Lesson Ten

Math 6080 (for the Masters Teaching Program), Summer 2020

Functions. Functions are created with ”def function(function inputs):” followed by the working code for the function, indented of course. For example,

```
def obvious():
    print(”this is an obviously dumb function”)
```

is a function without inputs, printing the pithy message after you enter

```
obvious()
```

Another (still dumb) function:

```
def less_obvious(number):
    print(number, ”is an interesting number”)
```

Prints a banal message with a function input, entered as:

```
less_obvious(5)
```

Remarks. A function can have multiple inputs (separated by commas).

The data type of the input of a function can be anything.

Uses. Functions are “modules” of code called upon to perform a specific purpose. Functions can recursively call upon themselves within themselves. This is incredibly powerful for mathematical applications.

Example. The factorial function can be defined recursively:

```
x = int(input(”Number to be factorialized: ”))
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n*factorial(n-1)
print(factorial(x))
```

Note. The return’command is a new feature. It was not present in the obvious examples, which simply printed an expression. Here, we want the function to send back an integer value when called, which is what the return command does.

Converting to binary. Here are two competing programs each of which will convert a number (entered by the user) to its binary (base 2) form.

The first uses a for loop and returns the binary digits in a string:

```
n = int(input(”Your number to be converted to binary: ”))
base2 = ””
while n > 0:
    base2 = str(n%2) + base2
    n = n//2
print(“the binary form of your number is,” base2)
```

Remark. All of this should be familiar, except for the str() command, which converts an integer into a string. In our case, the integer is 0 or 1, and it is converted to the string ‘0’ or ‘1’ in order to attach it (via the + sign). Run through the steps of the program on a sample number to understand how it works.

The next program creates an list of the digits of the binary form of the number, and does so with a recursive function:

```
n = int(input("Your number to be converted to binary: "))
def base2func(n):
    if n < 2 : return [n]
    else:
        return base2func(n//2) + [n%2]
print(base2func(n))
```

We'll discuss these in detail in class.

(Late Night) Exercise. Modify the second recursive function to prompt the user for a number **and** a base to convert it into (e.g. base 16).

```
n = int(input("Your number: "))
b = int(input("The base: "))
```

(You fill the rest in!)

Question. Why can't the first program be modified to an arbitrary base?

Or more positively: For which bases can the first program be modified to work?