Open in Colab

Download files from the internet

Here, you can see the files that are being used to build up the assignment.

```
# Install the libraries
!pip install ipywidgets
!pip install kaleido

# Grab your libary
!wget https://raw.githubusercontent.com/pattichis/lineart/main/lineart.py

# You only need to import the functions that you are using.

from IPython.display import HTML

import lineart
from lineart import cuteGraph, CreateVideo
```

```
    [illegible]
    Requirement already satisfied: decorator in /usr/local/lib/python3.9/dist-packag
    Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.9/dist-
    Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.9/d
    Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.9/dist
    Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.9/dist-pack
    Requirement already satisfied: Send2Trash>=1.5.0 in /usr/local/lib/python3.9/dis
    Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages
    Requirement already satisfied: prometheus-client in /usr/local/lib/python3.9/dis
    Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packag
    Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.9/dist-packag
```

```
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: pycparser in /usr/local/lib/python3.9/dist-packag
Installing collected packages: jedi
Successfully installed jedi-0.18.2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 10.5 MB/s eta 0:00:00
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
--2023-03-14 20:53:57--  https://raw.githubusercontent.com/pattichis/lineart/mai
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.1
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.
HTTP request sent, awaiting response... 200 OK
Length: 20043 (20K) [text/plain]
Saving to: ‘lineart.py’

lineart.py          100%[===================>]  19.57K  --.-KB/s    in 0.001s

2023-03-14 20:53:57 (27.8 MB/s) - ‘lineart.py’ saved [20043/20043]
```

Object Oriented Programming

The following code create a `cuteGraph` **object** and stores it in `NiceGr`.

```
NiceGr = cuteGraph()
```

`NiceGr` objects can be used to prepare plots.

`NiceGr.point()` can be used to specify a point using:

```
NiceGr.point(x=4, y=5, color='blue')
```

In this example, we want to plot a point with coordinates `(4, 5)` and the point is blue. Note that the colors are specified using strings.

We can then see the points on the graph using:

```
NiceGr.plotAll()
```

Assignment

Run the code below and plot three different points.

You can try different colors. Here is a list of colors:

- color="red"
- color="green"
- color="blue"
- color="yellow"

[ ] ↳ *1 cell hidden*

Points, lines, rectangles, and text

Plot points

We can plot multiple point by simply repeating the function call with more points:

```
NiceGr.point(x=1, y=2, color='red')
NiceGr.point(x=3, y=4, color='green')
```

Assignment

Plot three additional points.

[ ] ↳ *1 cell hidden*

Plot lines

Our objects can also produce lines and rectangles.

After creating the objects, you can plot a line segment using

```
NiceGr.lineseg(x1=0, y1=4, x2=2,  y2=6, color="yellow")
```

This command defines a line from `(x1, y1)=(0, 4)` to `(x2, y2)=(2, 6)`. The line is yellow.

Assignment

Run the code below.

Try three different lines with different colors.

[ ] ↳ *1 cell hidden*

Plot rectangles

Similarly, we can plot rectangles using:

```
NiceGr.rect(x1=0, y1=0, x2=2, y2=4, color="blue")
```

In this example, we have:

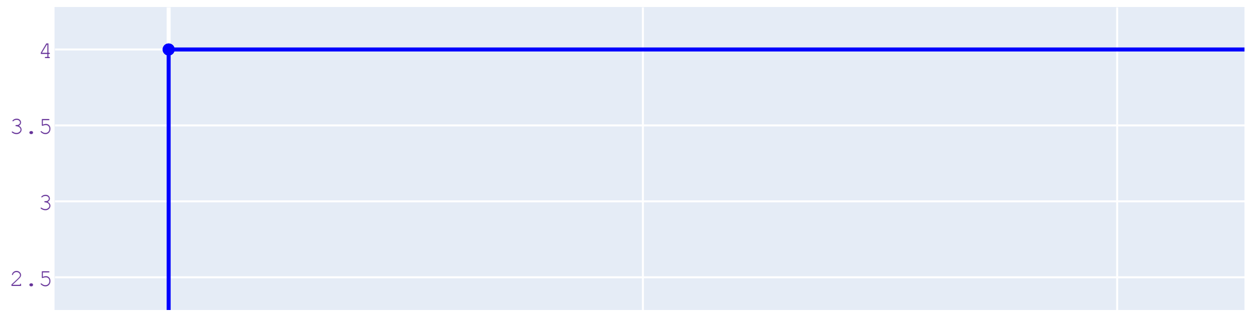
- (x1, y1) = (0, 0) is the lower-left corner of the recrtangle
- (x2, y2) = (2, 4) is the upper-right corner of the rectangle
- the rectangle color is blue.

