Programming Fundamentals Using Python

2018

Problem Set 13

Most recent updated: July 17, 2018

Objectives

1. Higher order function

Note: Solve the programming problems listed using your favorite text editor. Make sure you save your programs in files with suitably chosen names, **and try as much as possible to write your code with good style (see the style guide for python code)**. In each problem find out a way to test the correctness of your program. After writing each program, test it, debug it if the program is incorrect, correct it, and repeat this process until you have a fully working program. Show your working program to one of the cohort instructors.

Problems: Cohort sessions

1. Revise on Functions Write the following functions using while loop.

   - `def sum_naturals(n)`: this function returns the sum of natural numbers from 1 up to n.
   - `def sum_cubes(n)`: this function returns the sum of the cubes of natural numbers from 1 up to n.
   - `def pi_sum(n)`: this function computes pi using a series. The number of terms in the summation is specified by `n`.

     $$\pi \approx \frac{8}{1 \times 3} + \frac{8}{5 \times 7} + \frac{8}{9 \times 11} + \ldots \tag{1}$$

     Note that the denominator can be written as $(4i-3) \times (4i-1)$ for $i = 1, 2, \ldots$.

2. *Abstract function* Write a function that abstracts the three functions above under one function called `summation(n, term)` where `n` is an integer, which is the number of terms, and `term` is a function that computes the term of the summation. For example, one can pass on `cube(x)` that computes the cube of a number as an argument to `term`. Rewrite the three functions using the abstract function `summation(n, term)`.

3. *Abstracting method* Write a function `improve(update, close, guess = 1)`. The argument `update` and `close` are functions. The function `update` calculates and update the guess value of the solution. The function `close` checks if the guess is close to converging. The argument `guess` is the initial guess of the solution. Write a function `golden_update(guess)` that serves as the update function to calculate a golden ratio. You can use the following relations:

   $$\phi = 1 + \frac{1}{\phi} \tag{2}$$

   Write also a function `square_close_to(guess)` which checks if the square of the guess is close enough satisfying the following relationship:

   $$\phi^2 = \phi + 1 \tag{3}$$

   Reference: `https://en.wikipedia.org/wiki/Golden_ratio#Calculation`. You can make use of the following function definition to check if the square of ϕ is close to $\phi + 1$.

   ```
   def approx_eq(x, y, tolerance=1e-15):
           return abs(x - y) < tolerance
   ```

4. *Closure* Enclose the previous `golden_update()` and `square_close_to()` inside a function called `calculate_phi(tolerance)` where `tolerance` is an argument that is set to check if the guess is close.

5. *Newton's Method* Write a function `newton_update(f, df)` that returns an update function for Newton's method:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{4}$$

The arguments `f` and `df` are the function and its derivative respectively. The update function should take in one argument `x` which is the guess of the current iteration (x) and it should return the guess of the next iteration, i.e. x_{i+1}.

Write a function `find_zero(f, df)` that makes use of Newton's method to find the zero of a function. Write a function `square_root_newton(a)` that calculates the square root of the number `a`.

6. *lambda function* Rewrite some of the functions in Newton's method problem using lambda function.

End of Problem Set 13.