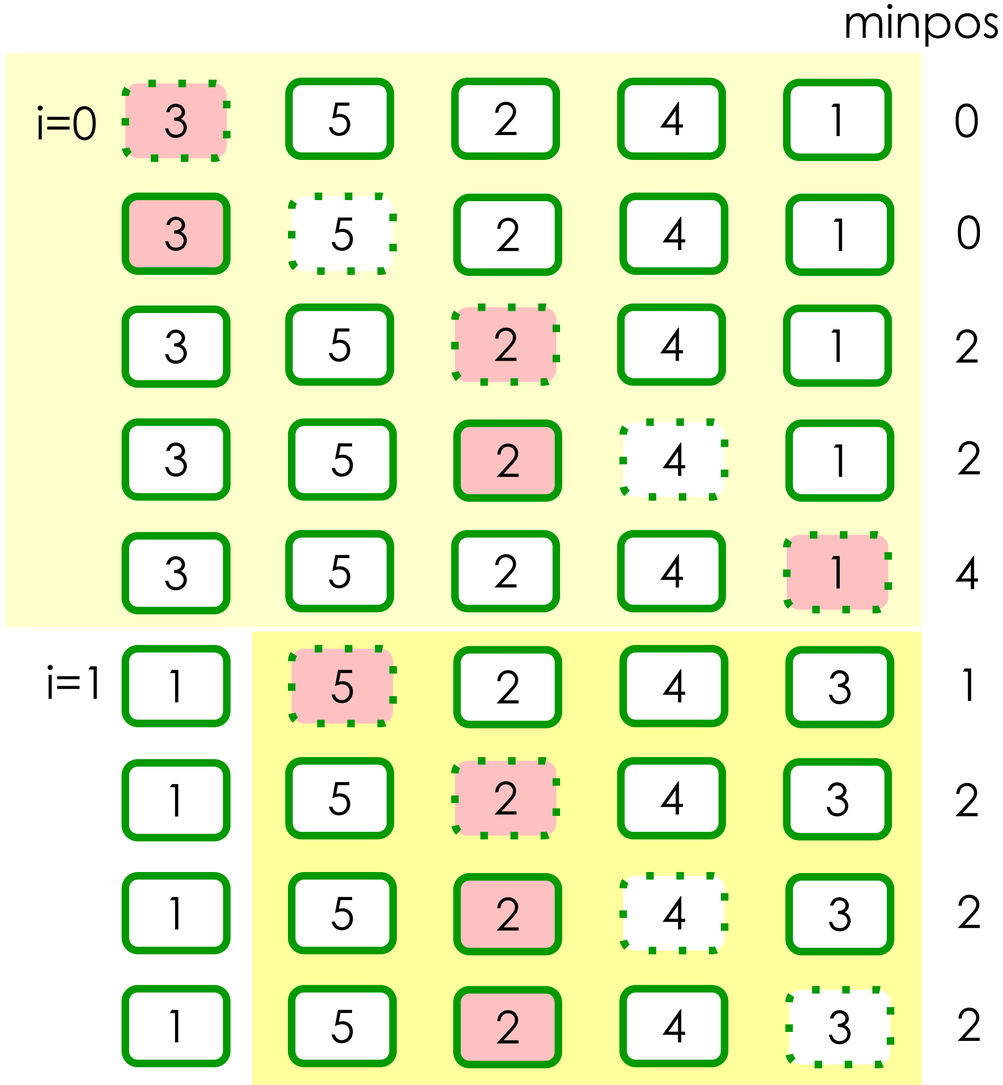
# 選択ソートを実装してみよう

# 選択ソート(Selection Sort)とは

- リストを先頭から最小の数を探し、ソート済みのグループに追加して順番に並び替えるアルゴリズム
- 特徴
  - 比較回数はn(n-1) / 2回，計算量はO($n^2$)

