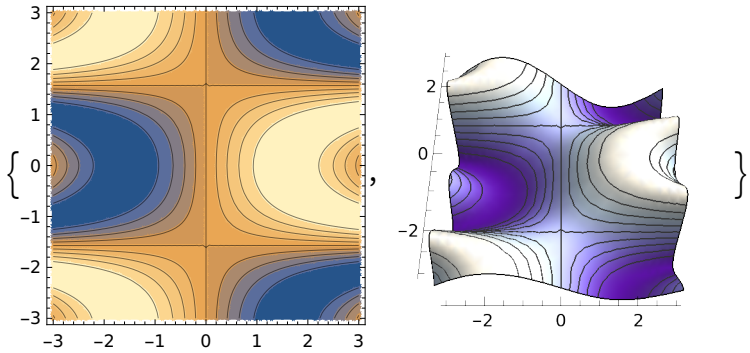
Plotting Functions of Two Variables with ContourPlot

1.ContourPlot is a commonly used command for visualizing a real-valued function of two variables.
2.A contour plot is a two-dimensional rendering of a three-dimensional surface.
3.It is much like a topographical map—it consists of the vertical projections of the contour lines onto the x-y plane.
4.The regions will be shaded according their relative height above (or below) the x-y plane;darker regions are lower and lighter regions are higher.

In[3]:=
```
{ContourPlot[Sin[x * Cos[y]], {x, -3, 3}, {y, -3, 3}], Plot3D[Sin[x * Cos[y]], {x, -3, 3},
  {y, -3, 3}, MeshFunctions → {#3 &}, Mesh → 9, ColorFunction → "LakeColors",
  ViewPoint → {0, -1, 2}, Boxed → False, Axes → {True, True, False}]}
```

Out[3]=



To produce a full contour plot,
we used ContourPlot in a manner identical to that of Plot3D. As a result,
above we see ContourPlot and a Plot3D of the same function,
showing the same level curves and using similar shading.

Formatting in ContourPlot

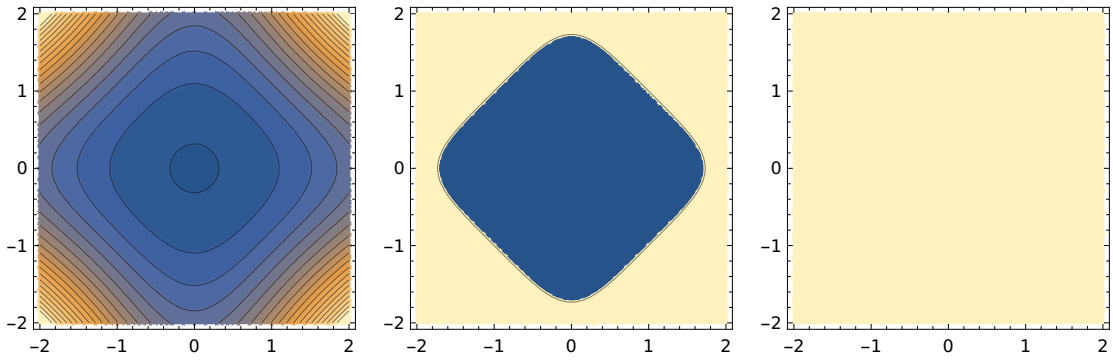
1.Contours:

The most commonly used option setting is Contours. Set it to a positive integer, say 20, and you will see 20 contour lines in the resulting plot.

In[4]:=
```
GraphicsRow@{ContourPlot[(1 + x^2) (1 + y^2), {x, -2, 2}, {y, -2, 2}, Contours → 20],
  ContourPlot[(1 + x^2) (1 + y^2), {x, -2, 2}, {y, -2, 2}, Contours → {4}],
  ContourPlot[(1 + x^2) (1 + y^2) = 4, {x, -2, 2}, {y, -2, 2}]}
```

Set: Tag Times in 4.99886 4.99886 is Protected.

Out[4]=

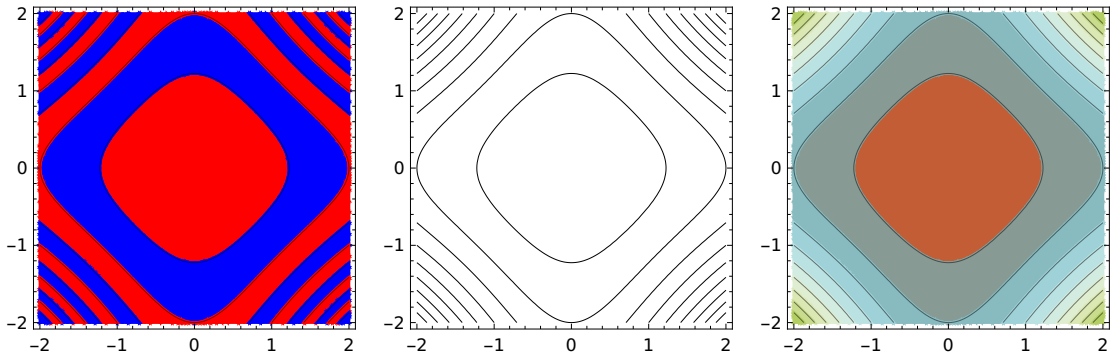


2.ContourShading:

The ContourShading option works much like the MeshShading option for Plot3D.

