

プログラムリスト 6.1 SymPyを用いて積分を計算する

```
In [31]: import sympy as sp          # (A1) SymPyライブラリをimport
x = sp.symbols('x')              # (B1) シンボリック変数xを定義

y1 = x**2 + 1/x                  # (B2) 関数 y1 = x^2 + 1/x を定義
y2 = sp.log(x) + sp.exp(x)      # (B3) 関数 y2 = ln(x) + e^x を定義
y3 = sp.sin(x) + sp.cos(x)      # (B4) 関数 y3 = sin(x) + cos(x) を定義

y1_int = sp.integrate(y1, x)     # (C1) y1 の積分を計算
y2_int = sp.integrate(y2, x)     # (C2) y2 の積分を計算
y3_int = sp.integrate(y3, x)     # (C3) y3 の積分を計算

print(f"y1 = x**2 + 1/x の積分: {y1_int}") # (D1) 計算結果を表示
print(f"y2 = ln(x) + e^x の積分: {y2_int}") # (D2) 計算結果を表示
print(f"y3 = sin(x) + cos(x) の積分: {y3_int}") # (D3) 計算結果を表示

y1 = x**2 + 1/x の積分: x**3/3 + log(x)
y2 = ln(x) + e^x の積分: x*log(x) - x + exp(x)
y3 = sin(x) + cos(x) の積分: sin(x) - cos(x)
```

プログラムリスト 6.2 SymPyで置換積分・部分積分の結果を確認する

```
In [32]: import sympy as sp          # (A1) SymPyライブラリをimport
x = sp.symbols('x')              # (B1) シンボリック変数xを定義

y1 = (2*x + 3)**4                # (B2) 例1: 置換積分で解ける関数
y2 = x**2*sp.sqrt(x**3+2)        # (B3) 例2: 典型的な置換積分 u=x^3+2
y3 = 3*x*sp.exp(2*x)            # (B4) 例3: 部分積分で解ける関数

y1_int = sp.integrate(y1, x)     # (C1) y1 の不定積分
y2_int = sp.integrate(y2, x)     # (C2) y2 の不定積分
y3_int = sp.integrate(y3, x)     # (C3) y3 の不定積分

simplified_y1_int = sp.simplify(y1_int) # (C4) y1 の不定積分を整理
simplified_y2_int = sp.simplify(y2_int) # (C5) y2 の不定積分を整理
simplified_y3_int = sp.simplify(y3_int) # (C6) y3 の不定積分を整理

# (D) 計算結果を表示
print(f"[1:] y1 = (2*x + 3)**4 の積分: {y1_int}")
print(f"> 式の整理後:\n {simplified_y1_int}")
print(f"[2:] y2 = x**2*sqrt(x**3+2) の積分: {y2_int}")
print(f"> 式の整理後:\n {simplified_y2_int}")
print(f"[3:] y3 = 3*x*exp(2*x) の積分: {y3_int}")
print(f"> 式の整理後:\n {simplified_y3_int}")

[1:] y1 = (2*x + 3)**4 の積分: 16*x**5/5 + 24*x**4 + 72*x**3 + 108*x**2 + 81*x
> 式の整理後:
x*(16*x**4 + 120*x**3 + 360*x**2 + 540*x + 405)/5
[2:] y2 = x**2*sqrt(x**3+2) の積分: 2*x**3*sqrt(x**3 + 2)/9 + 4*sqrt(x**3 + 2)/9
> 式の整理後:
2*(x**3 + 2)**(3/2)/9
[3:] y3 = 3*x*exp(2*x) の積分: (6*x - 3)*exp(2*x)/4
> 式の整理後:
3*(2*x - 1)*exp(2*x)/4
```

