An Introduction to TikZ

Presentation · November 2018

CITATIONS
0

READS
953

1 author:

Sudev Naduvath
Christ University, Bangalore
187 PUBLICATIONS **778** CITATIONS

SEE PROFILE

An Introduction to TikZ

Dr. N.K. Sudev

Department of Mathematics,
CHRIST (Deemed to be University),
Bangalore-560029, India.

05-07, November 2018

1 Setting Up the Environment

1.1 Necessary Codes in the Preamble

In the beginning, follow the usual steps as given below:

1. Define document class - *article* or *standalone* (the class for individual examples). (Use `\documentclass{article}` or `\documentclass{standalone}`).
2. Introduce the packages such as *graphicx*, *caption* and *subcaption*. This can be done by writing the command `\usepackage{graphicx,caption,subcaption}` in the preamble.
3. Include the Tikz package to the document by writing `\usepackage{tikz}` in the preamble.
4. Now we can include different tikz libraries according to our requirements.

Some of the very useful Tikz lbraries are:

1. *arrows* - to create and customise arrows.
2. *automata* - to draw finite state automata and Turing Machines.
3. *backgrounds* - to define backgrounds of pictures.
4. *calc* - to make complex coordinate caculations.
5. *calendar* - to display calendars.
6. *er* (entry relation) - to define nodes and edges between nodes with their attributes.
7. *mindmap* - to draw the main focus point, in the middle, with sub-points branching off.
8. *folding* - useful for producing real calendars.

9. *patterns*- defines patterns for filling areas.
10. *plothandlers, plotmarks* - to draw nodes and curves in different styles.
11. *shapes* - uses to define shapes (other than rectangle, circle and co-ordinate).
12. *snake* - to create curved lines.
13. *topaths* - to draw paths between nodes.
14. *trees*- draws tree diagrams.
15. *plothandlers, plotmarks* - to draw nodes and curves
16. *positioning* - to adjust positions of nodes and their coordinates and to create gridlines as in the graph sheets.
17. *decorations.markings* - for advanced customisations of figures and shapes

we can include the libraries using the command

```
\usetikzlibrary{
   intersections, arrows.meta,
   automata,er,calc,
   backgrounds,
   mindmap,folding,
   patterns,
   decorations.markings,
   fit,snakes,
   shapes,matrix,
   positioning,
   shapes.geometric,
   arrows,through
  }
```

1.2 Figure Environments

To draw a figure, we use the following tikz environment:

```
\begin{tikzpicture}
     Write your Tikz Code here...!!!
\end{tikzpicture}
```

To draw a figure with caption, we use the following tikz environment:

```
\begin{figure}[placement]
\centering
\begin{tikzpicture}
Write your Tikz Code here...!!!
\end{tikzpicture}
\end{figure}
```

To draw a figure with caption, we use the following tikz environment:

```
\begin{figure}[placement]
\centering
\begin{subfigure}[b]{0.45\textwidth}
\begin{tikzpicture}
Write your Tikz Code for the first figure here...!!!
\end{tikzpicture}
\caption{Subcaption for the first subfigure}
\end{subfigure}
\quad
\begin{subfigure}[b]{0.45\textwidth}
\begin{tikzpicture}
Write your Tikz Code for the second figure here...!!!
\end{tikzpicture}
\caption{Subcaption for the second subfigure}
\end{subfigure}
\caption{Your figure caption here}
\end{figure}
```

2 Drawing and Decorating Lines

1. To draw a line between, we can use the code:

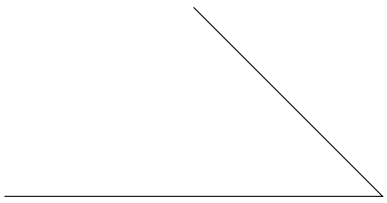
   ```
   \begin{tikzpicture}
   \draw (0,0) --(5,0);
   \end{tikzpicture}
   ```

   This will give us the output

2. The code

   ```
   \begin{tikzpicture}
   \draw (0,0) --(5,0) -- (2.5,2.5);
   \end{tikzpicture}
   ```

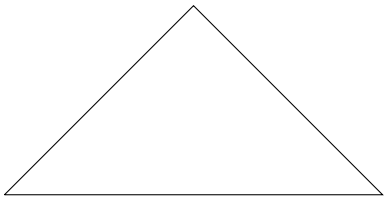
   gives us following output:

3. The code

   ```
   \begin{tikzpicture}
   \draw (0,0) --(5,0) -- (2.5,2.5) -- (0,0);
   \end{tikzpicture}
   ```

gives us following output:

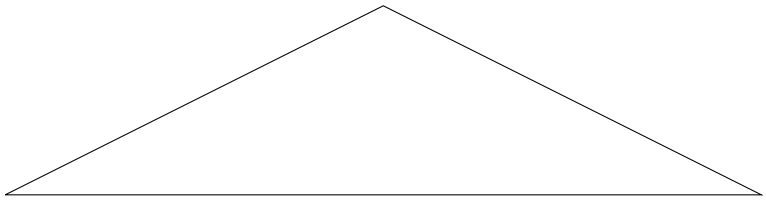


4. Scaling of the figures can be done as follows:

   (a) The code

   ```
   \begin{tikzpicture}[xscale=2]
   \draw (0,0) --(5,0) -- (2.5,2.5) -- (0,0);
   \end{tikzpicture}
   ```

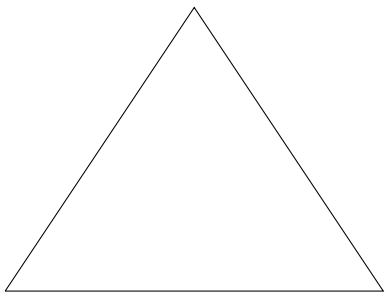
   gives us following output:



   (b) The code

   ```
   \begin{tikzpicture}[yscale=1.5]
   \draw (0,0) --(5,0) -- (2.5,2.5) -- (0,0);
   \end{tikzpicture}
   ```

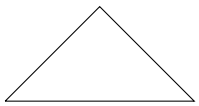
   gives us following output:



   (c) The code

   ```
   \begin{tikzpicture}[scale=0.5]
   \draw (0,0) --(5,0) -- (2.5,2.5) -- (0,0);
   \end{tikzpicture}
   ```

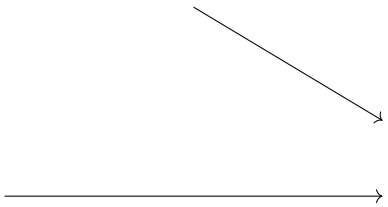
   gives us following output:



5. The code

```
\begin{tikzpicture}[yscale=1.5]
\draw [->] (0,0) --(5,0);
\draw [<-] (5,1) -- (2.5,2.5);
\end{tikzpicture}
```

gives us following output:



6. The code

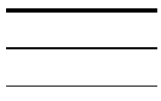
```
\begin{tikzpicture}
\draw [<->] (0,2) -- (0,0) -- (3,0);
\end{tikzpicture}
```

will create the figure

7. The thickness of the lines can be adjusted using the code:

```
\begin{tikzpicture}
\draw [ultra thick] (0,1) -- (2,1);
\draw [thick] (0,0.5) -- (2,0.5);
\draw [thin] (0,0) -- (2,0);
\end{tikzpicture}
```

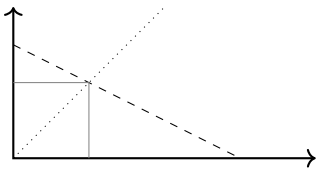
and the output will be as follows:



8. The option "help lines" is made specially to be fine gray lines for showing special points as shown below:

```
\begin{tikzpicture}
\draw [<->,thick] (0,2) -- (0,0) -- (4,0);
\draw [dashed]  (0,1.5) -- (3,0);
\draw [dotted] (0,0) -- (2,2);
\draw [help lines] (1,0) -- (1,1) -- (0,1);
\end{tikzpicture}
```

which yields the following diagram:



9. We can also decide line width using custom commands as follows

The code

```
\begin{tikzpicture}
\draw [line width=12] (0,0) -- (2,0);
\draw [line width=0.1cm] (0,1) -- (2,1);
\end{tikzpicture}
```

yields the following output:

10. We can also color the lines by using the code:

```
\begin{tikzpicture}
\draw [yellow] (0,2.5) -- (4,2.5);
\draw [cyan] (0, 2) -- (4,2);
\draw [purple] (0,1.5) -- (4,1.5);
\draw [green] (0,1) -- (4,1);
\draw [red] (0, 0.5) -- (4,0.5);
\draw [blue] (0,0) -- (4,0);
\end{tikzpicture}
```

and the output will be as follows:

Note: We have the following default colors: red , green , blue , cyan , magenta , yellow , black , gray , darkgray , lightgray , brown , lime , olive , orange , pink , purple , teal , violet and white .

In addition to these one can define new colors according to their preferences using the command `\definecolor{myblue}{rgb}{24,40,120}`

Here, the colour strips are created using the command

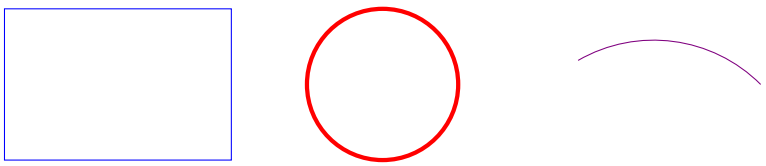
```
\begin{tikzpicture}
\draw [red, line width=6](0,0) -- (.5,0);
\end{tikzpicture}
```

