print() vs return

Many beginning programmers find the distinction between print and return very confusing, especially since we use print to show the returned value of a function like this: `print(countEven([1,3,8,9]))`. The print statement is fairly easy to understand. It takes a Python object and outputs a printed representation of it in the output window. You can think of the print statement as something that takes an object from the land of the program and makes it visible to the land of the human observer.

If you're confused, chances are it is not really about the print statement but about returned values and the evaluation of complex expressions. A function that returns a value is producing a value for use *by the program*, in particular for use in the part of the code where the function was invoked. Remember that when a function is invoked, control passes to the function, meaning that the function's code block is executed. But when the function returns, control goes back to the calling location, and a return value may come back with it.

If a returned value is for use *by the program*, what is it used for? There are three possibilities:

1. **Save it for later:**
   The returned value may be:
   - Assigned to a variable. For example, `x = countEven([1,3,8,9])`
   - Put in a list. For example, `myList.append(countEven([1,3,8,9]))`
   - Put in a dictionary. For example, `myDict[3] = countEven([1,3,8,9])`
2. **Use it in a more complex expression:**
   In that case, think of the return value as replacing the entire text of the function invocation. For example, if there is a line of code:

   ```
   x = square(square(3) + 7) - 5
   ```

   think of the return value 9 replacing the text `square(3)` in that invocation, so it becomes `square(9 + 7) -5`.
3. **Print it for human consumption:**
   For example, `print(countEven([1,3,8,9]))` outputs 1 to the output area. Note that, unless the return value is first saved as in possibility 1 (save it for later), it will be available only to the humans watching the output area, not to the program as it continues executing.

If your only purpose in running a function is to make an output visible for human consumption, there are two ways to do it. You can put one or more print statements inside the function definition and return the value for further use.

```
def countOdd(num_list):
    odd_count = 0
    for num in num_list:
        if num % 2 != 0:
            odd_count += 1
    print("The number of odd numbers in the list is",odd_count)

    return odd_count
```

The other possibility is to return a value from the function and print it. As you start to write larger, more complex programs, this will be **more typical**. Indeed the print statement will usually only be a temporary measure while you're developing the program. Eventually, you'll end up calling f and saving the return value or using it as part of a more complex expression.

```
def countOdd(num_list):
    odd_count = 0
    for num in num_list:
        if num % 2 != 0:
            odd_count += 1

    return odd_count


print(countOdd(num_list))    #OR
print("The number of odd numbers in the list is", countOdd(num_list))
```

You will know you've really internalized the idea of functions when you are no longer confused about the difference between print and return. Keep working at it until it makes sense to you!