Assignment

Run the code and plot two more rectangles.

```
# Create the object
NiceGr = cuteGraph()

# Add a rectangle to the graph
NiceGr.rect(x1=0, y1=0, x2=2, y2=4, color="blue")

# Add code for more rectangles


# Show the graph
NiceGr.plotAll()
```

Plots



Adding text

We can add text to the graph using strings.

Our strings can span multiple lines as given below:

```
text = """Student Names<br>
Class Name<br>
School Name<br>
Semester Year"""
```

Here, note that `<br>` defines a new line. To place the text centered at the origin, we use:

```
NiceGr.addText(x=0, y=0, text=text, color="black")
```

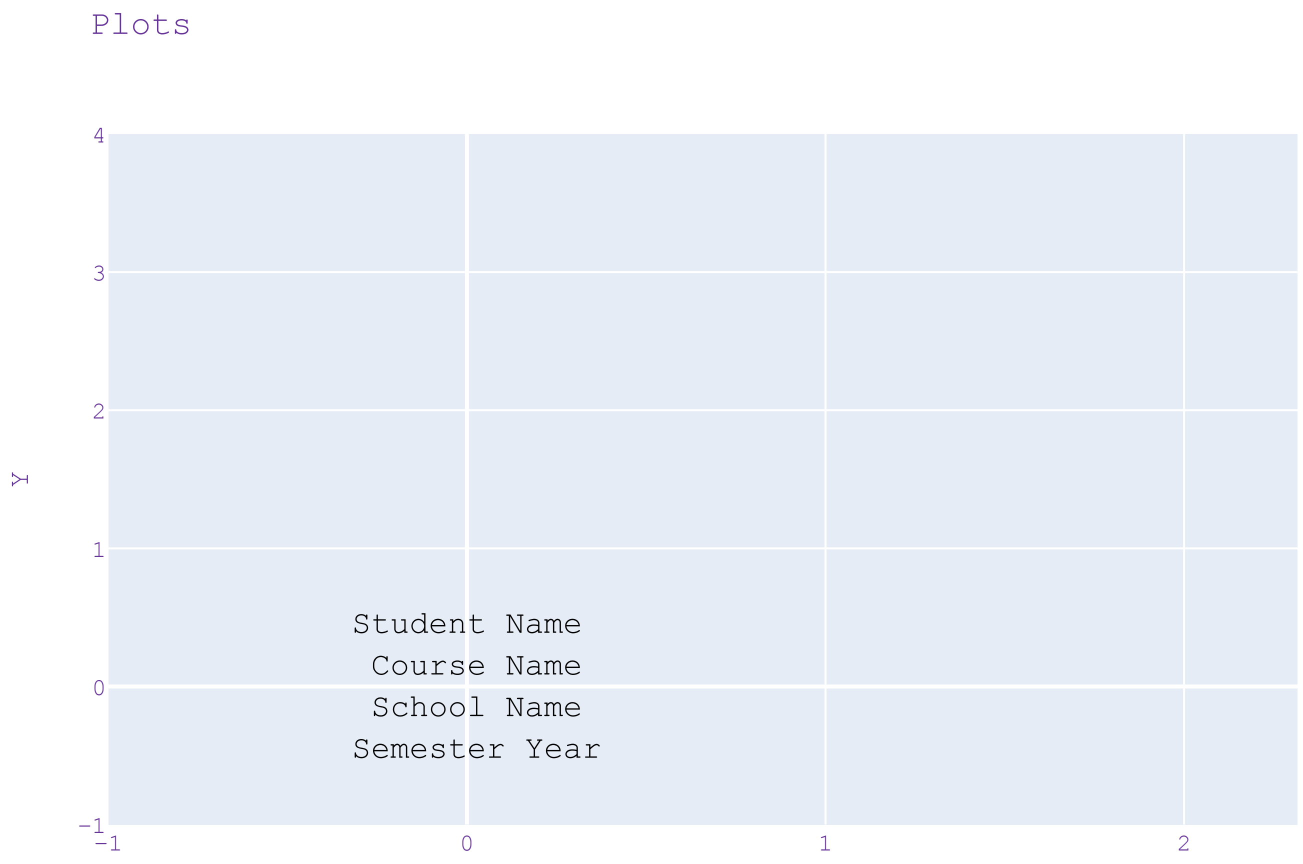
Assignment

Modify the code to enter your name.

If you cannot see the text, then click on the zoom controls on the upper-right hand corner of the image.

```
# Create the object
NiceGr = cuteGraph()

# Add text:
text = """Student Name<br>
Course Name<br>
School Name<br>
Semester Year"""

NiceGr.addText(x=0, y=0, text=text, color="black")
```

```
# Show the graph
NiceGr.plotAll()
```



