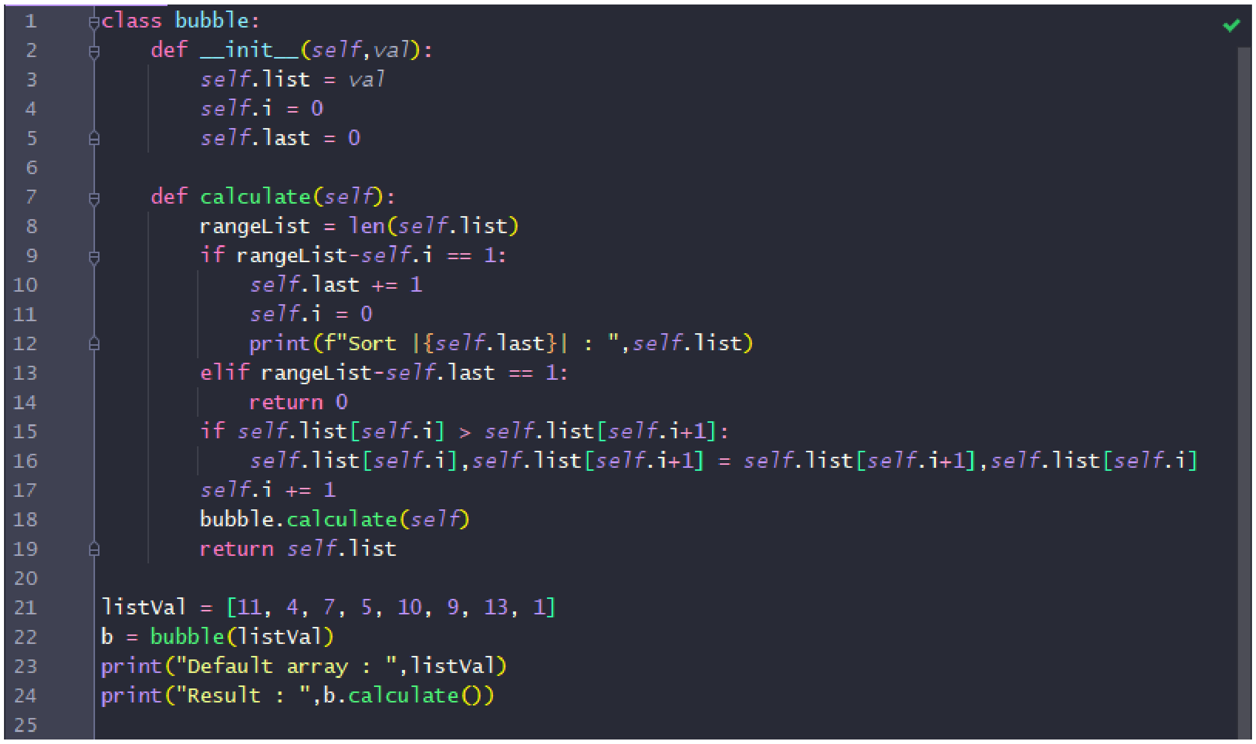
Sorting (Part 1)

EX1 :

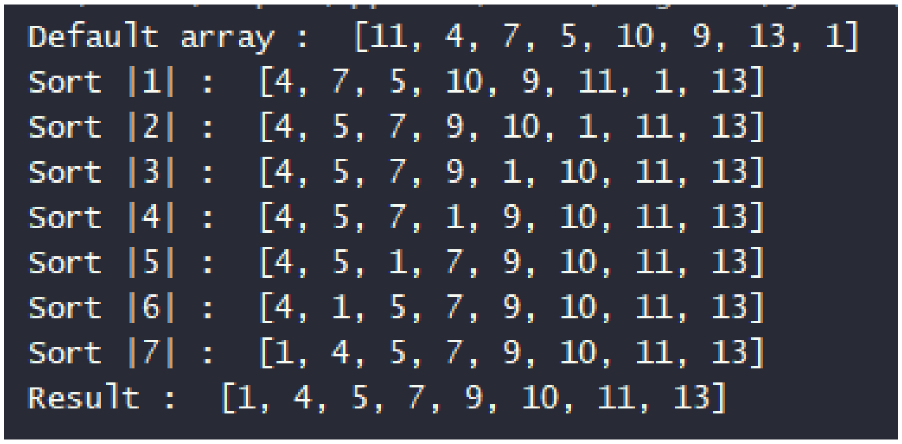
Code :

