Recitation 3 Notes:
30 September, 2022

Reminders:

- **MQ4 next Wednesday 10/5**
- **PS1 due next Wednesday**

Lecture 4 Recap: Simple Programs, Intro to Binary Numbers

1. *Simple Programs*

- Guess-and-check algorithms are one way to find a solution to a problem through exhaustive enumeration.
- Guess solution -> evaluate guess -> make educated adjustment to the guess and **repeat ….**
- Repeat these steps until you find a solution or have exhausted your set of possible solutions.
- Example programs: Guessing Square or Cube roots.

2. *Intro to Binary Numbers*

- Computers use binary numbers.
- Everything is stored in one of two states – either 0 or 1.
- Binary numbers are efficient and easy to perform operations on.
- A sequence of binary numbers e.g 00110011 is called a **sequence of bits.**
- Base 10 numbers can be converted to Binary numbers and visa versa.

First eight bits are the powers of two:
128, 64, 32, 16, 8, 4, 2, 1
So, in the first 8 bits we can store number up to (but not including) 256.

Example: Convert base 10 number 56 into binary representation.
56 = 00111000

Example2: Convert 00011001 into base 10.
00011001 = 1*1 + 1*8 + 1*16 = 25

Lecture 5 Recap: Floats, Fractions and Approximation Algorithms

1. *Floats*

- Python uses “floating points” to approximate real numbers.
- Operations on floats introduce a very small error.
- Many smaller errors turn into a bigger error.

2. *Fractions & Approximation*

- We use the same idea to store fractions in binary by raising 2 to the power of some negative number.

Ultimately, a computer represents everything in bits. So, numbers with many digits trailing the decimal are often approximated.
As a result, be careful when comparing and working with floats.

3. *Approximation Algorithms*

- Like guess and check but the goal is to find an answer that is considered “good enough”, and not necessarily exact.
- Guess an answer -> check if it’s “good enough” -> if not, make an educated change your guess -> repeat until your guess is “good enough”
- Key parameters: increment, epsilon, number of guesses etc…
- Remember to keep in mind what happens if you overshoot the close-enough stopping condition – don’t want an infinite loop.

Lecture 6 Recap: Bisection Search, Newton-Raphson

1. **Bisection Search**

- Search algorithm applied to problems with an inherent order to the range of possible answers (e.g an ordered list of numbers).
- Step to a simple binary search algorithm:
    - Guess the midpoint of the interval
    - If not the answer, check if answer is greater or less than the midpoint
    - Change interval
    - Repeat
- This method cuts the set of possible answer to check in half at each stage → logarithmic growth characteristic → more efficient algorithm

2. **Newton-Raphson**

- General approximation algorithm to find the roots of a polynomial in one variable
- Given polynomial function p(x), the goal is to solve for r such that p(r) = 0.
- N-R showed that:
    - If g is an approximation to the root, r, then

      $$g - p(g)/p'(g)$$

      Is a better approximation, where p’ is the derivative of p.

Lecture 7: Functions and Scope

Functions

- Functions capture computation within a black box.
- They allow us to reuse code and write programs in a more concise way.
- Functions take in input and return outputs.
- Inputs are cased as parameters of the function and outputs are returned using the return statement.
- Calling a function

  My_output = function_name(arg1, arg2, …, argN)
- When called, the entire function is replaced with the return value
- **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**.

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
- My python tutor does a good visualization of this https://pythontutor.com/.

6.100L Introduction to CS and Programming Using Python
Fall 2022