Advanced drawing examples

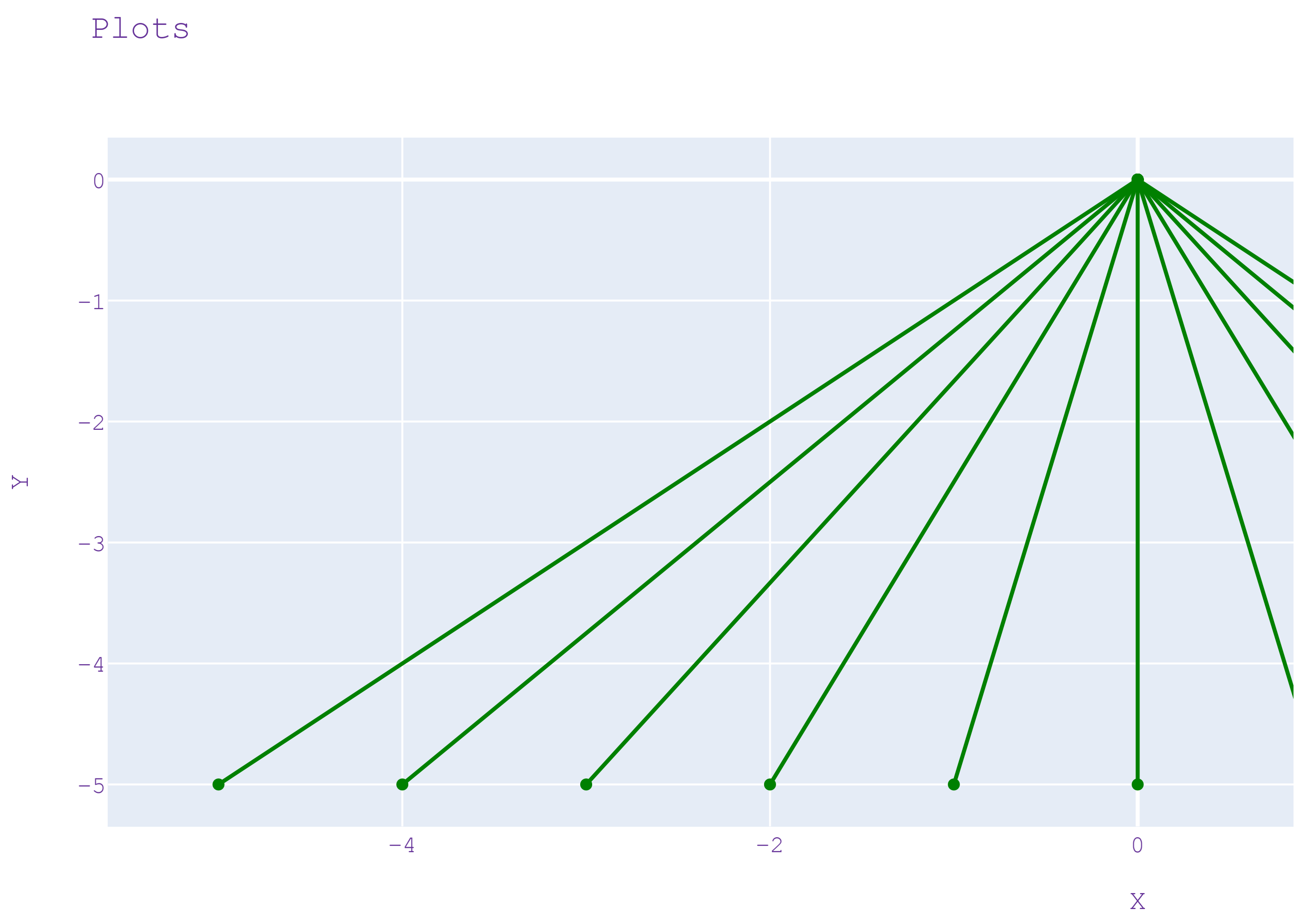
Lines from a given point

Suppose that the ending point is at the origin.

We can just move the x1 using a for loop to get all the lines to connect to the origin.

```
# Create the object
NiceGr = cuteGraph()

# change x1
for x1 in range(-5, +5+1):
  NiceGr.lineseg(x1=x1, y1=-5, x2=0, y2=0, color="green")
```

```
# Show the graph
NiceGr.plotAll()
```



Parallel lines

To generate parallel lines, we need to plot lines that have the same slope.

A simple way to do is to add the same `dx` and `dy` to every point.

The following example generates parallel line by moving x1:

```
# Parallel lines by varying x1
y1 = 0
dx = 1
dy = 1
for x1 in range(-5, +5+1):
  x2 = x1 + dx
  y2 = y1 + dy
  NiceGr.lineseg(x1=x1, y1=y1, x2=x2, y2=y2, color="green")
```

Assignment

Modify the code to draw parallel lines by varying x1.

