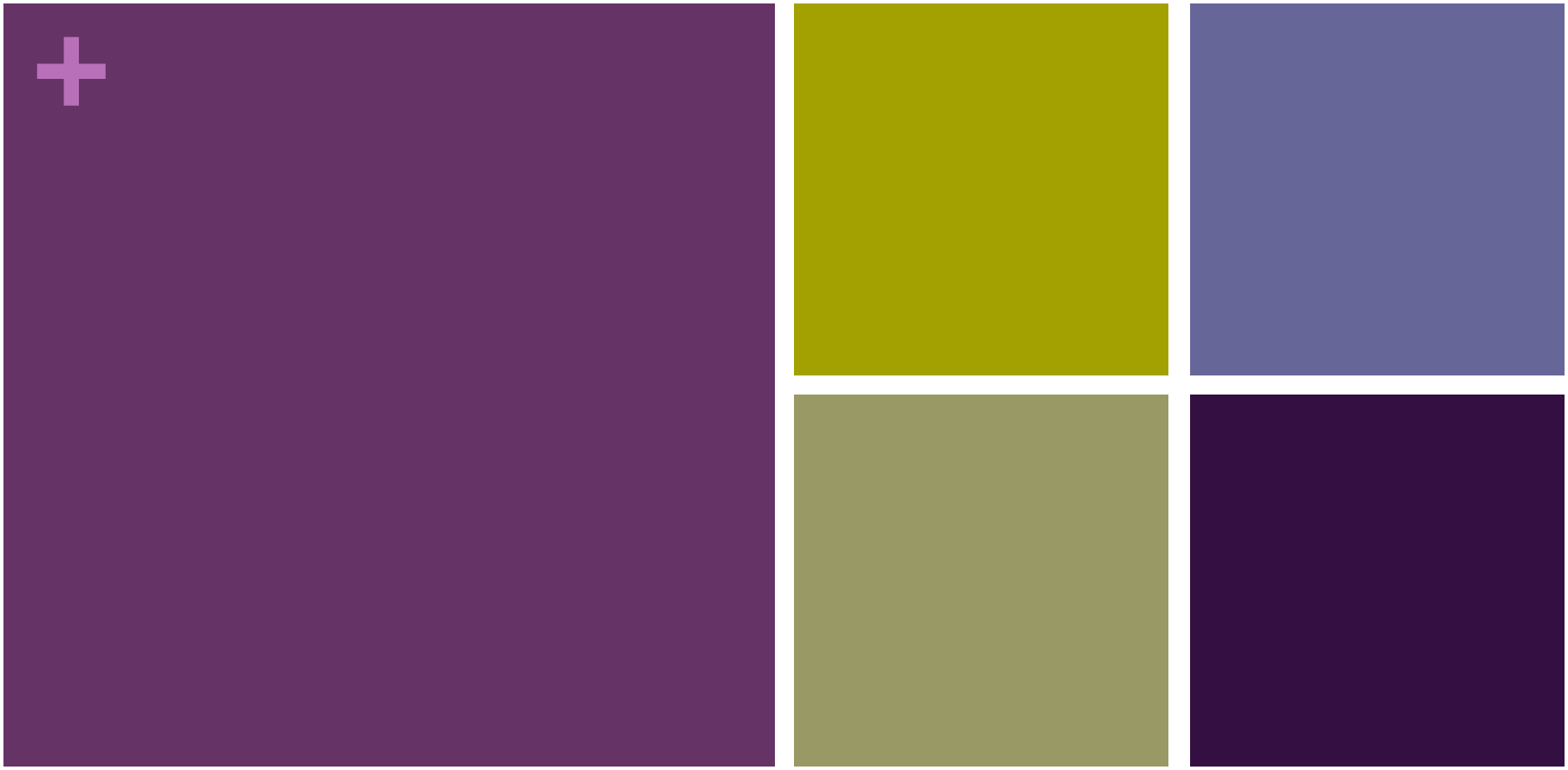
Review

www.pollev.com/python002



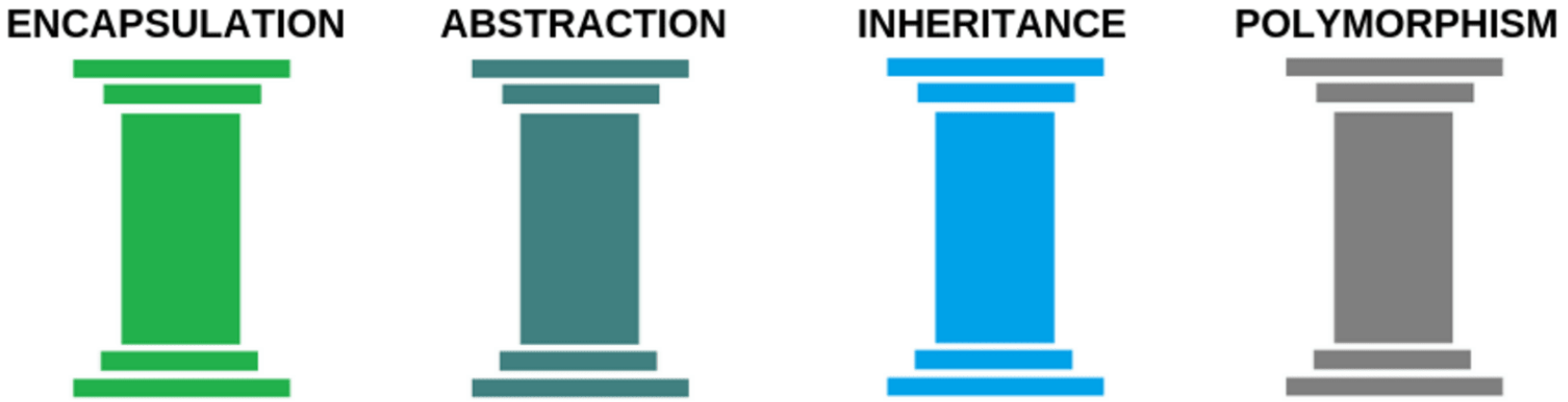
Object Oriented Programming

Overview

Procedural vs OOP

- Procedural programming is a method of writing software. It is a programming practice centered on the procedures or actions that take place in a program.
- Object-oriented programming is centered on objects.
  - Objects are created from abstract data types that encapsulate data and functions together.

4 Pillars of Object Oriented Programming



grouping of information	hiding of information	sharing of information	redefining of information

+

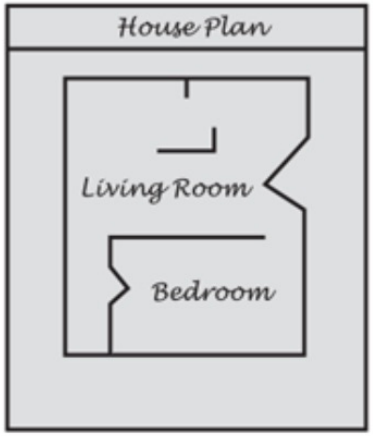
Classes and Objects

A class is a blueprint.

- A class is code that specifies the data attributes and methods for a particular type of object.
  - It is a description of an object's characteristics.
  - Classes are a blueprint that allow us to make many independent copies of objects that look or behave in similar ways.
- Each object that is created from a class is called an **instance** of the class.

Example

Blueprint that describes a house



Instances of the house described by the blueprint

Example 2

- Monster Blueprint:
  - Must have 2 or more eyes
  - Must have 4 or more limbs

- Draw a monster that follows the following blueprint.

- Link: https://bit.ly/3zDb6Ss

Car Class

class Car

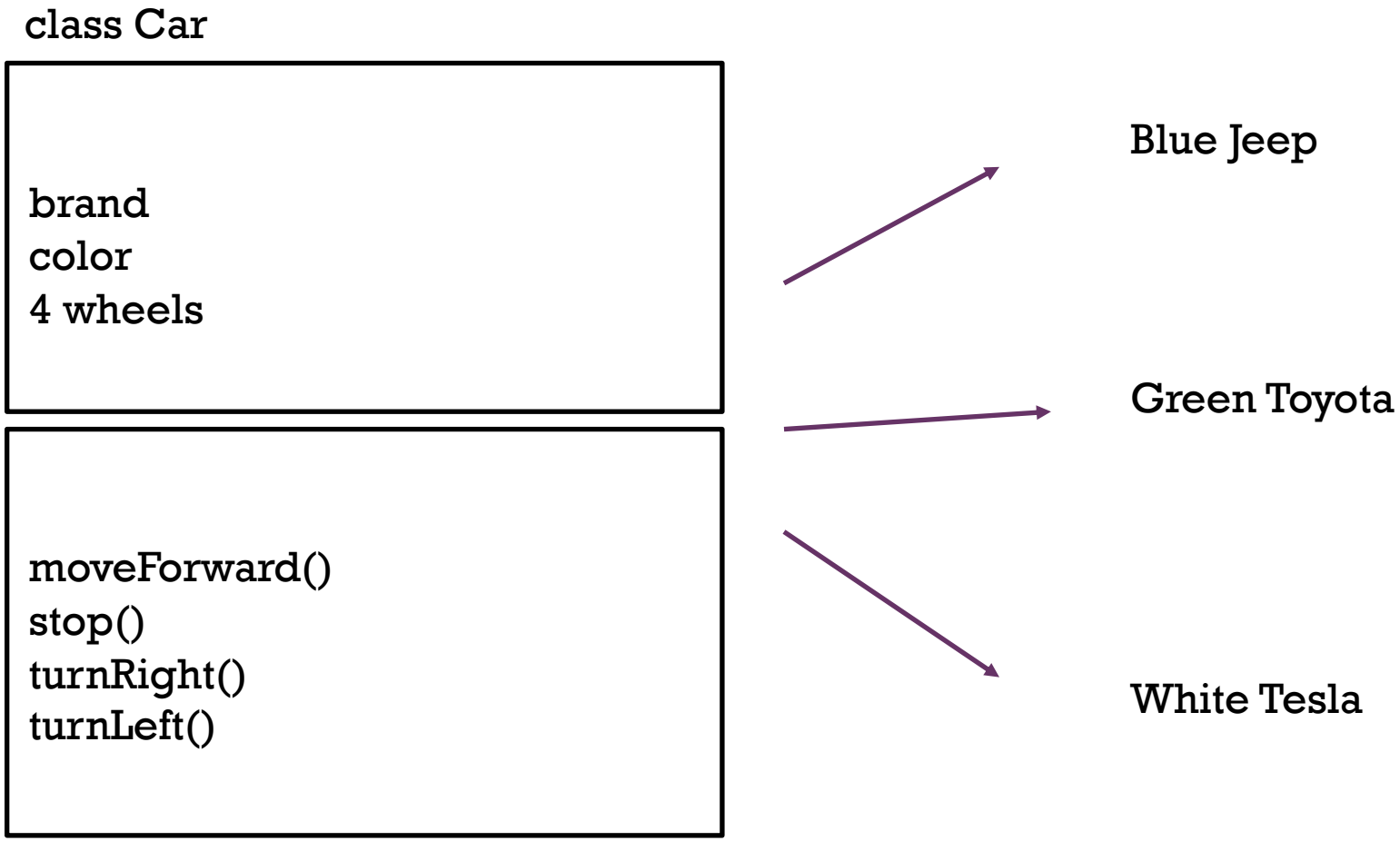
What do all cars have?
(aka attributes)

brand
color
4 wheels

What do all cars do?
(methods)

moveForward()
stop()
turnRight()
turnLeft()

Creating Car objects



Defining a class

```
# blueprint to create a car

class Car:


    # all cars have these variables (attributes) attached to them

    make = "Jeep"

    model = "Wrangler"

    miles = 0

    gas = 0

    gas_max = 15
```

Creating objects from a class

- Creating an object from a class is called **instantiation.**
- We create an object by using the name of the class followed by parenthesis.

```
car1 = Car()
```

- The variable `car1` is holding the memory address of where the object will be stored.
- You must define your class before you try to create an object!

