structures in Python (string, tuple, list, dictionary, array), with sample applications
  - Searching and sorting through arrays or lists
  - Handling exceptions
  - Classes, and object-oriented programming
  - (Time permitting) numerical methods: Newton Raphson, integration, vectors/matrices operations, continuous-time and discrete-event simulation

Modules

- A module is a collection of codes typically used to organize reusable functions
- A module can contain executable statements
- Useful in splitting a large program into multiple smaller pieces

- Creating a module
  - Use a text editor to create my_module.py file with function definitions and statements
  - Place the file in your working directory

  - BUT, first …

Available modules

- Presently, several modules are already available in Python
- Examples (search through https://www.w3schools.com/python/python_modules.asp):
  - **math**: built-in math module (over and above math functions such as max, abs)
    - Trignometry, etc.
  - **numpy**: Python library used for working with arrays
    - for problems in linear algebra, vectors, matrices, Fourier transforms, random nos.
  - **matplotlib**: low level graph plotting library
  - **scipy**: library of functions for scientific computation
    - Including functions for optimization, statistics and signal processing
  - Many more … including those for data processing, machine learning, etc.

Available module: random

- Several modules are available in Python

```
# Module-1, using module "random", & methods "randint" and "random"
# maxVal function returns larger of random numbers x and y
def maxVal(x, y):
    if x > y:
        return x
    else:
        return y
import random          # can work with random numbers
r1 = random.randint(1, 100) # "randint" is a "method"
r2 = random.randint(1, 100)
print(r1, r2)
print(maxVal(r1, r2))
# work with floating random numbers
r3 = random.random()*10  # random is module, also "method"
                # random method produces 0 <= float < 1.0
print(r3)
```

This is my_app.py

Available module: random

Essentially:
File name of module: random.py
Functions in module: randint, random

- Several modules are available in Python

```
# Module-1, using module "random", & methods "randint" and "random"
# maxVal function returns larger of random numbers x and y
def maxVal(x, y):
    if x > y:
        return x
    else:
        return y
import random          # can work with random numbers
r1 = random.randint(1, 100) # "randint" is a "method"
r2 = random.randint(1, 100)
print(r1, r2)
print(maxVal(r1, r2))
# work with floating random numbers
r3 = random.random()*10  # random is module, also "method"
                # random method produces 0 <= float < 1.0
print(r3)
```

This is my_app.py

Available module: random

Essentially:
File name of module: random.py
Functions in module: randint, random

- Several modules are available in Python

```
# Module-1, using module "random", & methods "randint" and "random"
# maxVal function returns larger of random numbers x and y
def maxVal(x, y):
    if x > y:
        return x
    else:
        return y
import random          # can work with random numbers
r1 = random.randint(1, 100) # "randint" is a "method"
r2 = random.randint(1, 100)
print(r1, r2)
print(maxVal(r1, r2))
# work with floating random numbers
r3 = random.random()*10  # random is module, also "method"
                # random method produces 0 <= float < 1.0
print(r3)
```

This is my_app.py

Output:

```
50 98
98
8.90243920837131
```

Available module: math

- Several modules are available in Python

```
# Module-2, using module "math", and methods such as 'sin(x)'
# compute min, max, abs of integers
import math
import random as rand# give the module a new name
x = min(5, 10, 25)          # 'min', 'max' function available in Python
y = max(5, 10, 25)          # don't need to import math for these
z = -x
print(x, y, z, abs(z))
#
r3 = rand.random() * 2 * math.pi
print(r3, math.sin(r3), math.cos(r3))
```

Available module: math

Essentially:
File name of module: math.py
Functions in module: sin, etc.

- Several modules are available in Python

```
# Module-2, using module "math", and methods such as 'sin(x)'
# compute min, max, abs of integers
import math
import random as rand# give the module a new name
x = min(5, 10, 25)          # 'min', 'max' function available in Python
y = max(5, 10, 25)          # don't need to import math for these
z = -x
print(x, y, z, abs(z))
#
r3 = rand.random() * 2 * math.pi
print(r3, math.sin(r3), math.cos(r3))
```

Output:

```
5 25 -5 5
5.305658970563565  -0.8291169447094159    0.5590752113944271
```

Modules – creating your own

Essentially:
File name of module: circle2.py
Data objects: pi
Functions in module: area, circumference, etc.

