[NEU_314_2019] Elin Ahlstrand EX 1

2. Boolean Indexing

- Using Boolean indexing and vector V from problem 1, write a single line of Julia code that will return all the values of V that are greater or equal to 30.

```
V[V .>= 30]

7-element Array{Int64,1}:
 30
 40
 50
 60
 50
 40
 30
```

- Using Boolean indexing and vector V from problem 1, write a single line of Julia code that will return all the values of V that are greater or equal to 30 *and* less or equal to 40.

```
V[30 .<= V.<= 40]

4-element Array{Int64,1}:
 30
 40
 40
 30
```

- Using Boolean indexing and the vectors t and V from problem 1, write a single line of Julia code that will return *the times* at which V is greater or equal to 30.

```
t[findall(V.>=30)]
7-element Array{Float64,1}:
 0.2
 0.4
 0.5
 0.6
 0.7
 0.8
 0.9
```

- Using Boolean indexing and the vectors t and V from problem 1, write a single line of Julia code that will return the number of elements of V that are greater or equal to 30.

```
length(t[findall(V.>=30)])
7
```

4.Boolean indexing

- Write Julia code that will calculate how many spikes were fired on Monday, but calculate this only for times when the current I was turned on (i.e., was bigger than 0).

INPUT

```
function spike_number(day::AbstractString)
      I = data[day]["I"];
      V = data[day]["V"];
      x=0;

      for i = 1:length(I)
            if I[i] .> 0
                  if V[i] .> 0
                  x +=1
                        end
            end
      end
      return x
end
```

OUTPUT

```
spike_number("Monday")
3
```

Also made a function that calculates the spike numbers on all days, and outputs this as an array.

INPUT

```
function spike_numbers(data::AbstractDict)

      days = collect(keys(data));
      days_spikes = zeros(length(days));
```

```
       for i=1:length(days)
             I = data[days[i]]["I"];
             V = data[days[i]]["V"];
             x=0;

             for i = 1:length(I)
                    if I[i] .> 0
                          if V[i] .> 0
                          x +=1
                          end
                    end
             end
             days_spikes[i]+= x
       end
       spike_stack = [days; days_spikes];
       spike_stack = reshape(spike_stack, (length(days),2))
       println(spike_stack)
end
```

OUTPUT

```
spike_numbers(data)
Any["Friday" 10.0; "Tuesday" 11.0; "Thursday" 4.0; "Wednesday" 1.0; "Monday" 3.0]
```

- Write Julia code that will calculate how many spikes *per second* were fired on Tuesday while the current was turned on (i.e., your code needs to figure out the duration of how long the injection current was on. It's ok to assume that timebins are 0.001 seconds long.)

INPUT

```
I = data["Tuesday"]["I"];
    spikes = spike_number("Tuesday");
    x=0;

    for i = 1:length(I)
          if I[i] .> 0
                x +=1
          end
    end
return spikes/(x*.001)
```

OUTPUT

```
5.5027513756878434
```

- Write a function that takes as inputs
  - A variable in the form of the `data` variable
  - A string representing the day of the week (e.g., “Tuesday”)

    And returns the spikes per second (i.e., firing rate) produced that day when the current was turned on.

```
function firing_rate(data::AbstractDict, day::AbstractString)

      I = data[day]["I"];
      spikes = spike_number(day);
      x=0;

      for i = 1:length(I)
            if I[i] .> 0
                  x +=1
            end
      end
      return spikes/(x*.001)
end
```

OUTPUT

```
firing_rate(data,"Tuesday")
5.5027513756878434
```

Also made a function that calculates the firing rates on all days, and outputs this as an array.

INPUT

```
function firing_rates(data::AbstractDict)
      days = collect(keys(data));
      numbers_spikes = zeros(length(days));

      for i=1:length(days)
```

```
          I = data[days[i]]["I"];
          spikes = spike_number(days[i]);
          x=0;
             for i = 1:length(I)
                   if I[i] .> 0
                         x +=1
                   end
             end
           numbers_spikes[i]+= spikes/(x*.001)
      end
      firing_stack = [days; numbers_spikes];
      firing_stack = reshape(firing_stack, (length(days),2))
      println(firing_stack)
end
```

OUTPUT

```
firing_rates(data)
Any["Friday" 5.0025; "Tuesday" 5.50275; "Thursday" 4.004; "Wednesday"
2.00401; "Monday" 3.003]
```

- Using the function you wrote in the previous bullet point, write a new function, that you'll call `average_rate()` and that takes a single input
  - A variable in the form of the `data` variable

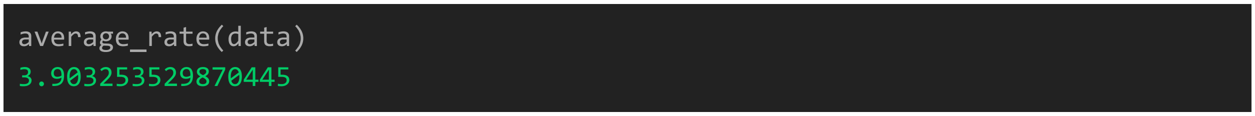
  That function should internally use a for loop to iterate over all the days of the week, and return the answer to: How many spikes per second were fired, if you average all the days of the experiment together, during times when the current was turned on?

INPUT

```
function average_rate(data::AbstractDict)

      days = collect(keys(data));
      x = 0;
           for i=1:length(days)
               day = days[i]
               x += firing_rate(day);
           end
      return x / length(days)
end
```

OUTPUT



```
average_rate(data)
3.903253529870445
```