3 Drawing Curves

1. The code

```
\begin{tikzpicture}
\draw [blue] (0,0) rectangle (3,2);
\draw [red, ultra thick] (6,1) circle [radius=1];;
\draw [violet] (10,1) arc [radius=2,start angle=45,end angle= 120];
\end{tikzpicture}
```

gives the output



2. We can make paths take smoother turns:

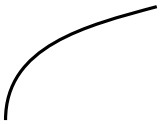
```
\begin{tikzpicture}
\draw [<->, rounded corners, thick, purple] (0,2) -- (0,0) -- (3,0);
\end{tikzpicture}
```

which gives you

3. We can do curves without plotting all the points. See an easy one:

```
\begin{tikzpicture}
\draw[very thick] (0,0) to [out=90,in=195] (2,1.5);
\end{tikzpicture}
```

which yields



4. The code

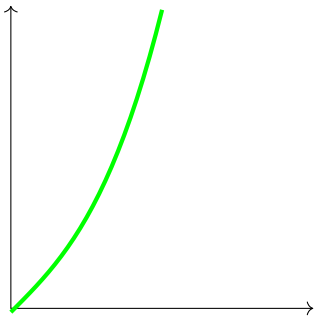
```
\begin{tikzpicture}
\draw [<->,thick, cyan] (0,0) to [out=90,in=180] (1,1)
to [out=0,in=180] (2,0) to [out=0,in=180] (3,1) to [out=0,in=-180] (4,1) ;
\end{tikzpicture}
```

gives the output

5. Tikz also has a math engine which enables you to plot functions:

```
\begin{tikzpicture}[xscale=2,yscale=2]
\draw [<->] (0,2) -- (0,0) -- (2,0);
\draw[green, ultra thick, domain=0:1] plot (\x, {\x*\x*\x+\x-0.025});
\end{tikzpicture}
```

gives you



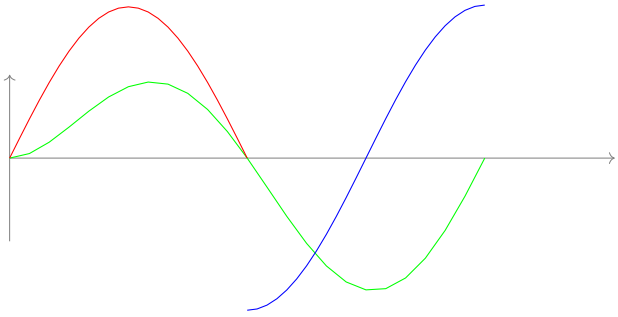
Many mathematical functions are available in default. Some of them are:

```
factorial(\x),
sqrt(\x),
pow(\x,y),
exp(\x),
ln(\x),
log10(\x),
log2(\x),
abs(\x),
mod(\x,y) - (x modulo y),
round(\x) - (rounds x to the nearest integer),
floor(\x),
ceil(\x),
sin(\x) - (sinx in degrees),
sin(\x r) - (sinx in radians),
cos(\x),
cos(\x,r),
tan(\x),
tan(\x,r),
min(\x,y,),
max(\x,y) etc.
```

6. The code

```
\begin{tikzpicture}
\draw [help lines, ->] (0,0) -- (8,0);
\draw [help lines, ->] (0,-1.1) -- (0,1.1);
\draw [green,domain=0:2*pi] plot (\x, {2*(sin(\x r)* ln(\x+1))/2});
\draw [red,domain=0:pi] plot (\x, {2*sin(\x r)});
\draw [blue, domain=pi:2*pi] plot (\x, {2*cos(\x r)*exp(\x/exp(2*pi))});
\end{tikzpicture}
```

will give us the following diagram.



4 Filling up Shapes

We can fill figures and shapes with colours. See the following illustrations.

The code

```
\begin{tikzpicture}
\draw [fill=red,ultra thick] (0,0) rectangle (1,1);
\draw [fill=red,ultra thick,red] (2,0) rectangle (3,1);
\draw [blue, fill=blue] (4,0) -- (5,1) -- (4.75,0.15) -- (4,0);
\draw [fill] (7,0.5) circle [radius=0.1];
\draw [fill=orange] (9,0) rectangle (11,1);
\draw [fill=white] (9.25,0.25) rectangle (10,1.5);
\end{tikzpicture}
```

gives the result:

We can define new colours according to our preference as mentioned earlier.

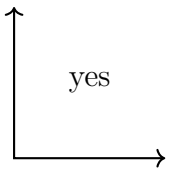
5 Putting Labels

In almost all figures we have to assign labels to the parts of the figure (marking angles, lengths etc.). (Here we have to use the library *positioning* for better placement of labels). To write labels at certain positions, we need to create nodes using `\node` command.

For example, the code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node at (1,1) {yes};
\end{tikzpicture}
```

gives us the output

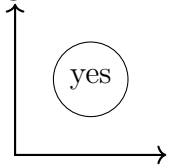


Note that the node is invisible until we define a shape for the node. We shall define shapes for nodes using `\node[shape]` command. For this we need to define the characters of the curves in the shape concerned using the `\tikzstyle{text} = {definition}` environment. We may have to include the libraries *shapes,snakes,matrix* etc. according to the requirements.

For example, the code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,circle] at (1,1) {yes};
\end{tikzpicture}
```

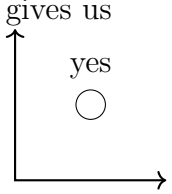
gives us



The code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,circle] at (1,1) [label={yes}] {};
\end{tikzpicture}
```

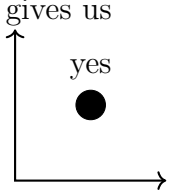
gives us



The code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,circle,fill] at (1,1) {yes};
\end{tikzpicture}
```

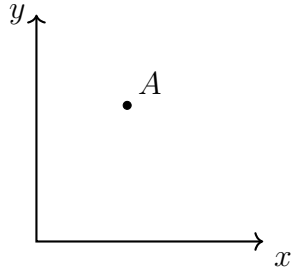
gives us



The code

```
\begin{tikzpicture}[xscale=3, yscale=3]
\draw [thick, <->] (0,1) -- (0,0) -- (1,0);
\node [below right] at (1,0) {$x$};
\node [left] at (0,1) {$y$};
\draw[fill] (.4,.6) circle [radius=.5pt];
\node[above right] at (.4,.6) {$A$};
\end{tikzpicture}
```

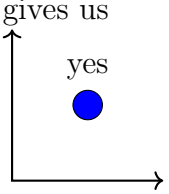
gives the output



The code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,circle,fill=blue] at (1,1) {yes};
\end{tikzpicture}
```

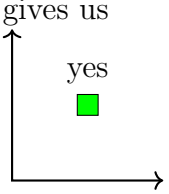
gives us



The code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,rectangle,fill=green] at (1,1) {yes};
\end{tikzpicture}
```

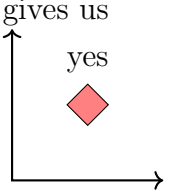
gives us



The code

```
\begin{tikzpicture}
\draw [thick, <->] (0,2) -- (0,0) -- (2,0);
\node[draw,diamond,fill=red!50] at (1,1) {yes};
\end{tikzpicture}
```

gives us



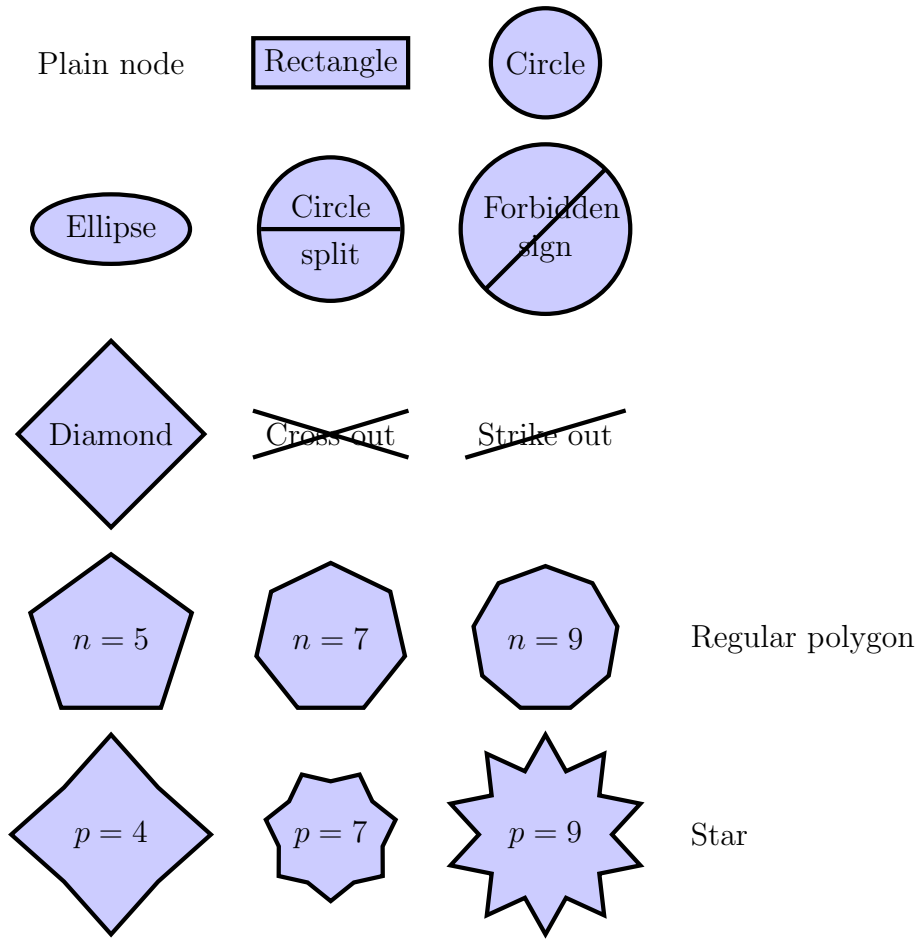
5.1 Node Shapes & Matrix Environment

The code

```
\begin{tikzpicture}[scale=2]
\tikzstyle{ann} = [draw=none,fill=none,right]
\matrix[nodes={draw, ultra thick, fill=blue!20},
row sep=0.3cm,column sep=0.5cm] {
\node[draw=none,fill=none] {Plain node}; &
\node[rectangle] {Rectangle}; &
\node[circle] {Circle};\\
```

```
\node[ellipse] {Ellipse};&
\node[circle split] {Circle \nodepart{lower} split};&
\node[forbidden sign,text width=4em, text centered]
{Forbidden sign};\\
\node[diamond] {Diamond};&
\node[cross out] {Cross out};&
\node[strike out] {Strike out};\\
\node[regular polygon,regular polygon sides=5] {$n=5$};&
\node[regular polygon,regular polygon sides=7] {$n=7$};&
\node[regular polygon,regular polygon sides=9] {$n=9$};&
\node[ann]{Regular polygon};\\
\node[star,star points=4] {$p=4$};&
\node[star,star points=7,star point ratio=0.8] {$p=7$};&
\node[star,star points=10] {$p=9$};&
\node[ann]{Star};\\
};
\end{tikzpicture}
```

gives us the result as follows:
the code

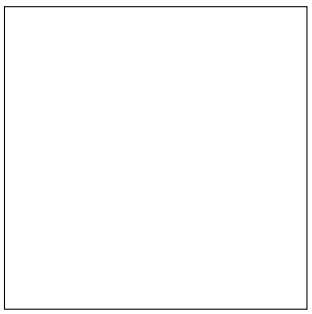


6 LaTeX Graphics using TikZ

1. The code

   ```
   \begin{tikzpicture}
   \draw (0,0) -- (4,0) -- (4,4) -- (0,4) -- cycle;
   \end{tikzpicture}
   ```

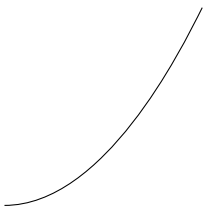
   gives us

2. The code

   ```
   \begin{tikzpicture}
   \draw (0,0) parabola (3,3);
   \end{tikzpicture}
   ```

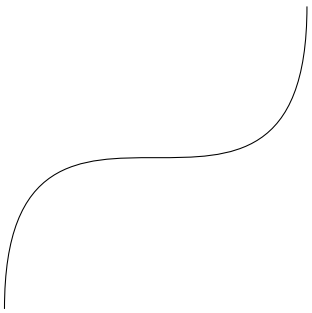
   gives us

3. To draw a curved line we use *control points*. We begin with our starting co-ordinate, then use two dots followed by the keyword *controls* and then the co-ordinates of our control points separated by an *and*.

