

予想してから動かす：問1・問2

同じ入力を毎回作り直し、元リストと戻り値を分けて記録してください。

問1 sorted() を別変数へ

```
durations = [25, 10, 25, 5]
result = sorted(durations)
print('original:', durations)
print('return:', result)
```

元リストと戻り値は、それぞれ何と表示されますか。

予想： _____

実行： _____

理由： _____

問2 sort() を別変数へ

```
durations = [25, 10, 25, 5]
result = durations.sort()
print('original:', durations)
print('return:', result)
```

問1と違うのは2行目だけ。2つの表示結果を予想してください。

予想： _____

実行： _____

理由： _____

よくある代入と受付順：問3・問4

問3の修正例は、失敗した変数を再利用せず新しい入力から試します。

問3 durations = durations.sort() を直す

```
durations = [25, 10, 25, 5]
durations = durations.sort()
print(durations)
```

出力は何ですか。元順が必要な場合と不要な場合の直し方も考えます。

予想： _____

実行： _____

理由： _____

問4 受付順を保って別一覧を作る

```
jobs = [('R1',25), ('R2',10), ('R3',25), ('R4',5)]
ordered = sorted(jobs, key=lambda item: item[1])
print('original:', jobs)
print('ordered:', ordered)
```

元のID順と、所要時間順のID順をそれぞれ書いてください。

予想： _____

実行： _____

理由： _____

同順位と空リスト：問5・問6

同じ25分のR1・R3の順と、戻り値の型に注目します。

問5 reverse=True で同順位も逆転する？

```
jobs = [('R1',25), ('R2',10), ('R3',25), ('R4',5)]
ordered = sorted(jobs, key=lambda x: x[1], reverse=True)
print('original:', jobs)
print('ordered:', ordered)
```

結果のID順を予想。25分のR1とR3はどう並びますか。

予想： _____

実行： _____

理由： _____

問6 空でも戻り値を区別

```
first=[]; a=sorted(first)
second=[]; b=second.sort()
print(first, a)
print(second, b)
```

元リスト2つと戻り値2つを予想してください。

予想： _____

実行： _____

理由： _____

説明練習：元順を残す理由と、同じキーの相対順を2文で説明しましょう。

解答・理由：問1～問3

実行結果だけでなく、英語公式仕様とどこが対応するか確認してください。

問1 新しいリストが戻る

```
original: [25, 10, 25, 5]
return:   [5, 10, 25, 25]
```

`sorted()` は新しいリストを返し、元のリストは受付順のままです。公式仕様の `new sorted list` に対応します。

問2 元リストを変更し、戻り値は None

```
original: [5, 10, 25, 25]
return:   None
```

`list.sort()` は in-place で並べ替え、returns None。Noneはこの操作の失敗を意味しません。

問3 誤代入の修正

```
print(durations) -> None
元順を残す: ordered = sorted(新しいdurations)
元順が不要: 新しいdurations.sort() (代入しない)
```

右辺の `sort()` が元リストを変更し、返した None が左辺へ代入されます。失敗後の None を再利用せず、どちらの修正も新しい入力から始めます。

解答・理由：問4～問6

同順位の理由をID順ではなく入力内の相対順で説明できれば合格です。

問4 元順と別一覧

元: R1 -> R2 -> R3 -> R4

結果: R4 -> R2 -> R1 -> R3

sorted() を選ぶと元のjobsを残せます。25分の2件は入力内と同じR1 -> R3の順です。

問5 降順でも同順位は安定

reverse=True: R1 -> R3 -> R2 -> R4

昇順を全体反転: R3 -> R1 -> R2 -> R4

公式Sorting HOWTOのOdds and Endsは、reverseでも安定性が保たれると説明しています。

問6 空リストの戻り値

sorted([]): 元=[] / 戻り値=[]

[].sort(): 元=[] / 戻り値=None

空でも戻り値の仕様は変わりません。

出典と検証環境

Python公式 <https://docs.python.org/3/howto/sorting.html#sorting-basics>

Python公式 <https://docs.python.org/3/library/functions.html#sorted>

Python公式 <https://docs.python.org/3/library/stdtypes.html#list.sort>

安定性 <https://docs.python.org/3/howto/sorting.html#sort-stability-and-complex-sorts>

降順 <https://docs.python.org/3/howto/sorting.html#odds-and-ends>

資料表示版 3.14.8 / 実行環境 3.12.14 / 参照日 2026-10-08