Krish Agarwal
21112016
2BSc DS A

Data Structures & Algorithms

CIA 3 Submission

Table of Content

Lab Programs					
Sr. No.	**Title**	**Description**	**LOC (self)**	**LOC (inspired)**	**Total**
1.	Lab1	1. Exploring List Methods 2. Implement Linear and Binary Search 3. Menu-Driven Program 4. Patterns	303	46	349
2.	Lab3	1. Increment and Decrement in 1 and 2 variables 2. Menu-Driven for the same 3. Matrix	121	0	121
3.	Lab4	1. Menu-Driven Program	292	18	310
4.	Lab5	1. Stack 2. Queue 3. Card Display	144	0	144
5.	Lab6	1. Same row/column problem 2. Bishop and rook problem 3. Chess Board	59	0	59
6.	Lab7	1. Linked List	118	51	169
7.	Lab8	1. Printing array 2. Travel possible/not possible question	32	0	32
8.	Lab9	1. Read an excel file 2. Add one column from dataset	17	0	17
9.	Lab Exam	1. Print odd numbers from 1-17	7	0	7
		Total:	1093	115	**1208**

Classwork					
Sr. No.	**Title**	**Description**	**LOC (self)**	**LOC (inspired)**	**Total**
1.	Menu- Driven	Menu- Driven program for all the sorts and searches	200	0	200
2.	BST	Binary Search Tree - insert and all 3 orders/traversals	60	24	84
3.	DFS	DFS - Vertices of a graph	0	44	44
4.	Patterns	Printing patterns in various orders	60	0	60
5.	Peel & Layer	Peeling and layering patterns	43	0	43
6.	LL + Stk + Que	Stack & Queue from LinkedList	137	0	137
7.	Heap Functions	All the functions of heap sort	32	40	72
8.	Menu- Driven: Heap	Menu- Driven program for the above	105	50	155
9.	Data Science from Scratch codes	1. Choosing random codes from given zip file and tweaking it	10	1055	1065
		Total:	647	1213	**1860**

Excel Files					
Sr. No.	Title	Description	LOC (self)	LOC (inpsired)	Total
1.	Sort Walkthroughs	Visual Representation of all sorts	0	0	0
2.	Visual Basic	Printing Patterns using VB	58	0	58
		Total:	58	0	58

Total LOC:	3126

Lines of code:

Self - 303
inspired - 46
Total - 349

Lab1

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS
Data Structures Using Python

Set 1

In [1]:
```
"""
Start and End Greet

------------------------------------------------------------
|This is an User defined program to display the prompt     |
|that are used before and after execution of each program. |
|                                                          |
|Developed by                                              |
|Krish Agarwal                                             |
------------------------------------------------------------
"""

#Creating of User Defined Function

def startgreet(var):
    '''                                     ///Lab  1/// '''

    print("\n    ----------------------------------", var, "----------------

    print("\n    -------------------- Hi there! Welcome to the Program -----


def endgreet():
    print("You have exited this program!\n")
    print("                  ///Thank You for Using this Program///")
```

Q1) Explore different methods in LIST function

In [2]:

```python
#Recalling of DocString
#Greeting at the beginning

variable1 = "SORTS"

print(startgreet.__doc__)

variable2 = startgreet(variable1)
print("\n")

# --- Creating a list and displaying it ---
initial_list = [1, 2, 3, 4, 5, 6, 3]
print("============================")
print("List:", initial_list)
print("============================")

# --- append function ---
initial_list.append(7)
print("\n1. Adding 7 to the list:", initial_list)

# --- count function ---

print("\n2. Counting the number of times 3 is in the list:", initial_list.co

# --- extend function ---
sec_list = [8, 9, 10]
initial_list.extend(sec_list)
print("\n3. Adding '8, 9, 10' to the list:", initial_list)

# --- index function ---
print("\n4. The posittion of element 7:", initial_list.index(7))

# --- insert function ---
initial_list.insert(0, 0)
print("\n5. Adding 0 the beginning:", initial_list)

# --- remove function ---
initial_list.remove(9)
print("\n6. Removing the 9 at the end:", initial_list)

# --- reverse banana ---
initial_list.reverse()
print("\n7. Reversing the order of the list:", initial_list)

# --- sort function ---
initial_list.sort()
print("\n8. Sorting the list in ascending order:", initial_list)
```

```
                                ///Lab  1///

    ---------------------------------- SORTS ----------------------------------
-

    -------------------- Hi there! Welcome to the Program -------------------
-
```

```
============================
List: [1, 2, 3, 4, 5, 6, 3]
============================

1. Adding 7 to the list: [1, 2, 3, 4, 5, 6, 3, 7]

2. Counting the number of times 3 is in the list: 2

3. Adding '8, 9, 10' to the list: [1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10]

4. The posittion of element 7: 7

5. Adding 0 the beginning: [0, 1, 2, 3, 4, 5, 6, 3, 7, 8, 9, 10]

6. Removing the 9 at the end: [0, 1, 2, 3, 4, 5, 6, 3, 7, 8, 10]

7. Reversing the order of the list: [10, 8, 7, 3, 6, 5, 4, 3, 2, 1, 0]

8. Sorting the list in ascending order: [0, 1, 2, 3, 3, 4, 5, 6, 7, 8, 10]
```

Q2) Create a list with 10 fixed numbers

In [3]:

```
fix_list = [12, 9, 14, 8, 6, 1, 11, 18, 6, 10]

#assigning the length of the list a variable
length = len(fix_list)
print(length)
```

```
10
```

Q3) Implement Linear Search

Note: Just displaying of the code. The menu-driven code is below in the menu-driven program.

In [4]:

```
def linearSearch2(fix_list, inp_array):

    for i in range(0, length):
        if fix_list[i] == inp_array:
            print("The number is present at position", i)
            return i
        return -1
```

In [5]:

```
inp_array = int(input("Enter the number you are looking for: "))
print()
linearSearch2(fix_list, inp_array)
```

```
Enter the number you are looking for: 5
```

Out[5]:

```
-1
```

Q4) Implement Binary Search

In [6]:

```python
# Binary Search in python
def binarySearch(array, x, low, high):

    # Repeat until the pointers low and high meet each other
    while low <= high:

        mid = low + (high - low)//2

        if array[mid] == x:
            return mid

        elif array[mid] < x:
            low = mid + 1

        else:
            high = mid - 1

    return -1


array = [3, 4, 5, 6, 7, 8, 9]

x = int(input("Enter the number you are searching for: "))
print("-"*40, "\n")

result = binarySearch(array, x, 0, len(array)-1)

if result != -1:
    print("Element is present at index " + str(result))
else:
    print("Not found")

```

```
Enter the number you are searching for: 14
----------------------------------------

Not found
```

Q5) Create a Menu for selecting different options

In [28]:

```
import random

#creeating a function for display and menu-driven program
def displayList():
    print("List:", fix_list)
    print(" ")

def menuDrivenProgram():
    print("*" * 20)
    print("0. Exit\n1. Linear Search\n2. Binary Search\n3. Bubble Sort\n4. S
    print("*" * 20)
    print(" ")

# --- Linear Search ---
def linearSearch(fix_list, inp_array):

    for i in range(0, length):
        if fix_list[i] == inp_array:
            print("The number is present at position", i)
            return i
        return -1

# --- Bubble Sort ---
# Binary Search in python
def binarySearch(array, x, low, high):

    # Repeat until the pointers low and high meet each other
    while low <= high:

        mid = low + (high - low)//2

        if array[mid] == x:
            return mid

        elif array[mid] < x:
            low = mid + 1

        else:
            high = mid - 1

    return -1

# --- Bubble Sort ---
def bubbleSort(fix_list):

    for i in range(0, length):
        for j in range(0, length-i-1):

            if fix_list[j] > fix_list[j+1]:

                temp = fix_list[j]
                fix_list[j] = fix_list[j+1]
                fix_list[j+1] = temp

# --- Selection Sort ---
def selectionSort(fix_list):
```

```
57
58      for i in range(0, length-1):
59          minPos = i
60
61          for j in range(i+1, length):
62
63              if fix_list[minPos] > fix_list[j]:
64                  minPos = j
65
66                  t = fix_list[minPos]
67                  fix_list[minPos] = fix_list[i]
68                  fix_list[i] = t
69
70  # --- Insertion Sort ---
71  def insertionSort(array):
72      for step in range(1, len(array)):
73          key = array[step]
74          j = step -1
75
76          while (j>= 0 and key<array[j]):
77              array[j+1] = array[j]
78              j = j - 1
79
80          array[j + 1] = key
81
82  #Recalling of DocString
83  #Greeting at the beginning
84
85  variable1 = "SORTS"
86
87  print(startgreet.__doc__)
88
89  variable2 = startgreet(variable1)
90  print("\n")
91
92  # --- Menu-Driven Program ---
93  while True:
94
95      menuDrivenProgram()
96
97      print("-" * 20)
98      option_input = int(input("Select an option:"))
99      print("-" * 20)
100     print(" ")
101
102     if option_input == 0:
103         endgreet()
104         break
105
106     elif option_input == 1:
107
108         inp_array = int(input("Enter the number you are searching for: "))
109         print("----------------------------------------")
110         linearSearch(fix_list, inp_array)
111         print(" ")
112
113     elif option_input == 2:
```

```
114
115         inp_bin = int(input("Enter the number you are searching for: "))
116         print("-"*40, "\n")
117
118         result = binarySearch(array, inp_bin, 0, len(array)-1)
119
120         if result != -1:
121             print("Element is present at index " + str(result), "\n")
122         else:
123             print("Not found\n")
124
125     elif option_input == 3:
126         bubbleSort(fix_list)
127
128     elif option_input == 4:
129         selectionSort(fix_list)
130
131     elif option_input == 5:
132         pass
133
134     elif option_input == 10:
135         displayList()
136
137     elif option_input == 11:
138         random.shuffle(fix_list)
```

