Abstract class

This is a blueprint for other classes, allowing you to create methods that must be implimented in a child class. An abstract method is a method declaration, but no implimentation. This acts as an interface for components to work together.

In [ ]:
```python
from abc import ABC, abstractmethod

class Polygon(ABC):

    @abstractmethod
    def sides(self):
        print("I have 0 sides")
```

In [ ]:
```python
#concrete implimentation of the Polygon class
class Triangle(Polygon):
    def hi(self):
        print("hi")

    def sides(self):   #must impliment this method
        #super().sides() #calling the base class method from a subclass
        print("I have 3 sides")


t = Triangle()
t.sides()
```

```
I have 3 sides
```

In [ ]:
```python
class Square(Polygon):
    def sides(self):
        print("I have 4 sides")
```

In [ ]:
```python
#advantages of having these common methods
#we know all polygons will impliment this method.
```

```python
drawings = [Square(),Triangle()]

for shapes in drawings:
    shapes.sides()
```

```
I have 4 sides
I have 3 sides
```