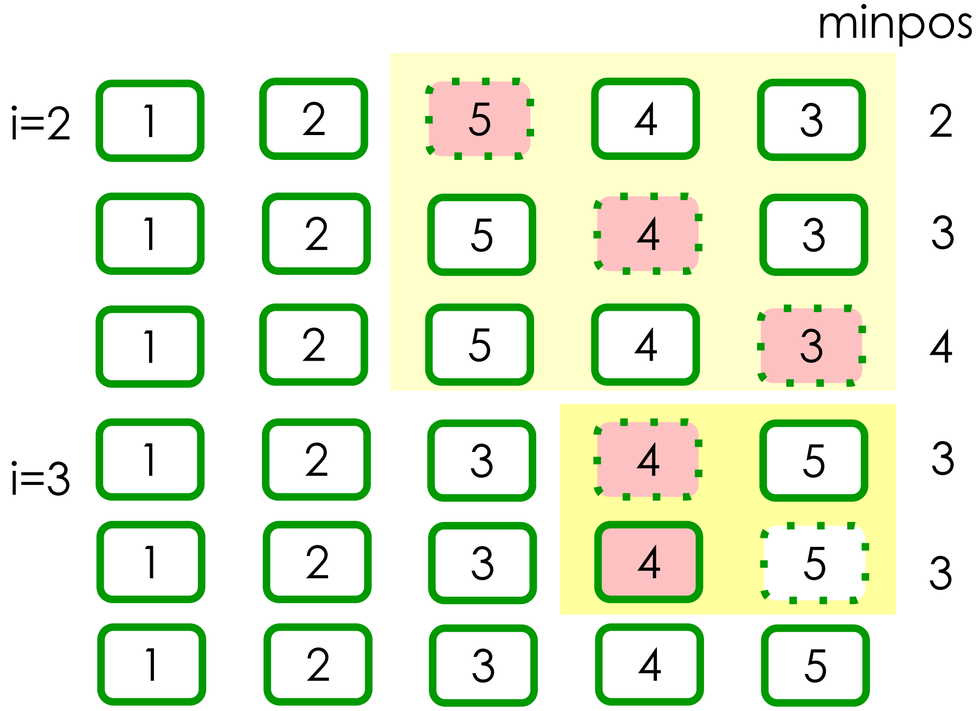
# （再掲）選択ソート（1/2）

- きちんと書くと、次のような手順となります
  1. リストの先頭から順に、要素を比較し、最も小さい要素がある場所（リストのインデックス）を求める
  2. 先頭の要素と最も小さい要素を入れ替える（リストの先頭は一番小さな要素となる）
  3. リストの先頭を除いた要素に対して、1からの手順を適用する
- 最小の要素を選択するので「選択ソート」と、呼ばれます



# （再掲）選択ソート（2/2）

- きちんと書くと、次のような手順となります
  1. リストの先頭から順に、要素を比較し、最も小さい要素がある場所（リストのインデックス）を求める
  2. 先頭の要素と最も小さい要素を入れ替える（リストの先頭は一番小さな要素となる）
  3. リストの先頭を除いた要素に対して、1からの手順を適用する
- 最小の要素を選択するので「選択ソート」と、呼ばれます



# （再掲）Python で定義してみると…

```
def selection_sort(lst):
    length = len(lst)
    for i in range(length):
        minpos = i
        for j in range(i + 1, length):
            if lst[j] < lst[minpos]:
                minpos = j
        if i != minpos:
            tmp = lst[i]
            lst[i] = lst[minpos]
            lst[minpos] = tmp
```

リストの先頭から要素を順に取り出します
この時、取り出したi番目の要素を、暫定の最小値とします

リストのi番目以降の要素を順にチェックしていき、最小値がどこにあるかを探します

最小値が見つかった場合には、i番目の要素と入れ替えます

```
lst = [3, 2, 5, 1, 4]
selection_sort(lst)
lst
```

```
[1, 2, 3, 4, 5]
```