///Lab 1///

---------- SORTS ----------

---------- Hi there! Welcome to the Program ----------

```
********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [6, 8, 9, 10, 6, 1, 12, 11, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
```

```
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:1
--------------------

Enter the number you are searching for: 6
----------------------------------------
The number is present at position 0

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:3
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:2
--------------------

Enter the number you are searching for: 6
----------------------------------------

Element is present at index 3

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
```

```
********************

--------------------
Select an option:10
--------------------

List: [1, 6, 6, 8, 9, 10, 11, 12, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:11
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [11, 10, 12, 14, 8, 9, 18, 6, 6, 1]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:4
--------------------

********************
0. Exit
```

```
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [1, 6, 6, 9, 8, 10, 11, 12, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:11
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [18, 6, 8, 10, 14, 6, 1, 11, 9, 12]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************
```

```
--------------------
Select an option:5
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [1, 6, 6, 8, 9, 10, 11, 12, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:0
--------------------

You have exited this program!

                  ///Thank You for Using this Program///
```

Set 2

Q1) Create two loops I,j and display I,j for all

In [8]:
```
print("-------------------------------------------------")
input_loop = int(input("Enter a number as the end point of the range: "))
print("-------------------------------------------------\n")

def loopOne(input_loop):
    for i in range (0, input_loop+1):
        print(i, end = " ")
    print(" ")

def loopTwo(input_loop):
    for j in range(input_loop, -1, -1):
        print(j, end = " ")
    print(" ")

loopOne(input_loop)
loopTwo(input_loop)
```

```
-------------------------------------------------
Enter a number as the end point of the range: 15
-------------------------------------------------

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

Q2) Display a full box

In [9]:
```
def fullBox(input_box):

    for i in range(input_box):
        for j in range(input_box):
            print("*", end=" ")
        print(" ")

input_box = int(input("Enter the order of the box: "))
print(" ")
fullBox(input_box)
```

```
Enter the order of the box: 5

* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Q3) Display a diagnol

Note: Referrence from the internet

In [10]:

```
def diagonal(input_diagonal):

    for i in range(input_diagonal):
        print(' '*i, end = '*\n')

input_diagonal = int(input("Enter the order of the diagnol:"))
print(" ")
diagonal(input_diagonal)
```

```
Enter the order of the diagnol:5

*
 *
  *
   *
    *
```

Q4) Display a Empty Box

In [11]:

```
def emptyBox(input_empty):

    for i in range(input_empty):
        for j in range(input_empty):

            if ((i==0) or (i==(input_empty-1)) or (j==0) or (j==(input_empty
                print("*", end = " ")

            else:
                print(" ", end = " ")

        print(" ")

input_empty = int(input("Enter the order of the box: "))
print(" ")
emptyBox(input_empty)
```

```
Enter the order of the box: 5

* * * * *
*       *
*       *
*       *
* * * * *
```

Q5) Display other diagonal

In [26]:

```python
def odiagonal(input_diagonal):

    for i in range(input_diagonal):
            for j in range(input_diagonal):
                if i + j == input_diagonal -1:
                    print("*", end = " ")
                else:
                    print(" ")

input_diagonal = int(input("Enter the order of the box: "))
print()

odiagonal(input_diagonal)
```

```
Enter the order of the box: 7







*





*





*





*





*





*





*
```

Q6) Display letter E

Note: Taken from the internet for refernce

In [17]:
```
inputETwo = 5

def letterETwo() :

    for i in range(0,inputETwo) :
        print("*",end="")
        for j in range(0,inputETwo) :
            if ( (i == 0 or i == inputETwo - 1) or (i == inputETwo // 2 and
                print("*",end="")
            else :
                continue
        print()

letterETwo()
```

```
******
*
****
*
******
```

Q7) Display Letter A

In [ ]:
```
1
```

Q8) Display letter X

In [21]:
```
def letterX(inputX):

    for i in range(inputX):
        for j in range(inputX):

            if (i==j) or (i+j == inputX-1):
                print("*", end=" ")

            else:
                print(" ", end=" ")
        print(" ")

inputX = int(input("Enter the order of the symbol:"))
print(" ")
letterX(inputX)
```

```
Enter the order of the symbol:7

*           *
  *       *
    *   *
      *
    *   *
  *       *
*           *
```

Q9) Display letter S

(optional)

In [ ]:
```
1
```

Q10) Display letter N

(optional)

In [ ]:
```
1
```

Lines of Code:

Self - 121
Total - 121

Lab 3 (Group 5 - Sakshi)

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS A
Data Structures Using Python

Q1

In [1]:

```python
def display():
    print(x)
    print(" ")

def menu():
    print("0. Exit")
    print("1. Increase x")
    print("2. Decrease x")
    print(" ")

print("-------------------")
x = int(input("Enter a number: "))
print("-------------------\n")

while True:

    menu()

    option_input = int(input("Select an option: "))
    print("-------------------\n")

    if option_input == 1:
        x += 1
        display()

    elif option_input == 2:
        x -= 1
        display()

    elif option_input == 0:
        print("Thank You for using this program!")
        break
```

```
-------------------
Enter a number: 8
-------------------

0. Exit
1. Increase x
2. Decrease x

Select an option: 1
-------------------

9

0. Exit
1. Increase x
2. Decrease x

Select an option: 2
-------------------

8

0. Exit
```

```
1. Increase x
2. Decrease x

Select an option: 2
-------------------

7

0. Exit
1. Increase x
2. Decrease x

Select an option: 1
-------------------

8

0. Exit
1. Increase x
2. Decrease x

Select an option: 0
-------------------

Thank You for using this program!
```

Q2

In [2]:

```
print("----------------------------")
var1 = int(input("Enter the x co-ordinate: "))
var2 = int(input("Enter the y co-ordinate: "))
print("----------------------------\n")

def displayQ2():
    print(var1, var2)
    print(" ")

def menu2():
    print("0. Exit")
    print("1. Increase x")
    print("2. Decrease x")
    print("3. Decrease y")
    print("4. Increase y\n")

while True:

    menu2()

    option_input = int(input("Select an option: "))
    print(" ")

    if option_input == 0:
        print("Thank You for using this program!")
        break

    if option_input == 1:
        var1 += 1
        displayQ2()

    elif option_input == 2:
        var1 = var1 - 1
        displayQ2()

    elif option_input == 3:
        var2 -= 1
        displayQ2()

    elif option_input == 4:
        var2 += 1
        displayQ2()
```

```
----------------------------
Enter the x co-ordinate: 6
Enter the y co-ordinate: 6
----------------------------

0. Exit
1. Increase x
2. Decrease x
3. Decrease y
4. Increase y

Select an option: 1
```

```
7 6

0. Exit
1. Increase x
2. Decrease x
3. Decrease y
4. Increase y

Select an option: 3

7 5

0. Exit
1. Increase x
2. Decrease x
3. Decrease y
4. Increase y

Select an option: 4

7 6

0. Exit
1. Increase x
2. Decrease x
3. Decrease y
4. Increase y

Select an option: 2

6 6

0. Exit
1. Increase x
2. Decrease x
3. Decrease y
4. Increase y

Select an option: 0

Thank You for using this program!
```

Q3

In [11]:

```
def displayQ3():
    print(array)
    print(" ")

def menu3():
    print("0. Exit")
    print("1. Increment")
    print("2. Decrement")
    print(" ")

while True:

    array = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]


    menu3()
    print()
    print(array)
    print()

    option_input = int(input("Select an option:"))
    print("-------------------------\n")

    if option_input == 0:
        print("Thank You for using this program!")
        break

    elif option_input == 1:

        inp_element = int(input("Which position to insert 8 at:"))
        print(" ")

        for i in range(len(array)):
            array[inp_element] = 8

    elif option_input == 2:

        inp_element = int(input("Which position to insert 8 at:"))
        print(" ")

        for i in range(len(array)):
            array[inp_element] = 0
```

```
0. Exit
1. Increment
2. Decrement


[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Select an option:1
-------------------------

Which position to insert 8 at:5

0. Exit
```

```
1. Increment
2. Decrement


[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Select an option:2
-------------------------

Which position to insert 8 at:5

0. Exit
1. Increment
2. Decrement


[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Select an option:0
------------------------

Thank You for using this program!
```

