

**Course - Section**

**Title:** Subtitle

Due on February 3<sup>rd</sup>, 2026 at 11:59 AM

*Instuctor, 09:00 AM*

**Author**

February 4<sup>th</sup>, 2026

**Problem 1: Introducing Problems (# pts.)**

Homework problems are placed on individual pages.

a. Instructions

b. Instructions

**Solution**

Solutions are placed below the problem statement, and can be split

**Part A**

into

**Part B**

different parts.

## Problem 2: Defining Features (# pts.)

Problems can include **inline math**:  $F_P = -b\dot{x}^2$  and **display math**:

$$\Delta E_{12} = Q_{12} + W_{12} \therefore W_{12} = Q_{12}.$$

## Solution

And so can solutions. Personally, I like to use `align*` environments (though you can use `gather*` environments) for multi-line math.

$$\begin{aligned} m_P \frac{d\dot{x}}{dt} &= F_P \\ m_P \frac{d\dot{x}}{dt} &= -b\dot{x}^2 \\ \frac{d\dot{x}}{\dot{x}^2} &= -\frac{b}{m_P} dt \\ \left[ -\frac{1}{\dot{x}} \right]_{v_0}^{\dot{x}} &= -\frac{b}{m_P} (t - t_0) \\ \dot{x} &= \left[ v_0^{-1} + \frac{b}{m_P} (t - t_0) \right]^{-1} \quad \square. \end{aligned}$$

## Code

We can also display code blocks using `minted` environments.

```

1 minutes_to_convert = 122
2
3 hours = int(minutes_to_convert / 60)
4 minutes = minutes_to_convert % 60
5
6 convert_label = " Minutes "
7 if minutes_to_convert == 1:
8     convert_label = " Minute "
9
10 hour_label = " Hours, "
11 if hours == 1:
12     hour_label = " Hour, "
13
14 minute_label = " Minutes"
15 if minutes == 1:
16     minute_label = " Minute"
17
18 print(
19     str(minutes_to_convert)
20     + convert_label
21     + "is the same as:\n"
22     + str(hours)
23     + hour_label
24     + str(minutes)

```

```
25 | + minute_label  
26 | )
```

### Problem 3: File insertion (# pts.)

We can also insert files at will. As we can see here, the problem statement provides code for us.

```
1 α = rand(10)
2
3 @show α
```

### Solution

I like to use the filepath convention `./code/pXX.jl` for code used in problem statements, `./code/sXX.jl` for code used in solutions, and solution program output in `./code/sXX.txt`. (This convention can be extended to problems with multiple parts by appending the part number to the filename: `./code/{s03a.jl, s03b.jl, etc}`.)

```
1 using Plots
2
3 α = rand(10)
4 β = rand(10)
5
6 @show α
7 @show β
8
9 scatter(α, β)
10 savefig("./images/s03.png")
11 gui()

1 α = [0.7854313695123665, 0.541860809065995, 0.3932797715417473, 0.07292701205128227,
   ↪ 0.9830814788972285, 0.05727320963460092, 0.679204187601832, 0.33258718643784635,
   ↪ 0.6212173497203207, 0.01750154068356402]
2 β = [0.767442608013935, 0.15189153412786238, 0.0923009409886506, 0.13252612177695922,
   ↪ 0.7067053536510514, 0.15395513872178057, 0.9238424646379806, 0.8877099471621314,
   ↪ 0.3425775926068372, 0.2894157129885663]
```

As you can see, our code outputs an image. I like to save them in `./images/sXX.png` (following the filepath naming convention we used for our code files). Let's display it here to finish off our solution. To display images, we use a combination of a **figure** environment and a **center** environment.

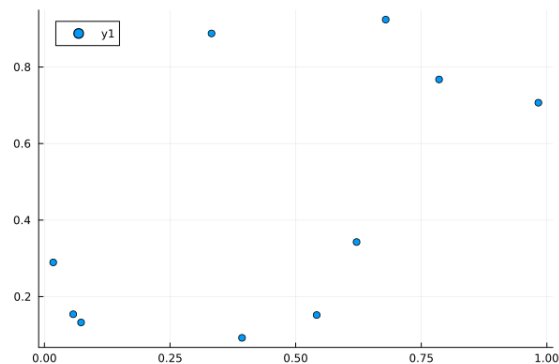


Figure 1: Data Plot

**Problem** Write number here: Write name of problem here (*# pts.*)

Write problem statement here.

**Solution**

Write solution here.