CS417 Programming Assignment #5

- Due: Monday, March 9th.
- Late penalty: Tue: -5%, Wed -10%, Thu -20%, Fri -50%, Sat -100%

Goals

For this assignment, you must create two modules (download these files which have starting code):

- `complex.py` , which defines the class `Complex`, representing a complex number.
- `julia.py` , which draws the Julia set.

Your tasks

1. Python includes a module `cmath` which handles complex arithmetic. In this assignment, you will make your own version.

   In case you haven't seen complex numbers before, I am posting a brief primer. It's available with the assignment files.

   Here are the methods of the `Complex` class:

Method	Description	Done?
`__init__`	Constructor. Initializes the real and imaginary parts of the complex number.	Yes
`plus`	Add `self + other`, and return the resulting sum, a new `Complex` instance.	Yes
`minus`	Subtract `self - other`, and return the resulting difference, a new `Complex` instance.	NO
`times`	Multiply `self x other`, and return the resulting product, a new `Complex` instance.	NO
`over`	Divide `self / other`, and return the resulting quotient, a new `Complex` instance.	NO
`conjugate`	Return the conjugate of `self`, a new `Complex` instance.	NO
`magnitude`	Return the magnitude of `self`, a `float`.	NO
`real`	Return the real part of `self`, a `float`.	Yes
`imag`	Return the imaginary part of `self`, a `float`.	NO
`equals`	Return `True` if both parts of `self` are equal to those of `other`, `False` otherwise.	NO
`__str__`	Return a string representation of `self`.	NO

The `__str__` method should be such that this code outputs `2.0 + 3.0i`:

```
z = Complex(2, 3)
print (z)
```

---

2. Implement the module `julia.py`, which lets a user explore the Julia set visually.

   A Julia set depends on a complex constant *c*. The Julia set consists of all the points *x y* that obey the convergence condition.

   To explore the Julia set, we try a grid of values *x y* and see if the convergence condition applies.

Convergence condition:

- Initially, set the complex number z to be (x + yi).
- Then, evaluate the following assignment repeatedly:
- $z = z^2 + c$.
- Equivalently, we can use the Complex class to do this:
- `z = z.times(z).plus(c)`
- After repeating this action many times, two things may happen:
  - *z* grows very large; the process did not converge, or
  - *z* stays small; the process is converging.

  If the process converges, the point *x y* is in the Julia set. If it doesn't, the point is not in the set.

  In practical terms, we can run

  - $z = z^2 + c$

  repeatedly, until one these things happen:

  - `z.magnitude() > 2` (so the point (*x y*) is *not* in the set), or
  - 100 iterations have been run (the point is *probably* in the set)

Starting Code

The starting code implements the following functions:

Function	Description	Completed?
`main`	Gets the constant c from the command line, if the user entered it. Then calls `show_set` to	NO

	display a rectangular window into the Julia set, and lets the user move the window. Code is provided to move the window to the right. You must complete this function, to move the window left, up, down, zoom in and zoom out.	
`show_set`	Displays a rectangular region of the Julia set.	Yes
`julia_converges`	Returns True if the point x y belongs to the Julia set, and False otherwise.	NO

Turning in Your Work

When you finish, go to mycourses.unh.edu, find CS417, find assignment 5, and click the “Submit” button. Then upload `complex.py` and `julia.py`.