```
Plots.GRBackend()
```

```julia
begin
    using PlutoUI
    using Images
    using ImageMagick
    using Plots
    gr()
end
```

A1: Plot an exponential curve with paramter τ and the numerical solution of the leaky bucket model with parameters C and λ on the same axes, with u = 0 and the same initial state

```
bucket_state_step (generic function with 1 method)
```

```julia
function bucket_state_step(v, Δt, C, λ, v0, u) # set up function bucket_state_step with paramaters v, Δt, C, λ, v0, u
Δv = (u - λ*(v - v0))*Δt/C # such that Δv = (u - λ*(v - v0))*Δt/C
end
```

```
0.0
```

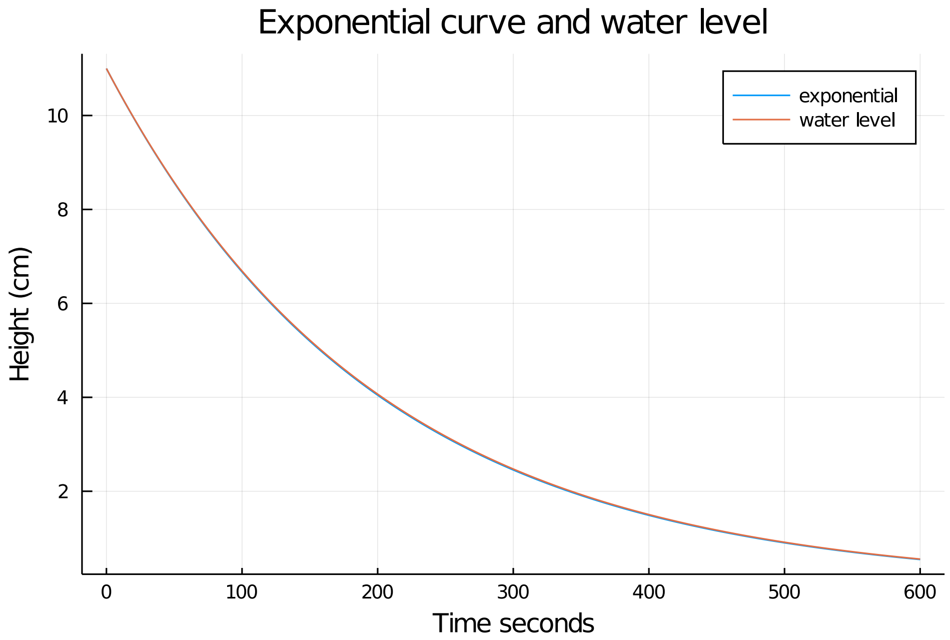
```julia
begin #establish first set of parameters for first plot
C = π*r^2 # C = pi*radius^2 where radius = r slider value
Δt = 0.05 # simulate in 50ms steps
T = 600.0 # duration of simulation in seconds
t = 0:Δt:T; # time iterator from 0 seconds to 600 seconds in 50ms steps
v = zeros(length(t)) # vector container for computed water level
v[1] = 11.0 # initial height = 11.0cm
v_rest = 0.0 # leak height = 0.0cm
u = 0.0 # input current = 0.0
end
```

C = 201.06192982974676

r slider, r = 8

λ slider, λ = 1

τ slider, τ = 200




```
begin
    plot(t, v[1]*exp.(-t./τ), # plot time iterator vs exponential curve mulitplied by initial height
        xlabel = "Time seconds", ylabel = "Height (cm)", label = "exponential", title = "Exponential curve and water level")
for i in 2:length(t) # for each iteration from the 2nd to the last iterations
v[i] = v[i-1] + bucket_state_step(v[i-1], Δt, C, λ, v_rest, u) # each iteration of v = the previous iteration + the bucket_state_step function
end
plot!(t,v, label = "water level") # add a plot of the time iterator vs water level to the same axes
end
```

A1.1: What is the relationship of the time constant τ of the analytical solution to the "biophysical" parameters C and λ? (Show this empirically)

By observation, as τ increases, C increases and/or λ decreases.

τ and C are directly proportional (τ = R*C) while τ and λ are inversely proportional (since R = 1/λ).

A1.2: What is the height of the water column above the leak channel at time t=τ relative to its initial height? (show this analytically and confirm by simulation)

Analytically: at t=τ the water column is 36.8% (0.36787944117 = 1/e) of its initial height above the leak channel

By simulation: when τ=300 and t=300, and when τ=200 and t=200, and when τ=100 and t=100, approximately 4.05cm out of 11cm remains, 4.05/11 = 0.368.

