BEE 4750/5750 Homework 0

Katerina Tang (kbt28)

2022-08-27

Problem 1

Problem 1.1

```
julia> function square_number(x)
           output = x.^2
           return output
       end
square_number (generic function with 1 method)
```

Problem 1.2

If $x = 5$, then $x^2 = 25$.

Problem 1.3

```
julia> x = LinRange(-10, 10, 100)
100-element LinRange{Float64}:
 -10.0,-9.79798,-9.59596,-9.39394,…,9.19192,9.39394,9.59596,9.79798,10.0

julia> y = square_number(x)
100-element Vector{Float64}:
 100.0
  96.00040812162027
  92.08244056728904
  88.24609733700643
  84.49137843077234
  80.81828384858687
  77.22681359044998
  73.71696765636158
  70.2887460463218
  66.94214876033055

  70.2887460463218
  73.71696765636158
  77.22681359044998
```

```
  80.81828384858687
  84.49137843077234
  88.24609733700643
  92.08244056728904
  96.00040812162027
 100.0

julia> plot(x, y, xlabel="x", ylabel="y = x^2", legend=false)
Error: UndefVarError: plot not defined
```

Problem 2

Problem 2.1

If $a \leq \sqrt{x}$, then

$$\frac{x}{a} \geq \frac{x}{\sqrt{x}} = \sqrt{x}.$$

If $a > \sqrt{x}$, then

$$\frac{x}{a} < \frac{x}{\sqrt{x}} = \sqrt{x}.$$

In both cases, $a \leq \sqrt{x} \leq \frac{x}{a}$.

Problem 2.2

```
julia> function newton_sqrt(x, err_tol)
           a = x

           while true
               root = 0.5 * (a + (x/a))
               if abs(root - a) < err_tol
                   return root
               end
               a = root
           end
       end
newton_sqrt (generic function with 1 method)
```

Using this method and an error tolerance of 10^{-8} gives $\sqrt{2} \approx 1.414213562373095$.

Problem 3

Problem 3.1

```
julia> a = rand(20)
20-element Vector{Float64}:
 0.9331393110382806
 0.38244659716069385
 0.41977073365946094
 0.8555305821357433
 0.8103009116740416
 0.6155412821813948
 0.6377846882948912
 0.3615148479227228
 0.07329921230608605
 0.062224160155968766
 0.7378285726719824
 0.8822537598110793
 0.7929164229169792
 0.016394654145352794
 0.24715780101110574
 0.7915704121937792
 0.46118191246416784
 0.37290593628954327
 0.19413114132083376
 0.2760736495169003
```

Problem 3.2

```
julia> function mean(vect)
           sum = 0

           for i in 1:length(vect)
               sum += vect[i]
           end

           return sum/length(vect)
       end
mean (generic function with 1 method)

julia> function demean(vect)
           return vect .- mean(vect)
       end
demean (generic function with 1 method)
```

We can test this with our random vector from part (1). ~~{.julia} julia> mean(a) 0.49619832944355047~~~~~~~~~~~~

```
julia> demean(a)
20-element Vector{Float64}:
  0.43694098159473016
 -0.11375173228285662
 -0.07642759578408953
  0.3593322526921928
  0.31410258223049115
  0.11934295273784434
  0.14158635885134074
 -0.13468348152082765
 -0.4228991171374644
 -0.4339741692875817
  0.24163024322843196
  0.3860554303675289
  0.2967180934734287
 -0.4798036752981977
 -0.24904052843244473
  0.29537208275022875
 -0.035016416979382625
 -0.1232923931540072
 -0.3020671881227167
 -0.2201246799266502
```

Problem 3.3

```
julia> b = zeros(10)
10-element Vector{Float64}:
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0
 0.0

julia> b[3:8] .= 1.0
6-element view(::Vector{Float64}, 3:8) with eltype Float64:
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0

julia> b
```

```
10-element Vector{Float64}:
 0.0
 0.0
 1.0
 1.0
 1.0
 1.0
 1.0
 1.0
 0.0
 0.0
```

Problem 3.4

```
julia> A = rand(5, 5)
5×5 Matrix{Float64}:
 0.207739  0.229437  0.61901   0.396694  0.588099
 0.270674  0.795164  0.384184  0.384746  0.368904
 0.742748  0.943977  0.916157  0.439151  0.209607
 0.992604  0.564283  0.30638   0.876965  0.0185846
 0.702607  0.980005  0.61078   0.5349    0.533109
```

```
julia> for i in 1:5
           A[:, i] .-= mean(A[:, i])
       end
```

Problem 4

Problem 4.1

Problem 4.2

Problem 4.3

References