```
# Create the object
NiceGr = cuteGraph()


# Parallel lines by varying x1
y1 = -9
dx = -8
dy =31
for x1 in range(-5, +5+1):
  x2 = x1 + dx
  y2 = y1 + dy
  NiceGr.lineseg(x1=x1, y1=y1, x2=x2, y2=y2, color="green")


# Show the graph
NiceGr.plotAll()
```

```
Plots
```

Generating Videos

Draw a house using line art

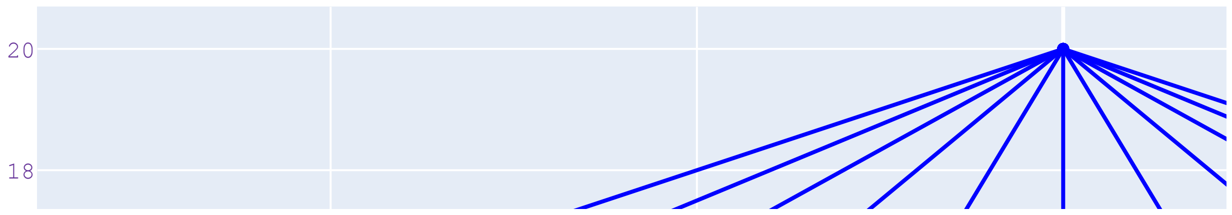
To generate a house, we put together the previous examples:

1. Build the roof using lines intersecting at a point.
2. Use rectangles to build the rest

Assignment

```
# Create the object
NiceGr = cuteGraph()

for x1 in range(-5, +5+1):
  NiceGr.lineseg(x1=x1, y1=15, x2=0, y2=20, color="Blue")

# Build the house
NiceGr.rect(x1=-5, y1=10, x2=+5, y2=15, color="purple")

# Show the graph
NiceGr.plotAll()
```

Plots



Generate frames using line art

To make videos, we first need to setup a grid for all remaining video frames.

