CMPT 383

Lecture 1: Functional Programming

Anders Miltner

2 Broad Classifications of Programs

- Imperative Programming
- Declarative Programming

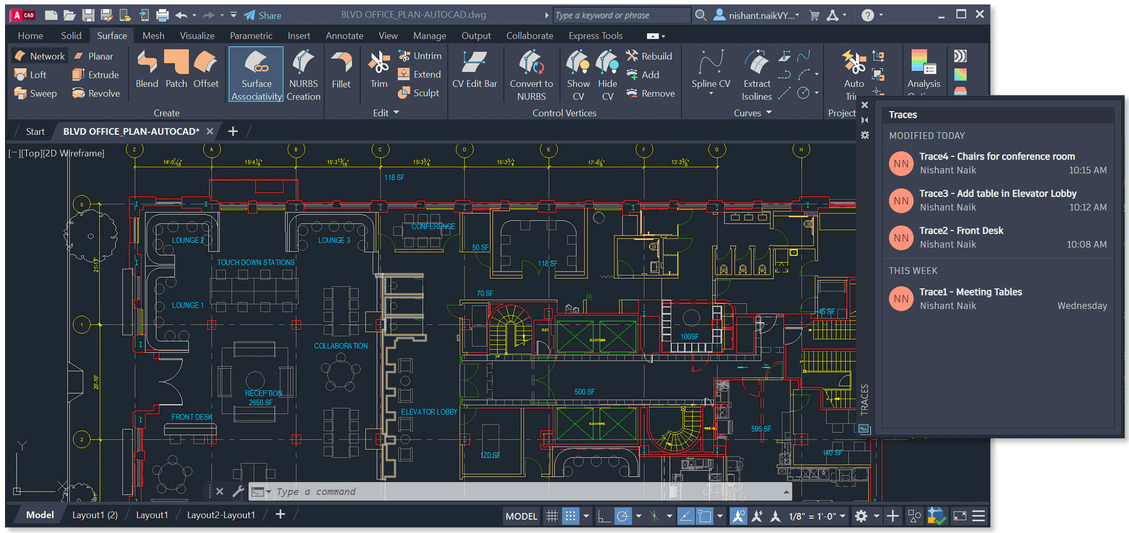
Imperative Programming

Telling the computer what to do

- An imperative program is a sequence of instructions for the computer to follow
- Variables represent things in memory, that the computer can manipulate
- Begin Program -> Execute a series of instructions -> End Program

Declarative Programming

- A declarative program describes something
  - HTML code describes a webpage
  - AutoCAD describes physical objects



Functional Programming

- A specific type of declarative programming
- In functional programming, you are describing mathematical objects and functions
- Functional Programming can happen in declarative languages too!
  - (And vice-versa, see the IO Monad lecture)

Mathematical Functions

Function (mathematics)

From Wikipedia, the free encyclopedia

"f(x)" redirects here. Not to be confused with f(x) (musical group).

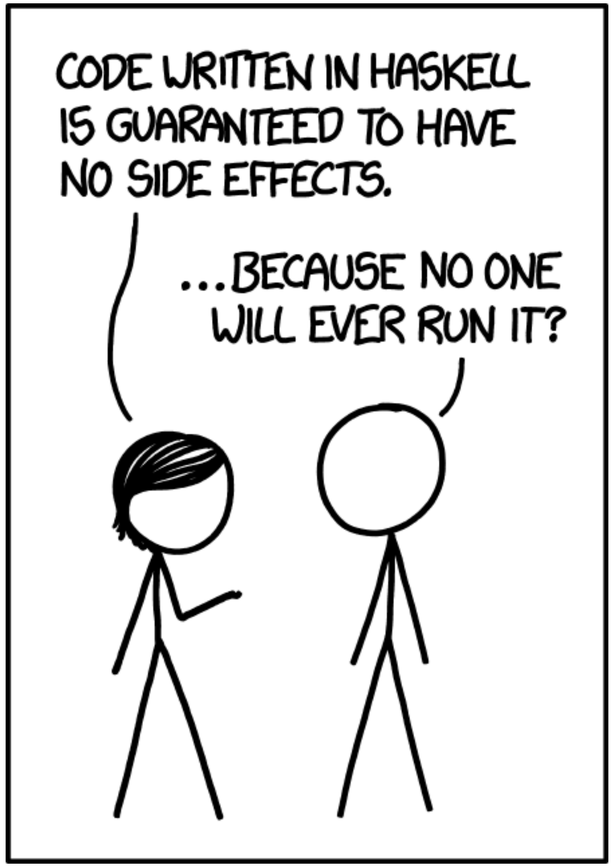
This article **needs additional citations for verification**. Please help improv
material may be challenged and removed.
Find sources: "Function" mathematics – news · newspapers · books · scholar · JSTOR *(Ju*

In mathematics, a **function** from a set X to a set Y assigns to each element of X exactly one element of Y.[1]

There is no “memory”

There is no “step 1”

It is unchanging


CODE WRITTEN IN HASKELL IS GUARANTEED TO HAVE NO SIDE EFFECTS.
...BECAUSE NO ONE WILL EVER RUN IT?

Non-Example

```
List<string> myList = new ArrayList<string>();
bool result0 = myList.isEmpty();
//result0 is true
myList.insert("First!");
bool result1 = myList.isEmpty();
//result1 is false
```

How can we achieve something like this in a functional language?

Haskell Version

```
let my_list_0 = []
let result_0 = is_empty my_list_0
-- result_0 is true
let my_list_1 = insert "First" my_list_0
let result_1 = is_empty my_list_1
-- result1 is false
```

Can we always do this?

- Yes, by the Church Turing Thesis
  - View Turing Machines as a stand-in for imperative styles
  - View the lambda calculus as a stand-in for functional styles

Imperative Code

```
int factorial(int n) {
  int res = 1;
  for (int i = 2; i <= n; i++) {
    res = res * i;
  }
  return res;
}
```

Functional Code

```
let factorial(n) =
  if n <= 2 then n
  else n*factorial(n-1)
```

Should we always do this?

- Honestly, no
- But it can be quite useful in certain places
- Functional programming is a useful tool for good programmers

First-Class Functions

- Historically, only values or objects are first class in OO / procedural languages
  - This is changing!!!
- Implications?
  - Functions can return functions
  - Functions can take functions as inputs
- Deeply understanding first-class functions and recursion will probably be the biggest hurdle in the Haskell portion of this class