

ゼロからわかるソートアルゴリズム

初心者のための図解：仕組みからPythonコードまで



ソート（並び替え）はプログラミングの基本です

データ分析、検索、アプリ開発など、あらゆる場面で「データを順番通りに整理する」処理が必要になります。本資料では、この仕組みの基本から実際のPythonコードまで、ブロックを動かす感覚で視覚的に理解していきましょう。



Sorting



アプローチ1：最小値を探して「入れ替える」



リスト全体の中から、最小値を探します。



見つかった最小値と、未整列部分の先頭の要素を入れ替えます。



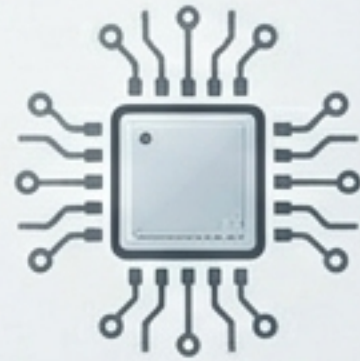
「最小値が確定した部分」を一つ右へ進め、残りの部分で再び最小値を探します。これを未整列の要素がなくなるまで繰り返します。

このアルゴリズムの「特徴と性能」



時間計算量: $O(n^2)$

どんなデータ状態であっても、必ず最後まで比較を行うため、常にこの時間がかかります。データの量が増えると時間がかかるタイプです。



空間計算量: $O(1)$

処理のために追加のメモリをほとんど使いません。



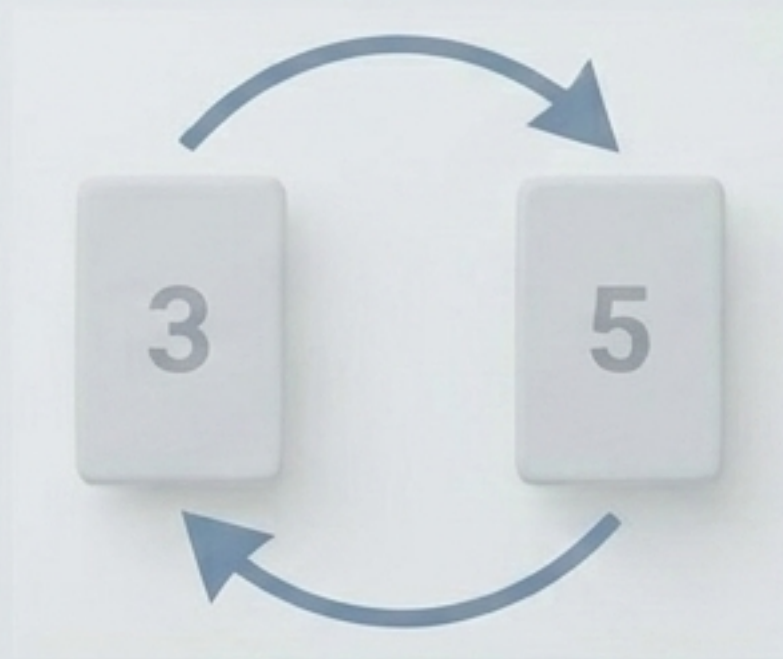
安定性: 不安定なソート

離れた場所にある要素同士を入れ替えるため、同じ値の要素の前後関係が崩れる可能性があります。

もう一つのアプローチ：「ずらして挿入する」

先ほどは「探して入れ替える」仕組みを見ましたが、ここからはもう一つの代表的な手法である挿入ソート（Insertion Sort）のPythonコードを見てみましょう。要素を右に「スライド」させ、正しい位置に「挿入」していくのが特徴です。

Selection（入れ替え）



Insertion（スライドと挿入）



Pythonでの実装全体像

```
def insertion_sort(arr):  
    n = len(arr)  
    for i in range(1, n):  
        key = arr[i]  
        j = i - 1  
        while j >= 0 and arr[j] > key:  
            arr[j + 1] = arr[j]  
            j -= 1  
        arr[j + 1] = key  
    return arr
```

たった十数行のコードで
実現できます！

ステップ1：挿入する値を取り出す

```
def insertion_sort(arr):  
    n = len(arr)  
    for i in range(1, n):  
        key = arr[i]  
        j = i - 1  
        while j >= 0 and arr[j] > key:  
            arr[j + 1] = arr[j]  
            j -= 1  
        arr[j + 1] = key  
    return arr
```

2番目の要素（インデックス1）からスタート。

挿入する値を取り出し、一時的に保存します。

比較対象となる左隣の要素のインデックスをセットします。



ステップ2：右へずらして、正しい位置へ挿入

```
def insertion_sort(arr):  
    n = len(arr)  
    for i in range(1, n):  
        key = arr[i]  
        j = i - 1  
        while j >= 0 and arr[j] > key:  
            arr[j + 1] = arr[j]  
            j -= 1  
        arr[j + 1] = key  
    return arr
```



keyより大きい要素を見つけたら...

