Disc 05

OOP

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

Look at the interpreter session to determine what class/instance attributes you may need to create

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

We also know that this initial value doesn't change, as subsequent calls to g still add 1 to what we pass in as the argument to g.

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

We also know that this initial value doesn't change, as subsequent calls to g still add 1 to what we pass in as the argument to g.

What does this assignment do? It's not clear yet, lets check out the next line

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

We also know that this initial value doesn't change, as subsequent calls to g still add 1 to what we pass in as the argument to g.

What does this assignment do? It's not clear yet, lets check out the next line

Now instead of adding 1, we add 5. Where did this 5 come from? Well we just assigned the instance variable of x called motto to be 5. So in the method g, we must be adding motto to what we pass in. Where else does motto do something? Where do we set it initially?

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

We also know that this initial value doesn't change, as subsequent calls to g still add 1 to what we pass in as the argument to g.

What does this assignment do? It's not clear yet, lets check out the next line

Now instead of adding 1, we add 5. Where did this 5 come from? Well we just assigned the instance variable of x called motto to be 5. So in the method g, we must be adding motto to what we pass in. Where else does motto do something? Where do we set it initially?

Bring all of the ideas together:

1. Create instance attribute motto and store initial value there.
2. Create function g which:
    1. Takes in one parameter (other than self)
    2. Adds the argument with motto
    3. Returns the sum

2.1 #4

Implement the `Yolo` class so that the following interpreter session works as expected. (Summer 2013 Final)

```
>>> x = Yolo(1)
>>> x.g(3)
4
>>> x.g(5)
6
>>> x.motto = 5
>>> x.g(5)
10
```

```
class Yolo:
    def __init__(self, x):
        self.motto = x

    def g(self, x):
        return self.motto + x
```

Look at the interpreter session to determine what class/instance attributes you may need to create

When we create an object we store the number 1 somewhere.

We then use this value when we call the method g (since 3 + 1 = 4)

We also know that this initial value doesn't change, as subsequent calls to g still add 1 to what we pass in as the argument to g.

What does this assignment do? It's not clear yet, lets check out the next line

Now instead of adding 1, we add 5. Where did this 5 come from? Well we just assigned the instance variable of x called motto to be 5. So in the method g, we must be adding motto to what we pass in. Where else does motto do something? Where do we set it initially?

Bring all of the ideas together:

1. Create instance attribute motto and store initial value there.
2. Create function g which:
    1. Takes in one parameter (other than self)
    2. Adds the argument with motto
    3. Returns the sum