- Name this file as circle2.py
- Each module is stored in a separate file
- Save it in your working directory

```
# MyOwnModule -1

# module for circles, spheres - named 'circle2'
# circle2.py placed in directory named \Documents\Python Scripts
"""Collection of functions related to circle, spheres"""
#
pi = 3.14159      # pi is assigned value when circle2 is 'import'ed
def area(radius):
    return(pi * (radius ** 2))
def circumference(radius):
    return(2 * pi * radius)
def sphereSurface(radius):
    return(4.0 * area(radius))
def sphereVolume(radius):
    return((4.0/3.0) * pi * (radius ** 3))
```

Modules – creating your own

Essentially:
File name of module: circle2.py
Data objects: pi
Functions in module: area, circumference, etc.

- Name this file as circle2.py
- Each module is stored in a separate file
- Save it in your working directory

```
# MyOwnModule -1
# module for circles, spheres
# placed in directory named \
"""Collection of functions re
#
pi = 3.14159      # pi is assi
def area(radius):
    return(pi * (radius ** 2)
def circumference(radius):
    return(2 * pi * radius)
def sphereSurface(radius):
    return(4.0 * area(radius)
def sphereVolume(radius):
    return((4.0/3.0) * pi * (
```

```
# "first.py" uses circle2 module
# imports circle2 to compute related values
import circle2
print(circle2.pi)
print(circle2.area(2.0))
print(circle2.circumference(2.0))
print(circle2.sphereSurface(2.0))
print(circle2.sphereVolume(2.0))
```

Output:

```
3.1416
12.5664
12.5664
50.2656
33.5104
```

Modules – creating your own

- Name this file as circle2.py
- Each module is stored in a separate file
- Save it in your working directory

```
# MyOwnModule -3
# module for circles, spheres – named "circle2"
# in directory named Python Scripts
"""Collection of functions related to circle, spheres"""
pi = 3.14159
def area(radius):
    return(pi * (
def circumference
    return(2 * pi
def sphereSurface
    return(4.0 *
def sphereVolume(
    return((4.0/3
```

```
# imports circle2 to compute related values
from circle2 import *
print(pi)
print(circle2.area(2.0))
```

Correct usage:
Print(area(2.0))

Output:

```
3.1416
NameError: name 'circle2' is not defined
```

Locating a module

- When a module is imported, Python interpreter looks for the module in the following sequence
  - The current directory, else
  - Each directory in the shell variable PYTHONPATH, else
  - Default directory in Linux /usr/local/lib/python/
- PYTHONPATH may be set in your Windows machine. To do so, see: https://www.geeksforgeeks.org/pythonpath-environment-variable-in-python/

Getting help on modules

- help(module):
- Displays the documentation associated with the object

```
>>> import math
>>> help(math)
Help on built-in module math:
NAME
    math
DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.
FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.
    acosh(x, /)
        Return the inverse hyperbolic cosine of x.
    asin(x, /)
        Return the arc sine (measured in radians) of x.
    asinh(x, /)
        Return the inverse hyperbolic sine of x.
    atan(x, /)
        Return the arc tangent (measured in radians) of x.
Etc.
```

Getting help on modules

- help(module):
- Displays the documentation associated with the object

```
>>> import circle2
>>> help(circle2)

NAME
    circle2 - Collection of functions related to circle, spheres
FUNCTIONS
    area(radius)
    circumference(radius)
    sphereSurface(radius)
    sphereVolume(radius)
DATA
    pi = 3.1416
FILE
    c:\users\bnjai\documents\python scripts\circle2.py
```

ICE 6.1 Section A

https://docs.google.com/forms/d/e/1FAIpQLSdZgS9uxYA4vtPEWYCxD_wo0E5Aic3FaN_DotBwti8sgrKY-Q/viewform?usp=sf_link