Problem 1

In the video lecture on Python functions we discussed variable *positional* arguments that use the special syntax `*args` to identify them. We can also have variable *keyword* arguments. These utilize the special syntax of `**kwargs`. The keyword arguments are passed into the function as a Python dictionary. For example, a function with the signature:

```
a_keyword_arg_function(**kwargs)
```

That is called with

```
a_keyword_arg_function(a=1, b=2, c='three')
```

will have a variable available for use inside the function

```
kwargs = {'a': 1, 'b': 2, 'c'='three'}
```

with this in mind, complete the function below that takes a `**kwarg` as an argument. You can assume the values of the inputs will always be numbers. The function should multiply all the given values together and output a Python string that has exactly the following formatting. Shown as example function calls and outputs.

```
multiply(a=1, b=2, c=3)
```

should return `'a * b * c = 6'`, and

```
multiply(x=3, y=1)
```

should return `'x * y = 3'`.

In [ ]:

```
def multiply(**kwargs):
    total = 1
    key1 = ""
    i = 0
    for key, value in kwargs.items():
        total *= value
        i +=1
        if i < len(kwargs):
            key1 += key + " * "
        elif i == len(kwargs):
            key1 += key + " = "

    return key1 + str(total)
```

In [ ]:

```
multiply(a=1, b=2, c=3)
```

Out[ ]:

```
'a * b * c = 6'
```

In [ ]:

```
multiply(x=4, y=3)
```

Out[ ]:

```
'x * y = 12'
```

Problem 2

Complete the function below. The function takes no arguments and returns a Python lambda function (yes, a function can return a function...) that implements the following mathematical operation

$$x + y^2$$

An example of how you'd call this funtion is

```
f = create_lambda()
```

followed by a call to the returned function for testing

```
f(x=1, y=2)
```

which would return `5`.

In [ ]:

```
def create_lambda():
    create_lambda = lambda x, y: x + y ** 2
    return create_lambda
```

In [ ]:

```
f = create_lambda()
```

In [ ]:

```
f(x=1, y=2)
```

Out[ ]:

```
5
```