Q4

In [2]:

```
matrix = [[0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0],
          [0, 0, 0, 0, 0]]
matrix
```

Out[2]:

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

Lines of Code:

Self - 292
Inspired - 18
Total - 310

Lab 4

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS A
Data Structures Using Python

In [2]:

```
"""
Start and End Greet

------------------------------------------------------------
|This is an User defined program to display the prompt     |
|that are used before and after execution of each program. |
|                                                          |
|Developed by                                              |
|Krish Agarwal                                             |
------------------------------------------------------------
"""

#Creating of User Defined Function

def startgreet(var):
    '''                                     ///Lab  4/// '''

    print("\n    ----------------------------------", var, "----------------

    print("\n    -------------------- Hi there! Welcome to the Program -----


def endgreet():
    print("You have exited this program!\n")
    print("                   ///Thank You for Using this Program///")
```

In [ ]:

```python
#importing random module for shuffling the list
import random

#Creating a list with fixed values
fix_list = [12, 9, 14, 8, 6, 1, 11, 18, 6, 10]

#assigning the length of the list a variable
length = len(fix_list)

#creating a function for display and menu-driven program
def displayList():
    print("List:", fix_list)
    print(" ")

def menuDrivenProgram():
    print("*" * 20)
    print("0. Exit\n1. Linear Search\n2. Binary Search\n3. Bubble Sort\n4. S
    print("*" * 20)
    print(" ")

# --- Linear Search ---
def linearSearch(fix_list, inp_array):

    for i in range(0, length):
        if fix_list[i] == inp_array:
            print("The number is present at position", i)
            return i
    return -1

# --- Bubble Sort ---
def bubbleSort(fix_list):

    for i in range(0, length):
        for j in range(0, length-i-1):

            if fix_list[j] > fix_list[j+1]:

                temp = fix_list[j]
                fix_list[j] = fix_list[j+1]
                fix_list[j+1] = temp

                print(fix_list)
    print(" ")

# --- Selection Sort ---
def selectionSort(fix_list):

    for i in range(0, length-1):
        minPos = i

        for j in range(i+1, length):

            if fix_list[minPos] > fix_list[j]:
                minPos = j

                t = fix_list[minPos]
```

```
                fix_list[minPos] = fix_list[i]
                fix_list[i] = t
                displayList()
    print(" ")

# --- Insertion Sort ---
def insertionSort(fix_list):

    for i in range(1, len(fix_list)): #We take the range from 1 as we consid

        save = fix_list[i] #assigning a variable to the position of elements

        j = i-1 #assigning a variable for the previous elements
        while j >=0 and save < fix_list[j] :
                fix_list[j+1] = fix_list[j] #similar to  [i-2] = [i-1]
                j -= 1 #moving elements one position ahead
        fix_list[j+1] = save

    print(fix_list)

#Recalling of DocString
#Greeting at the beginning

variable1 = "SORTS & SEARCHES"

print(startgreet.__doc__)

variable2 = startgreet(variable1)
print("\n")

# --- Menu-Driven Program ---
while True:

    menuDrivenProgram()

    print("-" * 20)
    option_input = int(input("Select an option:"))
    print("-" * 20)
    print(" ")

    if option_input == 0:
        endgreet()
        break

    elif option_input == 1:

        inp_array = int(input("Enter the number you are searching for: "))
        print("----------------------------------------")
        linearSearch(fix_list, inp_array)
        print(" ")

    elif option_input == 2:
        pass

    elif option_input == 3:
        bubbleSort(fix_list)

```

```
114     elif option_input == 4:
115         selectionSort(fix_list)
116 
117     elif option_input == 5:
118         pass
119 
120     elif option_input == 10:
121         displayList()
122 
123     elif option_input == 11:
124         random.shuffle(fix_list)
125         print(fix_list)
126         print(" ")
```

```
                                    ///Lab  1///

    ---------------------------------- SORTS & SEARCHES ----------------------
----------

    -------------------- Hi there! Welcome to the Program --------------------


********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle List
********************

--------------------
Select an option:10
--------------------

List: [12, 9, 14, 8, 6, 1, 11, 18, 6, 10]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle List
********************

--------------------
Select an option:11
--------------------

[9, 8, 18, 6, 12, 14, 11, 10, 6, 1]
```

```
********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle List
********************

--------------------
Select an option:11
--------------------

[18, 10, 6, 14, 12, 6, 8, 1, 9, 11]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle List
********************

--------------------
```

In [7]:

```python
import random

fix_list = [12, 9, 14, 8, 6, 1, 11, 18, 6, 10]
length = len(fix_list)

#creeating a function for display and menu-driven program
def displayList():
    print("List:", fix_list)
    print(" ")

def menuDrivenProgram():
    print("*" * 20)
    print("0. Exit\n1. Linear Search\n2. Binary Search\n3. Bubble Sort\n4. S
    print("*" * 20)
    print(" ")

# --- Linear Search ---
def linearSearch(fix_list, inp_array):

    for i in range(0, length):
        if fix_list[i] == inp_array:
            print("The number is present at position", i)
            return i
        return -1

# --- Bubble Sort ---
# Binary Search in python
def binarySearch(array, x, low, high):

    # Repeat until the pointers low and high meet each other
    while low <= high:

        mid = low + (high - low)//2

        if array[mid] == x:
            return mid

        elif array[mid] < x:
            low = mid + 1

        else:
            high = mid - 1

    return -1

# --- Bubble Sort ---
def bubbleSort(fix_list):

    for i in range(0, length):
        for j in range(0, length-i-1):

            if fix_list[j] > fix_list[j+1]:

                temp = fix_list[j]
                fix_list[j] = fix_list[j+1]
                fix_list[j+1] = temp
```

```
57
58  # --- Selection Sort ---
59  def selectionSort(fix_list):
60
61      for i in range(0, length-1):
62          minPos = i
63
64          for j in range(i+1, length):
65
66              if fix_list[minPos] > fix_list[j]:
67                  minPos = j
68
69                  t = fix_list[minPos]
70                  fix_list[minPos] = fix_list[i]
71                  fix_list[i] = t
72
73  # --- Insertion Sort ---
74  def insertionSort(array):
75      for step in range(1, len(array)):
76          key = array[step]
77          j = step -1
78
79          while (j>= 0 and key<array[j]):
80              array[j+1] = array[j]
81              j = j - 1
82
83          array[j + 1] = key
84
85  #Recalling of DocString
86  #Greeting at the beginning
87
88  variable1 = "SORTS & SEARCHES"
89
90  print(startgreet.__doc__)
91
92  variable2 = startgreet(variable1)
93  print("\n")
94
95  # --- Menu-Driven Program ---
96  while True:
97
98      menuDrivenProgram()
99
100     print("-" * 20)
101     option_input = int(input("Select an option:"))
102     print("-" * 20)
103     print(" ")
104
105     if option_input == 0:
106         endgreet()
107         break
108
109     elif option_input == 1:
110
111         inp_array = int(input("Enter the number you are searching for: "))
112         print("----------------------------------------")
113         linearSearch(fix_list, inp_array)
```

```
        print(" ")

    elif option_input == 2:

        inp_bin = int(input("Enter the number you are searching for: "))
        print("-"*40, "\n")

        result = binarySearch(fix_list, inp_bin, 0, len(fix_list)-1)

        if result != -1:
            print("Element is present at index " + str(result), "\n")
        else:
            print("Not found\n")

    elif option_input == 3:
        bubbleSort(fix_list)

    elif option_input == 4:
        selectionSort(fix_list)

    elif option_input == 5:
        pass

    elif option_input == 10:
        displayList()

    elif option_input == 11:
        random.shuffle(fix_list)
```

///Lab 4///

```
    ---------------------------------- SORTS & SEARCHES ----------------------
----------

    ------------------- Hi there! Welcome to the Program -------------------


********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [12, 9, 14, 8, 6, 1, 11, 18, 6, 10]

********************
0. Exit
1. Linear Search
```

```
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:1
--------------------

Enter the number you are searching for: 12
----------------------------------------
The number is present at position 0

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:3
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [1, 6, 6, 8, 9, 10, 11, 12, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
```

```
********************

--------------------
Select an option:2
--------------------

Enter the number you are searching for: 1
----------------------------------------

Element is present at index 0

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:11
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [8, 6, 6, 9, 12, 11, 1, 14, 18, 10]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:4
--------------------
```

```
********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [1, 6, 6, 8, 10, 9, 11, 12, 14, 18]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:11
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [11, 1, 14, 6, 6, 10, 12, 9, 18, 8]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
```

```
10. Display
11. Shuffle
********************

--------------------
Select an option:5
--------------------

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:10
--------------------

List: [11, 1, 14, 6, 6, 10, 12, 9, 18, 8]

********************
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Insertion Sort
10. Display
11. Shuffle
********************

--------------------
Select an option:0
--------------------

You have exited this program!

                  ///Thank You for Using this Program///
```