まだ実装できていない場合は
この演習時間内に実装しましょう！

# 選択ソートのプログラムを見る: Pythontutorで動かす

- 選択ソートを実行する selection_sort() 関数をPythontutor で見てみましょう。
- どのような動作をするか確認してみましょう。
  - xs はリスト
  - xs[i] は i 番目の要素

```
def selection_sort(lst):
    length = len(lst)
    for i in range(length):
        minpos = i
        for j in range(i + 1, length):
            if lst[j] < lst[minpos]:
                minpos = j
        if i != minpos:
            tmp = lst[i]
            lst[i] = lst[minpos]
            lst[minpos] = tmp
```

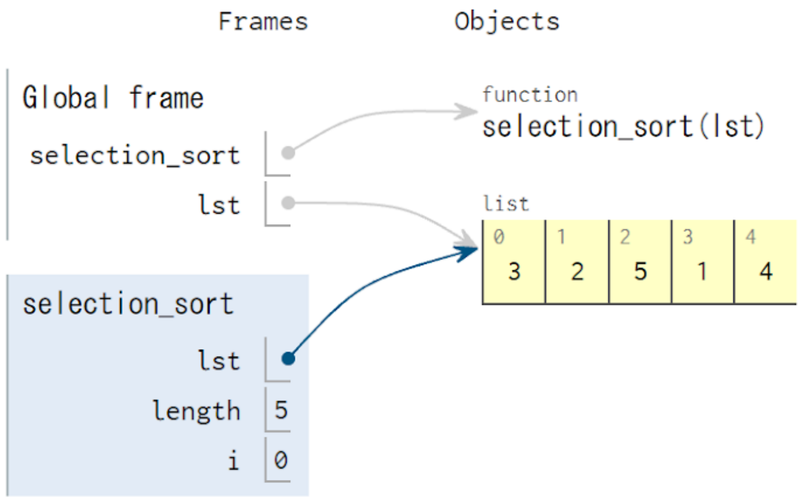
```
lst = [3, 2, 5, 1, 4]
selection_sort(lst)
lst
```

```
[1, 2, 3, 4, 5]
```

# 3 行目

Python 3.6
(known limitations)

```
1  def selection_sort(lst):
2      length = len(lst)
3      for i in range(length):
4          minpos = i
5          for j in range(i + 1, length):
6              if lst[j] < lst[minpos]:
7                  minpos = j
8          if i != minpos:
9              tmp = lst[i]
10             lst[i] = lst[minpos]
11             lst[minpos] = tmp
12
13 lst = [3, 2, 5, 1, 4]
14 selection_sort(lst)
15 lst
```

