Today we are going to learn to write our own custom functions.

You've already been using functions in Python, commands such as print() and len(). Today we learn to make our own custom functions.

First, let's back up: functions are blocks of code that you set up in a special way so that you can call them over and over from anywhere in your program. Typically you define a function at the top of your program and then call it from other places down below. You can also define functions in what is called a library, a Python program that you access from other programs. An example of this is the random library that we import when we want to do something involving random things.

Functions can accept optional "arguments," pieces of information that get handed to the function. Here are some examples:

```
print("go tigers!")
```

This seems obvious, but when we do a print call we are calling the print function and sending it one or more arguments, that is, things that we want to print. Here is another example:

```
len(myList)
```

Above we send a list variable myList to the "len" function, and it returns how many items are in the list.

Let's create a sample function. Type this into a new document:

```
def sayHi(name):
    print("Hello",name)
    print(name.center(20,"$"))
```

This is a silly function that says "hello" to whatever name you send it, then prints the name centered in 20 dollar signs. Test this by adding a line like this below the function:

```
sayHi("Mr. Hays")
```

Now run the program. Here is the output:

```
Hello Mr. Hays
$$$$$$Mr. Hays$$$$$$
```

How does this all work? We first define a function (the "def" line) called sayHi. This function has two lines of code in it (the different print lines). The function accepts one argument, a variable called "name".

So that's a function. Not so scary, really. Why do we want to use functions? Well, sometimes you will need to do certain things over and over again in a program and defining a function to do it in one place makes coding things easier.

(Continued on next page)

One more thing we have to learn about functions: a function can return information back to whoever is calling it (for example the len() function gives you how long something is), or it can just do things (for example the print() function prints whatever you give to it.).

Functions that give you back information have a return call in them. Here is an example:

```
def square_perimeter(side):
   return side*4
```

This new function, square_perimeter, calculates and returns the perimeter of a square. You could call it like this:

```
print(square_perimeter(10))
```

This would print out 40, because the perimeter of a square with sides of 10 is 4 times 10 or 40.

Today you are going to make and use three functions, some with a return and some without.

Today's assignment:

Create new file called day31_functions, save it to the Z: drive. Put your name in a comment at the top.

1.

Write a function called **printStars** that accepts a number and prints that many stars on a single line. You will start by writing this line:

```
def printStars(num):
```

Then, indented one level, write a for loop that prints num stars all on one line, then prints a blank line. Test your code by putting a line below your function that looks like this:

```
printStars(10)
```

Make sure this line is not inside of the function (it has to be unindented all the way to the left.) Here is sample output for the call printStars(10):

```
**********
```

Part 2:

Write a function called **printList** that accepts a list variable and a number, and then prints the list in that many columns. Start like this:

```
def printList(theList,columns):
```

Then, inside the function, write code that prints all of the items in the list (use a for loop) in the right number of columns (whatever the variable columns is). Put tabs between list items. At the end print a blank line so that you are safely out of column printing mode.

(Continued on next page)

Test the function by adding these lines below it, not indented:

```
myList = [1,3,5,7,9,11,13,15,17,19]
printList(myList,3)
print()
printList(myList,5)
```

You will get the following output:

```
1       3       5
7       9       11
13      15      17
19

1       3       5       7       9
11      13      15      17      19
```

Can you see how the above shows the list printed in three columns, then again in 5 columns?

Make your function work even if your list variable doesn't contain numbers. For example, test it with this:

```
myList=["a","b","c","d","e","f","g","h","i","j","k"]
printList(myList,4)
```

You should get the following:

```
a       b       c       d
e       f       g       h
i       j       k
```

Part 3:

Write a function called **primeCheck** that accepts a number, doesn't print anything, and uses a return call to say if a number is prime or not.

Start like this:

```
def primeCheck(num):
```

Then inside, check if num is prime. If it is return "prime", if it is not return "composite" (in math talk that means "not prime".) Your function should not print anything. We learned how to figure out if a number is prime in our day20 assignment. You should be able to copy some of that code. Test your code with the following:

```
print(400,primeCheck(400))
print(401,primeCheck(401))
```

The output should be:

```
400 composite
401 prime
```

Save your program on the Z: drive. Even if you do not get all three parts done, save what you have there for partial credit.