Referred from the Internet

In [8]:
```
def insertionSort(fix_list):

    # Traverse through 1 to len(arr)
    for i in range(1, len(fix_list)):

        key = fix_list[i]

        # Move elements of arr[0..i-1], that are
        # greater than key, to one position ahead
        # of their current position
        j = i-1
        while j >=0 and key < fix_list[j] :
                fix_list[j+1] = fix_list[j]
                j -= 1
        fix_list[j+1] = key

    print(fix_list)
```

In [9]:
```
insertionSort(fix_list)
```

```
[1, 6, 6, 8, 9, 10, 11, 12, 14, 18]
```

In [ ]:
```

```

Lines of Code:

Self - 144
Total - 144

Lab 5

Name: Krish Agarwal | Sakshi Modi
Registration Number: 21112016 | 21112022
Class: 2BSc DS A
Data Structures Using Python

Stack

In [1]:

```
"""
Start and End Greet

-------------------------------------------------------------
|This is an User defined program to display the prompt      |
|that are used before and after execution of each program.  |
|                                                           |
|Developed by                                               |
|Krish Agarwal                                              |
-------------------------------------------------------------
"""

#Creating of User Defined Function

def startgreet(var):
    '''                                     ///Lab  5/// '''

    print("\n    ----------------------------------", var, "----------------

    print("\n    -------------------- Hi there! Welcome to the Program -----


def endgreet():
    print("You have exited this program!\n")
    print("                   ///Thank You for Using this Program///")
```

In [7]:

```python
#importing of the random module
import random

#Stack
stack = []
global stack

#creeating a function for display and menu-driven program
def displayStack():
    print("Cards in hand:", stack)
    print(" ")

def menuDrivenProgram():
    print("*" * 20)
    print("0. Exit\n1. Push\n2. Pop\n3. Delete\n4. Pick a card\n10. Display"
    print("*" * 20)
    print(" ")

def populateArray(cards):
    stack.append(random.sample(range(1, 53), cards))

def appendCard(inp_cards):
    stack.append(inp_cards)

def pop():
    stack.pop(0)

#Recalling of DocString
#Greeting at the beginning
variable1 = "STACK"

print(startgreet.__doc__)

variable2 = startgreet(variable1)
print("\n")

# --- Menu-Driven Program ---
while True:

    menuDrivenProgram()

    print("-" * 20)
    option_input = int(input("Select an option:"))
    print("-" * 20)
    print(" ")

    if option_input == 0:
        endgreet()
        break

    if option_input == 10:
        displayStack()

    elif option_input == 1:
        cards = int(input("Enter the total number of cards in hand:"))
        populateArray(cards)
```

```
        print(" ")

    elif option_input == 2:
        pop()

    elif option_input == 3:
        pass

    elif option_input == 4:
        inp_cards = int(input("Enter the number of cards to be picked:"))
        appendCard(inp_cards)
```

```
                                    ///Lab  5///

    ---------------------------------- STACK ----------------------------------

    -------------------- Hi there! Welcome to the Program --------------------


********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:10
--------------------

Cards in hand: []

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:1
--------------------

Enter the total number of cards in hand:5

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************
```

```
--------------------
Select an option:10
--------------------

Cards in hand: [[41, 42, 45, 43, 9]]

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:2
--------------------

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:10
--------------------

Cards in hand: []

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:3
--------------------

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************
```

```
--------------------
Select an option:10
--------------------

Cards in hand: []

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:4
--------------------

Enter the number of cards to be picked:2
********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:10
--------------------

Cards in hand: [2]

********************
0. Exit
1. Push
2. Pop
3. Delete
4. Pick a card
10. Display
********************

--------------------
Select an option:0
--------------------

You have exited this program!

                  ///Thank You for Using this Program///
```

Queue

In [8]:
```
import random

def populate(total):

#generating random numbers inside the queue and
    que= random.sample(range(1,52),total)
    return que
```

In [9]:
```
# creating empty queue
que= []
```

In [10]:
```
tot=int(input("Enter number of elements you want in your queue:"))
que= populate(tot)
```

```
Enter number of elements you want in your queue:5
```

In [11]:
```
def display(que):
    return  que
```

In [12]:
```
display(que)
```

Out[12]:
```
[36, 32, 14, 4, 5]
```

In [13]:
```
def pop():
    que.pop(0)
    return que
```

In [14]:
```
display(que)
pop()
display(que)
```

Out[14]:
```
[32, 14, 4, 5]
```

In [15]:
```
def remove(number):
    for i in range(0,number):
        display(que)
        pop()
    return que
```

In [16]:
```
numbers=int(input("Enter number of elements you want to remove from your que
remove(numbers)
display(que)
```

```
Enter number of elements you want to remove from your queue:1
```

Out[16]:
```
[14, 4, 5]
```

In [17]:
```
def popStack():
    que.pop(len(que)-1)
    return que

```

In [18]:
```
display(que)
popStack()
display(que)
```

Out[18]: [14, 4]

In [19]:
```
def removeStack(number):
    for i in range(0,number):
        display(que)
        popStack()
    return que
```

In [20]:
```
numbers=int(input("Enter number of elements you want to remove from your que
removeStack(numbers)
display(que)
```

```
Enter number of elements you want to remove from your queue:1
```

Out[20]: [14]

In [30]:
```
from PIL import Image
# creating a object
def Show(no):
    no=str(no)
    location="C:/Users/KRISH/Desktop/Python/SEM2/Lab programs/Card Images.zi
    location=no+".GIF"
    disShow = Image.open(location)
    disShow.show()

```

In [31]:
```
1 Show(5)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_18636/1492421116.py in <module>
----> 1 Show(5)

~\AppData\Local\Temp/ipykernel_18636/2260477634.py in Show(no)
      5     location="C:/Users/KRISH/Desktop/Python/SEM2/Lab programs/Card Im
ages.zip/images"
      6     location=no+".GIF"
----> 7     disShow = Image.open(location)
      8     disShow.show()
      9

c:\users\krish\appdata\local\programs\python\python310\lib\site-packages\PIL
\Image.py in open(fp, mode, formats)
   3066
   3067     if filename:
-> 3068         fp = builtins.open(filename, "rb")
   3069         exclusive_fp = True
```

In [ ]:
```
1
```

Lines of Code:

Self - 59
Total - 59

Lab 6

Name: Gaurika Chopra | Krish Agarwal | Sakshi Modi
Registration Number: 21112009 | 21112016 | 21112022
Class: 2BSc DS A
Data Structures Using Python

Same row

In [11]:
```
def sameRow(row,col,newrow,newcol):
    if (row==newrow):
        print("Possible")
    return None
```

In [12]:
```
sameRow(5,6,6,7)
```

In [13]:
```
def sameCol(row,col,newrow,newcol):
    if (col==newcol):
        return ("Possible")
    return None
```

In [14]:
```
print(sameCol(5,5,6,5))
```

```
Possible
```

Rook

In [5]:
```
def Rook(row,col,newrow,newcol):

    if sameRow(row,col,newrow,newcol):
        return True
    elif sameCol(row,col,newrow,newcol):
        return True
    else:
        return("NOT POSSIBLE")
```

In [6]:
```
Rook(5, 6, 7, 8)
```

Out[6]:
```
'NOT POSSIBLE'
```

Bishop

In [15]:

```python
def kill(bishopX, bishopY, tokillX, tokillY) :
    # bishop's Position
    if (tokillX - bishopX == tokillY - bishopY) :
        print("Bishop can kill!")

    elif (-tokillX + bishopX == tokillY - bishopY):
         print("Bishop can kill!")

    else:
        print("No, Bishop has restricted movement!")
```

In [16]:

```python
bX= int(input("Enter the X coordinate of BISHOP: "))
bY= int(input("Enter the Y coordinate of BISHOP: "))
tkX= int(input("Enter the X coordinate of ToKILL-PAWN: "))
tkY= int(input("Enter the Y coordinate of ToKILL-PAWN: "))
print("-" * 40)
print(" ")

print("The coordinates of our bishop is: ", bX, ",", bY)
print("The coordinates of the pawn to kill: ", tkX,",", tkY)
print(" ")

kill(bX, bY, tkX, tkY)
```

