Insertion Sort

Advanced Higher Computing Science

Learning Objectives

By the end of this lesson you will be able to:

- ❏ Describe, exemplify, and implement a standard insertion sort algorithm

Insertion Sort – Ascending order

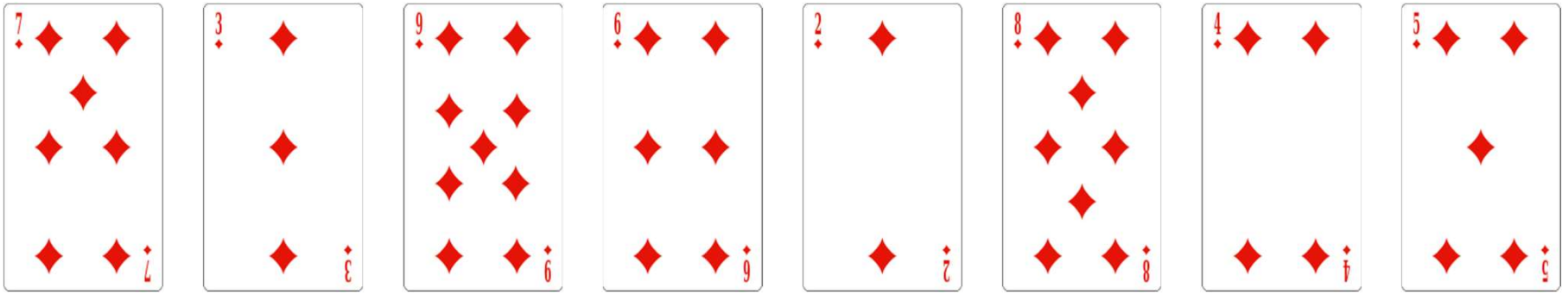
An **insertion sort** builds the list one element at a time

1. The current value to be inserted is stored
2. When an element is greater than the value to be inserted it is shifted one element to the right.
3. When the next element to be shifted is less than the value to be inserted then the value is placed at that point
4. The first element to be checked is item 1(assuming the first index is 0)

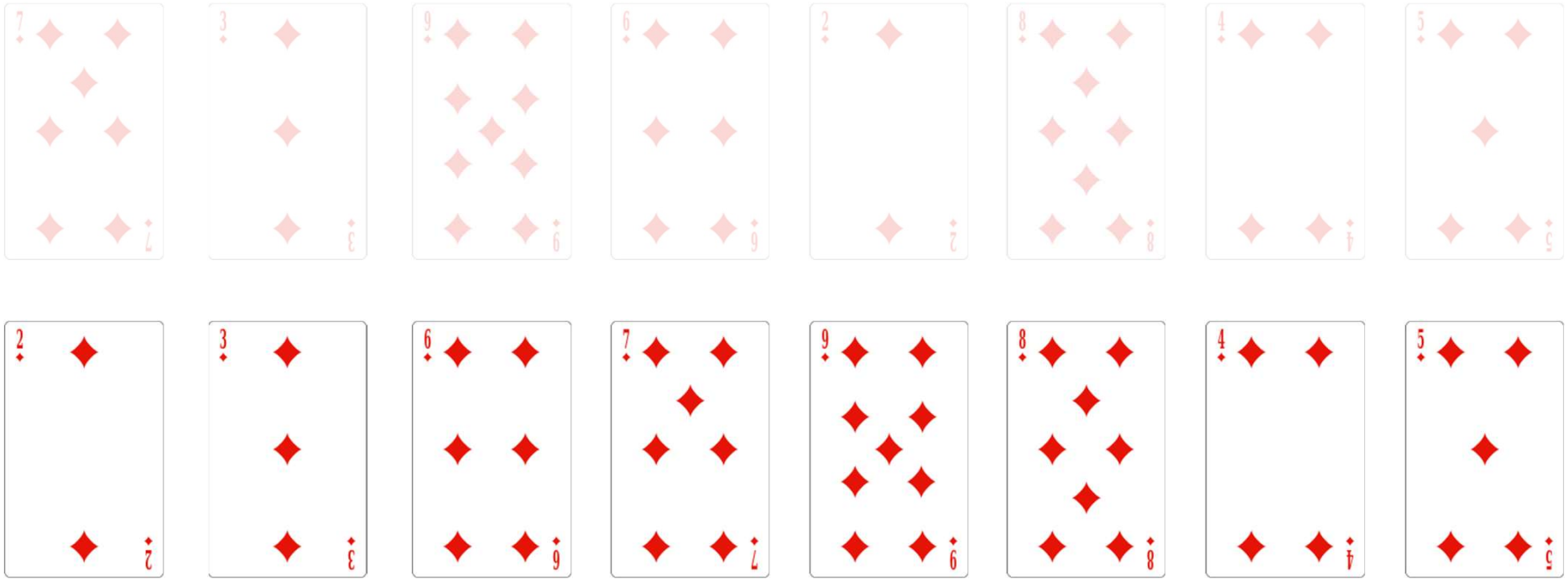
Ascending Insertion Sort - Algorithm

```
FOR index = 1 TO length(list) - 1
        currentval = list [index]
        position = index
        WHILE position > 0 AND (list [position-1] > currentval)
                list[position] = list [position-1]
                position -= 1
        END WHILE
SET list[position] = currentval
END FOR
```

