انواع مرتب سازی به شرح ذیل هستند :

- ✓ مرتب سازی حبابی (Bubble Sort)
- ✓ مرتب سازی انتخابی (Selection Sort)
- ✓ مرتب سازی درجی (Insertion Sort)
- ✓ مرتب سازی ادغامی (Merge Sort)
- ✓ مرتب سازی سریع (Quik Sort)

❖ مرتب سازی حبابی (Bubble Sort)

مرتب‌سازی حبابی یکی از الگوریتم‌هایی است که معمولاً در دوره‌های مقدماتی کلاس‌های علوم رایانه (Computer Sience) ارائه می‌شود ، چون به روشنی طرز کار مرتب‌سازی را نشان می‌دهد و همزمان درک آن نیز ساده و آسان است . مراحل مرتب‌سازی حبابی از طریق یک لیست و مقایسه عناصر مجاور صورت می‌گیرد . در این فرایند وقتی عناصر در ترتیب نادرستی باشند ، با هم دیگر تعویض می‌شوند . این عملیات روی آرایه‌ها تا زمانی که هیچ دو عنصر مجاوری در ترتیب نادرست نباشند ، ادامه می‌یابد . از آنجا که مرتب‌سازی حبابی به طور مکرر روی بخش مرتب نشده لیست انجام می‌یابد ، مقدار پیچیدگی آن در بدترین حالت برابر با O (n^2) است .

```python
def bubble_sort(arr):
    def swap(i, j):
        arr[i], arr[j] = arr[j], arr[i]

    n = len(arr)
    swapped = True

    x = -1
    while swapped:
        swapped = False
        x = x + 1
        for i in range(1, n-x):
            if arr[i - 1] > arr[i]:
                swap(i - 1, i)
                swapped = True

    return arr

List = [15, 56, 16, 10, 40]

print(bubble_sort(List))
```

❖ مرتب سازی انتخابی (Selection Sort)

مرتب‌سازی انتخابی کاملاً ساده است ؛ اما با این حال در اغلب موارد عملکردی بهتر از مرتب‌سازی حبابی دارد . اگر قرار است یکی از این دو روش را انتخاب کنید ، در اغلب موارد بدون بررسی می‌توان گفت که مرتب‌سازی انتخابی، گزینه بهتری است . ما در مرتب‌سازی انتخابی ، لیست یا آرایه ورودی خود را به دو بخش تقسیم می‌کنیم . یک بخش زیرمجموعه مرتب شده است و بخش دیگر زیرمجموعه‌ای است که همچنان منتظر مرتب‌سازی است . ابتدا کوچک‌ترین عنصر را در زیرمجموعه نامرتب انتخاب کرده و آن را در انتهای زیرمجموعه مرتب قرار می‌دهیم . از این رو به طور مرتب کوچک‌ترین عنصر نامرتب را بر می‌داریم و آن را در انتهای زیرمجموعه مرتب قرار می‌دهیم . این فرایند تا زمانی که کل عناصر به لیست مرتب انتقال یابند ادامه می‌یابد .

```python
def selection_sort(arr):
    for i in range(len(arr)):
        minimum = i

        for j in range(i + 1, len(arr)):
            # Select the smallest value
            if arr[j] < arr[minimum]:
                minimum = j

        # Place it at the front of the
        # sorted end of the array
        arr[minimum], arr[i] = arr[i], arr[minimum]

    return arr

List = [15, 56, 16, 10, 40]

print(selection_sort(List))
```

❖ مرتب سازی درجی (Insertion Sort)

مرتب‌سازی درجی از هر دو روش مرتب‌سازی حبابی و انتخابی که در بخش فوق اشاره کردیم هم سریع‌تر و هم ساده‌تر است . این روش تا حدود زیادی شبیه روش مرتب‌سازی کارت‌های بازی است که اغلب افراد به طور غریزی انجام می‌دهند . در این روش در هر تکرار یک عنصر از آرایه برداشته می‌شود. سپس موقعیت آن عنصر در آرایه در صورتی که مرتب باشد پیدا می‌شود . این فرایند تا زمانی که همه عناصر ورودی جایگاه خود را در آرایه مرتب بیابند تداوم می‌یابد .

```
def insertion_sort(arr, simulation=False):

    for i in range(len(arr)):
        cursor = arr[i]
        pos = i

        while pos > 0 and arr[pos - 1] > cursor:
            # Swap the number down the list
            arr[pos] = arr[pos - 1]
            pos = pos - 1
        # Break and do the final swap
        arr[pos] = cursor

    return arr

List = [15, 56, 16, 10, 40]

print(insertion_sort(List))
```