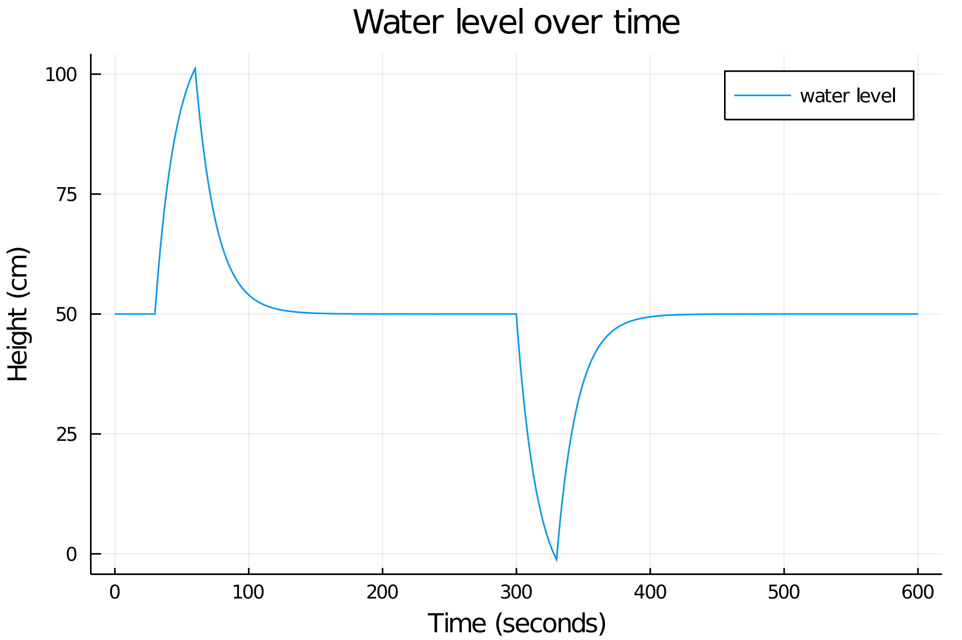
A2: Create an interactive simulation of a leaky bucket neuron that starts at rest potential and receives a 30-second burst of "synaptic input" after 30 seconds. Use a slider to control the amplitude of the input current. Allow the input to be positive or negative, ie it is possible to suck water out of the bucket. Assume that the bucket is tall enough and the leak channel is far enough up the side that the bucket cannot run dry or overflow. Note that water will leak backwards through the leak channel, into the bucket, if the water level drops below the channel. Note that we are starting to stretch the analogy between buckets and neurons. Soon it will break.

```
bucket_state_step_ (generic function with 1 method)
```

```julia
function bucket_state_step_(v_, Δt_, C_, λ_, v0_, u_) # set up function bucket_state_step_ with parameters v_, Δt_, C_, λ_, v0_, u_
Δv_ = (u_ - λ_*(v_ - v0_))*Δt_/C_ # such that Δv_ = (u_ - λ_*(v_ - v0_))*Δt_/C_
end
```

```julia
begin #establish second set of parameters for second plot
C_ = π*5.0^2 # C_ = pi*radius^2 where radius = 5.0
λ_ = 5.0 # λ_ = 5.0
Δt_ = 0.05 # simulate in 50ms steps
T_ = 600.0 # duration of simulation in seconds
t_ = 0:Δt_:T_; # time iterator from 0 seconds to 600 seconds in 50ms steps
v_ = zeros(length(t_)) # vector container for computed water level
v_[1] = 50.0 # initial height = 50.0cm
v_rest_ = 50.0 # leak height = 50.0cm
u_ = zeros(length(t_)) # vector container for variable input
    for i in 600:1200 # between 30 seconds and 60 seconds
        u_[i] = 1.0 # input is 1.0*"the value of the u slider"
    for i in 6000:6600 # between 300 and 330 seconds
        u_[i] = -1.0 # input is -1.0*"the value of the u slider"
        end
    end
end
```

u slider, u = 300




```
begin
    for i in 2:length(t_) # for each iteration from the 2nd to the last iterations
        v_[i] = v_[i-1] + bucket_state_step_(v_[i-1], Δt_, C_, λ_, v_rest_, u_[i]*u__)
# each iteration of v_ = the previous iteration + the bucket_state_step_ function
where the input is 1.0*"the value of the u slider" for 30s < t < 60s and -1.0*"the
value of the u slider" for 300s < t < 330s
    end
    plot(t_,v_, xlabel = "Time (seconds)", ylabel = "Height (cm)", label = "water
level", title = "Water level over time") # plot t_ vs v_
end
```