6.0001 Recitation 2 - Spring 2020

February 14

I. Administrivia

- PS2 due **Wednesday 2/19 @ 9PM**
- PS1 checkoff ongoing, due **2/24 @ 9PM**
- **Microquiz 1 Tuesday 2/18 in class**

II. Representing numbers in python

- **`int`:** representation of whole numbers
- **`float`**: approximation of real numbers.
    - Do not use == to test equality among floats as it has unexpected behavior. For example, if we create a variable x and add 0.1 to 10 times, it doesn't "equal" 1. Instead, to test equality among two floats check if they are within some small epsilon.
    - Exhaustive enumeration isn't feasible when working with floats because the possibility space is infinite.

III. Binary numbers

- Base 2 representation of a number
- Example: let's look at the number 38 in base 10 and 2
    - base 10: $3*10^1 + 8*10^0$
    - base 2: $\mathbf{1}*2^5 + \mathbf{0}*2^4 + \mathbf{0}*2^3 + \mathbf{1}*2^2 + \mathbf{1}*2^1 + \mathbf{0}*2^0 = 100110$

IV. Functions

- Motivation
    - Achieving abstraction / decomposition
    - A black box can be reused in the future
- Defining a function

```
def function_name(arg1, arg2, …, argN):
    #code
    return something
```

- Calling a function

```
function_name(arg1, arg2, …, argN)
```

- **print vs return**
    - **print:** for the user, just displays a value
    - **return**: for the computer and allows you to send values in a function back to other parts of your code

        - Nothing in the function will be executed after a return statement is executed.
        - Python's default **return** is **None**. None in most cases will make you sad, so make sure you return something when it needs to be returned.
- {code example} Function that multiplies by 5
    1. What's printed?
    2. What's the value of multiple_of_five?
    3. If you delete the return statement, what will be the value of multiple_of_five?
- {code example} Incorrect argument types
    - If you alter the argument types one of two things will happen:
        - Silently give a very wrong answer
        - Throw an error in your code

V. Scope

- Variable assignments are tracked in a **symbol table** or **stack frame** that maps variable names to their values
- When a function is **called**, a new stack frame is created.
- When the function returns, the stack frame pops off/is destroyed
- Last in, first out - just like a real-life stack!
- {code example}
    - The scope of "number" is within the function.
    - The scope of "two" is outside the function

VII. Recursion

- {code example} Factorial is an example of recursion
- {code example} Fibonacci: is an example of recursion

VII. Iteration

- **`for`** loops have a pre-specified range over which they run.

    `for i in range(x):`
        - `i` goes from 0 to x-1

    `for char in s:`
        - `char` is string that takes on the value of each character in `s`
- **`while`** loops have a condition that they check to determine if they should keep running. They run until the condition no longer evaluates to True.

```
counter = 0
while counter < 3:
```

```
        print(counter)
        counter += 1
```

- Converting between for and while loops
    - All for loops can be written as while loops
    - Not all while loops can be written as for loops
- **`range`**
    - take 1, 2, or 3 arguments
        - stop: the number the iteration stops at, NOT INCLUDED
        - start: the number the iteration starts at. INCLUDED
        - step: the number the value increases by each iteration (e.g. step size of 2 can be used to count just even numbers)
    - 1 argument: `range(stop)`
    - 2 arguments: `range(start, stop)`
    - 3 arguments: `range(start, stop, step)`
- **`break`** and **`continue`**
    - **break** terminates the innermost loop. The program jumps to code immediately after the loop
    - **continue** ends current iteration of the loop. The program jumps to the top of the loops and continues with the next iteration of the loop.

VIII. GuessAndCheck Algorithms

- The process is exhaustive enumeration
    - guess a value
    - check if solution is correct
    - keep guessing until solution found or all values guessed

- {code example} Finding square roots (from an earlier lecture)
    - Approximate solutions (good enough solution)
        - to approximate the square root of a number x, guess a number g. If g*g is close enough to x, you're done, otherwise your next guess is the average of g and x/g
            - Uses: while loops, comparing floats