その要素を一つ後ろ（右）にずらしていきます。

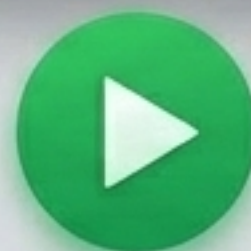
最終的に空いた正しい場所に、一時保存していたkeyを挿入します。

実際のデータでコードを動かしてみよう

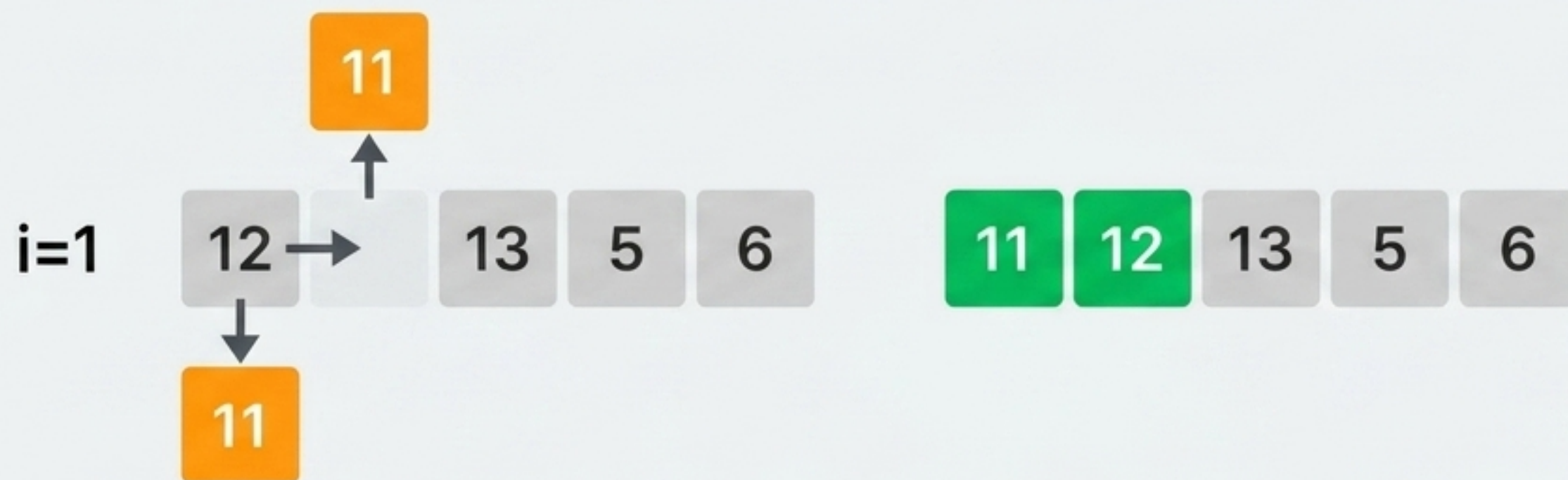
仕組みがわかったところで、実際のプログラムがどのように動くのか、ステップ・バイ・ステップで追ってみましょう。

12 11 13 5 6

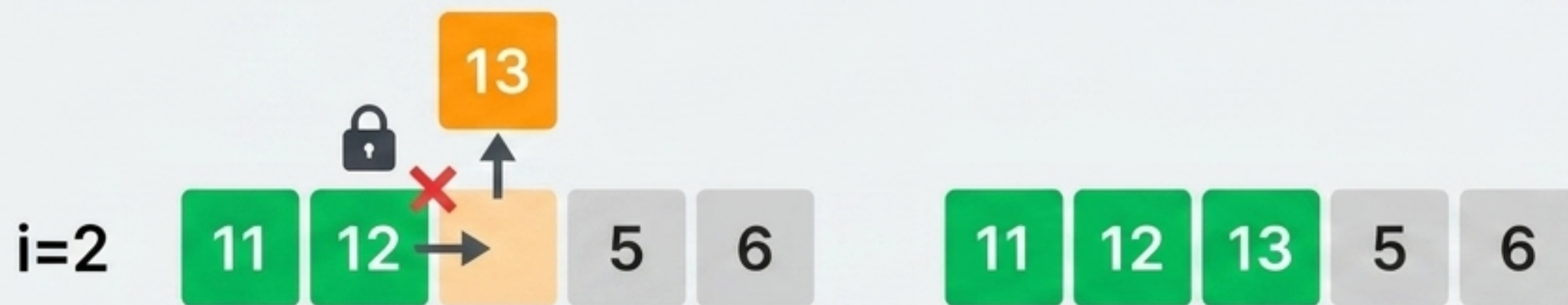
```
data = [12, 11, 13, 5, 6]  
sorted_data = insertion_sort(data)
```



