1. A pig gains weight according to a logistics curve such that the weight after t days is

$$w(t) = \frac{800}{1 + 3e^{-t/30}} \text{ lbs.}$$

(i) Show that the pig is gaining about 5 lbs/day at $t = 0$. What happens as t increases?

To show $w'(0) = 5$ use the Symbolics library in Julia.

In [1]:
```
using Symbolics
D(f,x)=expand_derivatives(Differential(x)(f))
@variables t
```

Out[1]:
```
1-element Vector{Num}:
 t
```

In [2]:
```
# Define the function and check it is correct
w(t)=M/(1+3*exp(-t/30))
M=800
w(t)
```

Out[2]:
```
800 / (1 + 3exp((-1//30)*t))
```

In [3]:
```
# Create the derivative dw(t)
dwtmp=D(w(t),t)
dws="dw(t)="*string(dwtmp)
eval(Meta.parse(dws))
dw(t)
```

Out[3]:
```
((80//1)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2)
```

In [4]:
```
# Evaluate dw(0)
dw(0)
```

Out[4]:
```
5.0
```

(ii) Suppose it costs 45 cents a day to keep the pig and the market price for pigs is 65 cents per pound, but is falling 1 cent per day. Find the optimal time to sell the pig.

Define the profit as

$$P(t) = p(t) \cdot w(t) - c(t)$$

where $p(t)$ is the price of the pig at time t and $c(t)$ is the cost of keeping the pig for t days.

Then solve $P'(t) = 0$ to find the optimal time to sell the pig.

In [5]:
```
# Create the profit function P(t) and display it
p(t)=65-t
c(t)=45*t
P(t)=p(t)*w(t)-c(t)
P(t)
```

Out[5]:
```
-45t + (800(65 - t)) / (1 + 3exp((-1//30)*t))
```

In [6]:
```
# Create the derivative dP(t)
dPtmp=D(P(t),t)
dPs="dP(t)="*string(dPtmp)
eval(Meta.parse(dPs))
dP(t)
```

Out[6]:
```
-45 + ((80//1)*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2) + -800 / (1 + 3exp((-1//30)*t))
```

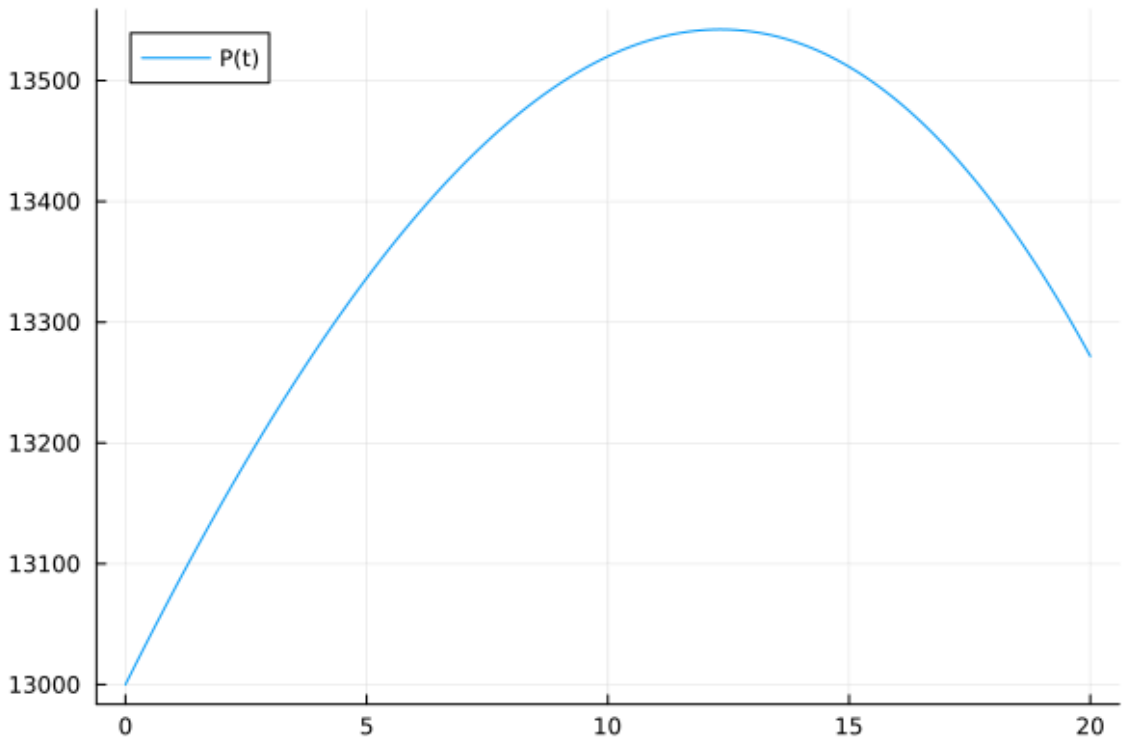
Since the $P'(t)$ is nonlinear use Newton's method to approximate a solution. First graph $P(t)$ to obtain a good initial guess of the maximum and then iterate

