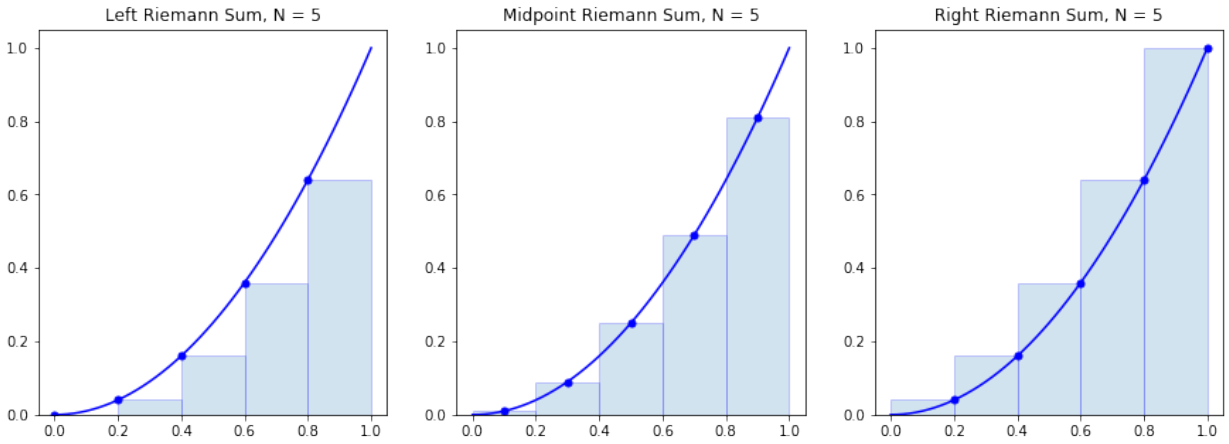
In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
```

We are now going to investigate the function $f(x) = x^2$, on the interval $[0, 1]$, using Riemann sums.

In [41]:
```
f = lambda x : x**2
a = 0; b = 1; N = 5
n = 10 # Use n*N+1 points to plot the function smoothly

x = np.linspace(a,b,N+1)
y = f(x)

X = np.linspace(a,b,n*N+1)
Y = f(X)

plt.figure(figsize=(15,5))

plt.subplot(1,3,1)
plt.plot(X,Y,'b')
x_left = x[:-1] # Left endpoints
y_left = y[:-1]
plt.plot(x_left,y_left,'b.',markersize=10)
plt.bar(x_left,y_left,width=(b-a)/N,alpha=0.2,align='edge',edgecolor='
b')
plt.title('Left Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,2)
plt.plot(X,Y,'b')
x_mid = (x[:-1] + x[1:])/2 # Midpoints
y_mid = f(x_mid)
plt.plot(x_mid,y_mid,'b.',markersize=10)
plt.bar(x_mid,y_mid,width=(b-a)/N,alpha=0.2,edgecolor='b')
plt.title('Midpoint Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,3)
plt.plot(X,Y,'b')
x_right = x[1:] # Left endpoints
y_right = y[1:]
plt.plot(x_right,y_right,'b.',markersize=10)
plt.bar(x_right,y_right,width=-(b-a)/N,alpha=0.2,align='edge',edgecolo
r='b')
plt.title('Right Riemann Sum, N = {}'.format(N))
```

Out[41]: Text(0.5, 1.0, 'Right Riemann Sum, N = 5')




In [40]:
```
dx = (b-a)/N
x_left = np.linspace(a,b-dx,N)
x_midpoint = np.linspace(dx/2,b - dx/2,N)
x_right = np.linspace(dx,b,N)

print("Partition with",N,"subintervals.")
left_riemann_sum = np.sum(f(x_left) * dx)
print("Left Riemann Sum:",left_riemann_sum)

midpoint_riemann_sum = np.sum(f(x_midpoint) * dx)
print("Midpoint Riemann Sum:",midpoint_riemann_sum)

right_riemann_sum = np.sum(f(x_right) * dx)
print("Right Riemann Sum:",right_riemann_sum)
```