トレース前半：近隣の整列



11を取り出す。12の方が大きいので右へずらし、11を先頭に挿入。

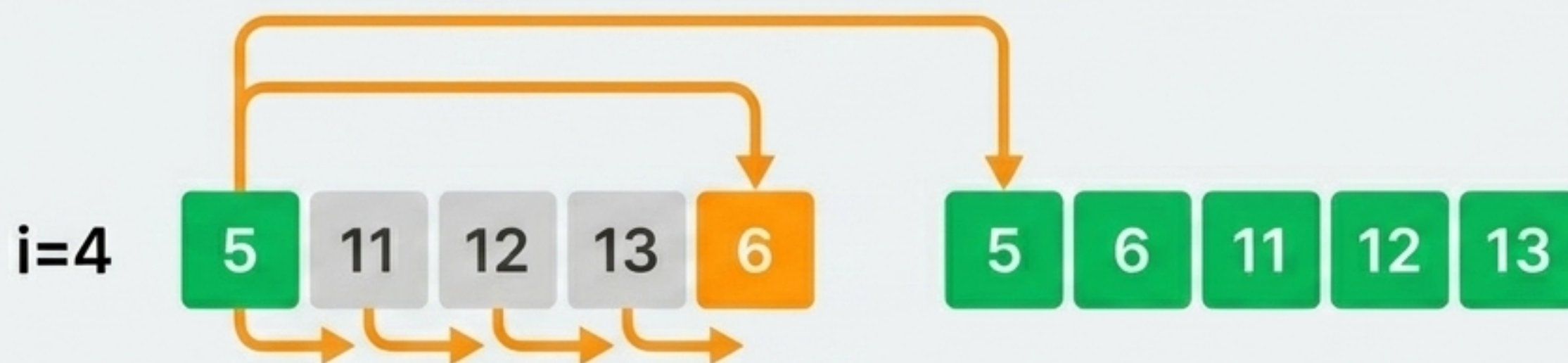


13を取り出す。左隣の12より大きいので、ずらさずにそのまま位置確定。

トレース後半：一気に前へスライド



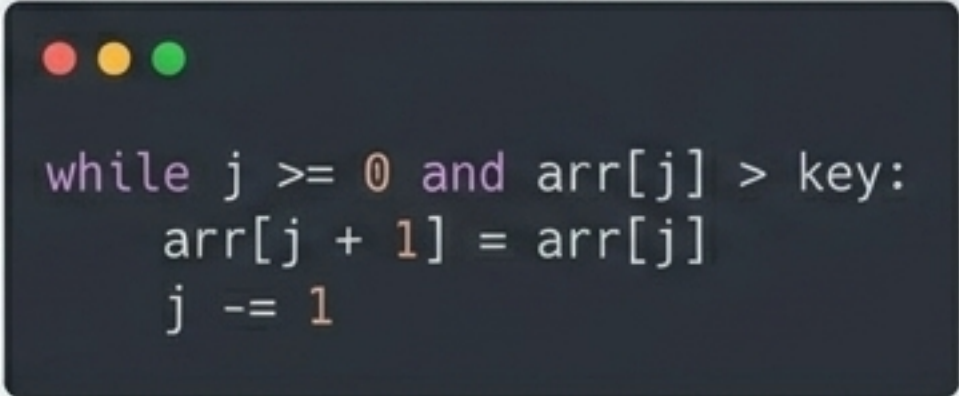
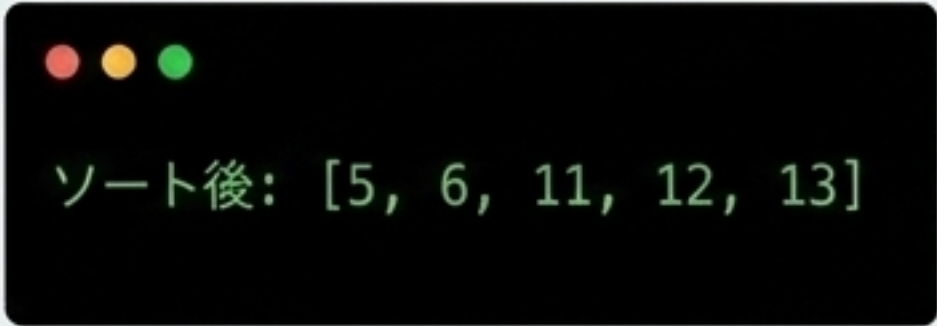
5を取り出す。自分より大きい要素を全て右へずらし、一気に先頭へ挿入。



6を取り出す。同様に大きい要素をずらし、5の隣に挿入。

5や6のような小さい数字は、空いたスペースを滑るように前の方へ進んでいきます。

ソートアルゴリズムの学習まとめ

アプローチ (Approach)	実装 (Implementation)	性能 (Performance)
		
未整列の部分から適切な位置を探し出し、ブロックを物理的に整理する。	whileループを使った「ずらし」と「挿入」の処理がカギ。	$O(n^2)$ 。シンプルだがデータ量には注意が必要。

仕組み（図）とコード（Python）を結びつけることで、アルゴリズムは確実に理解できます！