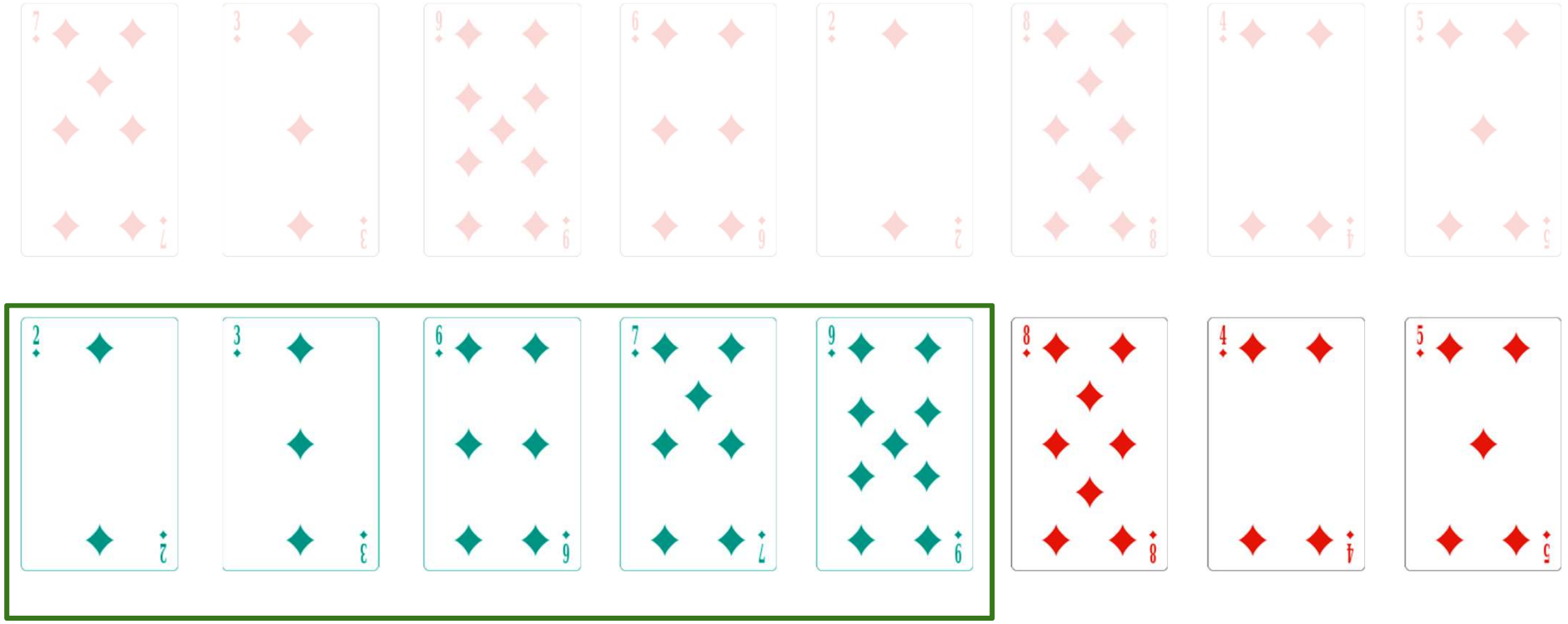
Insertion Sort - Cards



Insertion Sort – Cards (After 4 Passes)



Insertion Sort – Cards (After 4 Passes)



5 Sorted Elements

Python Implementation

```
for index in range (1,len(myList)):
    #store the value to be inserted into the array
    currentvalue = myList[index]
        position = index

        #shift the rest of the array one to the right
    while position > 0 and myList[position-1]>currentvalue:
        myList[position] = myList[position-1]
        position -=1

    #insert the value into the array
    myList[position] = currentvalue
```

What if we are using records/classes

Assuming we are just using public properties and using the following class structure:

```
class BMI():
    def __init__(self):
        self.height = 0.0
        self.weight = 0.0
        self.BMI = 0.0
```

What if we wanted to sort in order of BMI?

We will assume we have an array of these BMI objects held in an array called **myBMI**.

Python Implementation (records/classes)

```
for index in range (1,len(myBMI)):
    #store the record/object to be inserted into the array
    currentrecord = myBMI[index]
    position = index

    #shift the rest of the array one to the right
    while position > 0 and myBMI[position-1].BMI>currentrecord.BMI:
        myBMI[position] = myBMI[position-1]
        position -=1

    #insert the value into the array
    myBMI[position] = currentrecord
```

Sorting 2D Arrays

It isn't really all that different

You can apply any sort algorithm to a 2D array

If you consider that the following 2D array with names and 3 marks

Joe	8	2	1

Python Implementation (records/classes)

```
for index in range (1,len(myList)):
    currentvalue = myList[index]
        position = index

