Classes

[https://docs.python.org/3/tutorial/classes.htmlLinks to an external site.](https://docs.python.org/3/tutorial/classes.html)

Python organizes data and code in structures called classes.  The complete subject of object oriented programming is a bit too much to cover in this course, but we will use enough of it to understand how classes work and how to create them.

A Class is a structure that contains data and/or functions.  We have been using classes without really knowing it when we do things like

s="{:f}".format(x)

In this case the "{:f}" is an instance of the string class str.  The str class has the data that is the string itself, but it also has functions that operate on the string like format(), split(), replace(), lower() etc.  You call teh functions for the str class by using a "." after the string.

In python everything is a class, so almost everything has functions associated with it.

Background:

SPICE is a circuit simulation program that you will use extensively as an Electrical Engineer.  This assignment is the start of writing your own SPICE program.  First we need to be able to read a SPICE netlist.  A netlist is a file with a specific format that describes a circuit.  Each line of the file contains one component, its parameters and its connections.

We will start with a resistor.  A netlist line for a resistor looks like:

RXX <positive node> <negative node> <value>

For example "R1 VCC GND 1k" is a 1k ohm resistor named "R1" connected between nodes "VCC" and "GND"

In the same way, we can define inductors and capacitors

CXX <positive node> <negative node> <value>

LXX <positive node> <negative node> <value>

And a current supply

IXX <positive node> <negative node> <DC value> <AC magnitude=0> <AC frequency=0>

But for a current supply, the value can be either just the DC value or the ac parts are optional and default to 0

Component names can be any string, but must start with R,C,L,or I

Net names can be any string but "GND" is special and must be in all circuits.

Libraries

We have also been using libraries all along.  A library is simply a python file which contains functions like scanf, numpy, scipy.  To use those functions you import the file.

import hw05\_lib

Then the functions or classes are referenced by dot notation so to use the resistor class in your library, you reference it as hw05\_lib.resistor

If the library name is long you can give it a shorter alias like

import hw05\_lib as h5

then you can reference it as h5.resistor

1. Create a python library "hw05\_lib.py" in the src directory.
   1. Define a class named "resistor" to contain the information for a resistor
      1. data fields:
         1. name (string)
         2. positive node name (string)
         3. negative node name (string)
         4. value (float)
      2. functions:
         1. \_\_init\_\_(string)
            1. create a new resistor instance with values from the given string.
         2. current(float voltage)
            1. returns float current through the resistor for given voltage
         3. read(string)
            1. fills in the resistor class data from a string

If string does not start with an "R" or "r" raise an ValueError and print "Resistor must start with an R: "+ the offending string

If it is missing one of the values raise a ValueError and print "Resistor requires at least 4 values: "+ the offending string

* + - 1. print()
         1. prints out the contents of a resistor as

<name>\t<positive node>\t<negative node>\t<value (format %.2e)>\n

* 1. Define a library function (not part of resistor class)
     1. decodeValue(string)
        1. take a string value formatted according SPICE and return a float value
        2. argument is a string holding a value.  This string may have suffixes such as "1.5k" "2.3Meg" "1.5u"

valid suffixes are: T=1e12, G=1e9, Meg or X=1e6, k=1e3, m=1e-3, u=1e-6, n=1e-9, p=1e-12, f=1e-15, a=1e-18

they are case insensitive

* + - 1. If it is not a valid value the raise a ValueError and print "Invalid Value: "+ the offending string
  1. All of the above should raise exceptions as appropriate on failure

1. Write a test program "hw05a.py" that imports the library and prompts the user to enter the parameters of a resistor on a single line separated by any whitespace (any number of spaces or tabs) such as

R1 VCC GND 1k

* 1. prompt the user "Enter a resistor: "
  2. parse the string that the user inputs into a Resistor structure.
     1. If the input is invalid then print "Invalid input" and exit
     2. If the input is valid then print the resistor information using the print function
  3. prompt the user "Enter a voltage: "
     1. parse the input string in to a value.  The user may enter any valid SPICE or float value such as 1.23 or 5k.
     2. If the input is invalid then print "Invalid value" and exit
  4. Calculate the current using the current function and print it to the screen as in scientific notation with 2 decimal precision.
     1. "I = <value %.2e>\n"

1. Write a test program that takes a netlist file as a command line parameter and reads the file, creating a list of resistor classes from the lines of the file.
   1. ./hw05b.py <netlist filename>
   2. If no filename is given, exit with message "No netlist file given"
   3. Print each resistor in the file

You might like to use [https://pypi.org/project/scanf/  Links to an external site.](https://pypi.org/project/scanf/)to parse the values

And [https://docs.python.org/3/library/dataclasses.htmlLinks to an external site.](https://docs.python.org/3/library/dataclasses.html) for data structures in python

[Previous](https://weber.instructure.com/courses/560266/modules/items/9367109)[Next](https://weber.instructure.com/courses/560266/modules/items/9160705)