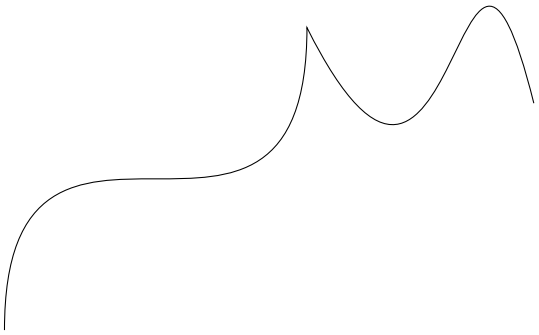
   For example, consider the command `\draw (0,0) .. controls (0,4) and (4,0) .. (4,4);` and the output will be as follows:

   ```
   \begin{tikzpicture}
   \draw (0,0) .. controls (0,4) and (4,0) .. (4,4);
   \end{tikzpicture}
   ```



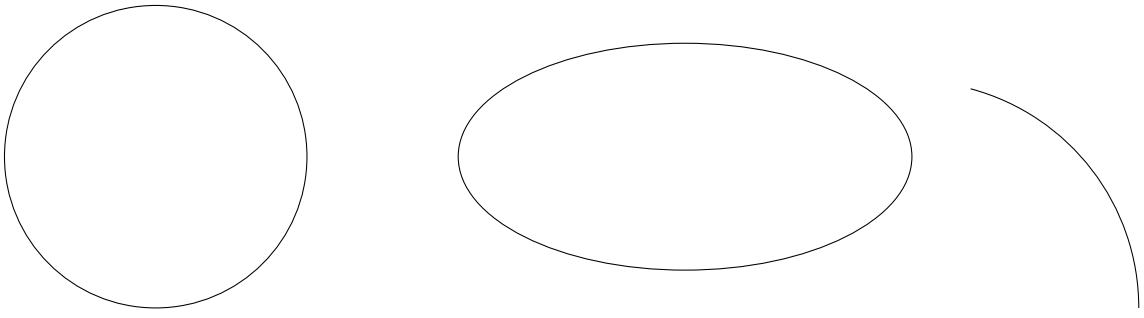
4. The code

```
\begin{tikzpicture}
\draw (0,0) .. controls (0,4) and (4,0) .. (4,4) .. controls (5,5)
and (6,2) .. (6,6);
\end{tikzpicture}
```

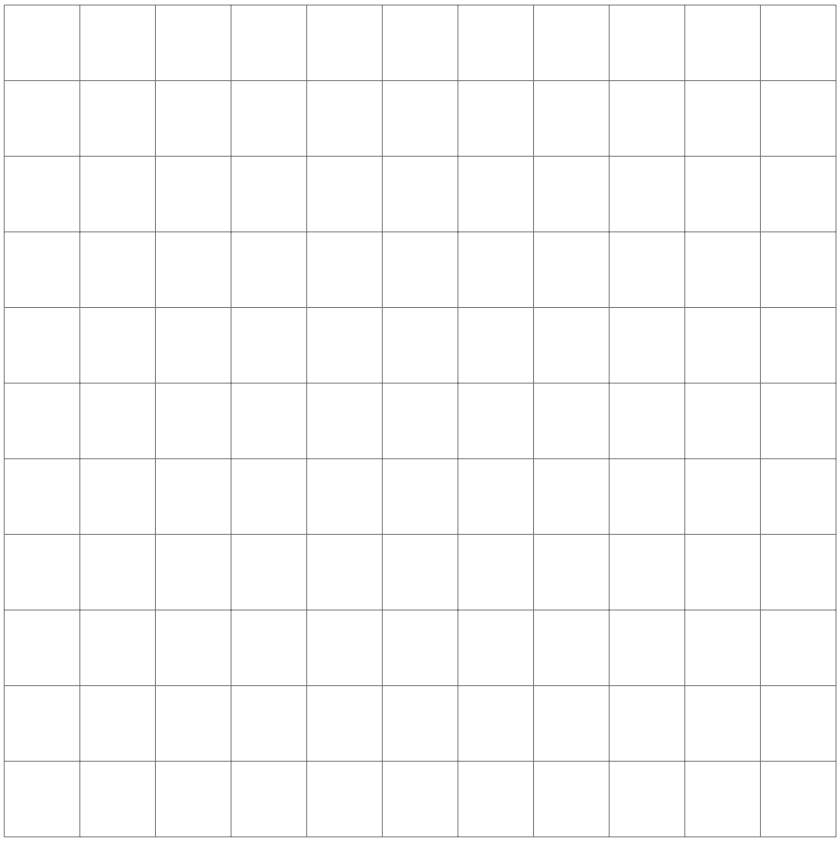


5. By the code

```
\begin{tikzpicture}
\draw (2,2) circle (2cm);
\draw (9,2) ellipse (2cm and 1.5cm);
\draw (15,1.5) arc (0:75:3cm);
\end{tikzpicture}
```



6. The code `\draw[step=1cm,gray,very thin] (-3,-3) grid (8,8);` wil give you the grid lines as follows:



7. Consider the code

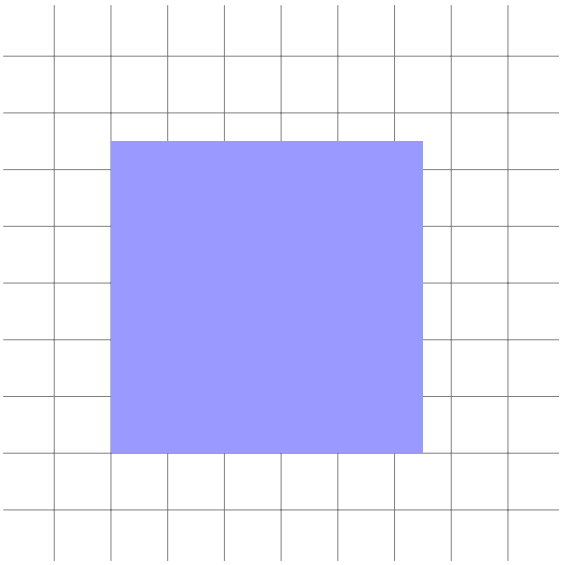
```
\begin{tikzpicture}
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\end{tikzpicture}
```

   the output will be:

8. We can fill areas in the above figure by adding the commands like `\fill[blue!40!white] (0,0) rectangle (4,4);` (see the out pu below). That is, the code

```
\begin{tikzpicture}
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\fill[blue!40!white] (-1,-1) rectangle (4.5,4.5);
\end{tikzpicture}
```

will give the output as follows:

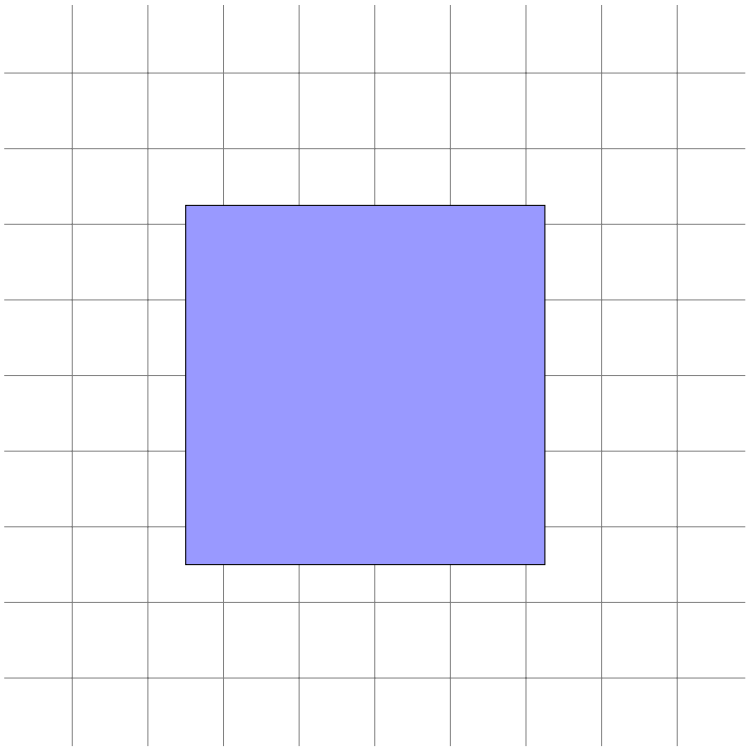


9. We can add boarder also using the `\filldraw` command. For example, the command

   That is, the code

```
\begin{tikzpicture}
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\filldraw[fill=blue!40!white, draw=black] (0,0) rectangle (4,4);
\end{tikzpicture}
```

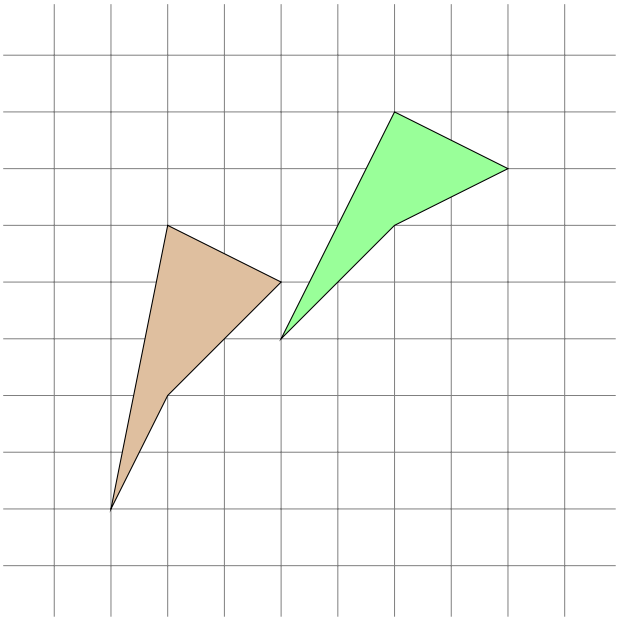
   will yield the figure



10. We can draw different shapes exactly as in a graph sheet and assign colours to these shapes. For example, the code

```
\begin{tikzpicture}
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\draw [fill=brown!50] (-1,-1) -- (0,1) -- (2,3) -- (0,4) -- cycle;
\draw [fill=green!40] (2,2) -- (4,4) -- (6,5) --(4,6) -- cycle;
\end{tikzpicture}
```

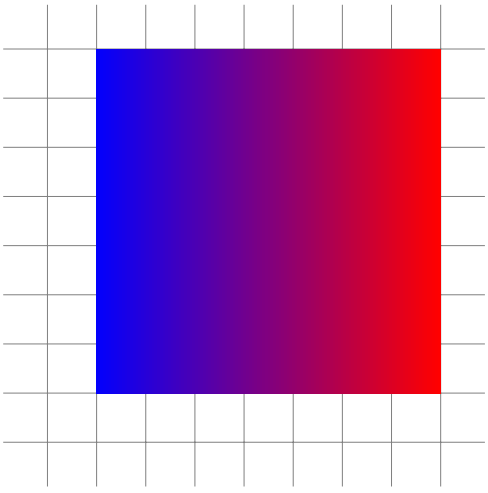
   will give us the following output:



11. Instead of assigning a single colour, we can assign some colour gradients also to the shapes using the `\shade` command. As an ilustraton, consider the following command:

```
\begin{tikzpicture}[scale=0.75]
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\shade[left color=blue,right color=red] (-1,-1) rectangle (6,6);
\end{tikzpicture}
```

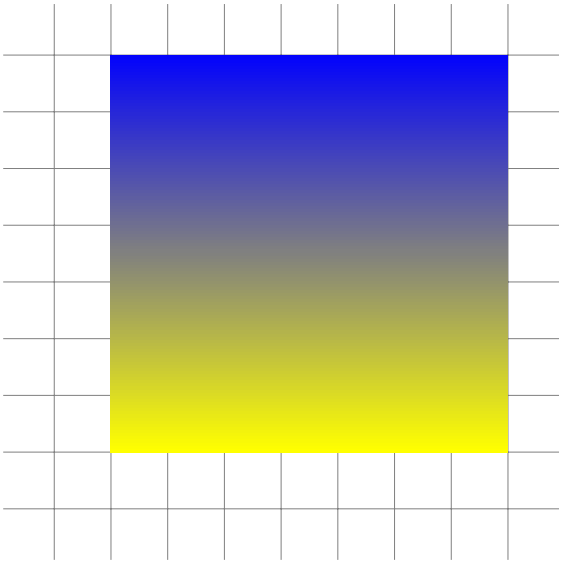
Here, we get the output



12. Consider the following command:

```
\begin{tikzpicture}[scale=0.75]
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\shade[top color=blue,bottom color=yellow] (-1,-1) rectangle (6,6);
\end{tikzpicture}
```

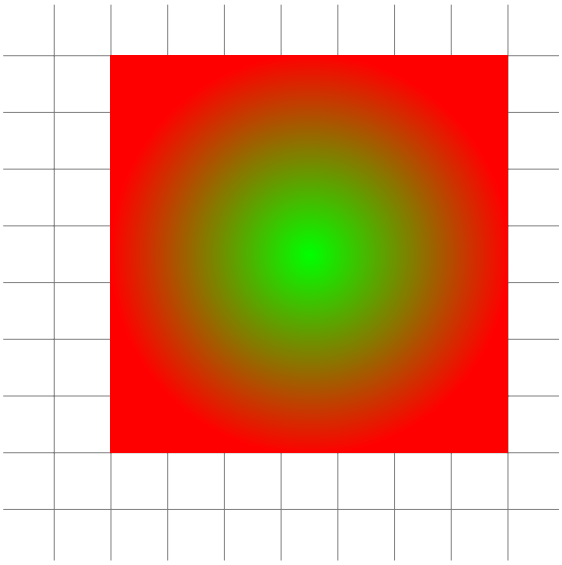
Here, we get the output



13. Consider the following command:

```
\begin{tikzpicture}[scale=0.75]
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\shade[inner color=green,outer color=red] (-1,-1) rectangle (6,6);
\end{tikzpicture}
```

Here, we get the output

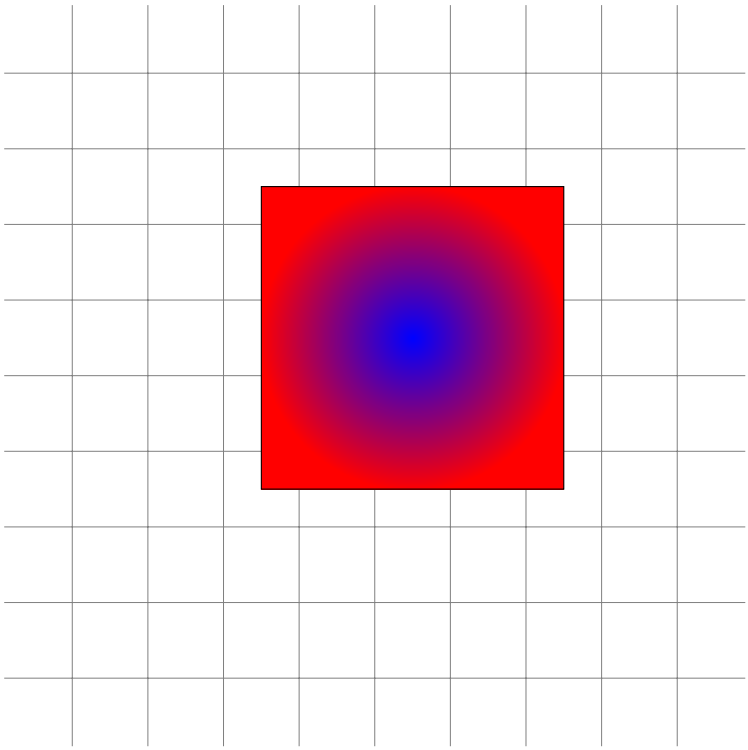


14. As explaied above, we can insert the boarders using `\shadedraw` command. See the example:

```
\begin{tikzpicture}
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\shadedraw[inner color=blue,outer color=red, draw=black] (0,0)
```

```
rectangle (4,4);
\end{tikzpicture}
```

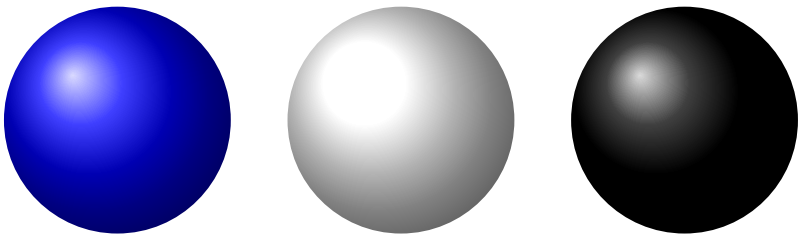
Here, we get the output



15. The `shade` can replace `\filldraw` and the output will be as follows.

```
\begin{tikzpicture}
\shade[shading=ball, ball color=blue]  (0,0) circle (2);
\shade[shading=ball, ball color=white] (5,0) circle (2);
\shade[shading=ball, ball color=black] (10,0) circle (2);
\end{tikzpicture}
```

Here, we get the output

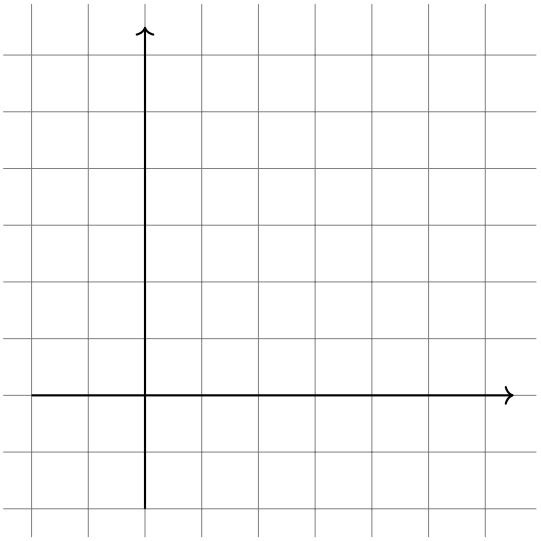


7 Coordinate System

16. The coordinate axes can be drawn as follows:

```
\begin{tikzpicture}%[scale=0.75]
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\draw[thick,->] (-2,0) -- (6.5,0);
\draw[thick,->] (0,-2) -- (0,6.5);
\end{tikzpicture}
```

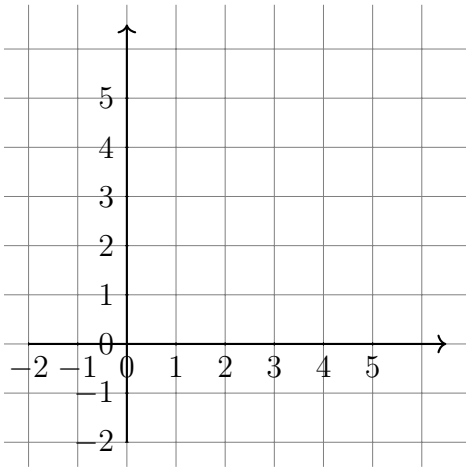
yields the figure as



17. Numbers/units can be marked o the coordinate axes as foows:

```
\begin{tikzpicture}%[scale=0.75]
\draw[thick,->] (-2,0) -- (6.5,0);
\draw[thick,->] (0,-2) -- (0,6.5);
\draw[step=1cm,gray,very thin] (-2.9,-2.9) grid (6.9,6.9);
\foreach \x in {-2,-1,0,1,2,3,4,5}
\draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north] {$\x$};
\foreach \y in {-2,-1,0,1,2,3,4,5}
\draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east] {$\y$};
\end{tikzpicture}
```

