

Introduction

One of the popular topics covered in elementary math classes was geometry and specifically the study of various 2-dimensional shapes. As you might remember, a big part of geometry is the calculation of the **perimeter** (distance around the edge of the shape) and **area** (space taken up within the inside of the shape) of those different shapes. For this assignment, you will be creating a Python program that serves as a perimeter and area calculator for various shapes. Specifically, it will have to work for the following shapes: rectangle, triangle, circle, trapezoid, and hexagon. Don't worry if you don't have those formulae memorized after all this time! Figure 1 further below in this document provides you with all the information you need to know to calculate the perimeter and area for each of these 5 shapes. Note: the distance around the edge of a circle is typically called the circumference, but for this assignment we will use the term perimeter to be consistent with the other geometric shapes.

In this assignment, you will get practice with:

- User input and casting
- Conditionals and nested conditionals
- Printing with formatting
- Evaluating mathematical expressions in Python
- Implementing code based on given instructions
- Importing and using pre-made Python libraries

File

For this assignment, you must create one (1) Python file named: ***shapes.py***. **Make sure you name this file exactly as instructed including having it all lowercase.**

shapes.py

Begin by prompting the user to type in the name of a shape. Determine if they type in one of the five (5) shape names: rectangle, triangle, circle, trapezoid, or hexagon. This must be case-insensitive meaning that these words in any case or combination of cases should work (i.e. RECTANGLE, Circle, HeXaGoN). If they did type in any of these five shape names, then prompt the user again to type in values for each of the dimensions associated with that shape (see Figure 1 below for reference). **Note: all dimensions and calculations must be unit-less so do not type "cm" or "in" or any other unit; just type numeric values.** For example, if they type in circle, then prompt them just for the radius; but if they typed in trapezoid, you have to then prompt them 5 more times to obtain the top, bottom, left, right, and height dimensions. Use the dimensions entered to calculate the perimeter and area for that shape. Print out the name of the shape and the computed perimeter and area, both of which must be **rounded to 2 decimal places**.

The program must also check for invalid input. If the user types in anything other than one of those five shapes when prompted for input the first time, the program must immediately end with the print line saying "*Invalid shape entered*". If the user types in zero (0) or a negative value for at least one of the dimensions for ANY shape, the program must print out a message saying "*Invalid value entered*". Note: you should finish all the prompts for the given shape before displaying this message even if an invalid dimension is already entered (see Example 4 in the **Test Case Examples** section further down).

Note: Python has a lot of built-in libraries that contain pre-made functions and variables that you can use very easily. The math library is one that you are encouraged to use in this assignment. To use this library, add the following line of code at the top of your assignment file:

```
import math
```

The value of π (pi) is stored in a variable in the math library and can be accessed using `math.pi`.

The square root function is also built into the math library and can be accessed using `math.sqrt(x)` where x is the value for which you want to find its square root.

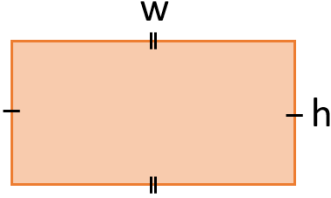
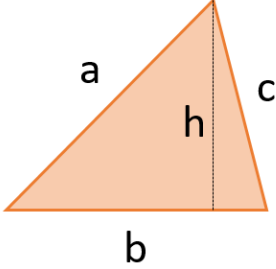
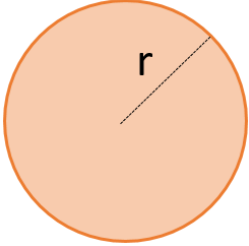
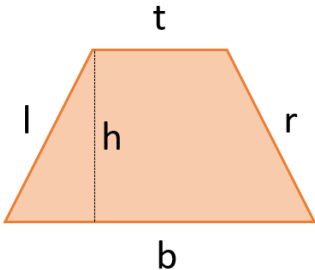
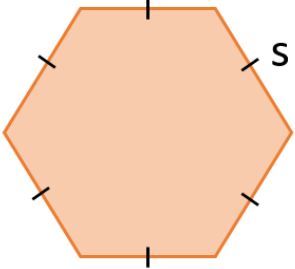
Shape	Diagram	Perimeter	Area
Rectangle		$(2 \times w) + (2 \times h)$	$w \times h$
Triangle		$b + a + c$	$\frac{b \times h}{2}$
Circle		$2 \times r \times \pi$	$\pi \times r^2$
Trapezoid		$t + r + b + l$	$\frac{(t + b)}{2} \times h$
Hexagon		$6 \times s$	$\frac{3 \times \sqrt{3}}{2} \times s^2$

Figure 1. The diagrams and formulae for perimeter and area for the shapes: rectangle, triangle, circle, trapezoid, and hexagon.

Important: Order of prompts

Some of the autograded tests are dependent on a specific order of the prompts for each of the dimensions. Please follow this order for the prompts.

Triangle

- b (base)
- a (side 2)
- c (side 3)
- h (height)

Trapezoid

- t (top)
- b (bottom)
- l (left side)
- r (right side)
- h (height)

The order does **not** matter for rectangles. Circles and hexagons only have one dimension each, so order is not a factor.

