Jennifer's Object-Oriented Programming Cheat Sheet

Dot Notation and Variable Lookup:

`<expression>.<name>`

Left side: <expression> tells us where we want to look (evaluates to either a class or an instance).
Right side: <name> is the variable we want to look up

Rules: If <expression> is a class:

1. Look up <name> within class. If we find it, we're done!
2. Otherwise, look up <name> in the parent class and repeat these steps.

If <expression> is an instance:

1. Look up <name> within the instance. If we find it, we're done!
2. Else, look up <name> within its class.
3. If we don't find it, look up <name> in the parent class and repeat steps 2-3.

Example that shows:

1. Dot notation must be used to refer to class/instance attributes.
2. The parent of methods is global.

```
x = 5
class A():
    x = 10
    def __init__(self):
        print(x)
>>> a = A()
5
```

__init__ method: When we have the name of a class followed by (), we are making an instance of that class and will call the class's init method.

__repr__ method: If we type in an instance in interactive mode, we will call its repr method and display the string we get back without quotes.

__str__ method: If we call print on an instance, then we will call its str method and display the string we get back with no quotes.

```
class A:
    def __init__(self):
        print('init')
    def __repr__(self):
        return 'repr'
    def __str__(self):
        return 'str'
```

```
>>> A()
init
repr
>>> x = A
>>> x()
init
repr
```

```
>>> print(A())
init
str
>>> x = A
>>> print(x)
init
str
```

Note: If we have defined a __repr__ method but not a __str__ method, then str will default to repr, meaning that calling print(obj.) will display its repr method.

```
class A:
    def __init__(self):
        print('init')
    def __repr__(self):
        return 'repr'
```

```
>>> A()
init
repr
```

```
>>> print(A())
init
repr
```

Invoking methods:

Conditions for when we pass in the instance implicitly as the first argument (both must be true):

1. LHS of dot notation is an **instance**
2. RHS of dot notation is a method **found in the class**

```
class A():
    def f():
        print('hi')
    def g(self): #Note that if we called this parameter
                 something other than self, the
                 examples would still work the same
                 way
        print('bye')

a = A()
a.h = lambda x: 'hello' #this function belongs to the
                        instance
```

```
>>> A.f()
hi
>>> a.f() #a is passed in implicitly as first parameter
Error (expected 0 arguments, but got 1 argument)
>>> A.g()
Error (expected 1 argument, but got 0 arguments)
>>> A.g(a) #Note that a.g() is equivalent to A.g(a)
bye
>>> a.g() #a is passed in implicitly as first parameter
bye
>>> a.h() #violated 2nd condition so a is not passed
          in implicitly as first parameter
Error (expected 1 argument, but got 0 arguments)
>>> a.h(5)
'hello'
```