To do this, right after creating the video, we use the `prepVideo()` function to make all video frames be the same.

Here is an example use:

```
NiceGr = cuteGraph()
NiceGr.prepVideo(minX=-5, minY=-5, maxX=10, maxY=10, magFactor=3)
```

Here, we tell the grid that we have that our house will be plotted approximately within the bounds of `(minX, minY)` to `(maxX, maxY)`. To make extra space for the names and the frontyard of the house, we make this frame three times larger using `magFactor=3`.

After we are done plotting each video frame, we save a picture using:

```
NiceGr.plotAll()
NiceGr.saveImage("frame2.png")
```

We will collect these frames to make a video!

Assignment

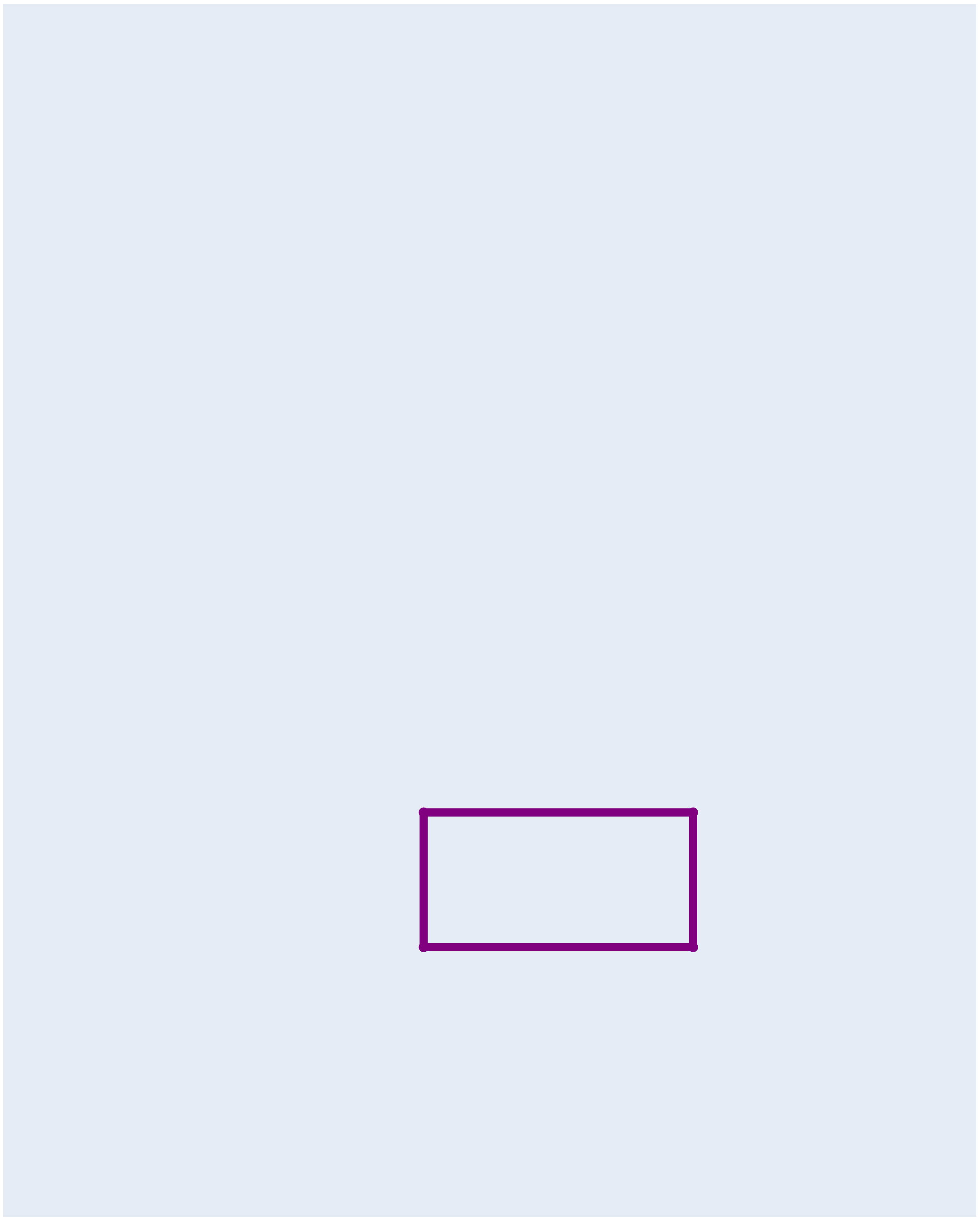
Feel free to experiment with different designs.

