PSYCH 420 - Assignment 2

Objective

- Write a function on Julia that estimates the fourth root (or the nth root) of a number, given a guess.
- As per feedback from assignment 1, code from python and Julia is submitted for a "side-by-side" comparison.
- The full julia file is available in this folder: Assignment 2 Julia Link
- The full python file is available in this folder: Assignment 2 Python Link

The Fourth Root Function

- A function to estimate the fourth root can be written using the Newton-Raphson method.
- Julia:

```
function fourth_root_manual(y:: Float64, guess:: Float64)
    # let the number we are finding be x.
    # we know y = x*x*x*x
    # 0 = x*x*x*x - y
    error= 1.e-6
    iterations = 0

    while (error <= abs(y - guess^4)) && (iterations < 100)
        guess = guess + (y - guess^4) / (4*guess^3)
        iterations += 1
    end

    return guess
end
```

- Python:

```
MAX_ITERATIONS: int = 100 # The maximum number of iterations allowed in this module.
TOLERANCE: float = 1.e-6 # The tolerance is 1x10^-6.

def fourth_root_manual(x: float, guess: float = 1):
""" Returns the fourth root of x, using the Newton-Raphson method. """
    iterations: int = 0
    while (TOLERANCE <= abs(x - guess**4)) and (iterations < MAX_ITERATIONS):
        derivative: float = (4 * (guess**3)) # derivative of x^4 is 4x^3
        guess = guess + (x - guess**4) / derivative # update guess using Netwon Raphson
        iterations += 1 # Update the number of iterations
    return guess
```

The Nth Root Function

- The fourth root function can be slightly modified to calculate the nth root.
- Julia:

```
    function nth_root_manual(y:: Float64, n:: Int64, guess:: Float64)
        error= 1.e-6
        iterations = 0

        while (error <= abs(y - guess^n)) && (iterations < 100)
            guess = guess + (y - guess^n) / (n*guess^(n-1))
            iterations += 1
        end

        return guess
    end
```

- Python:

```
def nth_root_manual(x: float, n: int, guess: float = 1):
    """ Returns the nth root of x, using the Newton Raphson method.
"""
    iterations: int = 0
    while (TOLERANCE <= abs(x - guess**n)) and (iterations <
MAX_ITERATIONS):
        derivative: float = n * (guess**(n-1)) # derivative of x^n is
(n)x^(n-1) by the Chain Rule.
        guess = guess + (x - guess**n) / derivative # update guess
using Netwon Raphson
        iterations += 1 # Update the number of iterations
    return guess
```

Outputs

- Terminal outputs from running the accompanying files.
- Julia:

```
pchaddha@Parmandeeps-MacBook-Pro juliapsych420 % julia
assignment_2_script.jl
The fourth root of 36.0 is 2.449489742783178.
The fifth root of 36.0 is 2.0476725110792193.
The fourth root estimation of 36.0 using Newton Raphson is
2.449489742783237.
```

```
The fifth root estimation of 36.0 using Newton Raphson is
2.0476725111078875.
```

- Python:

```
(root-oHblsZw_-py3.8) pchaddha@Parmandeeps-MacBook-Pro compNeuroIntro420 %
poetry run python ./juliapsych420/assignment_2.py
    The fourth root of 36 is 2.449489742783178.
    The fifth root of 36 is 2.0476725110792193.
    The fourth root estimation of 36 using Newton Raphson is
2.449489742783237.
    The fifth root estimation of 36 using Newton Raphson is
2.0476725111078875.
```