    while position>0 and myList[position-1].name> currentvalue.name :
        myList[position] = myList[position-1]
        position -=1

    myList[position] = currentvalue
```

This stores the current record

Ensures we are comparing the name field in each record

This is sorting on the Name field in ascending order

Standard Insertion Sort (2D Array)

```
for index in range (1,len(myList)):
    currentvalue = myList[index]
        position = index

    while position>0 and myList[position-1][0]>currentvalue[0]:
        myList[position] = myList[position-1]
        position -=1

    myList[position] = currentvalue
```

This stores the whole row of the 2D Array

Ensures we are comparing the first column in each row

This is sorting on the Name (which is in the first element) in ascending order

Standard Insertion Sort (2D Array)

```
for index in range(1, len(searchlist)):
    # stores each part of the current
    # row to be inserted
    #Name
    currentvalue1 = searchlist[index][0]
    #Mark2
    currentvalue2 = searchlist[index][1]
    #Mark3
    currentvalue3 = searchlist[index][2]

    # Store the position to be shifted to.
    position = index

    while position > 0 and searchlist[position - 1][0] > currentvalue1:
        searchlist[position][0] = searchlist[position - 1][0]
        searchlist[position][1] = searchlist[position - 1][1]
        searchlist[position][2] = searchlist[position - 1][2]
        position -= 1

    # store the value to be inserted into the array
    searchlist[position][0] = currentvalue1
    searchlist[position][1] = currentvalue2
    searchlist[position][2] = currentvalue3
```

These store each part of the 2D array

Ensures we are checking the first column in each row

This version of the algorithm stores each value in a row separately - can be seen as inefficient