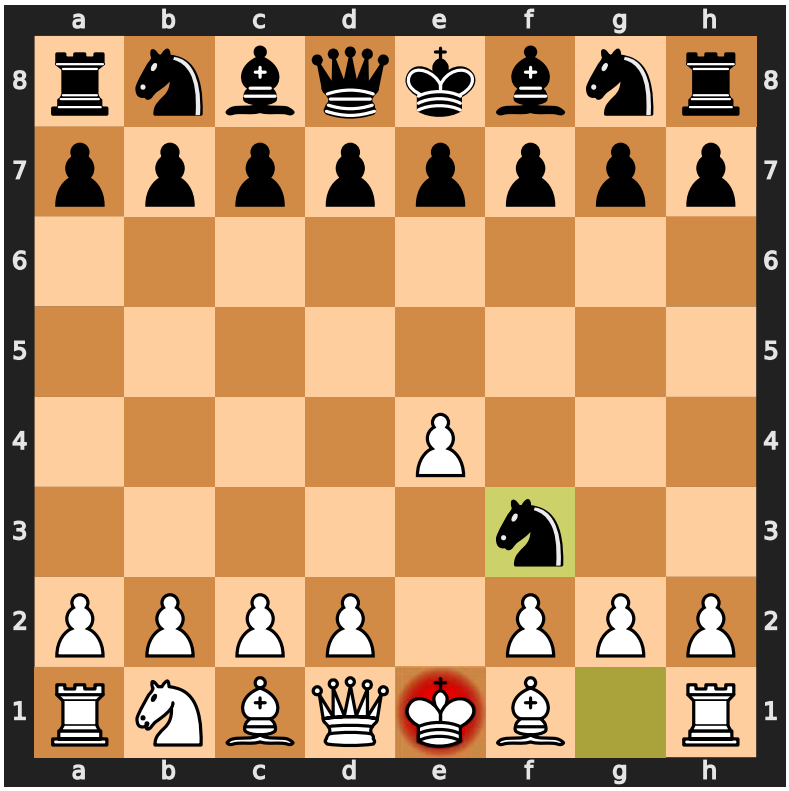
```
Enter the X coordinate of BISHOP: 5
Enter the Y coordinate of BISHOP: 6
Enter the X coordinate of ToKILL-PAWN: 7
Enter the Y coordinate of ToKILL-PAWN: 8
----------------------------------------

The coordinates of our bishop is:  5 , 6
The coordinates of the pawn to kill:  7 , 8

Bishop can kill!
```

Chess Board

In [17]:

```
import chess

board = chess.Board()

board.legal_moves

#Pawn
board.push_san("e4")

#Knight
Nf3 = chess.Move.from_uci("g1f3")
board.push(Nf3)

board
```

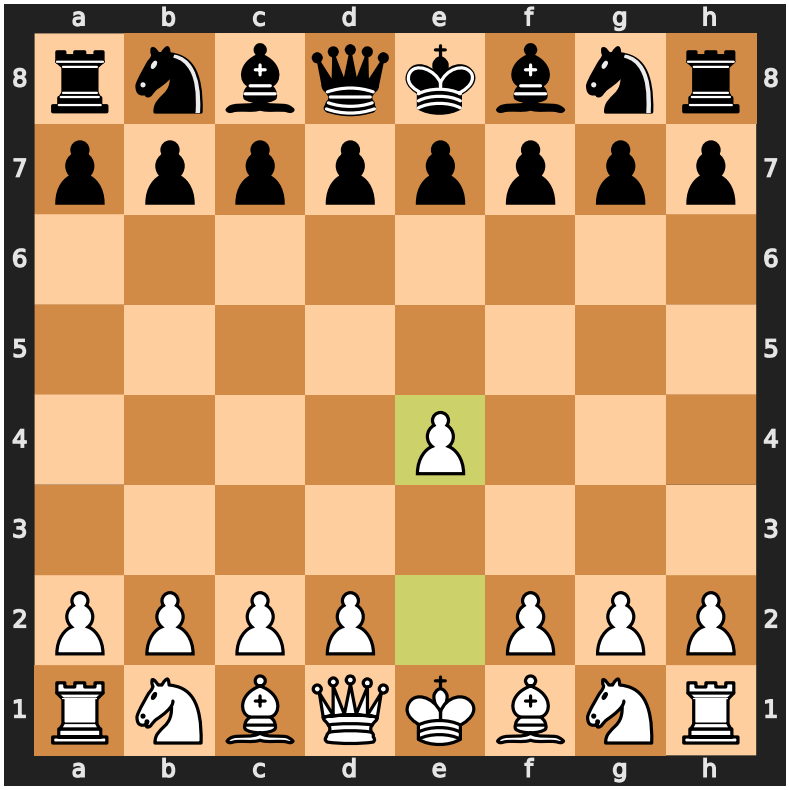
Out[17]:



In [18]:

```
#Unmaking the previous move
board.pop()

board
```

Out[18]:



In [ ]:

```
1
```

Lines of Code:

Self - 118
Inspired - 51
Total - 169

Lab 7

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS A
Data Structures Using Python

In [4]:

```
"""
Start and End Greet

------------------------------------------------------------
|This is an User defined program to display the prompt     |
|that are used before and after execution of each program. |
|                                                          |
|Developed by                                              |
|Krish Agarwal                                             |
------------------------------------------------------------
"""

#Creating of User Defined Function

def startgreet(var):
    '''                                     ///Lab  4/// '''

    print("\n    ----------------------------------", var, "----------------

    print("\n    -------------------- Hi there! Welcome to the Program -----


def endgreet():
    print("You have exited this program!\n")
    print("                   ///Thank You for Using this Program///")

def menuDrivenProgram():
    print("1.Insert at beginning\n2. Insert in between\n3. Insert at end\n4.
```

In [5]:

```python
# Create a node
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:

    def __init__(self):
        self.head = None

    # Insert at the beginning
    def insertAtBeginning(self, new_data):
        new_node = Node(new_data)

        new_node.next = self.head
        self.head = new_node

    # Insert after a node
    def insertAfter(self, prev_node, new_data):

        if prev_node is None:
            print("The given previous node must inLinkedList.")
            return

        new_node = Node(new_data)
        new_node.next = prev_node.next
        prev_node.next = new_node

    # Insert at the end
    def insertAtEnd(self, new_data):
        new_node = Node(new_data)

        if self.head is None:
            self.head = new_node
            return

        last = self.head
        while (last.next):
            last = last.next

        last.next = new_node

    # Deleting a node
    def deleteNode(self, position):

        if self.head is None:
            return

        temp = self.head

        if position == 0:
            self.head = temp.next
            temp = None
            return
```

Referred from net

```
57
58          # Find the key to be deleted
59          for i in range(position - 1):
60              temp = temp.next
61              if temp is None:
62                  break
63
64          # If the key is not present
65          if temp is None:
66              return
67
68          if temp.next is None:
69              return
70
71          next = temp.next.next
72
73          temp.next = None
74
75          temp.next = next
76
77      # Search an element
78      def search(self, key):
79
80          current = self.head
81
82          while current is not None:
83              if current.data == key:
84                  return True
85
86              current = current.next
87
88          return False
89
90      # Sort the linked list
91      def sortLinkedList(self, head):
92          current = head
93          index = Node(None)
94
95          if head is None:
96              return
97          else:
98              while current is not None:
99                  # index points to the node next to current
100                 index = current.next
101
102                 while index is not None:
103                     if current.data > index.data:
104                         current.data, index.data = index.data, current.data
105
106                     index = index.next
107                 current = current.next
108
109     # Print the linked list
110     def printList(self):
111         temp = self.head
112         while (temp):
113             print(str(temp.data) + " ", end="")
```

Referred from the net

```
            temp = temp.next

#Recalling of DocString
#Greeting at the beginning

variable1 = "Fixed List"

print(startgreet.__doc__)

variable2 = startgreet(variable1)
print("\n")

if __name__ == '__main__':

    l = LinkedList()
    l.insertAtEnd(1)
    l.insertAtBeginning(2)
    l.insertAtBeginning(3)
    l.insertAtEnd(4)
    l.insertAfter(l.head.next, 5)

    print('linked list:')
    l.printList()

    print("\nAfter deleting an element:")
    l.deleteNode(3)
    l.printList()

    print()
    item_to_find = 3
    if l.search(item_to_find):
        print(str(item_to_find) + " is found")
    else:
        print(str(item_to_find) + " is not found")

    l.sortLinkedList(l.head)
    print("Sorted List: ")
    l.printList()
```

///Lab 4///

```
    ---------------------------------- Fixed List ------------------------------
----

    -------------------- Hi there! Welcome to the Program --------------------


linked list:
3 2 5 1 4
After deleting an element:
3 2 5 4
3 is found
Sorted List:
2 3 4 5
```

In [ ]:

```
1
```

Lines of Code:

Self - 32
Total - 32

Lab 8

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS A
Data Structures Using Python

Q1) Printing the array

In [1]:
```
array = [[0, 1, 1, 0],
         [1, 0, 0, 0],
         [0, 1, 0, 0],
         [1, 1, 1, 0]]

for i in array:
    for j in i:
        print(j, end = " ")
    print(" ")
```

```
0 1 1 0
1 0 0 0
0 1 0 0
1 1 1 0
```

