Loops and Algorithms

NSC 1002 Workshop 4

17 Nov 2017

Michael Kelleher, Beth Wingate

1. Review
   - Review: Data and Conditionals
   - Review: Loops

2. Algorithms
   - What are algorithms?
   - Bisection

Data Types

Basic Data Types

- □ *float* - Floating point "real" number (e.g. 3.14)
- □ *int* - Integer number: (e.g. 1 or 9223372036854775807)
- □ *string* - Group of characters: (e.g. "Python", or "Hello World")

Collection Data Types

- □ List - mutable collection of homogeneous elements, surrounded by square brackets: [ ]
  - □ Example: `evens = [2, 4, 6, 8, 10]`
  - □ `print(evens[0], evens[-1])`
- □ Tuple - immutable collection of heterogeneous elements, defined by the "," surrounded by round brackets ( )
  - □ Example: `student1= ('James', 438)`
  - □ `print(student1[0], student1[-1])`

Conditionals

```
x = 5
y = 30
if x == y:
    print(’x equals y’)
elif x > y:
    print(’x is greater than y’)
else:
    print(’x less than y’)
```

Recall

- Relational operators used in conditional statements: $== \quad > \quad < \quad != \quad >= \quad <=.$
- Logicals used in conditional statements: *and*, *or*, and *not*
- Example: `(a != b and c==b)`
- Will be true when *a* does not equal *b*, and *b* is equal to *c*

Review: Loops

A “`for` loop” **iterates** *for* a specified number of iterations.

```
N = 5
sqrs = []
for i in range(N):
    sqrs.append(i**2)
print(sqrs)
```

```
[0, 1, 4, 9, 16]
```

Review: Loops

A “`for` loop” **iterates** *for* a specified number of iterations.

```
N = 5
sqrs = []
for i in range(N):
    sqrs.append(i**2)
print(sqrs)
```

[0, 1, 4, 9, 16]

A `while` loop **iterates** *while* a *condition* is true.

```
N = 5
sqrs = []
while len(sqrs) < N:
    sqrs.append(i**2)
print(sqrs)
```

[0, 1, 4, 9, 16]

Algorithms

- □ Algorithms are a set of instructions or rules for a computation
- □ For example: root finding, minimisation/maximisation, finding mean and variance, differential equation solving, ...
- □ Last time: Exhaustive enumeration (SLOW!), Heron of Alexandria's
- □ Today you will be working with two new root finding algorithms: Bisection and Newton-Raphson.
- □ These will all accomplish the same goal: the approximate square root of a number is found, but will vary in design, accuracy, and speed.

Root finding algorithms

The idea of root finding algorithms is to solve the equation

$$F(x) = 0.$$

In our example, if we are interested in finding the square root of a number x with the answer being x_0 then we are interested in

$$\sqrt{x} = x_0,$$

and thus the equation we're trying to solve with a root solver is,

$$\sqrt{x} - x_0 = 0, \tag{1}$$

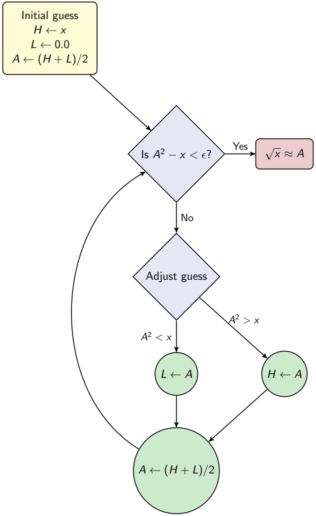
therefore,

$$F(x) = \sqrt{x} - x_0 = 0.$$

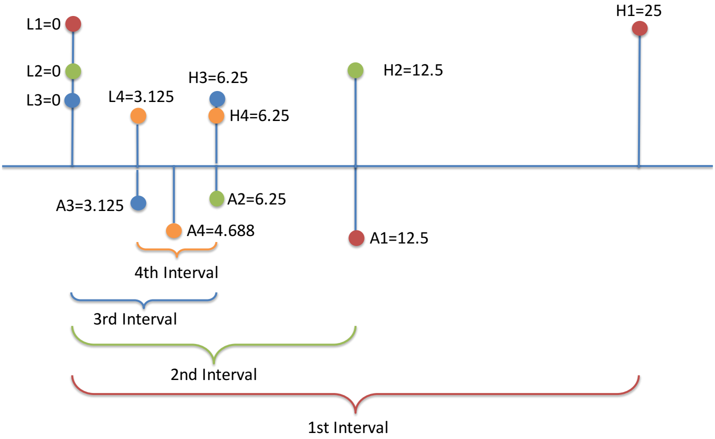
Or, if we square each side of equation (1) we can alternatively write this,

$$F(x) = x - x_0^2 = 0.$$

Bisection Method Design



Bisection Method Design



Questions?

Questions?

Now you have some time to work through worksheet examples