Creating Modules

Modules

1. Create a module, **stringinfo.py**, that contains the following functions.
   - *length*(s) – determine the number of characters in a string *s*. Returns the length.
   - *is_alphabetic*(s) – determines if a string s consists solely of alphabetic characters. Returns `True` or `False`.
   - *character_count*(s, c) – counts the number of instances of a character, *c*, in a string, *s*. Returns the number of occurrences.

   Test your functions on some strings input by the user. Note that some of these functions that you wrote are already included in Python. We will look at many of them soon.
2. Create a module, **geometry.py**, that contains the following functions. All return a single value.
   - *area_rectangle*(length, width)
   - *area_circle*(radius)
   - *circumference*(radius)
   - *volume_rectangular_prism*(length, width, height)
   - *volume_cylinder*(radius, height)
   - *surface_area_rectangular_prism*(length, width, height) - calls *area_rectangle*
   - *surface_area_cylinder*(radius, height) - calls *area_circle*, *area_rectangle* and *circumference*

   Write your main program in another file that imports your geometry module. Ask the user to select either a rectangular prism or a cylinder, then to enter relevant values. Calculate the volume and surface area of the selected figure.
3. Create a module, **lines.py**, that contains the following functions.
   - *slope*(x_1, y_1, x_2, y_2) – calculate the slope of the line through (x_1, y_1) and (x_2, y_2).
   - *y_intercept*(m, x, y) – calculate the y-intercept of a line with slope (m), passing through the point (x, y).
   - *slope_intercept_form*(m, b) – display the equation of the line with slope (m) and y-intercept (b) in the form $y=mx+b$
   - *standard_form*(m, b) – display the equation of the line with slope (m) and y-intercept (b) in the form $Ax+By=C$.

   Use your main program to display some equations of lines in both forms. Be sure to have your functions call other functions as appropriate.
4. Create a module, **radicals.py**, that contains the following functions.
   - *is_square_number*(n) – given a positive integer n, determine whether it is a perfect square. Return True or False.
   - *simplify_radical*(r) – given a square root, $\sqrt{r}$, for some positive integer r, rewrite it in the form $a\sqrt{b}$, where a and b are both positive integers. If $\sqrt{r}$ cannot be simplified, then $a=1$ and $b=r$. Call your *is_square_number* function to help. Return a and b.
   - *print_radical*(a, b) – display a radical. If you like, you can print the "square root symbol" using the command `print("\u221A")`, assuming your font supports Unicode. If $a=1$, it should be omitted. That is, $2\sqrt{3}$ is acceptable, but $1\sqrt{5}$ should display as $\sqrt{5}$ instead.

   Test your module using a variety of radical values.
5. Extend the functionality of your **lines.py** module by adding the following function:
   - *intersect*(m_1, b_1, m_2, b_2) – calculate the point of intersection of the lines $y=m_1x+b_1$ and $y=m_2x+b_2$.