Introduction to Functional Programming.

Using Python Language

Sriram Sankaranarayanan

February 25, 2018

Imperative Programming

What is *imperative programming*?

- Program is a sequence of commands.
- Manipulates *state* step by step.
- Program is a list of statements.
- Includes: conditional statements, while and for loops.
- Incudes: subroutines (also called "functions").

A Story

Congratulations! You just graduated and joined a top company as a Python developer.

Week 1: Your boss wants you to write a function:

- Input an integer x.
- Double the input.
- Add one to the result of the previous step.
- Then, multiply by itself (square it).
- Next, integer divide the result by 2.
- Next, decrement the result by 1
- Return the value so obtained.

Python Program

```python
def boss_crazy_fun_v1(x):
    x = 2 * x  # Double
    x = x + 1  # Add one
    x = x * x  # Square
    x = x // 2 # Integer divide by 2
    x = x - 1  # Decrement the result by 1
    return x   # Return
```

Story (continued)

Good Job!

1. However, our requirements changed.
2. Modify code to operate on list of integers.

Python Program

```
1  def boss_crazy_fun_v1(lst0):
2      lst1 = [] # This is the return list
3      for x in lst0:
4          x = 2 * x  # Double
5          x = x + 1  # Add one
6          x = x * x  # Square
7          x = x // 2 # Integer divide by 2
8          x = x - 1  # Decrement the result by 1
9          lst1.append(x)
10     return lst1    # Return
```

Story (continued)

You are fast! However, requirements changed (again).

The sequence of operations must now be repeated thrice.

Python Program

```
1  def boss_crazy_fun_v1(lst0):
2      lst1 = [] # This is the return list
3      for x in lst0:
4          for i in range(0,3):
5              x = 2 * x  # Double
6              x = x + 1  # Add one
7              x = x * x  # Square
8              x = x // 2 # Integer divide by 2
9              x = x - 1  # Decrement the result by 1
10         lst1.append(x)
11     return lst1   # Return
```

Imperative Programs

1. Program is a sequence of commands I give to a computer.
2. Think in terms of control flow and control structure.
3. Maintains state.
4. Accumulator variables for maintaining intermediate results.

Let's Think Functionally

What is functional programming?

1. Think in terms of functions.
2. Programming composition of functions.
3. Will avoid loops (for the most part!).
4. Instead we will think iter, map, reduce, filter.

Functional Programming Languages

Languages are based on *functional* programming:

1. LISP, Scheme, Racket,...
2. ML, SML, OCaml, F#, ...
3. Haskell
4. Scala

Languages that provide support for functional programming style:

1. Scripting: Perl, Python, Ruby, PHP, Javascript
2. Multipurpose: C++, Java
3. Query Languages: SQL
4. Special Purpose: R, Julia, ...

Plan

We will study functional programming in Python:

1. Functions and Composition.
2. Functions as Values, Lambdas and Higher Order Functions.
3. Applying Functions: Map, Reduce, Filter, Iter.
4. List Comprehensions in Python.
5. Types/Type Systems.

Functional Story

```
1  from functools import reduce
2  def boss_crazy_fun_v1(lst0):
3      # define the functions we would like to
4      double = lambda y:  2* y
5      add_one = lambda p: p + 1
6      square = lambda l: l * l
7      div2 = lambda w: w//2
8      decr = lambda z: z - 1
9      # here is the sequence of functions we wish to apply
10     seq = [double, add_one, square, div2, decr]
11     # Compose  the functions in list  using reduce
12     f = lambda x: reduce( lambda v, f: f(v), seq, x )
13     # map the function f three times to each value in lst0
14     return_val = map( lambda x: f(f(f(x))), lst0)
15     # return the result as a list
16     return list(return_val)
```