Bubble sort



```python
class bubble:
    def __init__(self,val):
        self.list = val
        self.i = 0
        self.last = 0

    def calculate(self):
        rangeList = len(self.list)
        if rangeList-self.i == 1:
            self.last += 1
            self.i = 0
            print(f"Sort |{self.last}| : ",self.list)
        elif rangeList-self.last == 1:
            return 0
        if self.list[self.i] > self.list[self.i+1]:
            self.list[self.i],self.list[self.i+1] = self.list[self.i+1],self.list[self.i]
        self.i += 1
        bubble.calculate(self)
        return self.list

listVal = [11, 4, 7, 5, 10, 9, 13, 1]
b = bubble(listVal)
print("Default array : ",listVal)
print("Result : ",b.calculate())
```

Result :

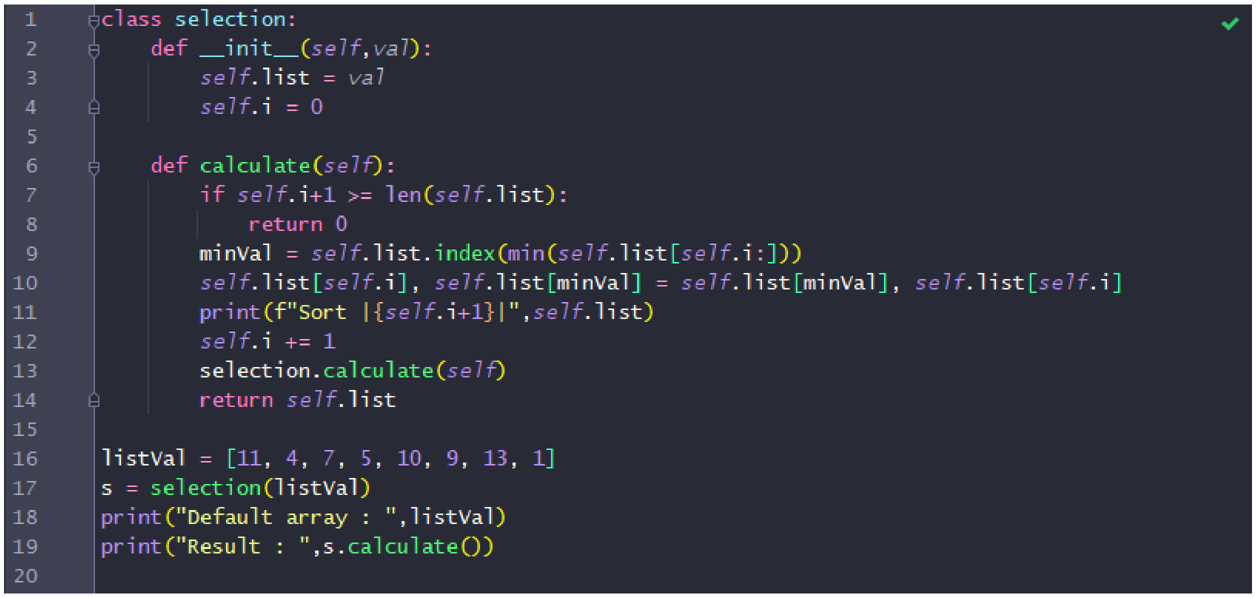


```
Default array :  [11, 4, 7, 5, 10, 9, 13, 1]
Sort |1| :  [4, 7, 5, 10, 9, 11, 1, 13]
Sort |2| :  [4, 5, 7, 9, 10, 1, 11, 13]
Sort |3| :  [4, 5, 7, 9, 1, 10, 11, 13]
Sort |4| :  [4, 5, 7, 1, 9, 10, 11, 13]
Sort |5| :  [4, 5, 1, 7, 9, 10, 11, 13]
Sort |6| :  [4, 1, 5, 7, 9, 10, 11, 13]
Sort |7| :  [1, 4, 5, 7, 9, 10, 11, 13]
Result :  [1, 4, 5, 7, 9, 10, 11, 13]
```

EX2 :

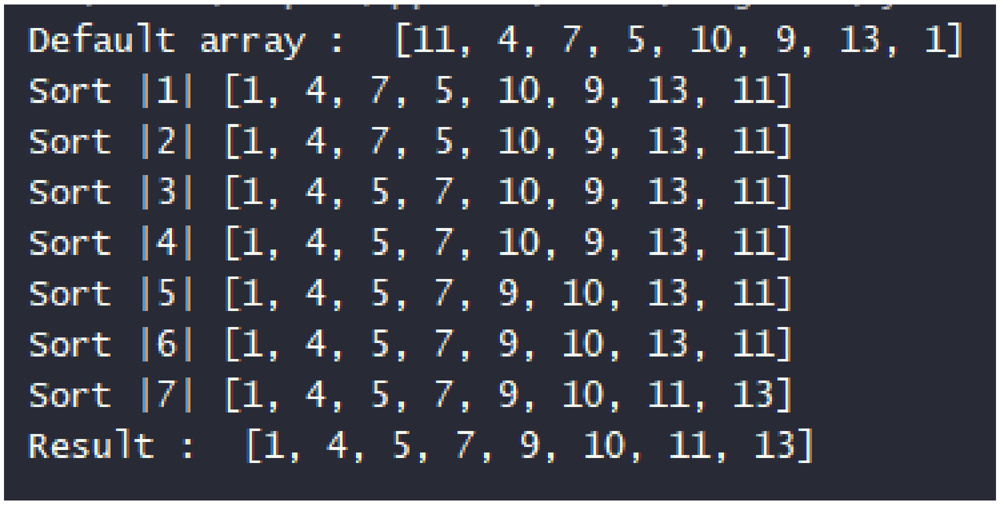
Code :

Selection sort



```python
class selection:
    def __init__(self,val):
        self.list = val
        self.i = 0

    def calculate(self):
        if self.i+1 >= len(self.list):
            return 0
        minVal = self.list.index(min(self.list[self.i:]))
        self.list[self.i], self.list[minVal] = self.list[minVal], self.list[self.i]
        print(f"Sort |{self.i+1}|",self.list)
        self.i += 1
        selection.calculate(self)
        return self.list

listVal = [11, 4, 7, 5, 10, 9, 13, 1]
s = selection(listVal)
print("Default array : ",listVal)
print("Result : ",s.calculate())
```

Result :

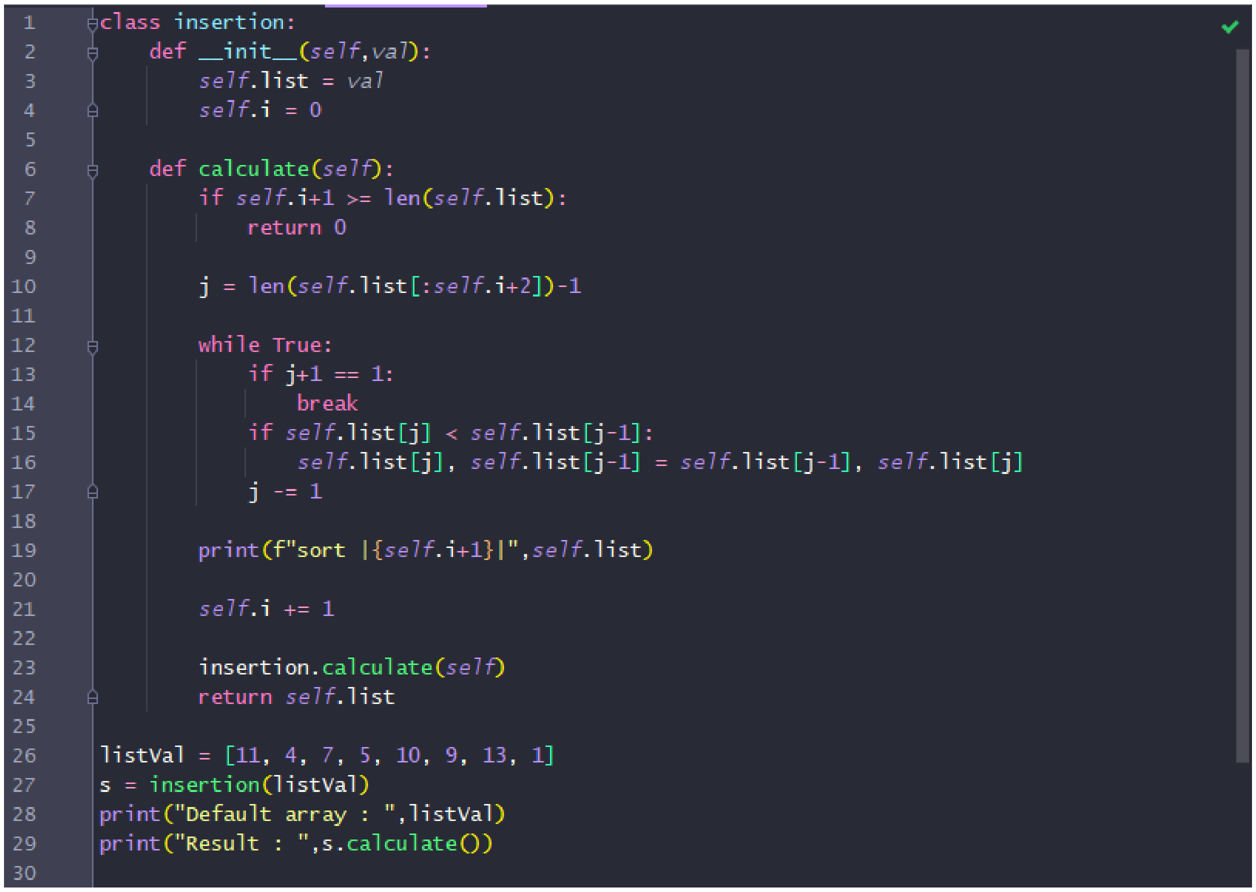


```
Default array :  [11, 4, 7, 5, 10, 9, 13, 1]
Sort |1| [1, 4, 7, 5, 10, 9, 13, 11]
Sort |2| [1, 4, 7, 5, 10, 9, 13, 11]
Sort |3| [1, 4, 5, 7, 10, 9, 13, 11]
Sort |4| [1, 4, 5, 7, 10, 9, 13, 11]
Sort |5| [1, 4, 5, 7, 9, 10, 13, 11]
Sort |6| [1, 4, 5, 7, 9, 10, 13, 11]
Sort |7| [1, 4, 5, 7, 9, 10, 11, 13]
Result :  [1, 4, 5, 7, 9, 10, 11, 13]
```