Accessing attributes within a class

- To access data within a class, we use the "dot syntax"

```
car1 = Car()

print(car1.brand)

print(car1.make)
```

Creating Multiple Instances

- One of the biggest advantages of defining classes is that you can make as many objects as you would like!
- Each instance of a class has its own set of data attributes.
  - Classes allow you to make many different independent copies.

Constructors

- Constructor functions are designed to run one time when an object is created.
- To set up a constructor function, you need to refer to it using a special name - '__init__' (two underscores before and after the word 'init')
- This function accepts a single argument which is a reference to the instance that is being created.

```
def __init__(self):

        print("New object being made!")
```

Methods within Classes

- In addition to attaching values to an object we can also attach functions to our objects as well.

```
def drive(self):

        print("Driving Car")
```

- The function is designed to accept the 'self' argument, just like the constructor function does. We call functions defined in this way as 'methods' of the object

- To use the method, we can use the dot notation to write

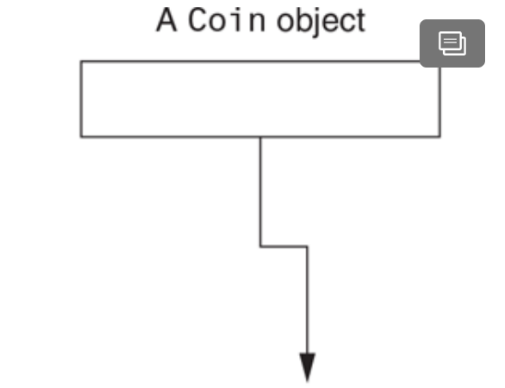
```
car1. drive()
```

Programming Challenge

- Design a class called **Coin** that stimulates a coin being flipped.
- The class should have an attribute called "sideup" to store whether the coin is "Heads" or "Tails"
- The class should have a method to toss the coin and randomly choose between heads or tails.

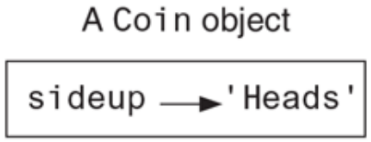
Workflow

1. An object is created in memory from the `Coin` class.



2. The `Coin` class's `__init__` method is called, and the `self` parameter is set to the newly created object

After these steps take place, a `Coin` object will exist with its `sideup` attribute set to `'Heads'`.



Programming Challenge

- Design a class called **CheckingAcccount** which has the following:
  - A constructor that accepts 4 arguments - a owner, account number, and balance
  - A method called ”view_balance". This method should accept no arguments and prints the account number and balance
  - A method called “withdraw” , with 1 argument, that removes a specified amount of money from the account
  - A method called “deposit”, with 1 argument, that adds a specified amount of money to the account

Getters and Setters

Getters

- A method that returns a value from a class's attribute but does not change it is known as an **accessor method**.
- Accessor methods provide a safe way for code outside the class to retrieve the values of attributes, without exposing the attributes in a way that they could be changed by the code outside the method.

Setters

- A method that stores a value in a data attribute or changes the value of a data attribute in some other way is known as a **mutator method**.
- Mutator methods can control the way that a class's data attributes are modified. They usually accept a new value as an argument