Q2) One step

In [30]:

```
.print("="*25)
inp1 = int(input("Enter the Source: "))
inp2 = int(input("Enter the Destination: "))
print("="*25, "\n")

#A/1 = "Pune"
#B/2 = "Jaipur"
#C/3 = "Kochi"
#D/4 = "Mumbai"

if array[inp1-1][inp2-1] == 1:
    print("Travel is possible")
    print("\n", array)

else:
    print("Travel is not possible")
    print("\n", array)
```

```
=========================
Enter the Source: 1
Enter the Destination: 4
=========================

Travel is not possible

 [[0, 1, 1, 0], [1, 0, 0, 0], [0, 1, 0, 0], [1, 1, 1, 0]]
```

Q3) Two Step

In [ ]:

```
print("="*25)
inp1 = int(input("Enter the Source: "))
inp2 = int(input("Enter the Destination: "))
print("="*25, "\n")

for i in range(0, len(array)):

```

Lines of Code:

Self - 17
Total - 17

Lab9

Name: Krish Agarwal | Krish Goyal | Abhishek Tiwari | Dhairyashil
Registration Number: 21112016 |21112015 | 21112038 | 21112006
Class: 2BSc DS A
Data Structures using Python

In [1]:
```
with open('note.txt') as f:
    lines = f.readlines()
    print(lines)
```

```
['1, Krish, 1000\n', '2, Sne, 2000\n', '3, Sakshi, 3000\n', '4, Shreyans, 400
0']
```

In [2]:
```
with open('note.txt') as f:
    line = f.readline()
    while line:
        line = f.readline()
        print(line)
```

```
2, Sne, 2000

3, Sakshi, 3000

4, Shreyans, 4000
```

In [3]:
```
with open('note.txt', encoding='utf8') as f:
    for line in f:
        print(line.strip())
```

```
1, Krish, 1000
2, Sne, 2000
3, Sakshi, 3000
4, Shreyans, 4000
```

In [4]:
```
f = open("C:/Users/KRISH/Desktop/Python/SEM2/Lab programs/note.txt", "r")
print(f.read())
```

```
1, Krish, 1000
2, Sne, 2000
3, Sakshi, 3000
4, Shreyans, 4000
```

In [7]:

```
1 myfile = open("note.txt")
2 txt = myfile.read()
3 print(txt)
4 myfile.close()
```

```
1, Krish, 1000
2, Sne, 2000
3, Sakshi, 3000
4, Shreyans, 4000
```

Lab Exam (Group2)

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS A
Data Structures Using Python

In [27]:
```
1 def func1(n):
2     for i in range(n+1):
3         if i%2!=0:
4             print(i, end = " ")
5 
6 n = int(input("Enter A number:"))
7 func1(n)
```

```
Enter A number:17
1 3 5 7 9 11 13 15 17
```

In [ ]:
```
1 def func2(n):
2     for i in range
```

Lines of Code:

Self - 200
Total - 200

Name: Krish Agarwal
Registration Number: 21112016
Class: 2BSc DS
Menu-Driven Program for all Sorts & Searches

In [1]:

```
def menuDrivenProgram():
    print("-" * 20)
    print("0. Exit")
    print("1. Linear Search")
    print("2. Binary Search")
    print("3. Bubble Sort")
    print("4. Selection Sort")
    print("5. Merge Sort")
    print("6. Insertion Sort")
    print("10. Display")
    print("11. Populate the Array")
    print("12. Shuffle the array")
    print("-" * 20)
    print(" ")
```

In [2]:

```python
# --- Importing necessary libraries ---
import random

# --- array ---
array = [92, 47, 40, 16, 82, 22, 12, 13, 56, 73]
length = len(array)

# --- Linear Search ---
def linearSearch(array, inp_array):

    for i in range(0, len(array)):
        if array[i] == inp_array:
            return True, i

    return False, -1

# --- Binary Search ---
def binarySearch(array, x, low, high):

    # Repeat until the pointers low and high meet each other
    while low <= high:

        mid = low + (high - low)//2

        if array[mid] == x:
            return mid

        elif array[mid] < x:
            low = mid + 1

        else:
            high = mid - 1

    return -1

# --- Bubble Sort ---
def bubbleSort(array):

    for i in range(0, len(array)):
        for j in range(0, len(array)-i-1):

            if array[j] > array[j+1]:

                temp = array[j]
                array[j] = array[j+1]
                array[j+1] = temp

# --- Selection Sort ---
def selectionSort(array, size):

        for step in range(size):
            mid_idx = step

            for i in range(step+1, size):
                if array[i]<array[mid_idx]:
                    mid_idx = i
```

```
57
58              (array[step], array[mid_idx]) = (array[mid_idx], array[step])
59
60  # --- Merge Sort ---
61  def mergeSort(array):
62      if len(array) > 1:
63
64          #  r is the point where the array is divided into two subarrays
65          r = len(array)//2
66          L = array[:r]
67          M = array[r:]
68
69          # Sort the two halves
70          mergeSort(L)
71          mergeSort(M)
72
73          i = j = k = 0
74
75          # Until we reach either end of either L or M, pick larger among
76          # elements L and M and place them in the correct position at A[p..r]
77          while i < len(L) and j < len(M):
78              if L[i] < M[j]:
79                  array[k] = L[i]
80                  i += 1
81              else:
82                  array[k] = M[j]
83                  j += 1
84              k += 1
85
86          # When we run out of elements in either L or M,
87          # pick up the remaining elements and put in A[p..r]
88          while i < len(L):
89              array[k] = L[i]
90              i += 1
91              k += 1
92
93          while j < len(M):
94              array[k] = M[j]
95              j += 1
96              k += 1
97
98  # --- Insertion Sort ---
99  def insertionSort(array):
100     for step in range(1, len(array)):
101         key = array[step]
102         j = step -1
103
104         while (j>= 0 and key<array[j]):
105             array[j+1] = array[j]
106             j = j - 1
107
108         array[j + 1] = key
109         print(array)
110
111 # --- Display ---
112 def displayArray():
113     print(array)
```

In [4]:

```
while True:

    menuDrivenProgram()

    option_input = int(input("Select an option:"))
    print(" ")

    # --- Exit ---
    if option_input == 0:
        print("\nYou have Exited the program!")
        break

    # --- Linear Search ---
    elif option_input == 1:

        inp_array = int(input("Enter the number you are searching for: "))
        print(" ")
        result,pos = linearSearch(array, inp_array)
        if result:
            print(inp_array," found at position [index] : ",pos)
            print()
        else:
            print("Number not found\n")

    # --- Binary Search ---
    elif option_input == 2:

        inp_bin = int(input("Enter the number you are searching for: "))
        print("-"*40, "\n")

        result = binarySearch(array, inp_bin, 0, len(array)-1)

        if result != -1:
            print("Element is present at index " + str(result), "\n")
        else:
            print("Not found\n")

    # --- Bubble Sort ---
    elif option_input == 3:
        bubbleSort(array)

    # --- Selection Sort ---
    elif option_input == 4:
        selectionSort(array, length)

    # --- Merge Sort ---
    elif option_input == 5:

        if __name__ == '__main__':
            mergeSort(array)

    # --- Insertion Sort ---
    elif option_input == 6:
        insertionSort(array)

    # --- Display ---
```

```
    elif option_input == 10:
        displayArray()
        print(" ")

    # --- Populating the array ---
    elif option_input == 11:

        rand_inp = int(input("Enter number of elements to add: "))
        print(" ")

        for i in range(0,rand_inp):
            array.append(random.randint(1, 99))

    # --- Shuffle the Array ---
    elif option_input == 12:

        random.shuffle(array)
```

```
--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[92, 47, 40, 16, 82, 22, 12, 13, 56, 73]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:1

Enter the number you are searching for: 40

40  found at position [index] :  2

--------------------
0. Exit
1. Linear Search
2. Binary Search
```

```
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:2

Enter the number you are searching for: 69
----------------------------------------

Not found

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:3

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[12, 13, 16, 22, 40, 47, 56, 73, 82, 92]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
```

```
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:12

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:4

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[12, 13, 16, 22, 40, 47, 56, 73, 82, 92]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:12

--------------------
0. Exit
1. Linear Search
2. Binary Search
```

```
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[22, 82, 40, 73, 47, 92, 12, 56, 13, 16]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:5

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[12, 13, 16, 22, 40, 47, 56, 73, 82, 92]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------
```

```
Select an option:12

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[56, 92, 82, 13, 22, 16, 40, 73, 47, 12]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:6

[56, 92, 82, 13, 22, 16, 40, 73, 47, 12]
[56, 82, 92, 13, 22, 16, 40, 73, 47, 12]
[13, 56, 82, 92, 22, 16, 40, 73, 47, 12]
[13, 22, 56, 82, 92, 16, 40, 73, 47, 12]
[13, 16, 22, 56, 82, 92, 40, 73, 47, 12]
[13, 16, 22, 40, 56, 82, 92, 73, 47, 12]
[13, 16, 22, 40, 56, 73, 82, 92, 47, 12]
[13, 16, 22, 40, 47, 56, 73, 82, 92, 12]
[12, 13, 16, 22, 40, 47, 56, 73, 82, 92]
--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------
```

```
Select an option:11

Enter number of elements to add: 3

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[12, 13, 16, 22, 40, 47, 56, 73, 82, 92, 80, 41, 76]

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:12

--------------------
0. Exit
1. Linear Search
2. Binary Search
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:10

[22, 56, 80, 76, 92, 47, 13, 40, 12, 82, 73, 41, 16]

--------------------
0. Exit
1. Linear Search
2. Binary Search
```

```
3. Bubble Sort
4. Selection Sort
5. Merge Sort
6. Insertion Sort
10. Display
11. Populate the Array
12. Shuffle the array
--------------------

Select an option:0


You have Exited the program!
```