EX3 :

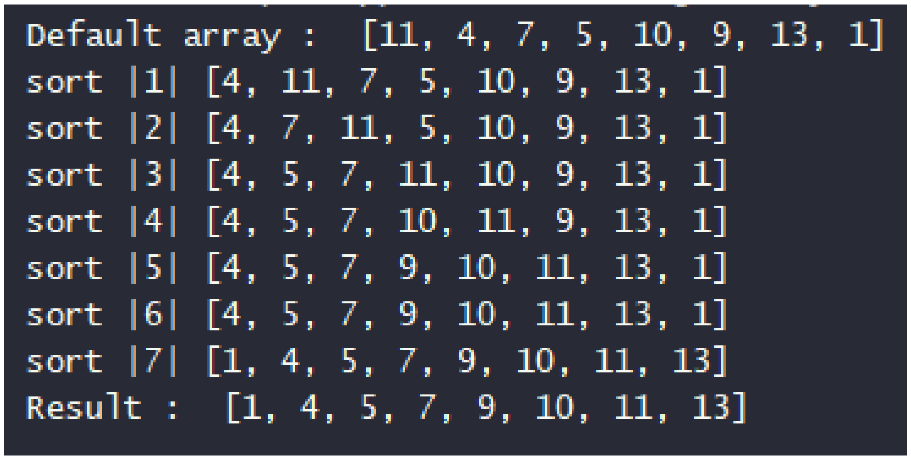
Code:

Insertion sort



```python
class insertion:
    def __init__(self,val):
        self.list = val
        self.i = 0

    def calculate(self):
        if self.i+1 >= len(self.list):
            return 0

        j = len(self.list[:self.i+2])-1

        while True:
            if j+1 == 1:
                break
            if self.list[j] < self.list[j-1]:
                self.list[j], self.list[j-1] = self.list[j-1], self.list[j]
            j -= 1

        print(f"sort |{self.i+1}|",self.list)

        self.i += 1

        insertion.calculate(self)
        return self.list

listVal = [11, 4, 7, 5, 10, 9, 13, 1]
s = insertion(listVal)
print("Default array : ",listVal)
print("Result : ",s.calculate())
```

Result:



```
Default array :  [11, 4, 7, 5, 10, 9, 13, 1]
sort |1| [4, 11, 7, 5, 10, 9, 13, 1]
sort |2| [4, 7, 11, 5, 10, 9, 13, 1]
sort |3| [4, 5, 7, 11, 10, 9, 13, 1]
sort |4| [4, 5, 7, 10, 11, 9, 13, 1]
sort |5| [4, 5, 7, 9, 10, 11, 13, 1]
sort |6| [4, 5, 7, 9, 10, 11, 13, 1]
sort |7| [1, 4, 5, 7, 9, 10, 11, 13]
Result :  [1, 4, 5, 7, 9, 10, 11, 13]
```

Explanation

From all above method which one is the best in term of performance?

- Bubble sort

From all above method which one is the best in term of resource management?

- Insertion sort

From all above method which one is the best in your opinion?

- Selection sort