# CS 162P
# Abstract Classes

An abstract class is a class that is never going to be instantiated by itself, but that defines behaviors for child classes that will be derived from it. An abstract class is any class which contains one or more abstract methods, and an abstract method is a method that does not have an implementation. Abstract classes are commonly used to define common interfaces for a variety of different objects.

Python does not provide abstract classes by default it is necessary to import the module Abstract Base Classes (ABC).

## Creating an Abstract Class

In the following example, vehicle is an abstract class that will be used to create multiple child classes. This matches the usage in real life where there are many types of things that are vehicles.

```
# get the definitions of Abstract Base Class
# and abstract method to use in defining Vehicle
from abc import ABC, abstractmethod

class Vehicle(ABC):
    @abstractmethod
    def getPower(self):
        pass

    @abstractmethod
    def getWheels(self):
        pass

class Bicycle(Vehicle):
    def getPower(self):
        return "pedal"

    def getWheels(self):
        return 2

class Sled(Vehicle):
    def getPower(self):
        return "gravity"

    def getWheels(self):
        return 0
```

Here Vehicle is derived from the Abstract Base Class (ABC) and has two abstract methods getPower() and getWheels(). There are two classes derived from Vehicle, Bicycle and Sled. They both must define methods that overwrite the abstract methods in Vehicle.

The following main tries to create an instance of Vehicle.

### Code

```
def main():
    object = Vehicle()
    print(object.getPower())
```

### Output

```
Traceback (most recent call last):
  File "/Users/jim/Desktop/Inheritance/main.py", line 42, in <module>
    main()
  File "/Users/jim/Desktop/Inheritance/main.py", line 30, in main
    object = Vehicle()
TypeError: Can't instantiate abstract class Vehicle with abstract
methods getPower, getWheels
```

This shows that it is not possible to create an object by instantiating Vehicle since it has abstract methods.

On the other hand, it is possible to create instances of children of Vehicle as shown here.

### Code

```
def main():
    bike = Bicycle()
    toy = Sled()

    print(bike.getPower())
    print(toy.getPower())
```

### Output

```
pedal
gravity
```