Intro to Functional Programming

Drew Ripberger

January 20, 2020

Our Language of Choice

Haskell

"it is a polymorphically statically typed, lazy, purely functional language, quite different from most other programming languages"
haskell.org

Why learn functional programming?

- Concise syntax
- Functional purity simplifies large codebases
- Recursion fits many problems extremely well
- Higher order functions are seen in numerous imperative languages
- More logically consistent with math
- Many functional concepts show up across languages

The Changes with Functional

To state it simply, everything is a function.

- No variables
- No loops

Instead functions and recursion can be leaned on for all of these common tasks like iteration.

An Example

Counting down from 50

Haskell

```haskell
main = mapM_ print [50,49..1]
```

Python

```python
for i in range(50,0,-1):
    print(i)
```

Haskell Breakdown

Haskell

```
main = mapM_ print [50,49..1]
```

Here *mapM_* is a function that maps the *print* function to each of the elements of the list *[50,49..1]*
This results in a countdown from 50 to 1 as such:
50
49
48
...
1

Python Breakdown

Python

```
for i in range(50,0,-1):
    print(i)
```

This Python code produces the same result, however in a vastly different way.
This results in the same countdown from 50 to 1:
50
49
48
...
1

Recursion in Functional

Determining if 22 shows up in a list of integers

Haskell

```
main = print $ has22 [2, 1, 2, 2]

has22 :: [] Int->Bool
has22 nums
          | null nums = False
          | length nums == 1 = False
          | nums!!0 == 2 && nums!!1 == 2 = True
          | otherwise = has22 $ tail nums
```

The Same Problem Imperatively

Python

```
def has22(nums):
  lastwastwo = False
  for i in nums:
    if i == 2:
      if lastwastwo:
        return True
      lastwastwo = True
    else:
      lastwastwo = False

  return False

print(has22([2,1,2,2]))
```

Functions in Math and Haskell

A Constant Function in Math

$$f(x) = 5$$

A Constant Function in Haskell

$$f = 5$$

Functions in Math and Haskell

A Linear Function in Math

$$f(x) = 2x$$

A Linear Function in Haskell

```
f x = 2 * x
```

The Critical Observation

Examples

```
name = "Mel Hoffert"
```

It is important to take notice of what this statement is saying.

In Python or any other imperative language, this would be saying that there exists a variable *name* that has the string "Mel Hoffert" stored in it. However, in a functional setting, *name* is a constant function.

In Summation

- Functional programming has seemingly few *technical* differences than any imperative language, but, these few differences have mass implications. What you will notice is that while some of you Python experience will transfer to Haskell, having to program without side-effects, without loops, and without variables will completely change the way you have to approach a problem.
- Not knowing immediately how to solve what we would consider simple problems in a language will get frustrating, but just imagine you are back, a few years ago learning how to use an *if* statement for the first time. Practice makes perfect.