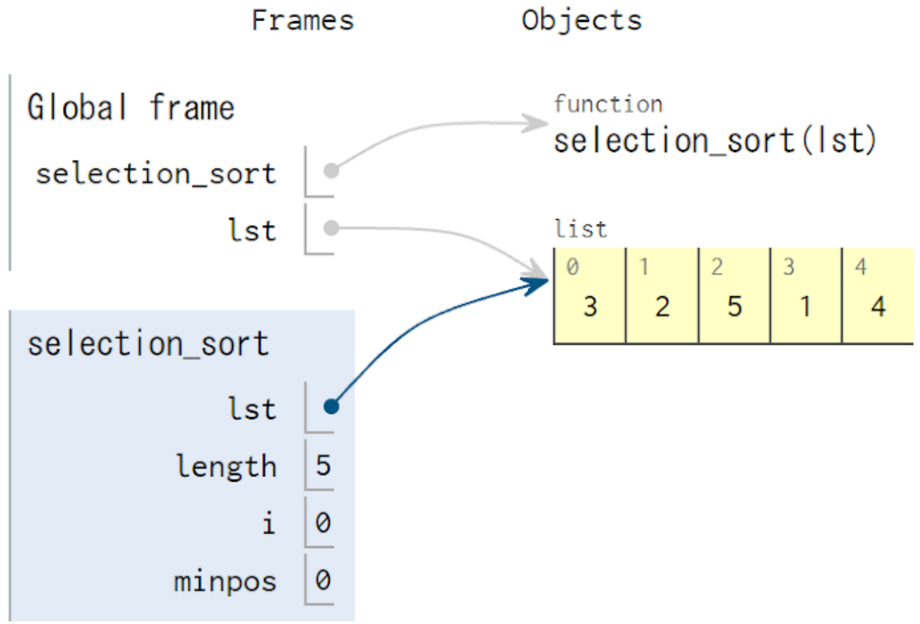



- `for i in range(length):`
  リストの要素を1つずつ調べるためのループです。

# 4ー5行目

Python 3.6
(known limitations)

```
1  def selection_sort(lst):
2      length = len(lst)
3      for i in range(length):
4          minpos = i
5          for j in range(i + 1, length):
6              if lst[j] < lst[minpos]:
7                  minpos = j
8          if i != minpos:
9              tmp = lst[i]
10             lst[i] = lst[minpos]
11             lst[minpos] = tmp
```

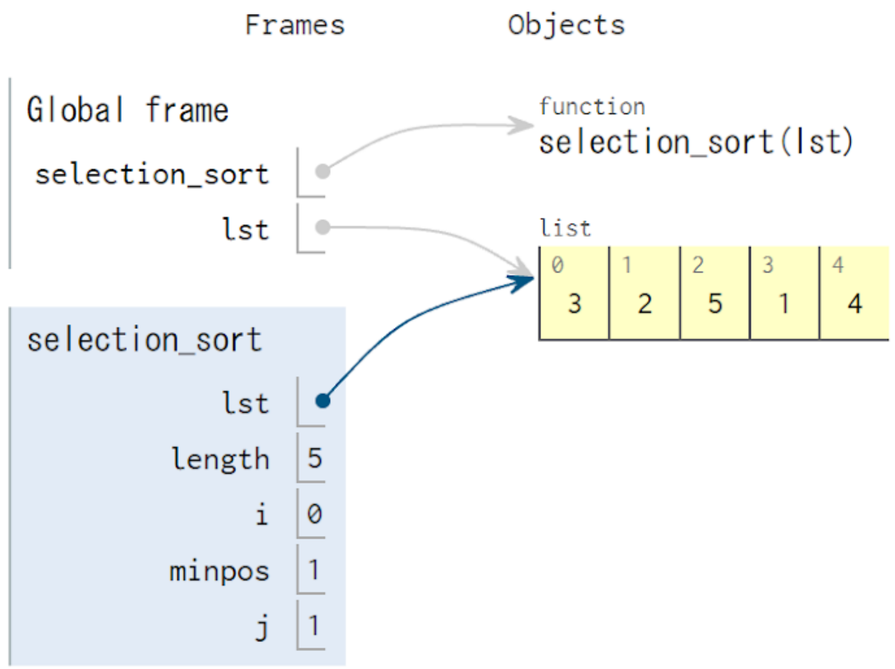



- 最小の要素のインデックス minposを、仮に i 番目とします（4行目）。
- minposとminposの右側にある全ての要素の中から、最も小さい要素を見つけるループです（5行目）。

# 6ー7行目

Python 3.6
(known limitations)

```
1  def selection_sort(lst):
2      length = len(lst)
3      for i in range(length):
4          minpos = i
5          for j in range(i + 1, length):
6              if lst[j] < lst[minpos]:
7                  minpos = j
8          if i != minpos:
9              tmp = lst[i]
10             lst[i] = lst[minpos]
11             lst[minpos] = tmp
```