In [ ]:

```
1
```

Lines of Code:

Self - 60
Inspired - 24

Binary Seach Tree

In [24]:

```
#creating a node
class Node:

    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

class BSTree:

    def __init__(self):
        self.root = None

    #insert at right position
    def insert(self, new_data):
        new_node = Node(new_data)

        #if the tree is empty, attach new node as root
        if (self.root == None):
            self.root = new_node
            return

        #code will only happen if tree is not empty
        curr = self.root

        while (curr!=None): #search for the dead end
            prev = curr

            if (curr.data>new_node.data):
                curr = curr.left
            else:
                curr = curr.right

        if (prev.data>new_node.data):
            prev.left = new_node
        else:
            prev.right = new_node

    def inOrder(self, rt):

        if (rt==None):
            return

        self.inOrder(rt.left)
        print(str(rt.data) + " ", end = "")
        self.inOrder(rt.right)

    def preOrder(self, rt):

        if (rt==None):
            return

        print(str(rt.data) + " ", end = "")
        self.preOrder(rt.left)
        self.preOrder(rt.right)

```

Referred from the net

```
    def postOrder(self, rt):

        if (rt==None):
            return

        self.postOrder(rt.right)
        self.postOrder(rt.left)
        print(str(rt.data) + " ", end = "")
```

In [30]:

```
if __name__ == '__main__':

    bst = BSTree()

    bst.insert(36)
    bst.insert(26)
    bst.insert(14)
    bst.insert(40)
    bst.insert(47)
    bst.insert(32)
    bst.insert(13)

    print("In Order Traversal:")
    bst.inOrder(bst.root)

    print("\n\nPre Order Traversal:")
    bst.preOrder(bst.root)

    print("\n\nPost Order Traversal:")
    bst.postOrder(bst.root)
```

```
In Order Traversal:
13 14 26 32 36 40 47

Pre Order Traversal:
36 26 14 13 32 40 47

Post Order Traversal:
47 40 32 13 14 26 36
```

In [ ]:

```
1
```

Lines of Code:

Inspired - 44
Total - 44

Depth First Search Algorithm (DFS ALgo)

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

In [1]:

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}

dfs(graph, '0')
```

```
0
1
3
4
2
2
```

Out[1]:

```
{'0', '1', '2', '3', '4'}
```

In [12]:

```python
def dfs(graph, start, visited):

    if visited[start] == 0:
        visited[start] = 1

    else:
        return visited

    print(start)
    print("Visited: {}".format(visited), "\n")

    for next in graph[start]:
        dfs(graph, next, visited)
    return visited


graph = {0: set([1, 2]),
         1: set([0, 3, 4]),
         2: set([0]),
         3: set([1]),
         4: set([2, 3])}

visited = [0, 0, 0, 0, 0]

dfs(graph, 0, visited)
```

```
0
Visited: [1, 0, 0, 0, 0]

1
Visited: [1, 1, 0, 0, 0]

3
Visited: [1, 1, 0, 1, 0]

4
Visited: [1, 1, 0, 1, 1]

2
Visited: [1, 1, 1, 1, 1]
```

Out[12]:

```
[1, 1, 1, 1, 1]
```

Lines of Code:

Self - 60
Inspired - 0

Patterns

In [1]:
```
for i in range(1, 10):
    if i%3!=0:
        print(i, end = " ")

    else:
        print(i)
```

```
1 2 3
4 5 6
7 8 9
```

In [41]:
```
num = 65

for i in range(1, 10):
    if i%3!=0:
        ch = chr(num)
        print(ch, end = " ")
        num +=1

    else:
        ch = chr(num)
        print(ch)
        num +=1
```

```
A B C
D E F
G H I
```

In [42]:

```python
n = int(input("Enter a number:"))
print("-"*20, "\n")

row = 1
j = 0

for i in range(1, n+1):
    if j<row:
        print(i, end = " ")
        j += 1

    else:
        print(" ")
        print(i, end = " ")
        row += 1
        j = 1
```

```
Enter a number:21
--------------------

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

In [52]:

```
n = int(input("Enter a number:"))
print("-"*20, "\n")

row = 1
j = 0
num = 65

for i in range(1, n+1):
    if j<row:
        ch = chr(num)
        print(ch, end = " ")
        j += 1
        num += 1

    else:
        print(" ")
        ch = chr(num)
        print(ch, end = " ")
        row += 1
        j = 1
        num += 1
```

```
Enter a number:66
--------------------

A
B C
D E F
G H I J
K L M N O
P Q R S T U
V W X Y Z [ \
] ^ _ ` a b c d
e f g h i j k l m
n o p q r s t u v w
x y z { | } ~
```

In [ ]:

```
n = int(input("Enter a number:"))
print("-"*20, "\n")

j = 0
row = 1


```

Lines of Code:

Self - 43
Total - 43

Peel and Layer

In [9]:

```python
def fullBox(input_box):

    for i in range(input_box):
        for j in range(input_box):
            print("*", end=" ")
        print(" ")

while True:

    print("0. Exit")
    print("1. Make a full Grid")
    print("2. Peel")
    print("3. Layer\n")

    option_inp = int(input("Select an option: "))
    print("--------------------\n")

    if option_inp == 0:
        print("Thank You for using the program.")
        print("Thank You!".center(100, " "))
        break

    elif option_inp == 1:
        input_box = int(input("Enter the order of the box: "))
        print(" ")
        fullBox(input_box)
        print(" ")

    elif option_inp == 2:
        input_box -= 1
        fullBox(input_box)
        print(" ")
        input_box -= 1
        fullBox(input_box)
        print(" ")

    elif option_inp == 3:
        input_box += 1
        fullBox(input_box)
        print(" ")
        input_box += 1
        fullBox(input_box)
        print(" ")
```

```
0. Exit
1. Make a full Grid
2. Peel
3. Layer

Select an option: 1
--------------------

Enter the order of the box: 1

*
```

```
0. Exit
1. Make a full Grid
2. Peel
3. Layer

Select an option: 1
--------------------

Enter the order of the box: 4

* * * *
* * * *
* * * *
* * * *

0. Exit
1. Make a full Grid
2. Peel
3. Layer

Select an option: 2
--------------------

* * *
* * *
* * *

* *
* *

0. Exit
1. Make a full Grid
2. Peel
3. Layer

Select an option: 3
--------------------

* * *
* * *
* * *

* * * *
* * * *
* * * *
* * * *

0. Exit
1. Make a full Grid
2. Peel
3. Layer

Select an option: 0
--------------------

Thank You for using the program.
                                                    Thank You!
```

In [ ]:

```
1
```

Lines of Code:

Self - 137
Total - 137

LinkedList + Stack + Queue

Linked List

In [1]:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def insertAtBeginning(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def insertAfter(self, prev_node, new_data):

        if prev_node is None:
            print("The number must be in the Linked List")
            return

        new_node = Node(new_data)
        new_data.next = prev_node.next
        prev_node.next = new_node

    def insertAtEnd(self, new_data):
        new_node = Node(new_data)

        if self.head is None:
            self.head = new_node
            return

        last = self.head
        while (last.next):
            last = last.next
        last.next = new_node

    def deleteNode(self, position):

        if self.head is None:
            return

        temp = self.head

        if position == 0:
            self.head = temp.next
            temp = None
            return

        for i in range(position - 1):
            temp = temp.next
            if temp is None:
                break

        if temp is None:
            return


```

```
        if temp.next is None:
            return

        next = temp.next.next
        temp.next = None
        temp.next = next

    def printList(self):
        temp = self.head
        while (temp):
            print(str(temp.data) + " ", end="")
            temp = temp.next

if __name__ == '__main__':

    l1 = LinkedList()
    l1.insertAtBeginning(1)
    l1.insertAtEnd(2)
    l1.insertAtEnd(3)
    l1.insertAtBeginning(0)
    l1.deleteNode(0)
    l1.printList()
```