What can you draw?

```
# Create the object
NiceGr = cuteGraph()
NiceGr.prepVideo(minX=-5, minY=-5, maxX=10, maxY=10, magFactor=3)

# Teach line above. Keep as one lesson. Fix the updates.
```

```
NiceGr.rect(x1=-5, y1=-10, x2=+5, y2=-5, color="Purple")

# Show the graph
NiceGr.plotAll()
NiceGr.saveImage("frame1.png")

# The door
NiceGr.setwidths(linewidth=3, pointwidth=1)
NiceGr.rect(x1=-1, y1=-10, x2=1, y2=-7, color="pink")

# The window
NiceGr.setwidths(linewidth=3, pointwidth=1)
NiceGr.rect(x1=+2, y1=-7, x2=2+1, y2=-7+1, color="silver")
NiceGr.rect(x1=+-3, y1=-7, x2=-3+1, y2=-7+1, color="silver")
NiceGr.plotAll()
NiceGr.saveImage("frame2.png")

# The roof
for x1 in range(-5, +5+1):
  NiceGr.lineseg(x1=x1, y1=-5, x2=0, y2=0, color="light blue")

# Show the graph
NiceGr.plotAll()
NiceGr.saveImage("frame3.png")

# Add a front:
y1 = -15
dx = 5
dy = 4
for x1 in range(-15, +10):
  x2 = x1 + dx
  y2 = y1 + dy
  NiceGr.lineseg(x1=x1, y1=y1, x2=x2, y2=y2, color="green")



y1 = -15
dx = -5
dy = 4
for x1 in range(-10, +15):
  x2 = x1 + dx
  y2 = y1 + dy
  NiceGr.lineseg(x1=x1, y1=y1, x2=x2, y2=y2, color="green")
# Show the graph
NiceGr.plotAll()
NiceGr.saveImage("frame4.png")

# Add text:
text = """BAK<br>
Math Strategies<br>
ECA<br>
```

```
2023"""

NiceGr.addText(x=15, y=0, text=text, color="black")

# Show the graph
NiceGr.plotAll()
NiceGr.saveImage("frame5.png")
```



Create a video using generated video frames

We can create a list of all of the video frame images using:

```
file_list = ['frame1.png', 'frame2.png', 'frame3.png', 'frame4.png', 'frame5.png']
```

We need to define a name for our video:

```
video_name = "video2.mp4"
```

To display the video, we simply use:

```
HTML(CreateVideo(video_name, file_list, fps=0.5))
```

Here, note that `fps` refers to the number of frames per second that we are displaying. At 0.5, it means that there is a delay of half a second between video frames.

Assignment

Try the following:

1. Slow down the video display by changing fps.
2. Speed up the video display.
3. Change the order of the video frames.

```
file_list = ['frame1.png', 'frame2.png', 'frame3.png', 'frame4.png', 'frame5.png']

video_name = "video2.mp4"
HTML(CreateVideo(video_name, file_list, fps=2))
```

```
Compressed video2.mp4 into temp_video.mp4
```