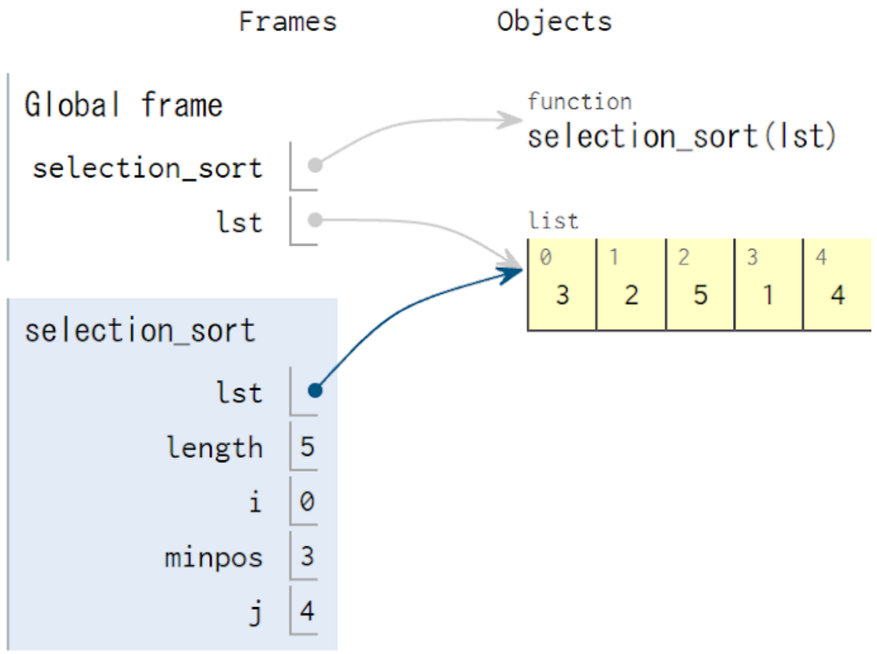



- ある要素(j番目)が一番小さい要素(仮)より小さい場合(6行目)
- j番目を一番小さい要素(仮)とします．(7行目)

# 8ー11行目

Python 3.6
(known limitations)

```
1  def selection_sort(lst):
2      length = len(lst)
3      for i in range(length):
4          minpos = i
5          for j in range(i + 1, length):
6              if lst[j] < lst[minpos]:
7                  minpos = j
8          if i != minpos:
9              tmp = lst[i]
10             lst[i] = lst[minpos]
11             lst[minpos] = tmp
```




- 5-7行目のループを終わると、minposはi番目とそれより右にあるすべての要素のうち、一番小さい要素の番号になります。
- もしi番目の要素が最も低い要素（minpos　番目）でない場合（8行目）
- i番目の要素とminpos番目の要素を入れ替える（9-11行目）

# 選択ソートの比較回数と交換回数

- ソートのループ実行中、具体的に何回計算が行われているか確認しましょう。
- 要素の比較回数と交換回数を調べるため、右の赤字のプログラムを追加します。

```
def selection_sort(lst):
    length = len(lst)
    cp = 0
    sw = 0
    for i in range(length):
        minpos = i
        for j in range(i + 1, length):
            cp += 1
            if lst[j] < lst[minpos]:
                minpos = j
        if i != minpos:
            sw += 1
            tmp = lst[i]
            lst[i] = lst[minpos]
            lst[minpos] = tmp
print("compare:", cp, "swap:", sw)
```

# （再掲）選択ソートの計算量は？

- 右の赤のパート（要素の比較）が何回行われるかを考えてみます
  - 外側のfor文：n－1 回
  - 内側のfor文：n－i－1 回
  - 従って、合計回数は…

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

$$O(n^2)$$

```
def selection_sort(lst):
    length = len(lst)
    for i in range(length):
        minpos = i
        for j in range(i + 1, length):
            if lst[j] < lst[minpos]:
                minpos = j
        if i != minpos:
            tmp = lst[i]
            lst[i] = lst[minpos]
            lst[minpos] = tmp
```

# 入力による交換回数の違い

- 選択ソートは常に一定の比較回数です
- リストの長さ n = 5 ならば 10回

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \frac{5(5-1)}{2} = 10$$

- 一方、交換回数はリストの内容によって異なります
  - 次のような数列をselection_sort()に入力してみましょう

[3, 2, 5, 1, 4]
[1, 2, 3, 4, 5]
[3, 5, 2, 1, 4]