1. Fill the missing pieces of the `Calculator` class

Fill ____ pieces of the `Calculator` implemention in order to pass the assertions.

In [1]:
```
class Calculator:
    def __init__(self, var1, var2):
        self.var1 = var1
        self.var2 = var2

    def calculate_power(self):
        return self.var1 ** self.var2

    def calculate_sum(self, var3):
        return self.var1 + self.var2 + var3
```

In [2]:
```
calc = Calculator(2, 3)
assert calc.calculate_power() == 8
assert calc.calculate_sum(4) == 9
```

2. Finalize `StringManipulator` class

Fill ____ pieces and create implementation for `stripped_title()` .

In [28]:
```
class StringManipulator:
    """Docstring of StringManipulator"""

    category = "Manipulator"

    def __init__(self, original):
        self.string = original

    def reverse_words(self):
        words = self.string.split()
        self.string = ' '.join(reversed(words))

    def make_title(self):
        # Create implementation for this
        self.string = self.string.title()


    def get_manipulated(self):
        return self.string
```

In [30]:
```
assert StringManipulator.__doc__ == 'Docstring of StringManipulator'
assert StringManipulator.category == 'Manipulator'

str_manip = StringManipulator('cOOL pyThON')

str_manip.reverse_words()
assert str_manip.get_manipulated() == 'pyThON cOOL'
```

```
str_manip.make_title()
assert str_manip.get_manipulated() == 'Python Cool'
```

3. Create `Dog` class

Create `Dog` class which has the following specification:

- Dogs consume their energy by barking and gain energy by sleeping
- A fresh `Dog` instance has 10 units of energy
- `Dog` has a method `sleep` which gives 2 units of energy
- `Dog` has a method `bark` which consumes 1 unit of energy
- `Dog` has a method `get_energy` which returns the amount of energy left

In [33]:

```
class Dog:
    # Your implementation here

    def __init__ (self):
        self.energy = 10

    def sleep(self):
        self.energy = self.energy + 2

    def bark(self):
        self.energy = self.energy - 1

    def get_energy(self):
        return self.energy
```

In [34]:

```
doge = Dog()
assert doge.get_energy() == 10

doge.bark()
doge.bark()
doge.bark()
assert doge.get_energy() == 7

doge.sleep()
assert doge.get_energy() == 9

another_doge = Dog()
assert another_doge.get_energy() == 10
```