+

Programming Challenges

Programming Challenge

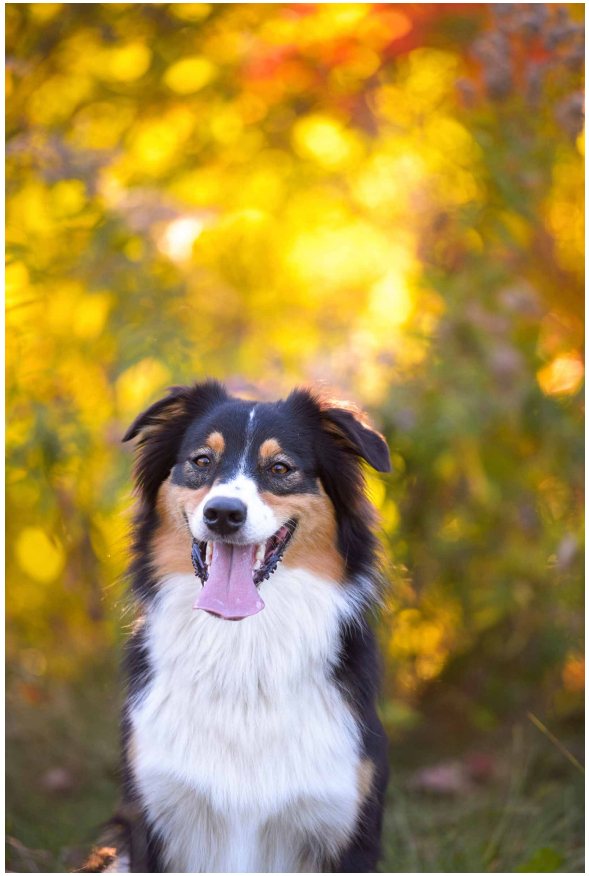
- Write a class named **RetailItem** that holds data about an item in a retail store. The class should store the following data in attributes: item description, units in inventory, and price. These attributes should be created by a constructor function

- Once you have written the class, write a program that creates three RetailItem objects and stores the following data in them

	Description	Units in Inventory	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

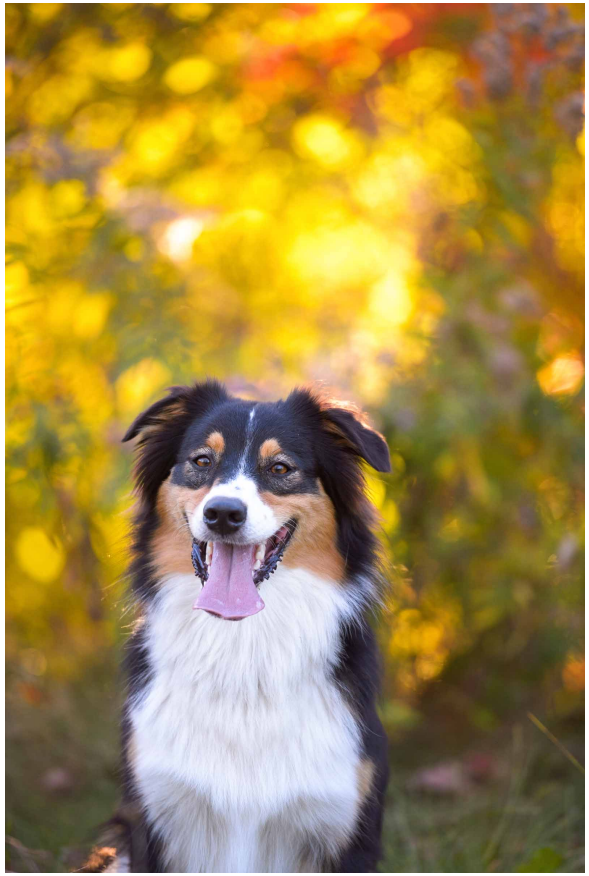
Programming Challenge

- Write a class named **Pet**, which should have the following data attributes:
  - _ _name (for the name of a pet)
  - _ _animal_type (for the type of animal that a pet is. Example values are 'Dog', 'Cat', and 'Bird')
  - _ _age (for the pet's age)
- The Pet class should have an _ _init_ _ method that creates these attributes



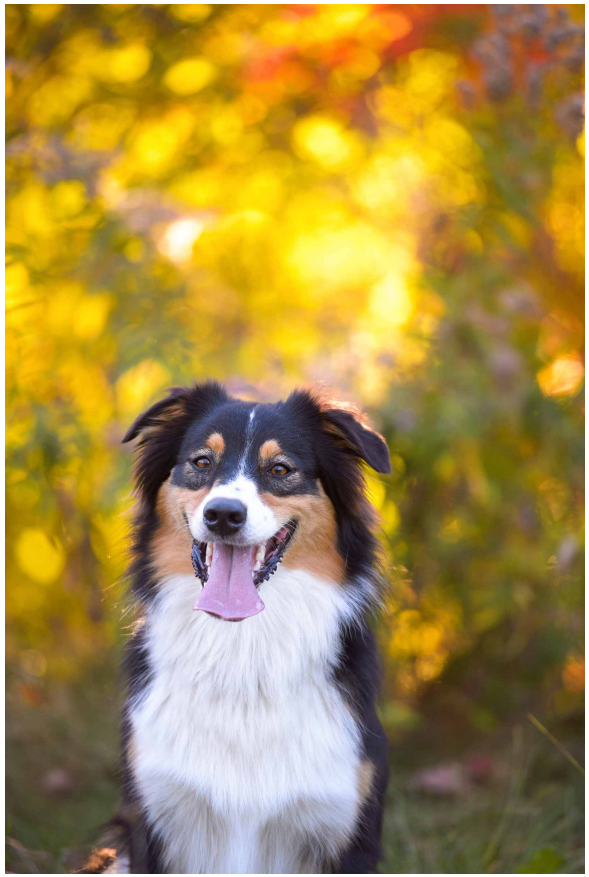
Programming Challenge

- It should also have the following methods:
- set_name: This method assigns a value to the _ _name field.
- set_animal_type: This method assigns a value to the _ _animal_type field.
- set_age: This method assigns a value to the _ _age field.
- get_name: This method returns the value of the _ _ name field.
- get_animal_type: This method returns the value of the _ _animal_type field.
- get_age: This method returns the value of the _ _age field.



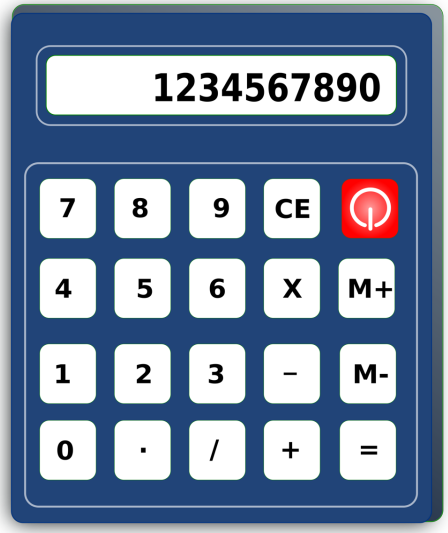
Programming Challenge

- Once you have written the class, write a program that creates an object of the class and prompts the user to enter the name, type, and age of his or her pet. This data should be stored as the object's attributes.
- Use the object's accessor methods to retrieve the pet's name, type, and age and display this data on the screen.



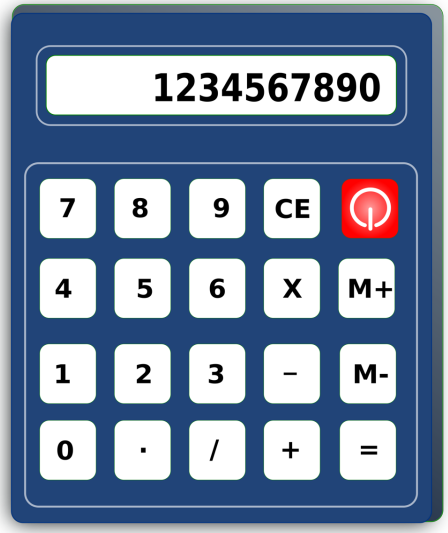
Programming Challenge

- Write a class named **Calculator**, that holds a series of methods:
- add: This method takes 2 arguments and returns the sum
- sub: This method takes 2 arguments and returns the difference
- div: This method takes 2 arguments and returns the quotient
- mult: This method takes 2 arguments and returns the product



Programming Challenge

- Prompt the user for an expression. You can always assume the user will enter valid expressions.
  - The numbers and math operator are separated by a space
- Use your Calculator Class methods to evaluate the expression.




```
>>>

Enter an expression: 2 * 3

2 * 3 = 6

>>>
```