PACKAGE 3:

Exercise 1

Light Switch Problem

This week's exercise centres around a classic logic puzzle:

A lot of switches

I have a board of light switches, numbered 0,1,2,...,1023.
Each light switch can be either on or off. All switches are initially off.

Step 1:
I flip all of the switches starting at 0.
At this point, all of the light switches are on.

Step 2:
I flip every second switch, starting at 0.
At this point, lights 0,2,4,6,8,... are off.
Lights 1,3,5,7,9,... are still on.

Step 3:
I flip every third switch, starting at 0.
So I flip switches 0,3,6,9,12,...
that is, if a switch is on, I flip it to off.
If a switch is off, I flip it to on.
...

Step 1023:

I flip every 1023'rd switch, starting at 0.
So I flip 0 and 1023.

Question: At this point, which switches are on and which are off?

Solve the Problem

Your first task is to try to solve the puzzle using nothing but your own brain (and possibly a pen & paper). Can you figure out the pattern? (there's a way of finding the answer without actually walking through 1023 steps).

Building a switch

Build a class **LightSwitch** with the following properties:

- When I create a switch, I should be able to set it's default state by inputting a string "**on**" or "**off**".
- I should be able to use a switch's **turn_on** method to turn it on.
- **turn_off** should work the same way, but in the opposite direction.

- I should also be able to call a **flip** method to flip its state (if it's on, it turns off; if it's off, it turns on).
- If I print a switch, it should print either **I am on** or **I am off** (whichever is currently true).
- If I try to perform an illegal operation on a switch (e.g., turn a switch on when it's already on), nothing should happen.

One other thing (hope you read the instructions before you started coding). The switch shouldn't hold its state in a string. There's a better data type that a switch can use to keep track of its state.

Building a switch board

Build a class **SwitchBoard** with the following properties:

- When I create a switchboard, I should be able to set the number of switches it contains.
- All switches should start in the "**off**" position.
- If I print a switchboard, it should print something along the lines of: "The following switches are on: 0 2 4 6 8".
- The which switch method should return a list of integers representing the switches that are on, in order(e.g., [1,3,5,7,9]).
- If I call flip(n) with n as an integer, it should flip the state of the n'th lightswitch.
- If I call flip **every(n)** with n as an integer, it should flip the state of every n'th lightswitch, starting at 0. So flip **every(2)** would flip switches 0, 2, 4, 6, etc.
- The method **reset()**, should turn all switches off.
- If I ask the switchboard to flip a switch which doesn't exist, nothing should happen (it shouldn't crash).

You can add any extra methods you need to make the code work as described above.

Now check your solution

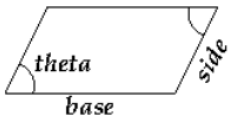
Now, write a piece of global code that solves the problem as stated above.

Exercise 2

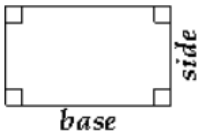
Shapes

Consider the following 4 shapes:

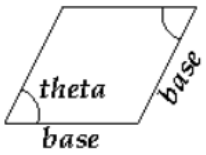
Parallelogram: a four sided figure with parallel pairs of sides. A Parallelogram is defined by the lengths of its two pairs of sides (labelled base and side in the picture below) and the interior angle (in degrees) between adjacent sides (labelled theta in the picture below).



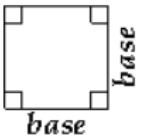
Rectangle: a parallelogram with four right angles. A Rectangle is defined by the lengths of its two pairs of sides (labelled base and side in the picture below).



Rhombus: a parallelogram with four equal sides. A Rhombus is defined by the length of its sides (labelled base in the picture below) and the interior angle (in degrees) between adjacent sides (labelled theta in the picture below).



Square: a parallelogram with four equal sides and four right angles; both a Rectangle and a Rhombus. A Square is defined by the length of its sides (labelled base in the picture below).



Your first task is to figure out the IS-A hierarchy of these shapes. Draw it out in a piece of paper. Are there any instances of multiple parents?

Your Task

You must write four classes: **Parallelogram**, **Rectangle**, **Rhombus** and **Square**, ensuring that the class hierarchy follows your diagram from the previous step. You may create additional classes if you wish (are there any good reasons why you might want to do this?). The parameters of **__init__** methods should always be input in the following order: (base, side, theta) (theta being given in degrees), though of course, not every class' init will take all three. So for example, a Rectangle will only take (base, side). Objects of these classes must have the following methods:

- **area()** - returns the area of the shape
    - Note: The area of a parallelogram is computed by **base ∗ side ∗ sin(theta)**
    - Warning: function **math.sin** in Python expects its argument to be an angle given in radians - take a look at the function **math.radians** to convert between degrees and radians.
- **bst()** - returns a list of three floats: **[base, side, theta]**. Even if a shape doesn't need one of the parameters for its input, it should still be able to return it. (e.g., a 10 x 10 square would return: [10.0, 10.0, 90.0]).
- When printed, each shape should return a string with text in the following format "I am a shape with area area". Four example, a 10 x 10 square would return the string: "I am a Square with area 100".

Be Lazy

One way to solve this would be to write four completely independent classes, and have each class completely implement all of their own functions. This would be a bad idea (why?). If you use inheritance correctly, you should find the exercise much simpler. Remember, you should never calculate something when you can just get another method to do the work for you.

Hint: It's possible to set more than one parent for your class. Are there any shapes here for which that would be a sensible thing to do?