プログラムリスト 6.3 SymPyで一次関数とその不定積分を描画する

```
In [33]: import numpy as np          # (A1) numpyライブラリをimport
import matplotlib.pyplot as plt     # (A2) matplotlibライブラリをimport
import sympy as sp                  # (A3) SymPyライブラリをimport
```

```

x = sp.symbols('x') # (B1) シンボリック変数xを定義

def f(x): # (H1) 被積分関数f(x)を定義
    return 1/2 * x

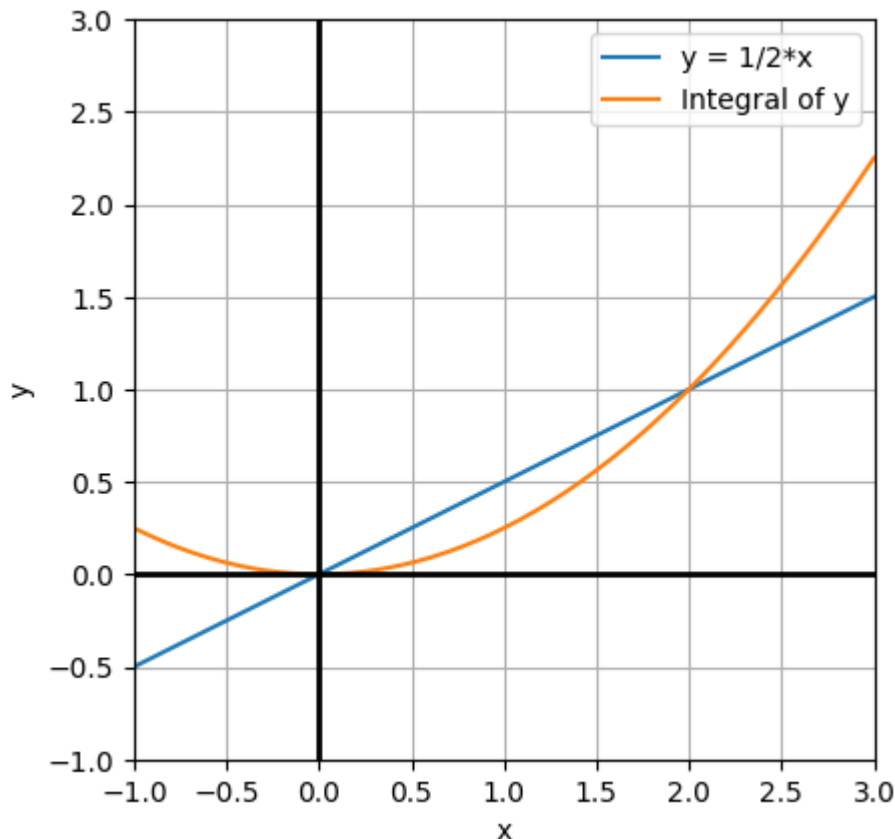
y_int = sp.integrate(f(x), x) # (C1) 関数f(x)の不定積分を計算
print(f"1/2*x の積分: {y_int}") # (D) 積分結果の関数を表示
y_int_np = sp.lambdify(x, y_int, 'numpy') # (C2) numpy形式に変換

x_plot = np.linspace(-1, 3, 200) # (B2) -1~3まで200点を生成
y_plot = f(x_plot) # (C3) 関数f(x)の値を計算
y_int_plot = y_int_np(x_plot) # (C4) 積分関数  $\int y \, dx$  の値を計算
plt.plot(x_plot, y_plot, label="y = 1/2*x") # (E1) 被積分関数の描画
plt.plot(x_plot, y_int_plot, label="Integral of y") # (E2) 積分関数の描画

plt.grid(True) # (F1) グリッド線を表示
plt.xlabel("x") # (F2) x軸ラベルの設定
plt.ylabel("y") # (F3) y軸ラベルの設定
plt.axvline(x=0, color='black', linewidth=2) # (F4) y軸の線を太く表示
plt.axhline(y=0, color='black', linewidth=2) # (F5) x軸の線を太く表示
plt.legend() # (F6) 凡例を表示
plt.legend() # (F6) 凡例を表示
plt.xlim(-1, 3) # (F7) x軸の範囲を設定
plt.ylim(-1, 3) # (F8) y軸の範囲を設定
plt.gca().set_aspect('equal') # (F12) x軸とy軸のスケールを揃える
plt.show() # (G) グラフを表示

```

1/2*x の積分: 0.25*x**2



プログラムリスト 6.4 SymPyを用いて定積分を計算する

```

In [34]: import sympy as sp # (A1) SymPyライブラリをimport
x = sp.symbols('x') # (B1) シンボリック変数xを定義

def f(x): # (H1) 被積分関数 f(x) を定義
    return 1/2 * x

```

```
y_int = sp.integrate(f(x), (x, 1, 2)) # (C1) 区間[1, 2]での定積分を計算
print(y_int)                         # (D1) 結果を表示
```

0.7500000000000000

プログラムリスト 6.5 1から10までの整数を加算する

In [35]:

```
sum = 0 # (C1) 合計値を保存する変数sumを0に初期化
for i in range(1, 11): # (C2) 1から10までの整数を順に取り出す
    sum = i + sum # (C3) sumにiを加算する
print(f"加算合計: {sum}") # (D1) 結果を表示
```

加算合計: 55

プログラムリスト 6.6 矩形近似と台形近似で定積分を近似する

In [36]:

```
import numpy as np # (A1) numpyライブラリをimport

def f(x): # (H1) 被積分関数 f(x) = 1/2 * x を定義
    return 1/2 * x

# 積分区間 [a, b] と分割数 n, 短冊幅 h
a = 1 # (B1) 区間の下限
b = 2 # (B2) 区間の上限
n = 10 # (B3) 分割数
h = (b - a) / n # (C1) 短冊幅

# 矩形近似による積分
sum_rect = 0 # (C1) 合計値を保存する変数sum_rectを0に初期化
for k in range(0, n): # (C2) kを0からn-1まで変化させる
    xk = a + k * h # (C3) 各区間の左端を利用
    sum_rect = sum_rect + f(xk) * h # (C4) 短冊矩形の面積をsum_rectに順次加算

# 台形近似による積分
sum_trap = 0 # (C5) 合計値を保存する変数sum_rectを0に初期化
for k in range(0, n): # (C6) kを0からn-1まで変化させる
    xk_L = a + k * h # (C7) 各区間の左側のx座標
    xk_R = a + (k+1) * h # (C8) 各区間の右側のx座標
    sum_trap = sum_trap + (f(xk_L) + f(xk_R)) * h * 1/2 # (C9) 台形の面積をsum_rectに順次加算

print(f"矩形近似による積分値: {sum_rect:.5f}") # (D1) 矩形近似による結果を出力
print(f"台形近似による積分値: {sum_trap:.5f}") # (D2) 台形近似による結果を出力
```

矩形近似による積分値: 0.72500

台形近似による積分値: 0.75000

プログラムリスト 6.7 フーリエ級数展開を用いた方形波近似

In [37]:

```
import numpy as np # (A1) numpyライブラリをimport
import matplotlib.pyplot as plt # (A2) matplotlibライブラリをimport
x = np.linspace(0, 3*np.pi, 500) # (B1) xの範囲を0から3πまで500点で生成

square_wave = np.sign(np.sin(x)) # (C1) 周期2πの方形波を定義
y = 4/np.pi * (np.sin(x) # (C2) フーリエ級数展開(1, 3, 5次まで)
                + 1/3*np.sin(3*x)
                + 1/5*np.sin(5*x))

plt.plot(x, square_wave, label="Square Wave") # (E1) 元の方形波を描画
plt.plot(x, y, label="Approx (N=5)") # (E2) フーリエ級数近似を描画

plt.grid(True) # (F1) グリッド線を表示
plt.xlabel("x") # (F2) x軸ラベルの設定
```

```

plt.ylabel("y") # (F3) y軸ラベルの設定
plt.axhline(0, color='black', linewidth=2) # (F4) x軸線を太く表示
plt.axvline(0, color='black', linewidth=2) # (F5) x軸の線を太く表示
plt.legend() # (F6) 凡例を表示
plt.ylim(-1.3, 1.3) # (F8) y軸の範囲
ticks = np.arange(0, 3*np.pi + np.pi, np.pi) # (F9) x軸目盛の範囲
labels = [ "0", "pi", "2pi", "3pi"] # (F10) x軸目盛にpi(π)表記を使用
plt.xticks(ticks, labels) # (F11) x軸をπで表示
plt.show() # (G) グラフを表示

```

