Homework 1: Python Numeric Functions

See Webcourses and the syllabus for due dates.

General Directions

This homework should be done individually. Note the grading policy on cooperation carefully if you choose to work together (which we don't recommend).
In order to practice for exams, we suggest that you first write out your solution to each problem on paper, and then check it typing that into the computer.
You should take steps to make your Python code clear, including using symbolic names for important constants.
Tests that are provided in hw1-tests.zip, which consists of several python files with names of the form `test_`f`.py`, where f is the name of the function you should be writing. Your function f should go in a file named f`.py` and the function itself should be named f. These conventions will make it possible to test using pytest.
Pytest is installed already on the Eustis cluster. To install pytest on your own machine, execute the following command from the shell (or terminal window):

```
pip3 install -U pytest
```

See pytest.org's installation page for more details.
After you have pytest installed, and after you have written your solution for a problem that asks for a function named f, you can run pytest on our tests for f by executing at a command line

`pytest test_`f`.py > `f`_tests.txt`

which puts the output of the testing into the file f`_tests.txt`.
You can also run pytest from within IDLE. To do that first edit a test file with IDLE (so that IDLE is running in the same directory as the directory that contains the files), then from the Run menu select "Run module" (or press the F5 key), and then execute the following statements:

```
import pytest
pytest.main(["test_f.py", "--capture=sys"])
```

which should produce the same output as the command line given above. Then you can copy and paste the test output into the file f`_tests.txt` to hand in.

What to turn in

For problems that ask you to write a Python function, upload your code as an ASCII file with suffix `.py`, and also upload the output of running our tests (as an ASCII file with suffix `.txt`).
In your Python code files, include your name (and any statements about cooperation or collaboration as required by the grading policy, as needed) as comments at the top of the file. Don't edit the output of our tests.

Problems

1. (6 points) [Programming] Define a Python function, `moons(years)`, which takes a number, `years`, and returns return the number of lunar months that a person that age has lived.

   For this problem we will follow many cultures and use the "sidereal month" as the definition of a lunar month, which is the time it takes for the moon to return to the same (apparent) place in the sky. A sidereal month lasts 27.321661 days. A year is 365.25 days long.

Tests for this problem appear in Figure 1. Run the tests as indicated above and save the output in a `.txt` file to hand in.

```
# $Id: test_moons.py,v 1.1 2019/01/11 16:48:42 leavens Exp $
from moons import moons
from math import isclose
Sidereal_Month = 27.321661 #days
Days_in_Year = 365.25
def test_moons():
    assert isclose(moons(1), Days_in_Year / Sidereal_Month)
    assert isclose(moons(18), 18 * Days_in_Year / Sidereal_Month)
    assert isclose(moons(19), 19 * Days_in_Year / Sidereal_Month)
    assert isclose(moons(20), 20 * Days_in_Year / Sidereal_Month)
    assert isclose(moons(30), 30 * Days_in_Year / Sidereal_Month)
    assert isclose(moons(40), 40 * Days_in_Year / Sidereal_Month)
    assert isclose(moons(50), 50 * Days_in_Year / Sidereal_Month)
    # oldest human lived 122 years and 164 days
    assert isclose(moons(122.449315), 122.449315 * Days_in_Year / Sidereal_Month)
    # some turtles and a clam have lived 500 years
    assert isclose(moons(500), 500 * Days_in_Year / Sidereal_Month)
    # a bristlecone pine is thought to be about 5000 years old
    assert isclose(moons(5000), 5000 * Days_in_Year / Sidereal_Month)
```

Figure 1: Tests for `moons`, found in the file `test_moons.py`.

Remember to turn in both your file `moons.py` and the output of running our tests. These should be submitted to webcourses as ASCII text files that you upload.

2. (8 points) [Programming] Define a Python function, `discriminant(a,b,c)` that when given 3 numbers, `a`, `b`, and `c`, returns the quadratic discriminant $\mathtt{b}^2 - 4 \cdot \mathtt{a} \cdot \mathtt{c}$.

   Tests for this problem appear in Figure 2.

```
# $Id: test_discriminant.py,v 1.1 2019/01/12 05:00:06 leavens Exp leavens $
from discriminant import discriminant
from math import isclose
def test_discriminant():
    assert isclose(discriminant(0, 0, 0), 0)
    assert isclose(discriminant(3, 4, 5), -44)
    assert isclose(discriminant(3.0, 5.0, 4.0), -23.0)
    assert isclose(discriminant(1.0, 10.0, 1.0), 96.0)
    assert isclose(discriminant(27.0, 2.0, 1.0), -104.0)
    assert isclose(discriminant(3.14, 2.781, 0), 7.733961000000001)
    assert isclose(discriminant(6.0, 10.0, -1.0), 124)
    assert isclose(discriminant(1.5e-2,6.7789,3.1e-4), 45.95346661000001)
```

Figure 2: Tests for `discriminant`, found in the file `test_discriminant.py`.

Remember to turn in both your file `discriminant.py` and the output of running our tests. These should be submitted to webcourses as ASCII text files that you upload.

3. (8 points) [Programming] A "light-second" is a unit of length, the distance that light travels in a second; it is defined to be exactly 299,792,458 meters. Define a Python function, `lightSecs(miles)` that when given a number of miles returns the equivalent of that length in light-seconds.

   Note that a mile is 1609.344 meters.

   Tests for this problem appear in Figure 3.

```
# $Id: test_lightSecs.py,v 1.1 2019/01/12 05:00:06 leavens Exp leavens $
from lightSecs import lightSecs
from math import isclose
def test_lightSecs():
    assert isclose(lightSecs(1), 5.368193752225749e-06)
    assert isclose(lightSecs(299792458 / 1609.344), 1.0)
    assert isclose(lightSecs(1000), 0.005368193752225748)
    # distance to Mars (on average) from earth is about 140 million miles
    assert isclose(lightSecs(140e6), 751.5471253116048)
    # distance to Voyager 1 on 1/11/2019 at about 11:28pm
    assert isclose(lightSecs(13491749492), 72426.32532954932)
    # approximate distance to the closest star, Proxima Centari
    assert isclose(lightSecs(25e15), 134204843805.6437)
```

Figure 3: Tests for `lightSecs`, found in the file `test_lightSecs.py`.

Remember to turn in both your file `lightSecs.py` and the output of running our tests. These should be submitted to webcourses as ASCII text files that you upload.

4. (10 points) [Programming] Define a Python function, `marsweight(earthweight)` that when given a person's weight in pounds (of force) on Earth, `earthweight`, returns the number of pounds (of force) that a scale (brought from earth) would register for that person on the surface of Mars.

   You can calculate the weight by using Newton's law of gravitation, which says that the gravitational force (which is what weight is) between two masses m_1 and m_2 separated by a distance of r meters is given by

   $$F = G \cdot m_1 \cdot m_2 / r^2. \tag{1}$$

   The mass of Mars is approximately $0.64171e24$ kg[1] The radius of Mars is approximately 3389.5 kilometers. Here the force would be in units of Newtons (N). Newton's gravitational constant, G, is equal to $6.764e-11\ N \cdot m/kg^2$. On Earth, a pound (of weight) is equivalent to 0.453592 kg (of mass). Conversely, 1 Newton (N) of force is equivalent to 0.224809 pounds of force

   Hint: convert the given weight to a mass (in kg), then use Newton's law of gravitation to solve for the force (in N) on the surface of Mars between Mars and that mass, then covert back to pounds of force.

   Tests for this problem appear in Figure 4 on the next page.

   Remember to turn in both your file `marsweight.py` and the output of running our tests. These should be submitted to webcourses as ASCII text files that you upload.

Points

This homework's total points: 32.

[1] These facts about Mars are from `https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html`.

```
# $Id: test_marsweight.py,v 1.1 2019/01/12 05:00:06 leavens Exp leavens $
from marsweight import marsweight
from math import isclose

def test_marsweight():
    assert isclose(marsweight(1), 0.3801305841547296, rel_tol=2e-2)
    assert isclose(marsweight(120), 45.615670098567556, rel_tol=2e-2)
    assert isclose(marsweight(150), 57.019587623209425, rel_tol=2e-2)
    assert isclose(marsweight(200), 76.02611683094592, rel_tol=2e-2)
```

Figure 4: Tests for `marsweight`, found in the file `test_marsweight.py`.