Test Case Examples

```
Enter a 2D shape: circle
Enter the radius of the circle: 4.7
Shape: circle
Perimeter: 29.53
Area: 69.40
```

Example 1: the user typed in 'circle' as the shape and was then prompted for the radius for which they in 4.7. The perimeter (circumference) and area were computed and displayed, rounded to 2 decimal places.

```
Enter a 2D shape: triangle
Enter the base length of the triangle: 9.2
Enter the 2nd side length of the triangle: 5.8
Enter the 3rd side length of the triangle: 6.3
Enter the height of the triangle: 4.7
Shape: triangle
Perimeter: 21.30
Area: 21.62
```

Example 2: The perimeter and area of a triangle were computed and displayed, rounded to 2 decimal places.

```
Enter a 2D shape: HEXaGon
Enter the side length of the hexagon: 7.25
Shape: HEXaGon
Perimeter: 43.50
Area: 136.56
```

Example 3: The perimeter and area of a hexagon were computed and displayed, rounded to 2 decimal places. Note that the mixture of cases in the shape did not affect the program.

```
Enter a 2D shape: rectangle
Enter the length of the rectangle: -3.0
Enter the width of the rectangle: 5.9
Invalid value entered
```

Example 4: An invalid (negative) dimension was given so the message "Invalid value entered" was displayed. Note that the invalid value was given in the first of two dimension prompts but the second prompt was still given.

```
Enter a 2D shape: oval
Invalid shape entered
```

Example 5: An invalid shape was entered (oval is not one of the shapes to be included in this program) so the message "Invalid shape entered" was displayed.

Tips and Guidelines

- Constants must be named with all uppercase letters, i.e. *PI*.
- Variables should be named in all lowercase and contain underscores between words for multi-word names, i.e. *rad* or *valid_shape*.
- Add comments throughout your code to explain what each section of the code is doing and/or how it works.
- When you submit your code on Gradescope, look at the autograded test scores. If any tests are failing (i.e. 5/7 means that out of 7 tests, 5 are passing and 2 are failing), it means that your code isn't completely correct.
- Test your code with a variety of different input values and check that it's functioning correctly for all cases. It is your responsibility to test your code for correctness.

Rules

- Read and follow the instructions carefully.
- Submit the shapes.py file as described in the Files section of this document and no additional files.
- Submit the assignment on time. Late submissions will not be accepted or will be marked as 0, except where late coupons are used.
- Forgetting to submit a finished assignment is **not** a valid excuse for missing a due date.
- Make regular backups of your work and store it in multiple places, i.e. USB key, on the cloud, etc. Experiencing technical issues resulting in lost or corrupted files is **not** a valid excuse for missing a due date.
- Submissions must be done on Gradescope. They will **NOT** (under any circumstances) be accepted by email.
- You may re-submit your code as many times as you would like. Gradescope uses your last submission as the one for grading by default. There are no penalties for re-submitting. However, re-submissions that come in after the

due date **will** be considered late and marked as 0, unless you have enough late coupons remaining, in which case late coupons will be applied.

- The top of **each** file you create must include your name, your Western ID, the course name, the date on which you **created** the file (no need to change it each time you modify your code), as well as a description of what the file does. This information is **required**. You must follow this format:

```
"""
```

```
*****
```

```
CS 1026 - Assignment X – Interest Calculator
```

```
Code by: Jane Doe
```

```
Student ID: jdoe123
```

```
File created: October 12, 2024
```

```
*****
```

```
This file is used to calculate monthly principal and interest amounts for a  
given mortgage total. It must calculate these values over X years using  
projected variations in interest rates. The final function prints out all the  
results in a structured table.
```

```
"""
```

- Assignments will be run through a similarity checking software to check for code that looks very similar to that of other students. Sharing or copying code in any way is considered plagiarism and may result in a mark of zero (0) on the assignment and/or reported to the Dean's Office. Plagiarism is a serious offence. Work is to be done **individually**.

What constitutes plagiarism

Plagiarism, or academic dishonesty, comes in a variety of forms including:

- sending any portion of your code to peer(s)
- letting peer(s) look at your code
- using any portion of code from peer(s)
- using ChatGPT or other AI platforms to generate any portion of code or comments

- using code that you found online
- paying someone to do your work
- being paid to do someone else's work
- leaving your work on a computer unattended where peers may be able to find it
- uploading your code to an online repository where peers may be able to access it

Submission

Due: Wednesday, October 2, 2024 at 11:59pm

You must submit the one (1) file (*shapes.py*) to the Assignment 1 submission page on Gradescope. There are several tests that will automatically run when you upload your files. You will see the score of the tests but you will not see what is being tested. It is recommended that you create your own test cases to check that the code is working properly in many different scenarios.

Marking Guidelines

The assignment will be marked as a combination of your auto-graded tests and manual grading of your comments, formatting, etc. Below is a breakdown of the marks for this assignment:

[17 marks] Auto-graded tests

[1 mark] Header section with name, student ID, course info, creation date, and description of file

[1 mark] Comments throughout code

[1 mark] Meaningful variable names

Total: 20 marks

IMPORTANT:

A mark of zero (0) will be given for tests that do not run on Gradescope. It is your responsibility to ensure that your code runs on Gradescope (not just on your computer). Your files must be named correctly, and you must follow all instructions carefully to ensure that everything runs correctly on Gradescope.

A mark of zero (0) will be given for hardcoding results or other attempts to fool the autograder. Your code must work for **any** test cases.

The weight of this assignment is 6% of the course mark.