

プログラムリスト 3.1 グラフ描画のための配列を生成するPythonプログラム

```
In [8]: import numpy as np          # (A1) numpyライブラリをimport
x = np.linspace(-2, 2, 5)        # (B1) xの範囲を-2から2まで5点で定義
y = 2*x                          # (C1) 関数  $y = 2^x$  を定義
print("x=", x)                  # (D1) 配列xの中身を出力
print("y=", y)                  # (D1) 配列yの中身を出力
```

```
x= [-2. -1.  0.  1.  2.]
y= [-4. -2.  0.  2.  4.]
```

プログラムリスト 3.2 指数関数のグラフを描く Python プログラム

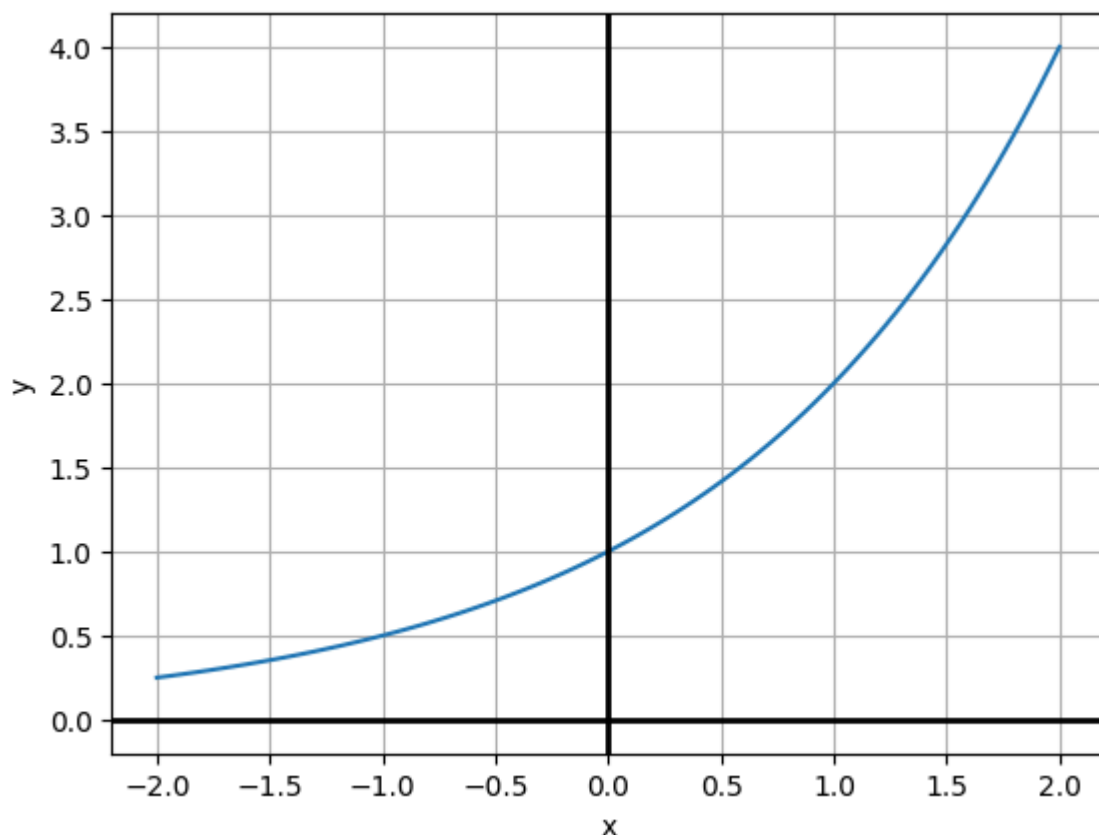
```
In [9]: import numpy as np          # (A1) numpyライブラリをimport
import matplotlib.pyplot as plt   # (A2) matplotlibライブラリをimport

x = np.linspace(-2, 2, 100)       # (B1) xの範囲を-2から2まで100点で定義
y = 2**x                          # (C1) 関数  $y = 2^x$  を定義

plt.plot(x, y)                   # (E1) グラフを描画

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.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.show()                       # (G) グラフを表示
```



プログラムリスト 3.3 複数の指数関数を描画する Python プログラム

```
In [10]: 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
```

```

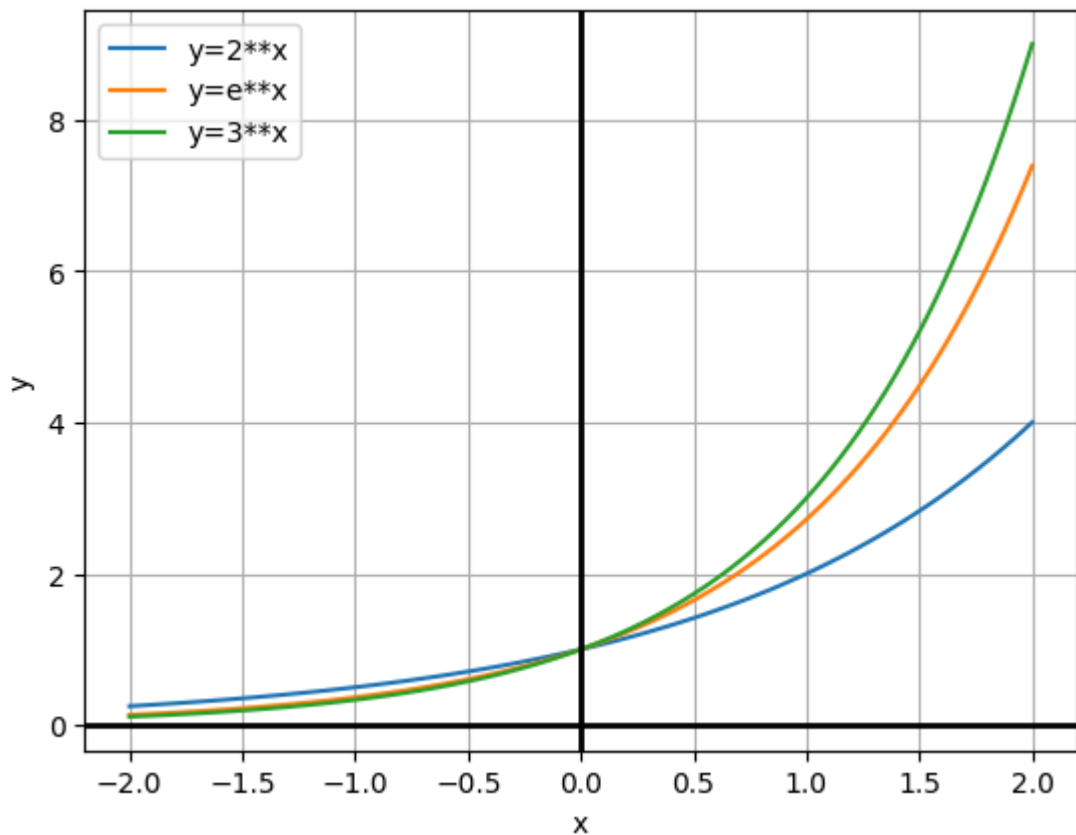
x = np.linspace(-2, 2, 100)      # (B1) xの範囲を-2から2まで100点で定義
y1 = 2**x                        # (C1) 関数  $y = 2^x$  を定義
y2 = np.exp(x)                   # (C2) 関数  $y = e^x$  を定義
y3 = 3**x                        # (C3) 関数  $y = 3^x$  を定義

plt.plot(x, y1, label="y=2**x")  # (E1)  $y = 2^x$  のグラフを描画
plt.plot(x, y2, label="y=e**x")  # (E2)  $y = e^x$  のグラフを描画
plt.plot(x, y3, label="y=3**x")  # (E3)  $y = 3^x$  のグラフを描画

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.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.show()                       # (G) グラフを表示

```



In [20]:

```

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

#変更部分始まり
x = np.linspace(-2, 0.5, 100)    # (B1) xの範囲を-2から2まで100点で定義
#変更部分終わり

y1 = 2**x                        # (C1) 関数  $y = 2^x$  を定義
y2 = np.exp(x)                   # (C2) 関数  $y = e^x$  を定義
y3 = 3**x                        # (C3) 関数  $y = 3^x$  を定義

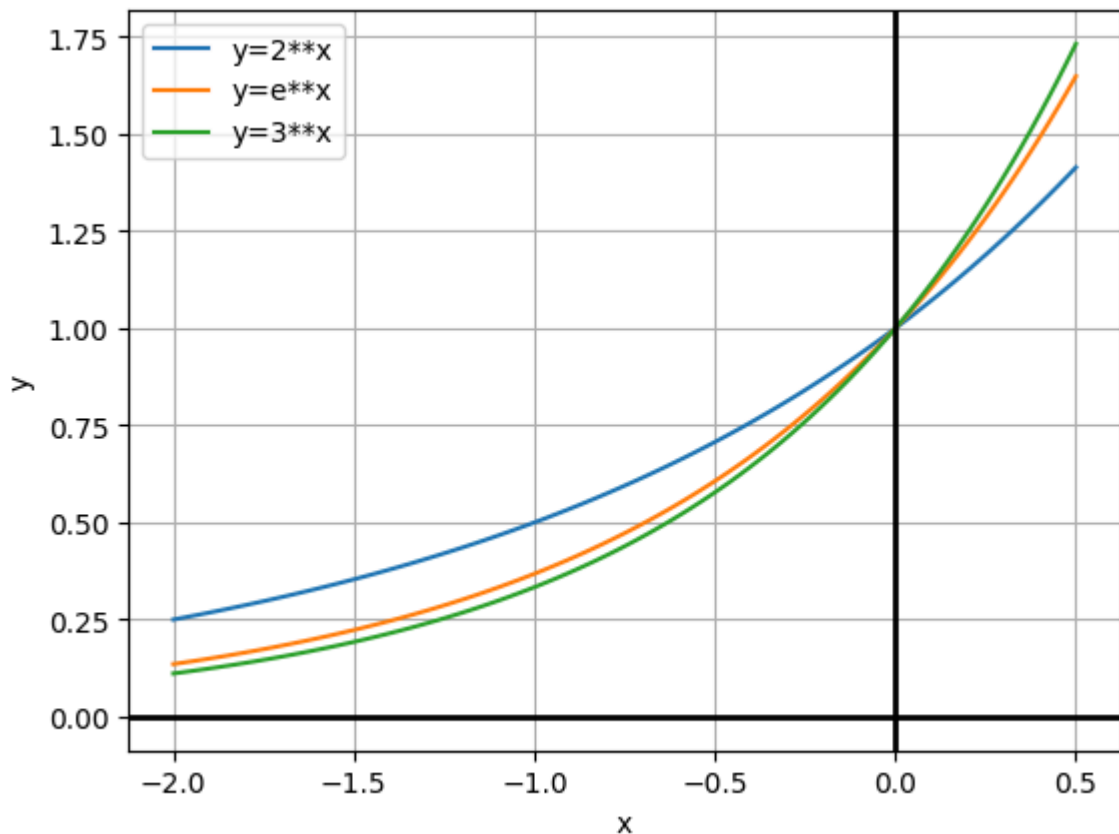
plt.plot(x, y1, label="y=2**x")  # (E1)  $y = 2^x$  のグラフを描画
plt.plot(x, y2, label="y=e**x")  # (E2)  $y = e^x$  のグラフを描画
plt.plot(x, y3, label="y=3**x")  # (E3)  $y = 3^x$  のグラフを描画

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.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.show() # (G) グラフを表示
```



プログラムリスト 3.4 複数の対数関数を描画する Python プログラム

In [12]:

```
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

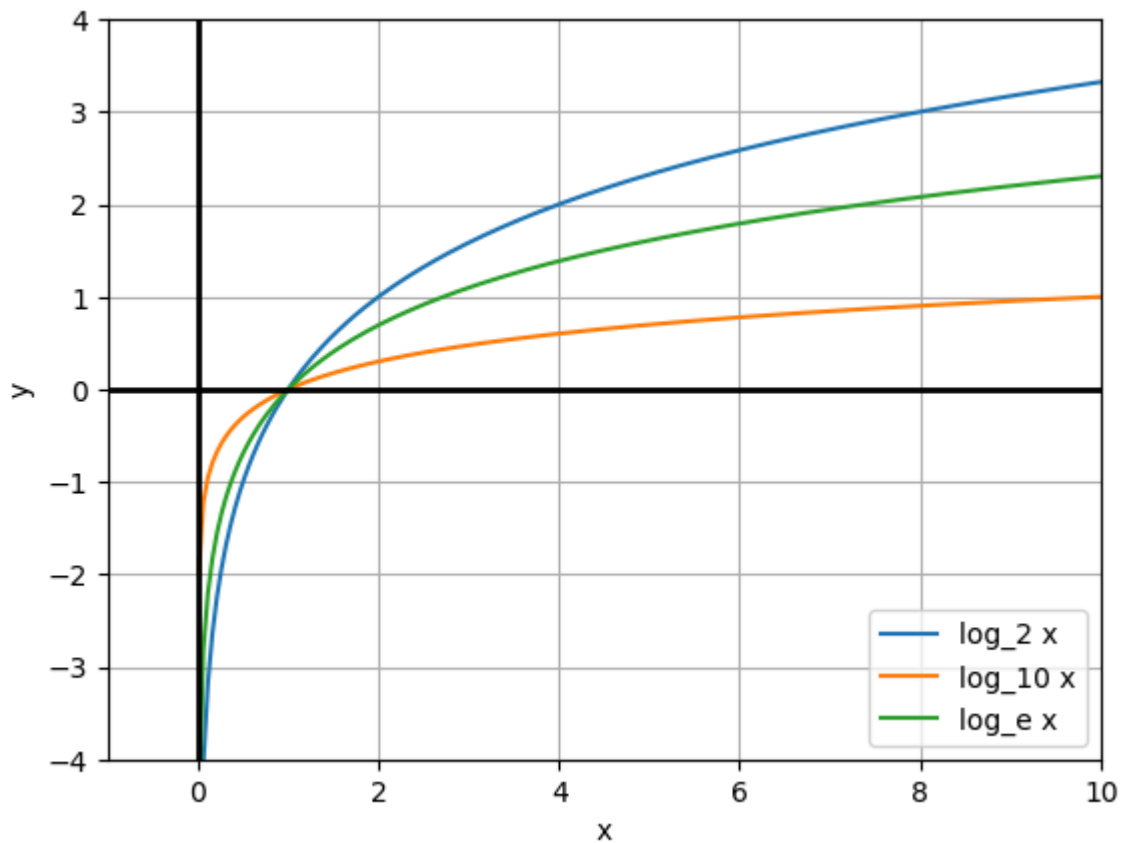
x = np.linspace(0.01, 10, 200) # (B1) xの範囲を0.01から10まで200点で定義 (0は不可)

y1 = np.log2(x) # (C1) 関数 y = log_2 x を定義
y2 = np.log10(x) # (C2) 関数 y = log_10 x を定義
y3 = np.log(x) # (C3) 関数 y = log_e x (自然対数) を定義

plt.plot(x, y1, label="log_2 x") # (E1) y = log_2 x のグラフを描画
plt.plot(x, y2, label="log_10 x") # (E2) y = log_10 x のグラフを描画
plt.plot(x, y3, label="log_e x") # (E3) y = log_e x のグラフを描画

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.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.xlim(-1, 10) # (F7) x軸の表示範囲を設定
plt.ylim(-4, 4) # (F8) y軸の表示範囲を設定

plt.show() # (G) グラフを表示
```



プログラムリスト 3.5 複数の指数関数を片対数グラフに描画する Python プログラム

In [13]:

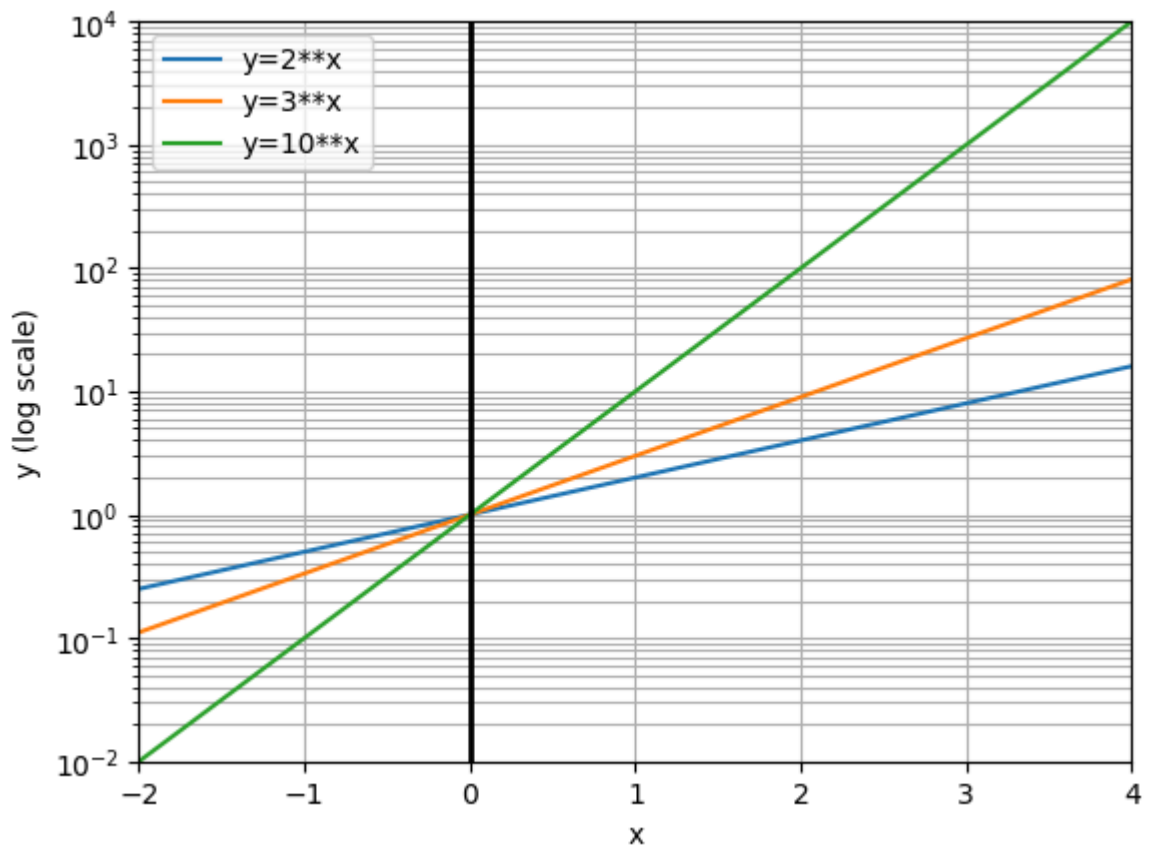
```
import numpy as np          # (A1) numpyライブラリをimport
import matplotlib.pyplot as plt # (A2) matplotlibライブラリをimport

x = np.linspace(-2, 4, 200) # (B1) xの範囲を-2から4まで200点で定義

y1 = 2**x                   # (C1) 関数  $y = 2^x$  を定義
y2 = 3**x                   # (C2) 関数  $y = 3^x$  を定義
y3 = 10**x                  # (C3) 関数  $y = 10^x$  を定義

plt.semilogy(x, y1, label="y=2**x") # (E1)  $y = 2^x$  の片対数グラフを描画
plt.semilogy(x, y2, label="y=3**x") # (E2)  $y = 3^x$  の片対数グラフを描画
plt.semilogy(x, y3, label="y=10**x") # (E3)  $y = 10^x$  の片対数グラフを描画

plt.grid(True, which='both') # (F1) 主/補助目盛にグリッドを描画
plt.xlabel("x")              # (F2) x軸ラベルを設定
plt.ylabel("y (log scale)")  # (F3) y軸ラベルを設定
plt.axvline(x=0, color='black', linewidth=2) # (F4) y軸の線を太く表示
plt.legend()                 # (F6) 凡例を表示
plt.xlim(-2, 4)              # (F7) x軸の範囲を設定
plt.ylim(10**-2, 10**4)      # (F8) y軸の範囲を設定 (対数目盛)
plt.show()                   # (G) グラフを表示
```



プログラムリスト 3.6 べき乗と平方根関数を片対数・両対数グラフに描画するPythonプログラム

```
In [14]: 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(0.1, 10, 200)         # (B1) xの範囲を0.1から10まで200点で定義 (sqrt(x)に備え0は過

y1 = x**2                             # (C1) 関数 y = x^2 を定義
y2 = np.sqrt(x)                       # (C2) 関数 y = sqrt(x) を定義

# 片対数グラフ (y軸対数)
plt.semilogy(x, y1, label="y=x**2")    # (E1) y = x^2 を片対数で描画
plt.semilogy(x, y2, label="y=sqrt(x)") # (E2) y = sqrt(x) を片対数で描画

plt.grid(True, which='both')           # (F1) 主/補助目盛にグリッドを描画
plt.xlabel("x")                        # (F2) x軸ラベルを設定
plt.ylabel("y (log scale)")            # (F3) y軸ラベルを設定
plt.legend()                           # (F6) 凡例を表示
plt.xlim(0.1, 10)                     # (F7) x軸の範囲を設定
plt.ylim(10**-1, 10**2)               # (F8) y軸の範囲を設定 (対数目盛)
plt.show()                             # (G) グラフを表示

# 両対数グラフ (x, y軸とも対数)
plt.loglog(x, y1, label="y=x**2")      # (E1) y = x^2 を両対数で描画
plt.loglog(x, y2, label="y=sqrt(x)")   # (E2) y = sqrt(x) を両対数で描画

plt.grid(True, which='both')           # (F1) 主/補助目盛にグリッドを描画
plt.xlabel("x (log scale)")            # (F2) x軸ラベルを設定
plt.ylabel("y (log scale)")            # (F3) y軸ラベルを設定
plt.legend()                           # (F6) 凡例を表示
plt.xlim(0.1, 10)                     # (F7) x軸の範囲を設定
plt.ylim(10**-1, 10**2)               # (F8) y軸の範囲を設定 (対数目盛)
plt.show()                             # (G) グラフを表示
```