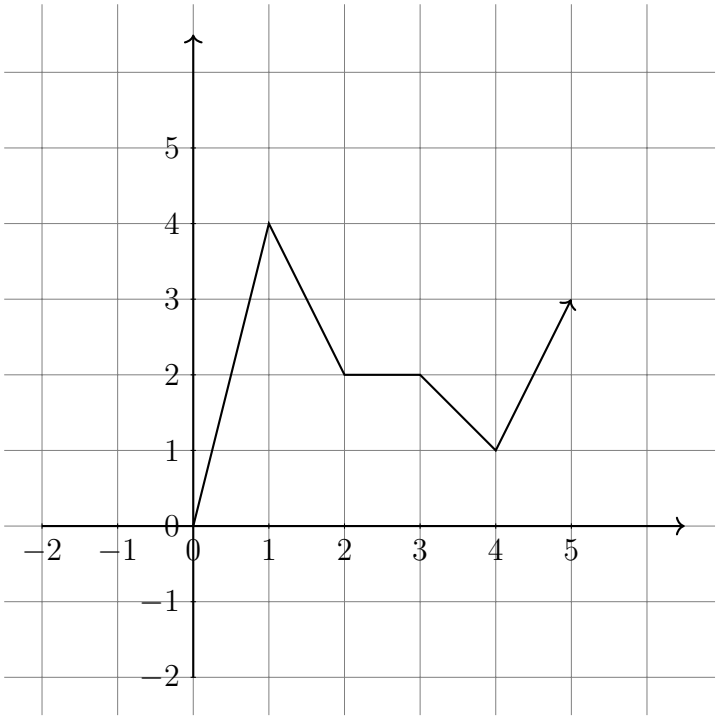
yields the figure as



18. Graphs can also be plotted in the above grid environment as exlained below:

```
\draw[step=1cm,gray,very thin] (-2.5,-2.5) grid (6.9,6.9);
\draw[thick,->] (-2,0) -- (6.5,0);
\draw[thick,->] (0,-2) -- (0,6.5);
\foreach \x in {-2,-1,0,1,2,3,4,5}
\draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north] {$\x$};
\foreach \y in {-2,-1,0,1,2,3,4,5}
\draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east] {$\y$};
\draw [thick,->] (0,0) -- (1,4) -- (2,2)-- (3,2) -- (4,1) -- (5,3);
\end{tikzpicture}
```

yields the figure as



8 Flow Charts

To create the shapes, we use in flowcharts, the libraries `shapes.geometric` and `arrows`. Just add these libaries in the preamble.
To begin with, we need to define the required shapes using
`\tikzstyle{shape-name}=[attribute1,atribute2,...]`:

1. The start & stop buttonscan be defined as follows:

```
\tikzstyle {ss} = [rectangle, rounded corners, minimum width=3cm,
minimum height=1cm,text centered, draw=black, fill=red!30]
```

Now, the command `\node (start) [ss] {Start};` givesus the output

2. The input-output boxes can be defined as follows:

```
\tikzstyle{io}=[trapezium, trapezium left angle=70, trapezium right
angle=110, minimum width=3cm, minimum height=1cm, text centered,
draw=black, fill=blue!30]
```

3. Now the process boxes and decision boxes can be created as follows:

```
\tikzstyle{proc}=[rectangle, minimum width=3cm, minimum height=1cm,
text centered, draw=black, fill=orange!30]
```

and

```
\tikzstyle{dec} = [diamond, minimum width=3cm, minimum height=1cm,
text centered, draw=black, fill=green!30]
```

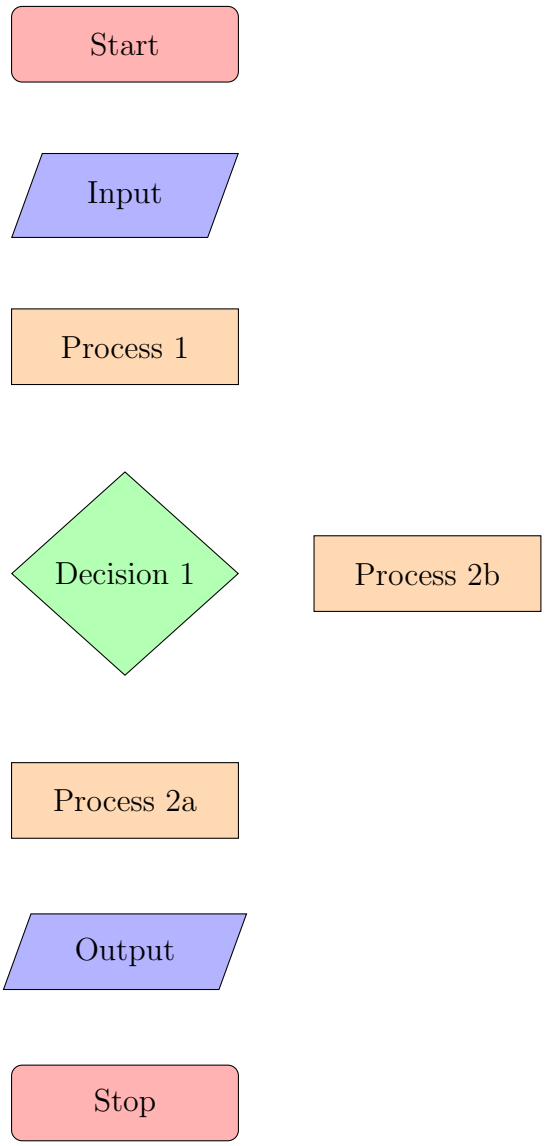
4. Now, it remains to define arrows which can be done as foows:

```
\tikzstyle{arrow} = [thick,->,>=stealth]
```

5. Now, the commands

```
\begin{tikzpicture}
\node (start) [ss] at (0,14) {Start};
\node (in1) [io] at (0,12) {Input};
\node (pro1) [proc] at (0,10) {Process 1};
\node (dec1) [dec] at (0,7) {Decision 1};
\node (pro2a) [proc] at (0,4) {Process 2a};
\node (pro2b) [proc] at (4,7) {Process 2b};
\node (out1) [io] at (0,2) {Output};
\node (stop) [ss] at (0,0) {Stop};
\end{tikzpicture}
```

give us the following output

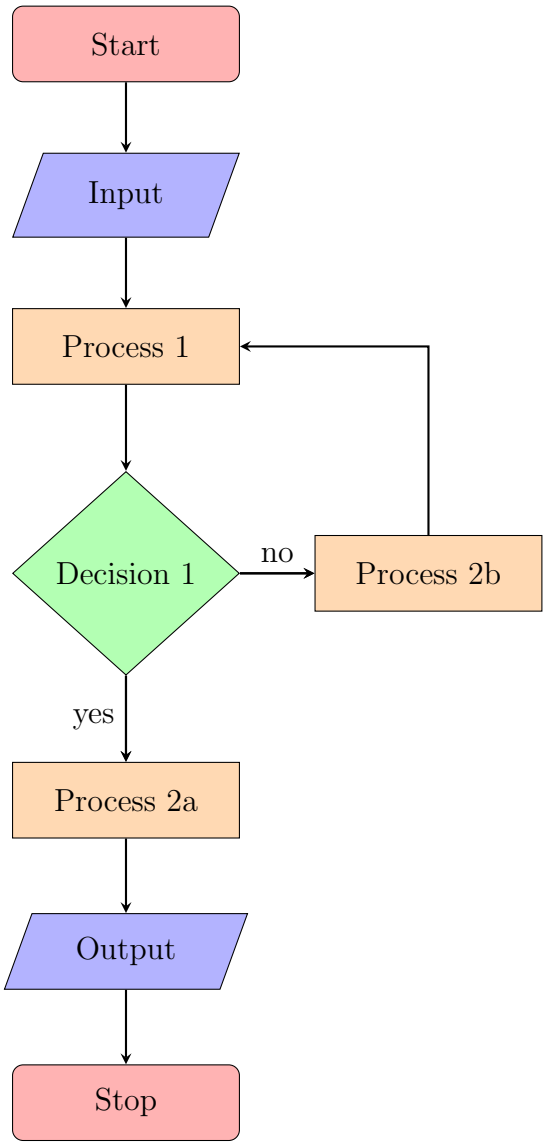


6. Now, the commands

```
\begin{tikzpicture}
\node (start) [ss] at (0,14) {Start};
\node (in1) [io] at (0,12) {Input};
\node (pro1) [proc] at (0,10) {Process 1};
\node (dec1) [dec] at (0,7) {Decision 1};
\node (pro2a) [proc] at (0,4) {Process 2a};
\node (pro2b) [proc] at (4,7) {Process 2b};
\node (out1) [io] at (0,2) {Output};
\node (stop) [ss] at (0,0) {Stop};
\draw [arrow] (start) -- (in1);
\draw [arrow] (in1) -- (pro1);
\draw [arrow] (pro1) -- (dec1);
\draw [arrow] (dec1) -- (pro2a);
\draw [arrow] (dec1) -- (pro2b);
```

```
%\draw [arrow] (dec1) -- node {yes} (pro2a);
%\draw [arrow] (dec1) -- node {no} (pro2b);
\draw [arrow] (dec1) -- node[anchor=east] {yes} (pro2a);
\draw [arrow] (dec1) -- node[anchor=south] {no} (pro2b);
%\draw [arrow] (pro2b) -- (pro1);
\draw [arrow] (pro2b) |- (pro1);
\draw [arrow] (pro2a) -- (out1);
\draw [arrow] (out1) -- (stop);
\end{tikzpicture}
```

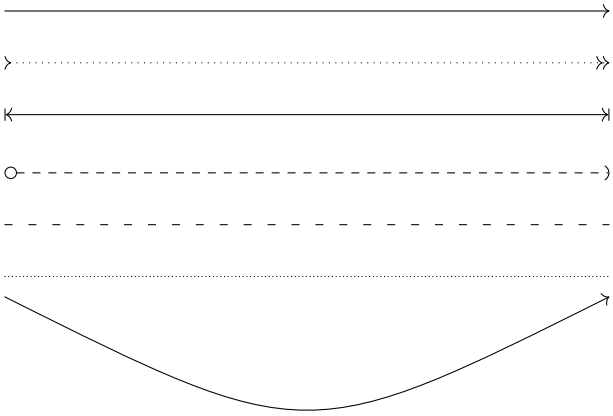
give us the following output



9 More on Shapes

1. Some special types of arrows can be created using the following commands:

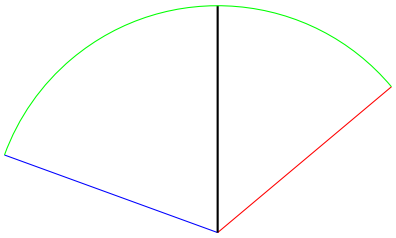
```
\draw[->] (0,0) -- (14,0);
\draw[dotted, >->>] (0,0) -- (4,0);
\draw[|<->|] (0,0) -- (4,0);
\draw[dashed, o-)] (0,0) -- (4,0);
\draw[loosely dashed] (0,0) -- (4,0);
\draw[densely dotted] (0,0) -- (4,0);
\draw[->] (0,0) .. controls (2,-1) .. (4, 0);
```



2. We can also draw figures using the polar coordinates also. For example, consider the command:

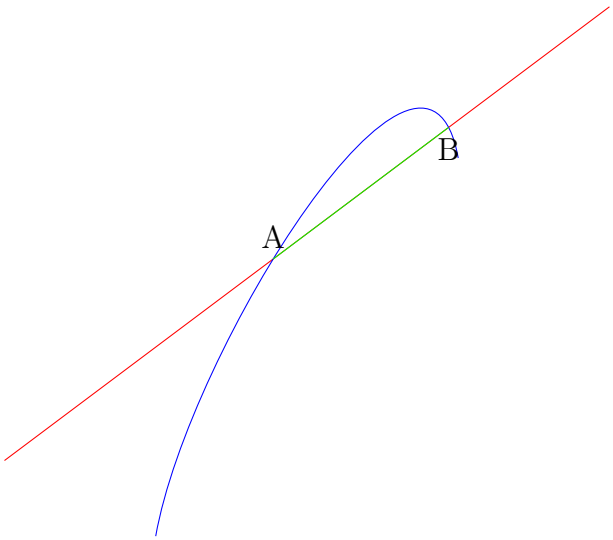
```
\begin{tikzpicture}
\draw[color=red] (0,0) -- (40:1);
\draw[color=blue] (0,0) -- (160:1);
\draw[thick] (0,0) -- (90:1);
\draw[color=green] (40:1) arc (40:160:1);
\end{tikzpicture}
```

   Here, we get the figure



3. Use of the `intersections` library can be illustrated as follows:

```
\begin{tikzpicture}
% Draw to path and give a name to them
\draw [red, name path={red line}] (0,0) -- (4,3);
\draw [blue, name path={blue curve}] (1,-0.5) to[out=80, in=100] (3,2);
% use the intersections on a path to giv them coordinates
% and draw a line between them
\draw [green, name intersections={of=red line and blue curve,
by={first intersect, second intersect}}]
(first intersect) -- (second intersect);
% one can use the coordinates furtheron
\node [above] at (first intersect) {A};
\node [below] at (second intersect) {B};
\end{tikzpicture}
```

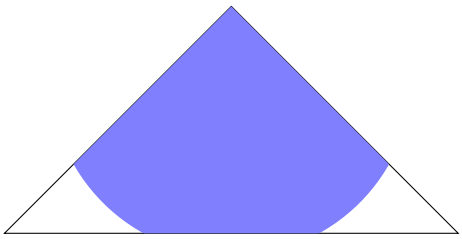


4. A path is a series of straight and curved line segments. We can draw paths, fill them colours or can use for clipping of subsequent drawings. After a `\clip` command, all subsequent drawings are clipped, only the parts inside the clipping region are drawn.

   For example,

   ```
   \begin{tikzpicture}
   \path[clip, draw] (1,1)--(2,2)--(3,1)--cycle;
   \path[fill=blue!50] (2, 1.7) circle (.8);
   \end{tikzpicture}
   ```

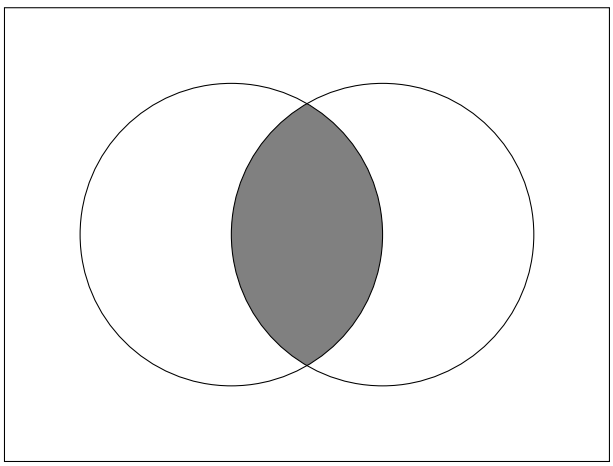
   yields the output



5. We can use the *scope* environment to restrict the effect of clipping, as shown in the following example:

```
\begin{tikzpicture}
\draw (-2, 1.5) rectangle (2, -1.5);
\begin{scope}
\clip (-0.5, 0) circle (1);
\clip ( 0.5, 0) circle (1);
\fill[color=gray] (-2,1.5)rectangle (2,-1.5);
\end{scope}
\draw (-0.5, 0) circle (1);
\draw ( 0.5, 0) circle (1);
\end{tikzpicture}
```

yields the output



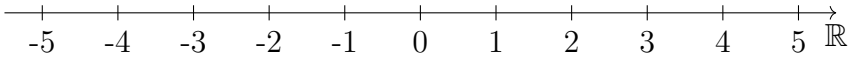
6. We can also create loops (like loops in algorithms/programs), as explained below:

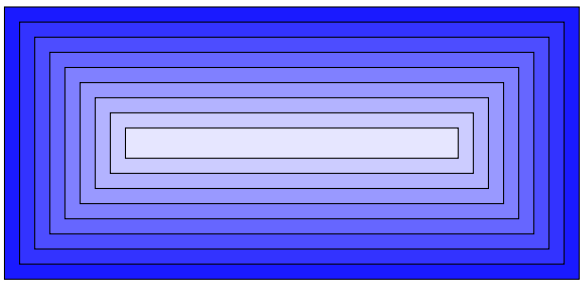
The codes

```
\draw[->] (-5.5,0) -- (5.5,0) node [below] {$\mathbb{R}$};
\foreach \x in {-5,...,5}
\draw (\x, 0.1) -- (\x, -0.1) node [below] {\x};
%
%
\foreach \x in {1,3,...,10}
```

```
\shade[ball color=red!\x 0!green] (\x,0) circle (3mm);
%
%
\foreach \x in {9,...,1}
\draw[fill=blue!\x0] (-0.1*\x - 1, -0.1*\x )
rectangle (0.1*\x + 1, 0.1*\x );
```

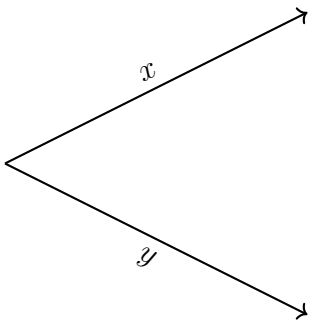
respectively give the following diagrams





7. Labeling of lines can be done as given below:

```
\begin{tikzpicture}
\draw[->,thick] (0,0) -- (4,2) node[pos=.5,sloped,above] {$x$};
\draw[->,thick] (0,0) -- (4,-2) node[pos=.5,sloped,below] {$y$};
\end{tikzpicture}
```

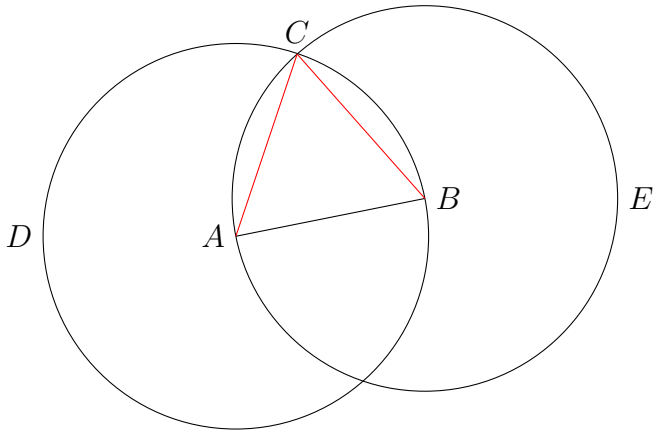


8. Figures including shapes can be drawn as follows:

```
\begin{tikzpicture}[scale=2]
\coordinate [label=left:$A$] (A) at (0,0);
\coordinate [label=right:$B$] (B) at (1.25,0.25);
\draw (A) -- (B);
\node (D) [draw,circle through=(B),label=left:$D$] at (A) {};
\node (E) [draw,circle through=(A),label=right:$E$] at (B) {};
\coordinate[label=above:$C$] (C) at (intersection 2 of D and E);
```

```
\draw [red] (A) -- (C);
\draw [red] (B) -- (C);
\end{tikzpicture}
```

generates the diagram (note that the library *through* is to be used here.)

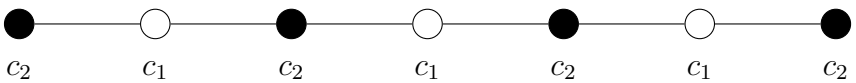


10 Graph Theory Using Tikz

Tikz creates amazingly neat and beautiful and graphs in graph theory. To draw a path, we can use the following Tikz code.

```
\tikzstyle{every node}=[draw,shape=circle,node distance=1.8cm]
\begin{tikzpicture}
\node [fill,label=below:$c_2$] (1) {};
\node [label=below:$c_1$] (2) [right of=1] {};
\node [fill,label=below:$c_2$] (3) [right of=2] {};
\node [label=below:$c_1$] (4) [right of=3] {};
\node [fill,label=below:$c_2$] (5) [right of=4] {};
\node [label=below:$c_1$] (6) [right of=5] {};
\node [fill,label=below:$c_2$] (7) [right of=6] {};
\draw
(1) -- (2)
(2) -- (3)
(3) -- (4)
(4) -- (5)
(5) -- (6)
(6) -- (7);
\end{tikzpicture}
```

and the output will be as follows:



Note that the commands such as `[right of=a]`, `[left of=a]`, `[above of=a]` and `[below of=a]` can be used to position shapes in flowcharts also.

The command `node distance=1.8cm` creates a distance 1.8cm between two adjacent nodes.

10.1 Graphs Using Polar Coordinates

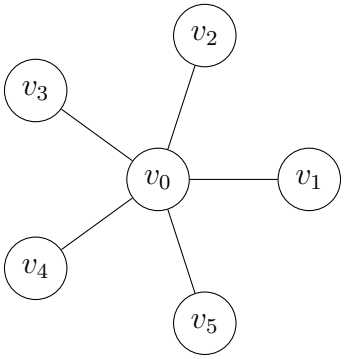
1. To draw cycles and/or vertices in a cyclic manner, it is better to use polar coordinates system in Tikz. Look at the following code:

```
\begin{tikzpicture}
\tikzstyle{every node}=[draw,shape=circle];
\node (v0) at (0:0) {$v_0$};
\node (v1) at (   0:2) {$v_1$};
\node (v2) at (  72:2) {$v_2$};
\node (v3) at (2*72:2) {$v_3$};
\node (v4) at (3*72:2) {$v_4$};
\node (v5) at (4*72:2) {$v_5$};
\draw (v0) -- (v1)
(v0) -- (v2)
(v0) -- (v3)
(v0) -- (v4)
(v0) -- (v5);
\end{tikzpicture}
```

   Note that in the command line `\node (v0) at (0:0) {$v_0$};`, `(v0)` defines the name of the corresponding vertex. Putting names for vertices is essential for drawing edges between vertices. Now, `at (a:b)` defines the position of that point at an angle a^{deg} from the horizontal and radius b points.

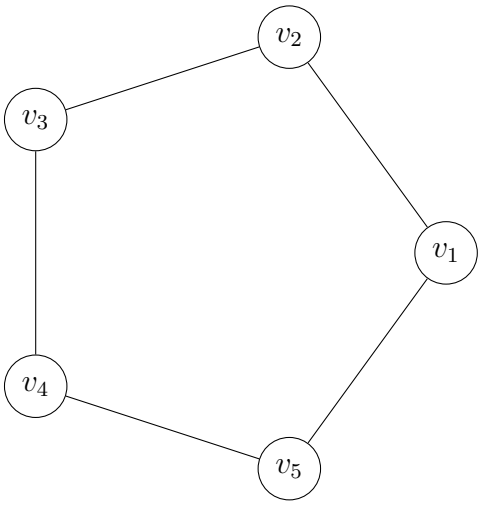
   The command `\tikzstyle{every node}=[draw,shape=circle];` defines the attributes of the vertices. Also, while drawing the edges, note that the semicolon is placed only after the last edge is defined.

   Then, the above code will provide us the following output:



2. A cycle on five vertices can be created using the command

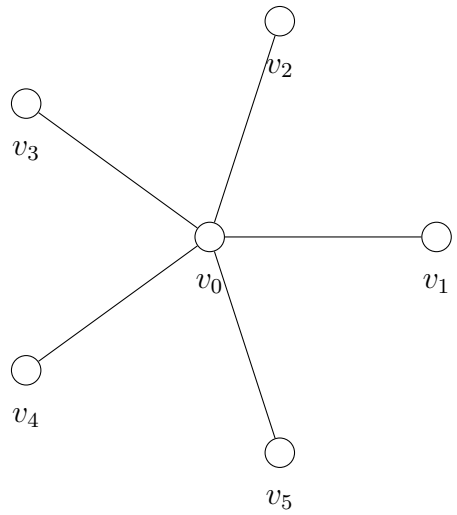
```
\begin{tikzpicture}
\tikzstyle{every node}=[draw,shape=circle];
%\node (v0) at (0:0) {$v_0$};
\node (v1) at (0:2) {$v_1$};
\node (v2) at (72:2) {$v_2$};
\node (v3) at (2*72:2) {$v_3$};
\node (v4) at (3*72:2) {$v_4$};
\node (v5) at (4*72:2) {$v_5$};
\draw (v1) -- (v2)
(v2) -- (v3)
(v3) -- (v4)
(v4) -- (v5)
(v1) -- (v5);
\end{tikzpicture}
```



3. Labeling can be done in a different way so that labels appear outside of the nodes. See the example:

```
\begin{tikzpicture}
\tikzstyle{every node}=[draw,shape=circle];
\node (v0) at (0:0) [label=below:$v_0$] {};
\node (v1) at (   0:3) [label=below:$v_1$] {};
\node (v2) at (  72:3) [label=below:$v_2$] {};
\node (v3) at (2*72:3) [label=below:$v_3$] {};
\node (v4) at (3*72:3) [label=below:$v_4$] {};
\node (v5) at (4*72:3) [label=below:$v_5$] {};
\draw (v0) -- (v1)
(v0) -- (v2)
(v0) -- (v3)
(v0) -- (v4)
(v0) -- (v5);
\end{tikzpicture}
```

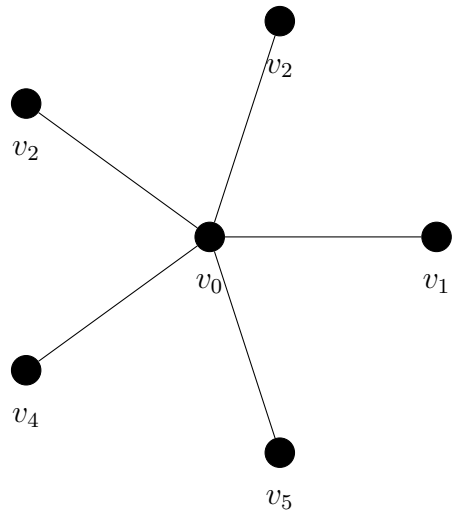
which gives the output as



4. Labeling can be done in a different way so that labels appear outside of the nodes. See the example:

```
\begin{tikzpicture}
\tikzstyle{every node}=[draw,shape=circle];
\node (v0) at (0:0) [fill,label=below:$v_0$] {};
\node (v1) at (   0:3) [fill,label=below:$v_1$] {};
\node (v2) at (  72:3) [fill,label=below:$v_2$] {};
\node (v3) at (2*72:3) [fill,label=below:$v_3$] {$v_3$};
\node (v4) at (3*72:3) [fill,label=below:$v_4$] {$v_4$};
\node (v5) at (4*72:3) [fill,label=below:$v_5$] {$v_5$};
\draw (v0) -- (v1)
(v0) -- (v2)
(v0) -- (v3)
(v0) -- (v4)
(v0) -- (v5);
\end{tikzpicture}
```

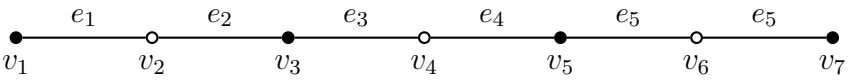
which gives the outputas



We can also customise the vertex attributes with the entire document globally using some commands like `\newcommand{\vertex}{\node[vertex]}` in the preamble. Also, the tikz-style codes can be written in square brackets as shown in the following.

```
\begin{figure}[h!]
\centering
\begin{tikzpicture}[auto,node distance=1.8 cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries}]
\vertex [fill,label=below:$c_2$] (1) {};
\vertex [label=below:$c_1$] (2) [right of=1] {};
\vertex [fill,label=below:$c_2$] (3) [right of=2] {};
\vertex [label=below:$c_1$] (4) [right of=3] {};
\vertex [fill,label=below:$c_2$] (5) [right of=4] {};
\vertex [label=below:$c_1$] (6) [right of=5] {};
\vertex [fill,label=below:$c_2$] (7) [right of=6] {};
\path[every node/.style={font=\sffamily\small}]
(1) edge node {} (2)
(2) edge node {} (3)
(3) edge node {} (4)
(4) edge node {} (5)
(5) edge node {} (6)
(6) edge node {} (7);
\end{tikzpicture}
\caption{The path $P_7$ with $\chi^+$colouring}\label{pathmaxorder}
\end{figure}
```

The output of the above will be as follows:

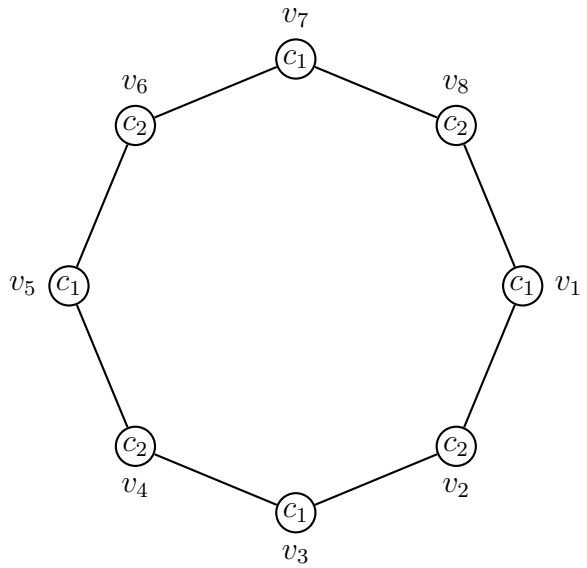


The code `\path` can be used in place of `\draw` so that labeling and colouring becomes easier.

The code

```
\begin{tikzpicture}[auto,node distance=1.5 cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries}]
\vertex (v1) at (0:3) [label=right:$c_1$]{$v_1$};
\vertex (v2) at (315:3)[label=below:$c_2$]{$v_2$};
\vertex (v3) at (270:3) [label=below:$c_1$]{$v_3$};
\vertex (v4) at (225:3)[label=below:$c_2$]{$v_4$};
\vertex (v5) at (180:3)[label=left:$c_1$]{$v_5$};
\vertex (v6) at (135:3)[label=above:$c_2$]{$v_6$};
\vertex (v7) at (90:3)[label=above:$c_1$]{$v_7$};
\vertex (v8) at (45:3) [label=above:$c_2$]{$v_8$};
\path
(v1) edge (v2)
(v1) edge (v8)
(v2) edge (v3)
(v3) edge (v4)
(v4) edge (v5)
(v5) edge (v6)
(v6) edge (v7)
(v7) edge (v8);
\end{tikzpicture}
```

creates the cycle

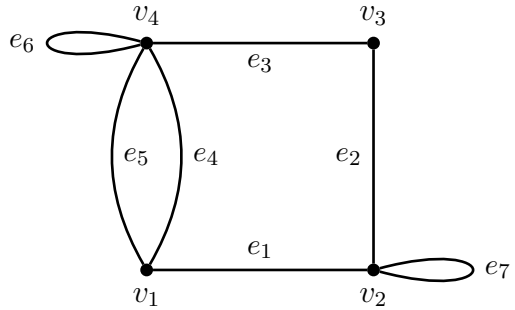


Multiple edges and loops can be created using commands as shown in the following Tikz code

```
\begin{tikzpicture}[auto,node distance=2.5cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries}]
\vertex (v1) at (0,0) [fill,label={below:$v_1$}]{};
\vertex (v2) at (3,0) [fill,label={below:$v_2$}]{};
\vertex (v3) at (3,3) [fill,label={above:$v_3$}]{};
\vertex (v4) at (0,3) [fill,label={above:$v_4$}]{};
\path[every node/.style={font=\sffamily\small},line width=1pt]
```

```
(v1) edge node {$e_1$} (v2)
(v2) edge node {$e_2$}(v3)
(v3) edge node {$e_3$}(v4)
(v4) edge [bend left] node {$e_4$} (v1)
(v4) edge [bend right] node {$e_5$} (v1)
(v4) edge [loop left,looseness=100] node {$e_6$} (v4)
(v2) edge [loop right,looseness=100] node {$e_7$} (v2)
;
\end{tikzpicture}
```

which gives the following output:



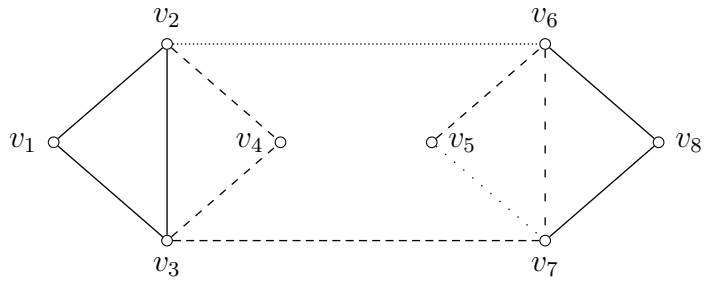
The codes `loop left`, `loop right`, `loop above`, `loop below`, `loop above right` etc. can be used for correct positioning of the loop. The command `losseness=a` decides the shape of the loop.

Graphs with dashed and/or dotted edges can be created using the commands as given below:

```
\begin{tikzpicture}
\vertex (1) at (0,0) [label={left:$v_1$}] {};
\vertex (2) at (0.75,0.65) [label={above:$v_2$}] {};
\vertex (3) at (0.75,-0.65) [label={below:$v_3$}] {};
\vertex (4) at (1.5,0) [label={left:$v_4$}] {};
\vertex (5) at (2.5,0) [label={right:$v_5$}] {};
\vertex (6) at (3.25,0.65) [label={above:$v_6$}] {};
\vertex (7) at (3.25,-0.65) [label={below:$v_7$}] {};
\vertex (8) at (4,0) [label={right:$v_8$}] {};
\path[every node/.style={font=\sffamily\small},line width=1pt]
(1) edge node {} (2)
(1) edge node {} (3)
(2) edge node {} (3)
(2) edge [dashed] node {} (4)
(3) edge [dashed] node {} (4)
(2) edge [densely dotted] node {} (6)
(3) edge [densely dashed] node {} (7)
(5) edge [dashed] node {} (6)
(5) edge [loosely dotted] node {} (7)
(6) edge [loosely dashed] node {} (7)
(6) edge node {} (8)
(7) edge node {} (8)
```

```
;
\end{tikzpicture}
```

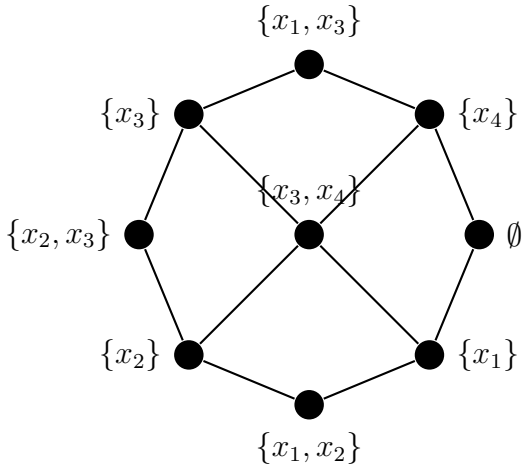
This will yields the figure:



Set-labels can be assigned to the graphs by using the Tikz code given below:

```
\begin{tikzpicture}[auto,node distance=1.5cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries}]
%
\node [circle,fill] (u1) at (0:2.25) [label={right:$\emptyset$}]{};
\node [circle,fill] (u2) at (315:2.25) [label={right:$\{x_1\}$}]{};
\node [circle,fill] (u3) at (270:2.25) [label={below:$\{x_1,x_2\}$}]{};
\node [circle,fill] (u4) at (225:2.25) [label={left:$\{x_2\}$}]{};
\node [circle,fill] (u5) at (180:2.25) [label={left:$\{x_2,x_3\}$}]{};
\node [circle,fill] (u6) at (135:2.25) [label={left:$\{x_3\}$}]{};
\node [circle,fill] (u7) at (90:2.25) [label={above:$\{x_1,x_3\}$}]{};
\node [circle,fill] (u8) at (45:2.25) [label={right:$\{x_4\}$}]{};
\node [circle,fill] (u) at (0:0) [label={above:$\{x_3,x_4\}$}]{};
\path[every node/.style={font=\sffamily\small}]
(u1) edge node {} (u2)
(u2) edge node {} (u3)
edge node {} (u)
(u3) edge node {} (u4)
(u4) edge node {} (u5)
edge node {} (u)
(u5) edge node {} (u6)
(u6) edge node {} (u7)
edge node {} (u)
(u7) edge node {} (u8)
(u8) edge node {} (u1)
edge node {} (u)
;
\end{tikzpicture}
```

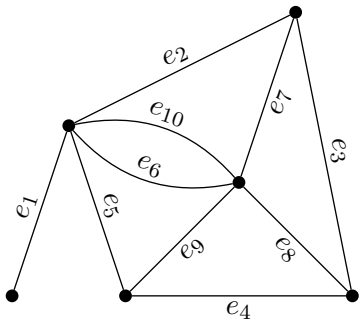
The above code will produce a figure as given below:



In a similar manner, we can assign labels to edges also. For example,

```
\begin{tikzpicture}[scale=1.5,auto,node distance=1.5cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries},el/.style = {inne
every label/.append style = {font=\tiny}]
\vertex [fill] (1) at (0,0) {};
\vertex [fill] (2) at (2,0) {};
\vertex [fill] (3) at (1.5,2.5)  {};
\vertex [fill] (4) at (-0.5,1.5)  {};
\vertex [fill] (5) at (1,1) {};
\vertex [fill] (6) at (-1,0) {};
\path[every node/.style={font=\sffamily\small},line width=0.5pt]
(1) edge node [el,below] {$e_4$} (2)
(2) edge node [el,above] {$e_3$} (3)
(3) edge node [el,above] {$e_2$} (4)
(4) edge node [el,above] {$e_5$} (1)
(4) edge node [el,above] {$e_1$} (6)
(5) edge node [el,below] {$e_9$} (1)
(5) edge node [el,below] {$e_8$} (2)
(5) edge node [el,below] {$e_7$} (3)
(5) edge node [el,above] {$e_6$} (4)
;
\end{tikzpicture}
```

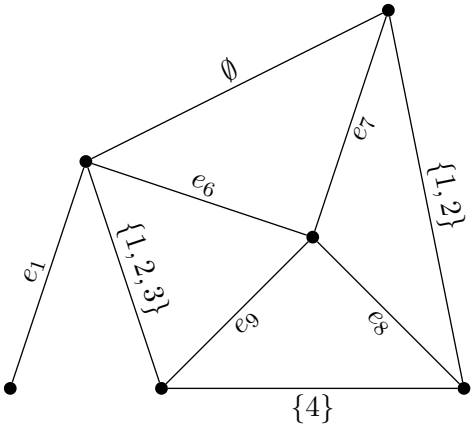
yields the figure



Edges can be given set-labels by using the commands as shown in the below Tikz code:

```
\begin{tikzpicture}[scale=1.5,auto,node distance=1.5cm,
thick,main node/.style={circle,draw,font=\sffamily\Large\bfseries},el/.style = {inne
every label/.append style = {font=\tiny}]
\vertex [fill] (1) at (0,0) {};
\vertex [fill] (2) at (2,0) {};
\vertex [fill] (3) at (1.5,2.5)  {};
\vertex [fill] (4) at (-0.5,1.5)  {};
\vertex [fill] (5) at (1,1) {};
\vertex [fill] (6) at (-1,0) {};
\path[every node/.style={font=\sffamily\small},line width=0.5pt]
(1) edge node [el,below] {$\{4\}$} (2)
(2) edge node [el,above] {$\{1,2\}$} (3)
(3) edge node [el,above] {$\emptyset$} (4)
(4) edge node [el,above] {$\{1,2,3\}$} (1)
(4) edge node [el,above] {$e_1$} (6)
(5) edge node [el,below] {$e_9$} (1)
(5) edge node [el,below] {$e_8$} (2)
(5) edge node [el,below] {$e_7$} (3)
(5) edge node [el,above] {$e_6$} (4)
;
\end{tikzpicture}
```

which gives the output



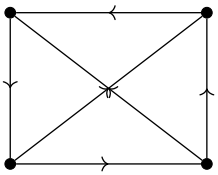
Note that the package *er* is necessary to use `el`.

11 Directed graphs Using Tikz

When use the command `arrow` or the symbol `->`, normally arrow marks will appear at the end of the edges. Using certain edge decorations, we can decide the position of the arrow mark. For example, see the following code:

```
\begin{tikzpicture}[x=1.3cm, y=1cm,every edge/.style={draw,postaction={decorate,
decoration={markings,mark=at position 0.5 with {\arrow{>}}}}}]
\vertex (1) at (0,0) [fill] {};
\vertex (2) at (2,0) [fill] {};
\vertex (3) at (2,2) [fill] {};
\vertex (4) at (0,2) [fill] {};
\path[every node/.style={font=\sffamily\small},line width=0.5pt]
(1) edge (2)
(2) edge (3)
(3) edge (4)
(4) edge (1)
(1) edge (3)
(2) edge (4)
;
\end{tikzpicture}
```

The output will be



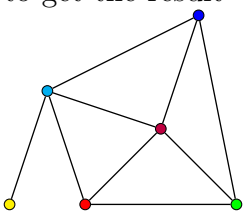
11.1 vertex and Edge Coloring

We can assign colors to the vertices by giving the attribute `fill=color name` in the attribute square bracket of definition the corresponding vertex. See the following code for example:

```
\begin{tikzpicture}
\vertex [fill=red] (1) at (0,0) {};
\vertex [fill=green] (2) at (2,0) {};
\vertex [fill=blue] (3) at (1.5,2.5)  {};
\vertex [fill=cyan] (4) at (-0.5,1.5)  {};
\vertex [fill=purple] (5) at (1,1) {};
\vertex [fill=yellow] (6) at (-1,0) {};
\path[every node/.style={font=\sffamily\small},line width=0.5pt]
(1) edge (2)
(2) edge (3)
(3) edge (4)
```

```
(4) edge (1)
(4) edge (6)
(5) edge (1)
(5) edge (2)
(5) edge (3)
(5) edge (4)
;
\end{tikzpicture}
```

to get the result

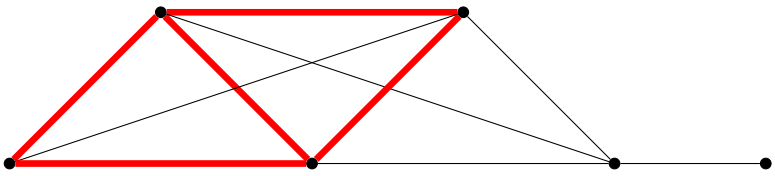


In a similar manner, we can assign colors to the edges also. See the following example:

The code

```
\begin{tikzpicture}
\vertex (u3) at (0,0) [fill]{3};
\vertex (u4) at (2,0) [fill]{4};
\vertex (u5) at (4,0) [fill]{5};
\vertex (u6) at (5,0) [fill]{6};
\vertex (u1) at (1,1) [fill]{2};
\vertex (u2) at (3,1) [fill]{1};
\path
(u1) edge [red,line width= 2.5pt ]  (u2)
(u1) edge [red,line width= 2.5pt](u3)
(u1) edge [red,line width= 2.5pt](u4)
(u1) edge (u5)
(u2) edge (u3)
(u2) edge[red,line width= 2.5pt] (u4)
(u2) edge (u5)
(u3) edge[red,line width= 2.5pt] (u4)
(u4) edge (u5)
(u5) edge (u6)
;
\end{tikzpicture}
```

yields the output



References

[1] J. Cremer, (2011). A very minimal introduction to TikZ, https://cremeronline.com/LaTeX/minimaltikz.pdf

[2] M Hellmund, (2009). PGF/TikZ - Graphics for LaTeX, www.math.uni-leipzig.de/~hellmund/LaTeX/pgf-tut.pdf

[3] A. Mertz, W. Slough (2007). Graphics with TikZ, The PracTeXJournal, **2007**(1), 22 pages.

[4] T. Tantau, (2007). Tikz and PGF Manual, Ver-2.10.