```
Partition with 10 subintervals.
Left Riemann Sum: 0.2850000000000001
Midpoint Riemann Sum: 0.33249999999999996
Right Riemann Sum: 0.385
```

All threee of these Riemann sumns approximate $\int_0^1 x^2 dx$. As we discussed in class,

$$\int_0^1 x^2 dx = \frac{x^3}{3}|_0^1 = \frac{1}{3}.$$

Note that the left Riemann sums underestimates the integral, the right Riemann sums overestimates it, and the midpoint Riemann sum is almost exactly right on the money.

Now let's consider a more complicated function to integrate. We shall approximate

$$\int_0^1 \frac{x}{1+x^2} dx.$$

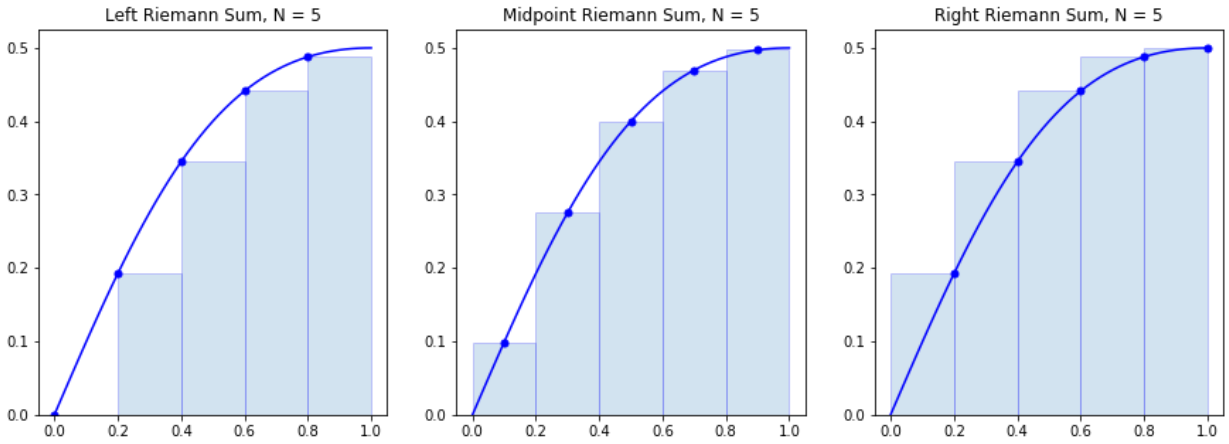
Computing the anti-derivative, we see that this integral equals

$$\frac{1}{2}\log(1+x^2)|_0^1 = \frac{\log(2)}{2} = .346\ldots$$

We shall now employ Riemann sums.

In [45]:
```
f = lambda x : x/(1+x**2)
a = 0; b = 1; N = 5
n = 10 # Use n*N+1 points to plot the function smoothly

x = np.linspace(a,b,N+1)
y = f(x)

X = np.linspace(a,b,n*N+1)
Y = f(X)

plt.figure(figsize=(15,5))

plt.subplot(1,3,1)
plt.plot(X,Y,'b')
x_left = x[:-1] # Left endpoints
y_left = y[:-1]
plt.plot(x_left,y_left,'b.',markersize=10)
plt.bar(x_left,y_left,width=(b-a)/N,alpha=0.2,align='edge',edgecolor='
b')
plt.title('Left Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,2)
plt.plot(X,Y,'b')
x_mid = (x[:-1] + x[1:])/2 # Midpoints
y_mid = f(x_mid)
plt.plot(x_mid,y_mid,'b.',markersize=10)
plt.bar(x_mid,y_mid,width=(b-a)/N,alpha=0.2,edgecolor='b')
plt.title('Midpoint Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,3)
plt.plot(X,Y,'b')
x_right = x[1:] # Left endpoints
y_right = y[1:]
plt.plot(x_right,y_right,'b.',markersize=10)
plt.bar(x_right,y_right,width=-(b-a)/N,alpha=0.2,align='edge',edgecolo
r='b')
plt.title('Right Riemann Sum, N = {}'.format(N))
```

Out[45]: Text(0.5, 1.0, 'Right Riemann Sum, N = 5')




In [46]:
```
dx = (b-a)/N
x_left = np.linspace(a,b-dx,N)
x_midpoint = np.linspace(dx/2,b - dx/2,N)
x_right = np.linspace(dx,b,N)

print("Partition with",N,"subintervals.")
left_riemann_sum = np.sum(f(x_left) * dx)
print("Left Riemann Sum:",left_riemann_sum)

midpoint_riemann_sum = np.sum(f(x_midpoint) * dx)
print("Midpoint Riemann Sum:",midpoint_riemann_sum)

right_riemann_sum = np.sum(f(x_right) * dx)
print("Right Riemann Sum:",right_riemann_sum)
```

```
Partition with 5 subintervals.
Left Riemann Sum: 0.2932233254303209
Midpoint Riemann Sum: 0.3482550971134317
Right Riemann Sum: 0.3932233254303209
```

We see that all three sums are pretty close, but the Midpoint Riemann sums is the most accurate in this case.

Let us now approximate

$$\int_0^1 \frac{100}{\sqrt{1-x^2}} dx.$$

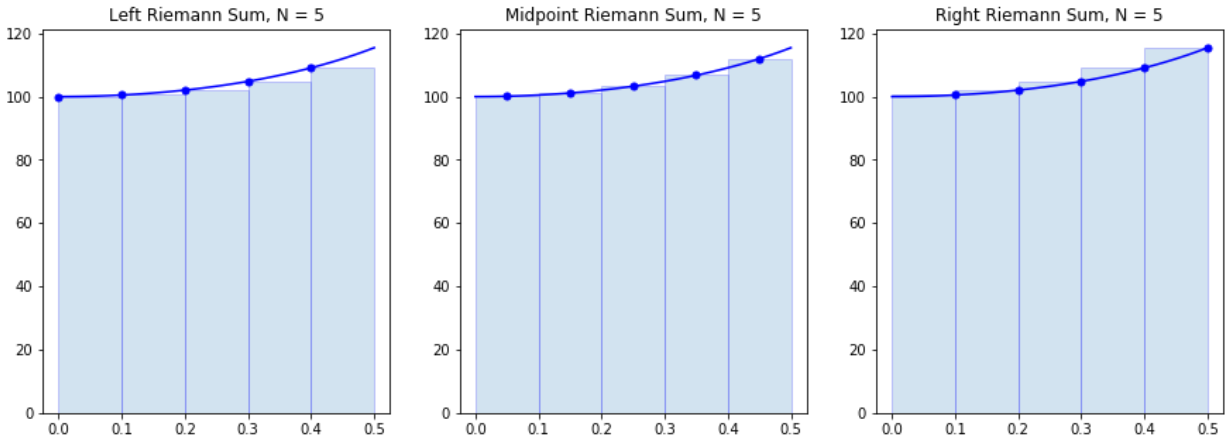
Once again, we can compute the anti-derivative and obtain

$$\int_0^{\frac{1}{2}} \frac{100}{\sqrt{1-x^2}} dx = 100 \sin^{-1}(x)|_0^{\frac{1}{2}} = 100 \cdot \frac{\pi}{6} = 52.33...$$

Let's see what the approximation by Riemann sums yields.

In [55]:
```
f = lambda x : 100/np.sqrt(1-x**2)
a = 0; b = 1/2; N = 5
n = 10 # Use n*N+1 points to plot the function smoothly

x = np.linspace(a,b,N+1)
y = f(x)

X = np.linspace(a,b,n*N+1)
Y = f(X)

plt.figure(figsize=(15,5))

plt.subplot(1,3,1)
plt.plot(X,Y,'b')
x_left = x[:-1] # Left endpoints
y_left = y[:-1]
plt.plot(x_left,y_left,'b.',markersize=10)
plt.bar(x_left,y_left,width=(b-a)/N,alpha=0.2,align='edge',edgecolor='
b')
plt.title('Left Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,2)
plt.plot(X,Y,'b')
x_mid = (x[:-1] + x[1:])/2 # Midpoints
y_mid = f(x_mid)
plt.plot(x_mid,y_mid,'b.',markersize=10)
plt.bar(x_mid,y_mid,width=(b-a)/N,alpha=0.2,edgecolor='b')
plt.title('Midpoint Riemann Sum, N = {}'.format(N))

plt.subplot(1,3,3)
plt.plot(X,Y,'b')
x_right = x[1:] # Left endpoints
y_right = y[1:]
plt.plot(x_right,y_right,'b.',markersize=10)
plt.bar(x_right,y_right,width=-(b-a)/N,alpha=0.2,align='edge',edgecolo
r='b')
plt.title('Right Riemann Sum, N = {}'.format(N))
```

Out[55]: Text(0.5, 1.0, 'Right Riemann Sum, N = 5')




In [56]:
```
dx = (b-a)/N
x_left = np.linspace(a,b-dx,N)
x_midpoint = np.linspace(dx/2,b - dx/2,N)
x_right = np.linspace(dx,b,N)

print("Partition with",N,"subintervals.")
left_riemann_sum = np.sum(f(x_left) * dx)
print("Left Riemann Sum:",left_riemann_sum)

midpoint_riemann_sum = np.sum(f(x_midpoint) * dx)
print("Midpoint Riemann Sum:",midpoint_riemann_sum)

right_riemann_sum = np.sum(f(x_right) * dx)
print("Right Riemann Sum:",right_riemann_sum)
```

```
Partition with 5 subintervals.
Left Riemann Sum: 51.6503282932075
Midpoint Riemann Sum: 52.32797429759466
Right Riemann Sum: 53.19733367700002
```

Once again, the Riemann sum approximation is very close to the exact answer.

Since the midterm is coming up, let's review graphing a bit. One of the functions that came up above is

$$y = \frac{x}{1+x^2}.$$

Let's graph this function by using the techniques we learned earlier in the semester.

There are no vertical asymtotes since we are not dividing by $0$ anywhere. The horizontal asymtotes are on the $x$-axis since

$$\lim_{x \to \pm\infty} \frac{x}{1+x^2} = 0.$$

The $x$-intercept is at $(0,0)$, and so is the $y$-intercept.

Taking the derivative we obtain

$$f'(x) = \frac{1-x^2}{(1+x^2)^2} = \frac{(1-x)(1+x)}{(1+x^2)^2}.$$

We can see from this that $f$ is decreasing on $(-\infty, -1)$, increasing on $(-1,1)$, and decreasing again on $(1,\infty)$.

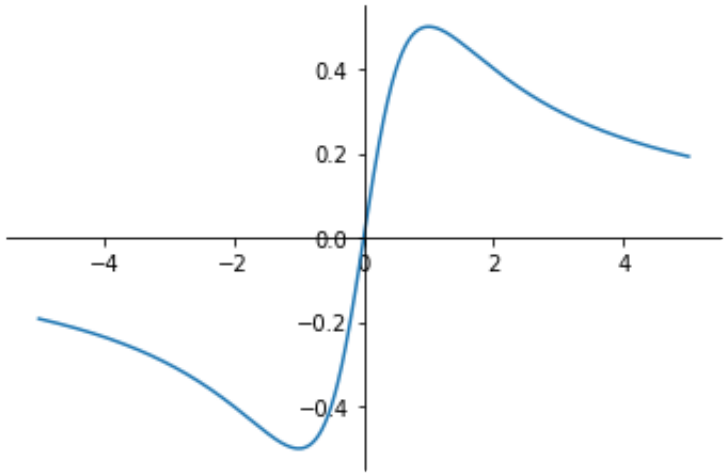
The critical points are $1, -1$. Checking to the left and to the right reveals that $-1$ is a local minimum and $1$ is a local maximum. Computing the second derivative we obtain

$$f''(x) = \frac{2x(x-\sqrt{3})(x+\sqrt{3})}{(1+x^2)^3}.$$

From this we deduce that there are inflection points at $x = 0$ and $x = \pm\sqrt{3}$. We can also see that $f''(x) < 0$ on $(-\infty, -\sqrt{3})$, concave up on $(-\sqrt{3}, 0)$, concave down on $(0, \sqrt{3})$ and concave up on $(\sqrt{3}, \infty)$.

In [59]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 1000)
y = x/(1+x**2)

ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
plt.plot(x,y)
```

Out[59]:
```
[<matplotlib.lines.Line2D at 0x120254790>]
```




As we can see, the graph is completely consistent with our observations above.

In [ ]: