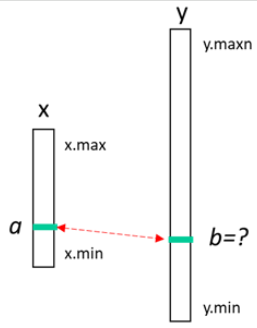
Interpolation

回顧過去使用 Python 程式碼開發的眾多工具，推薦一個既常用又實用的神奇工具：`np.interp`。

如圖所示，傳統上，我們可以利用比例關係來線性縮放純量或向量。例如，可以計算將 a 投射到 b 的值，或反過來從 b 投射到 a 的值。



(b-y.min)/(y.max-y.min) = (a-x.min)/(x.max-x.min)
b = (a-x.min)*(y.max-y.min)/(x.max-x.min) + y.min

本範例展示了 `np.interp` 函數的多種應用，這是 NumPy 庫中一個非常實用的插值工具。

比例縮放（Scale Conversion）

np.interp(<*predict_points*>, <*source_range*>, <*target_range*>)

如前所述，線性縮放純量或向量的操作可以通過 `np.interp` 來實現。而需要計算的資料點可以是純量，也可以是向量。

```
np.interp(0.5, (0, 1), (df['SIDD'].min(), df['SIDD'].max())) # scalar conversion

np.interp((0.1, 0.5, 0.9), (0, 1), (df['SIDD'].min(), df['SIDD'].max())) # vector conversion
```

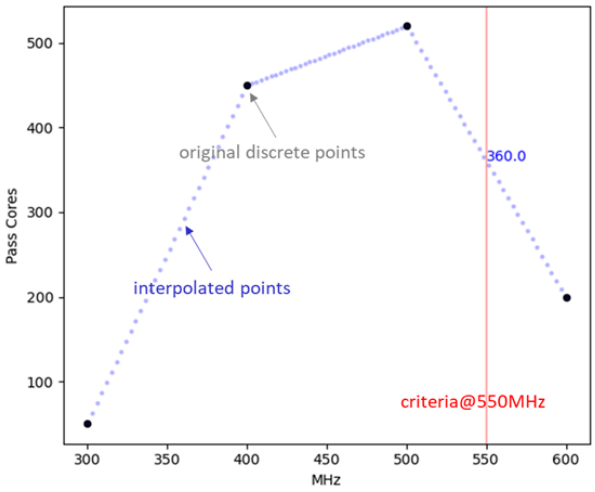
歸一化（Normalization）

運用相同原理，將目標數值範圍替換為 [0, 1]，即可實現數值的 [0, 1] 歸一化。

```
np.interp(1, (df['SIDD'].min(), df['SIDD'].max()), (0, 1)) # normalize to (0, 1)

np.interp((1, 3, 5), (df['SIDD'].min(), df['SIDD'].max()), (0, 1)) # normalize to (0, 1)
```

從離散點到分段線性（From Discrete Points to Piecewise Linear）

在許多實際案例中，我們經常需要根據手頭的離散數據推估未知區間的關係，例如假設線性關係並進行插值預測。如下圖所示，假設有一個多核心計算機的例子，當我們需要判定 550MHz 時的 Pass Core 數量，而測試樣本又不存在時。

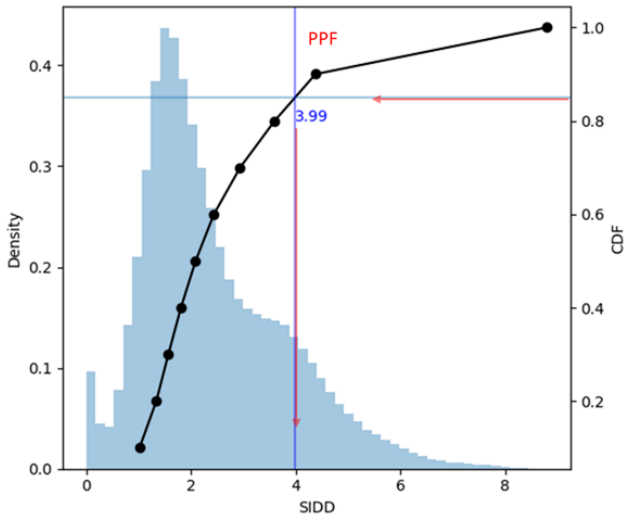


np.interp(<*predict_points*>, <*source_points*>, <*target_points*>)

```
x = np.array([300, 400, 500, 600])
y = np.array([50,  450, 520, 200])

tx = np.linspace(x.min(), x.max(), 100)
ty = np.interp(tx, x, y)

p = np.interp(550, x, y) # predict pass core# at 550MHz

plt.figure(figsize=(6,5))
plt.scatter(x, y, c='k', s=20)
plt.scatter(tx, ty, c='b', s=5, alpha=0.2)
plt.axvline(550, c='r', alpha=0.3)
plt.text(550, p, f'{p}', c='b')
plt.xlabel('MHz')
plt.ylabel('Pass Cores')
plt.tight_layout()
```

反函數（Inverse Function, PPF）

我們對數據的機率密度函數 (PDF, Probability Density Function) 積分可得累積分佈函數 (CDF, Cumulative Distribution Function)。當需要從機率值反推對應數值時，即為 PPF (Percent-Point Function) 的實作。



np.interp(<*predict_points*>, <*y_points*>, <*x_points*>)

```python
x = np.array([1.02, 1.34, 1.57, 1.81, 2.09, 2.44, 2.94, 3.6 , 4.38, 8.8 ])
y = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])

p = np.interp(0.85, y, x) # predict SIDD at CDF=0.5

plt.figure(figsize=(6,5))
ax1 = plt.subplot(111)
ax1.hist(df['SIDD'], bins=50, density=True, alpha=0.4)
ax1.set_xlabel('SIDD')
ax1.set_ylabel('Density')
ax2 = plt.twinx(ax1)
ax2.set_ylabel('CDF')
ax2.plot(x, y, c='k', marker='o')
ax2.axhline(0.85, alpha=0.4)
ax2.axvline(p, alpha=0.4, c='b')
ax2.text(p, 0.8, f'{p:.2f}', c='b')
plt.tight_layout()
```