In[2]:=
```
GraphicsRow@
 {ContourPlot[(1 + x^2) (1 + y^2), {x, -2, 2}, {y, -2, 2}, ContourShading → {Red, Blue}],
  ContourPlot[(1 + x^2) (1 + y^2), {x, -2, 2}, {y, -2, 2}, ContourShading → None],
  ContourPlot[(1 + x^2) (1 + y^2), {x, -2, 2}, {y, -2, 2}, ColorFunction → "IslandColors"]}
```

Out[2]=



Note :

you may set this option to None. As was the case with Plot3D,
the ColorFunction may be set to any named color gradient.

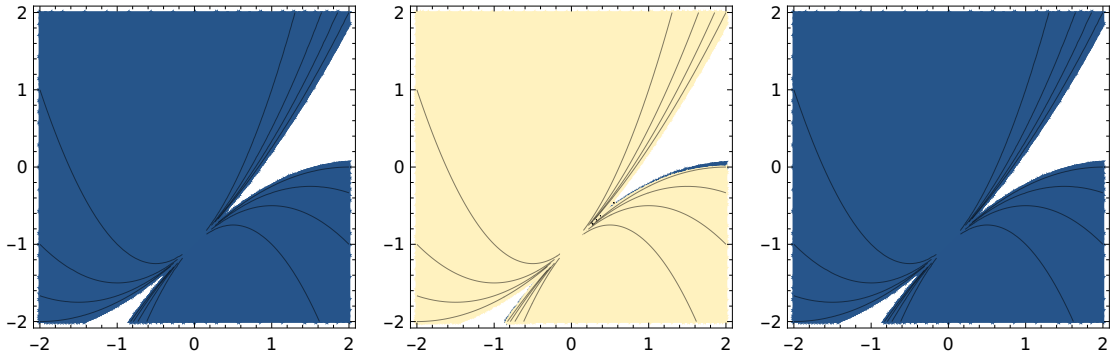
3.PlotPoints,MaxRecursion and Exclusions:

PlotPoints and **MaxRecursion**, are employed to improve image quality.

The **Exclusions** option provides another means of dealing with points of discontinuities of a function.

In[1]:=
```
GraphicsRow@{ContourPlot[x^2/(1-x+y), {x, -2, 2}, {y, -2, 2}],
  ContourPlot[x^2/(1-x+y), {x, -2, 2}, {y, -2, 2}, PlotPoints → 30, MaxRecursion → 3],
  ContourPlot[(x^2)/(1-x+y), {x, -2, 2}, {y, -2, 2}, Exclusions → {y == x-1}]}
```

Out[1]=



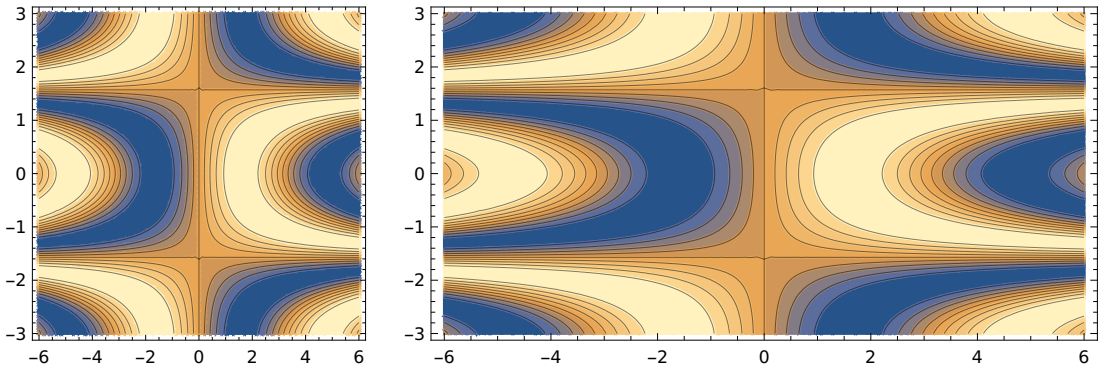
4.AspectRatio:

One uses **AspectRatio** to adjust the relative dimensions of a graphic produced by ContourPlot.

Note : By default, a ContourPlot will be square.

In[2]:=
```
GraphicsRow@{ContourPlot[Sin[x*Cos[y]], {x, -6, 6}, {y, -3, 3}],
  ContourPlot[Sin[x*Cos[y]], {x, -6, 6}, {y, -3, 3}, AspectRatio → Automatic]}
```

Out[2]=



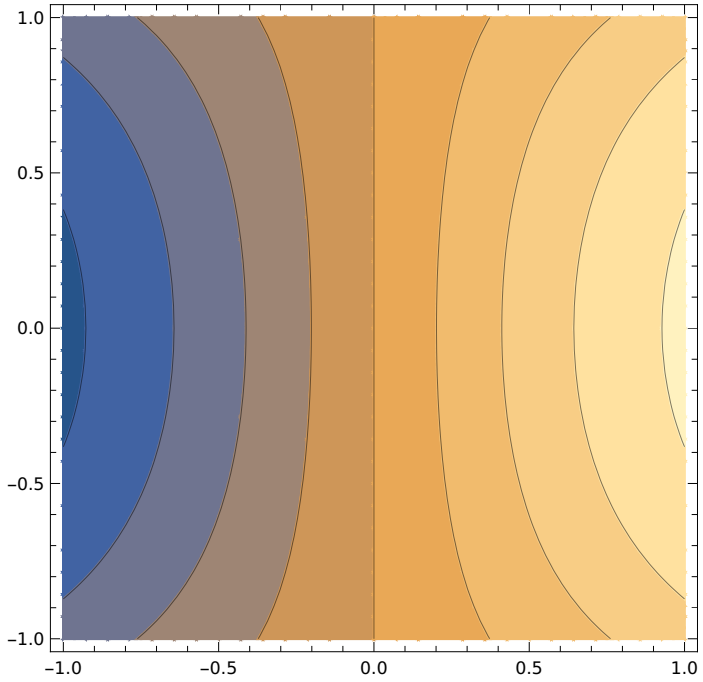
5.ContourLabels:

We can see the value of the z coordinate for all points on that curve by positioning the tip of the cursor along a level curve.

The option setting **ContourLabels→ Automatic** can be used to place these values directly onto the graphic.

In[3]:=
```
ContourPlot[Sin[x * Cos[y]], {x, -1, 1}, {y, -1, 1}, ContourLabels → Automatic]
```

Out[3]=

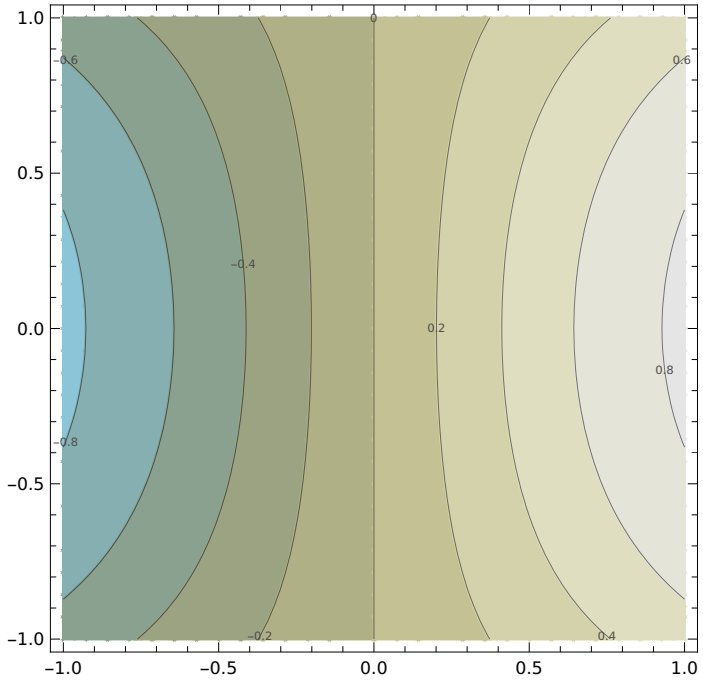


6.ColorFunction:

Displaying the z coordinate at the point ,(x, y), but makes the text gray in a six point font.

In[4]:=
```
ContourPlot[Sin[x * Cos[y]], {x, -1, 1}, {y, -1, 1},
 ContourLabels → (Style[Text[#3, {#1, #2}], GrayLevel[.3], 6] &),
 ColorFunction → "LightTerrain"]
```

Out[4]=



7.MeshFunctions:

To superimpose Mesh curves on top of the level curves.

In[5]:=
```
ContourPlot[x^2 - (4 * x * y / y^2 + 1), {x, -2, 2}, {y, -2, 2},
 MeshFunctions → {#1^2 + 2 #2^2 &}, Mesh → {{1}}, MeshStyle → Directive[Thick, Yellow]]
```

Out[5]=