$$t_{n+1} = t_n - g(t)/g'(t)$$

where $g(t) = P'(t)$ and $g'(t) = P''(t)$.

In [7]:
```
using Plots
plot(P,0:0.1:20,label="P(t)")
```

Out[7]:



From the graph it appears the $t_0 = 12$ is a reasonable initial guess for Newton's method.

In [8]:
```
g(t)=dP(t)
# Create the derivative dg(t)
dgtmp=D(g(t),t)
dgs="dg(t)="*string(dgtmp)
eval(Meta.parse(dgs))
dg(t)
```

Out[8]:
```
(-(80//1)*exp((-1//30)*t) - (8//3)*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp
((-1//30)*t))^2) + ((16//1)*(65 - t)*(exp((-1//30)*t)^2)) / ((1 + 3exp((-
1//30)*t))^3) + ((-80//1)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2)
```

In [9]:
```
# Perform Newton's iterations
tn=12.0
for n=1:5
    tn=tn-g(tn)/dg(tn)
    println("t_$n=",tn)
end
topt=tn
Popt=P(topt)
println("\nThe optimal time to sell the pig is ",topt)
println("The optimal profit is ",Popt)
```

```
t_1=12.337004792474968
t_2=12.334891525596113
t_3=12.334891442632154
t_4=12.334891442632161
t_5=12.334891442632161

The optimal time to sell the pig is 12.334891442632161
The optimal profit is 13542.357422246736
```

(iii) The parameter 800 represents the eventual mature weight M of the pig. Thus,

$$w(t) = \frac{M}{1 + 3e^{-t/30}} \text{ lbs.}$$

Perform a sensitivity analysis for the parameter M. Compute $S(t, M)$ and $S(P, M)$ evaluated at $M = 800$. Here P is the profit obtained at the optimal time t to sell the pig.

In [10]:
```
# Change M to a variable
@variables M
```

Out[10]:
```
1-element Vector{Num}:
 M
```

Note the above idea to change M to a variable won't work in Pluto because in order to keep track of dependencies Pluto doesn't allow one to change the definition of a variable in the middle of a calculation. For Pluto, one would need

to start over with a new function $\widetilde{w}(t)$ and a different name for the variable M. Here we are using JupyterLab which works the same as at the Julia command line so we can simply redefine M.

In [11]:
```
# Verify p(t) not has M in it
w(t)
```

Out[11]:
```
M / (1 + 3exp((-1//30)*t))
```

In [12]:
```
# Recompute dP(t) and g(t)
dPtmp=D(P(t),t)
dPs="dP(t)="*string(dPtmp)
eval(Meta.parse(dPs))
g(t)=dP(t)
g(t)
```

Out[12]:
```
-45 + ((1//10)*M*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2) +
(-M) / (1 + 3exp((-1//30)*t))
```

To compute dt/dM use implicit differentiation and the fact that $g(t) = 0$. Thus,

$$\frac{dg}{ds} = \frac{\partial g}{\partial t}\frac{dt}{dM} + \frac{\partial g}{\partial M} = 0$$

implies

$$\frac{dt}{dM} = -\frac{\partial g}{\partial M} \Big/ \frac{\partial g}{\partial t}$$

In [13]:
```
dgdM=D(g(t),M)
```

Out[13]:
```
((1//10)*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2) + -1 / (1 +
3exp((-1//30)*t))
```

In [14]:
```
dgdt=D(g(t),t)
```

Out[14]:
```
((1//50)*M*(65 - t)*(exp((-1//30)*t)^2)) / ((1 + 3exp((-1//30)*t))^3) + ((-
1//10)*M*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2) + (-(1//10)*M*exp((-
1//30)*t) - (1//300)*M*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^
2)
```

In [15]:
```
# Now evaluate S(t,M) at t=tn and M=800
StMtmp=substitute(-M/t*dgdM/dgdt,[t=>topt,M=>800])
StM=eval(Symbolics.toexpr(StMtmp))
println("S(t,M)=",StM)
```

```
S(t,M)=0.4329428182079427
```

To compute $S(P, M)$ note that

$$\frac{dP}{dM} = \frac{\partial P}{\partial t}\frac{dt}{dM} + \frac{\partial P}{\partial M}$$

and at the optimal time t that $\partial P/\partial t = 0$. Therefore

$$S(P,M) = \frac{M}{P}\frac{\partial P}{\partial M}\Big|_{t=t_n, M=800}$$

In [16]:
```
P(t)
```

Out[16]:
```
-45t + (M*(65 - t)) / (1 + 3exp((-1//30)*t))
```

In [17]:
```
dPdM=D(P(t),M)
```

Out[17]:
```
(65 - t) / (1 + 3exp((-1//30)*t))
```

In [18]:
```
eval(Symbolics.toexpr(substitute(dPdM,[t=>topt,M=>800])))
```

Out[18]:
```
17.621784421456482
```

In [19]:
```
SPMtmp=substitute(M/P(t)*dPdM,[t=>topt,M=>800])
SPM=eval(Symbolics.toexpr(SPMtmp))
println("S(P,M)=",SPM)
```

```
S(P,M)=1.0409877023336132
```

Note that it is also possible to approximate the sensitivity by considering $M = 801$ and recomputing the optimal time and then using

$$\frac{dt}{dM} \approx \frac{\Delta t}{\Delta M} \qquad \text{and} \qquad \frac{dP}{dM} \approx \frac{\Delta P}{\Delta M}$$

We do that now as an optional check on the above answer.

In [20]:
```
# Optional way of approximating S(t,M) and S(P,M)
M=801
dPtmp=D(P(t),t)
dPs="dP(t)="*string(dPtmp)
eval(Meta.parse(dPs))
g(t)=dP(t)
g(t)
```

Out[20]:
```
-45 + ((801//10)*(65 - t)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))^2) + -
801 / (1 + 3exp((-1//30)*t))
```

In [21]:
```
dgtmp=D(g(t),t)
dgs="dg(t)="*string(dgtmp)
eval(Meta.parse(dgs))
dg(t)
```

Out[21]:
```
((801//50)*(65 - t)*(exp((-1//30)*t)^2)) / ((1 + 3exp((-1//30)*t))^3) + (-
(801//10)*exp((-1//30)*t) - (267//100)*(65 - t)*exp((-1//30)*t)) / ((1 + 3e
xp((-1//30)*t))^2) + ((-801//10)*exp((-1//30)*t)) / ((1 + 3exp((-1//30)*t))
^2)
```

In [22]:
```
# Perform Newton's iterations
tn=12.0
for n=1:5
    tn=tn-g(tn)/dg(tn)
    println("t_$n=",tn)
end
tnew=tn
println("\nThe new optimal time to sell the pig is ",tnew)
```

```
t_1=12.34375594742852
t_2=12.341557751357138
t_3=12.341557661617747
t_4=12.341557661617747
t_5=12.341557661617747

The new optimal time to sell the pig is 12.341557661617747
```

In [23]:
```
StMapp=800/topt*(tnew-topt)/(M-800)
println("The approximate S(t,M)=",StMapp)
```

```
The approximate S(t,M)=0.43234877366143437
```

In [24]:
```
SPMapp=800/Popt*(P(tnew)-Popt)/(M-800)
println("The approximate S(P,M)=",SPMapp)
```

```
The approximate S(P,M)=1.040998778405237
```

Note that the approximate values for $S(t, M)$ and $S(P, M)$ computed using $M = 801$ are close to the values obtained using implicit differentiation and the chain rule in the earlier calculation. This consistency suggests both answers are correct.

In [ ]: