SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

15 Functional Programming in Python

Dr. Spencer Smith

Faculty of Engineering, McMaster University

February 7, 2018

15 Functional Programming in Python

- Administrative details
- Functional programming
- Functional programming in Python
  - Defining functions
  - List comprehension
  - Map
  - Filter
  - Fold (Reduce)
  - Anonymous functions
  - Partial functions

Administrative Details

- Assignment 1 solution in repo
- Assignment 2
  - Part 1: February 12, 2018
  - Partner Files: February 18, 2018
  - Part 2: March 2, 2018
  - Steps have been written out
  - When working with object, either create or use references
  - Use exact names and cases
  - Exceptions take one argument, a string
  - Exceptions in `Exceptions.py`
  - `Data.accessProg`, not `Data_accessProg` or `Data.Data_accessProg`
  - Two sample input files in repo
  - Added `Data_getC`

Functional Programming

- Computation is treated a the evaluation of mathematical functions (not CS subroutines)
  - No state
  - No mutable data
  - Programming with expressions, not statements
- No side effects
- Easier to reason about than imperative or OO code
- Functions are a first order data type
  - Can pass functions as arguments
  - Can return functions
- Origin with lambda calculus
- There is a focus on list processing

Functional Programming in Python

- Python was an imperative/OO language first
- Other functional languages (like Haskell) have pattern matching
- Python is dynamically typed
- Cannot inspect the type of a function

List Comprehension

- Examples from Learn You a Haskell for Great Good
- Gries and Schneider notation for set comprehension: $\{x : T | R : E\}$
  - x is the dummy variable
  - E is an expression
  - R is a predicate
- Modified version: $\{x : T | R \wedge P : E\}$
  - P is a predicate (filter)
- Python code: `[E for x in R if P]`
  - R is a sequence (list)

List Comprehension Examples

$\{x : T | R \wedge P : E\}$ (set) to `[E for x in R if P]` (sequence)

- $\{x : \mathbb{N} | x \in [1..10] : x^2\}$
  ```
  [x**2 for x in range(1, 11)]
  ```
- $\{x : \mathbb{N} | x \in [1..10] \wedge x^2 \geq 12 : x^2\}$
  ```
  [x**2 for x in range(1, 11) if x**2 >= 12]
  ```
- A list or radii for a seq circles of CircleT (A1-2017)
  ```
  radii=[c.radius() for c in circles]
  ```
- $[S_0, S_1, ..., S_{|S|-1}]$ to
  $[S_0.\text{eval}(x), S_1.\text{eval}(x), ..., S_{|S|-1}.\text{eval}(x)]$ (A2-2018)
  ```
  [s.eval(x) for s in S]
  ```

List Comprehension to find List Length

```
def length(xs):
    return sum([? for x in xs])
```

What should ? be to return the length of xs?

Similar to how we write count mathematically:
$+(i : \mathbb{N} | x \in xs : 1)$

List Comprehension Examples Cont'd

- Write a function `rep(x, n)` that returns a list of `n` elements, where each element is `x`
- Write a function that takes a list of integers (`xs`) and replaces each odd number greater or equal to 10 with `"BANG!"` and each odd number that's less than 10 with `"BOOM!"`
  - What is the basic structure for the list comprehension?
    [E for x in R if P]
  - What are R and P?
    [E for x in xs if odd(x)]
  - How do you write conditional expressions in Python?
    `x = true_value if condition else false_value`
  - What is E?
    ["BOOM!" if $x < 10$ else "BANG!" for x in xs if odd(x)]

List Comprehension Two Lists

- Given the lists
  - `nouns = ["smurf", "frog", "dwarf"]`
  - `adjectives = ["lazy", "grouchy", "scheming"]`
- Write a list comprehension that concatenates all the adjectives with all the nouns

Remove Everything but Uppercase

Write a function `removeNonUpperCase(st)` that takes a string `st` and returns the string that results by removing all non upper case letters

How would you build the sequence of `['A', 'B', ..., 'Z']`?

```
[chr(i) for i in range(ord('A'),ord('Z')+1)]
```

Nested List Comprehension

Given a list of several lists of numbers, remove all odd numbers without flattening the list.

```
xxs =
[[1,3,5,2,3,1,2,4,5],[1,2,3,4,5,6,7,8,9],[1,2,...
```

Map

- Examples from Learn You a Haskell for Great Good
- Mathematical model:
  - map : $(a \to b) \times$ seq of $a \to$ seq of b
  - map : $(a \to b) \times [a] \to [b]$
- Python code: `map(func, seq)`

Map Example

```
def add3(x):
    return x + 3

list(map(add3, [1, 5, 3, 1, 6]))
```

- What do you think will be printed?
- What is the type of `add3`?
- What type does `map` return in this case?

Anonymous Function Example

```
list(map(lambda x: x + 3, [1, 5, 3, 1, 6]))
```

or

```
add3 = lambda x: x + 3
list(map(add3, [1, 5, 3, 1, 6]))
```

- `lamda` followed by list of arguments: expression
- Write code to add `'!'` to every string in a list of strings