```
1 2 3
```

Queue

In [2]:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Queue:
    def __init__(self):
        self.front = None
        self.rear = None

    def insert(self, new_data):
        new_node = Node(new_data)

        if self.rear == None:
            self.rear = new_node
            self.front = new_node

        self.rear.next = new_node
        self.rear = new_node

    def delete(self):
        temp = self.front
        self.front = temp.next

        if (self.front == None):
            self.rear = None

    def printList(self):
        temp = self.front

        while (temp!=None):
            print(str(temp.data) + " ", end = " ")
            temp = temp.next
```

Stack

In [3]:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Stack:
    def __init__(self):
        self.top = None

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.top
        self.top = new_node

    def pop(self):
        pre_pop = self.top
        self.top = self.top.next
        pre_pop.next = None
        return pre_pop.next

    def printList(self):
        temp = self.top

        while (temp):
            print(str(temp.data) + " ", end = " ")
            temp = temp.next
```

Lines of Code:

Self - 32
Inspired - 40
Total - 72

Heap Sort

In [5]:

```python
# --- ReHeapUp ---
def reheapUp(array, pos):
    if pos == 0:
        return

    p = (pos - 1)//2

    if (array[p]>array[pos]):
        return
    else:
        array[p], array[pos] = array[pos], array[p]

    reheapUp(array, p)

# --- ReHeapDown ---Referred
def reheapDown(array, pos):

    if pos>=len(array):
        return

    c1 = 2*pos + 1 #one child
    c2 = 2*pos + 2 #second child

    if c1 >= len(array):
        print(pos, c1)
        return

    if c2 >= len(array):
        print(pos, c2)
        return

    else:
        print(pos, c1, c2)

        if array[c1]>array[c2]:
            print(pos, c1, c2)
            l = c1

        else:
            l = c2

    if l>pos:
        array[l], array[pos] = array[pos], array[l]
        return reheapDown(array, l)

# --- Build Heap ---
def buildHeap(array):

    n = len(array)

    for i in range(1, n):
        reheapUp(array, i)

# --- Insert Heap ---
def insertHeap(array, i):
    n = len(array)
```

```
    array.append(i)
    reheapUp(array, n)
    print("Insert %d"%i)

# --- Delete Heap ---Referred
def deleteHeap(array, inp_del):

    result = search(array, inp_del)
    last = -1

    array[result], array[last] = array[last], array[result]
    print(array)
    array.pop(last)
    reheapDown(array, result)
    print()
    return array

# --- Search ---
def search(array, k):

    for i in range(0, len(array)):
        if array[i] == k:
            return i

    return -1
```

Lines of Code:

Self - 105
Inspired - 50
Total - 155

Heap (menu-driven)

In [1]:
```
1 array = [108, 2, 5, 1, 6, 93, 12]
```

In [2]:

```
def menuDrivenProgram():
    print("1. Insert Heap")
    print("2. Build Heap")
    print("3. Reheap Up")
    print("4. Reheap Down")
    print("5. Delete Heap")
    print("6. Search an element")
    print("10. Display Heap as array")
    print("0. Exit\n")

def displayHeap():
    print(array)
    print()

# --- ReHeapUp ---
def reheapUp(array, pos):
    if pos == 0:
        return

    p = (pos - 1)//2

    if (array[p]>array[pos]):
        return
    else:
        array[p], array[pos] = array[pos], array[p]

    reheapUp(array, p)

# --- ReHeapDown --- Referred
def reheapDown(array, pos):

    if pos>=len(array):
        return

    c1 = 2*pos + 1 #one child
    c2 = 2*pos + 2 #second child

    if c1 >= len(array):
        print(pos, c1)
        return

    if c2 >= len(array):
        print(pos, c2)
        return

    else:
        print(pos, c1, c2)

        if array[c1]>array[c2]:
            print(pos, c1, c2)
            l = c1

        else:
            l = c2

    if l>pos:
```

```
        array[l], array[pos] = array[pos], array[l]
        return reheapDown(array, l)

# --- Build Heap ---
def buildHeap(array):

    n = len(array)

    for i in range(1, n):
        reheapUp(array, i)

# --- Insert Heap ---
def insertHeap(array, i):
    n = len(array)
    array.append(i)
    reheapUp(array, n)
    print("Insert %d"%i)

# --- Delete Heap --- Referred
def deleteHeap(array, inp_del):

    result = search(array, inp_del)
    last = -1

    array[result], array[last] = array[last], array[result]
    print(array)
    array.pop(last)
    reheapDown(array, result)
    print()
    return array

# --- Heap Sort ---Referred
def heapSort(arr):
    n = len(arr)

    # Build max heap
    for i in range(n//2, -1, -1):
        heapify(arr, n, i)

    for i in range(n-1, 0, -1):
        # Swap
        arr[i], arr[0] = arr[0], arr[i]

        # Heapify root element
        heapify(arr, i, 0)

# --- Search ---
def search(array, k):

    for i in range(0, len(array)):
        if array[i] == k:
            return i

    return -1
```

In [4]:

```
while True:

    menuDrivenProgram()

    print("-"*30)
    input_option = int(input("Select an option: "))
    print("-"*30, "\n")

    if input_option == 0:
        print("Thank you for using the program!".center(115, " "))
        break

    elif input_option == 10:
        displayHeap()

    elif input_option == 1:
        inp_heap = int(input("Enter the number to be added: "))
        print()
        insertHeap(array, inp_heap)

    elif input_option == 2:

        buildHeap(array)

    elif input_option == 3:

        reheapUp(array, pos)

    elif input_option == 4:

        reheapDown(array, pos)

    elif input_option == 5:

        inp_del = int(input("Enter the number to be deleted: "))
        deleteHeap(array, inp_del)

    elif input_option == 6:

        inp_search = int(input("Enter the number to be deleted: "))
        search(array, inp_search)

    else:
        pass
```

```
1. Insert Heap
2. Build Heap
3. Reheap Up
4. Reheap Down
5. Delete Heap
6. Search an element
10. Display Heap as array
0. Exit

------------------------------
Select an option: 10
```

```
------------------------------

[108, 2, 5, 1, 6, 93, 12]

1. Insert Heap
2. Build Heap
3. Reheap Up
4. Reheap Down
5. Delete Heap
6. Search an element
10. Display Heap as array
0. Exit

------------------------------
Select an option: 0
------------------------------

                                        Thank you for using the program!
```

In [ ]:

```
1
```

Lab9

Name: Krish Agarwal | Krish Goyal | Abhishek Tiwari | Dhairyashil
Registration Number: 21112016 |21112015 | 21112038 | 21112006
Class: 2BSc DS A
Data Structures using Python

Probability

In [1]:
```
1 pip install matplotlib
```

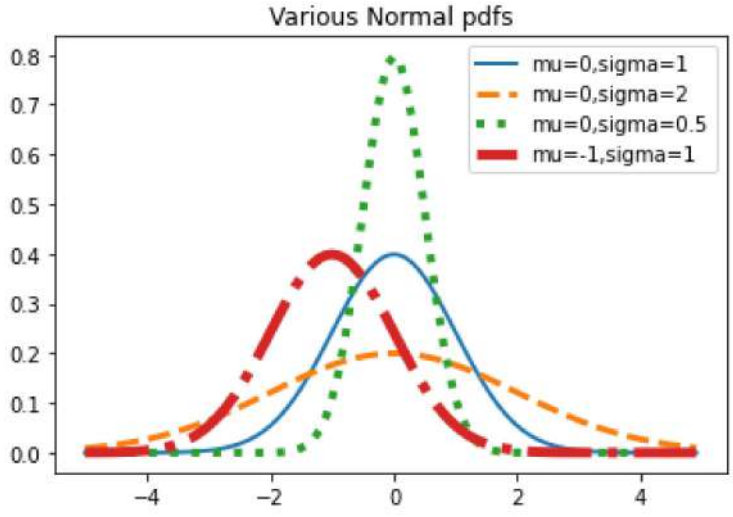
```
Requirement already satisfied: matplotlib in c:\users\krish\appdata\local\progr
ams\python\python310\lib\site-packages (3.5.2)Note: you may need to restart the
kernel to use updated packages.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)


Requirement already satisfied: fonttools>=4.22.0 in c:\users\krish\appdata\loca
l\programs\python\python310\lib\site-packages (from matplotlib) (4.33.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\krish\appdata\local
\programs\python\python310\lib\site-packages (from matplotlib) (3.0.6)
Requirement already satisfied: packaging>=20.0 in c:\users\krish\appdata\local
\programs\python\python310\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\krish\appdata\loca
l\programs\python\python310\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: numpy>=1.17 in c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages (from matplotlib) (1.22.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\krish\appdata\l
ocal\programs\python\python310\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\krish\appdata\local\pro
grams\python\python310\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\krish\appdata\local\pr
ograms\python\python310\lib\site-packages (from matplotlib) (9.1.1)
Requirement already satisfied: six>=1.5 in c:\users\krish\appdata\local\program
s\python\python310\lib\site-packages (from python-dateutil>=2.7->matplotlib)
(1.16.0)
```

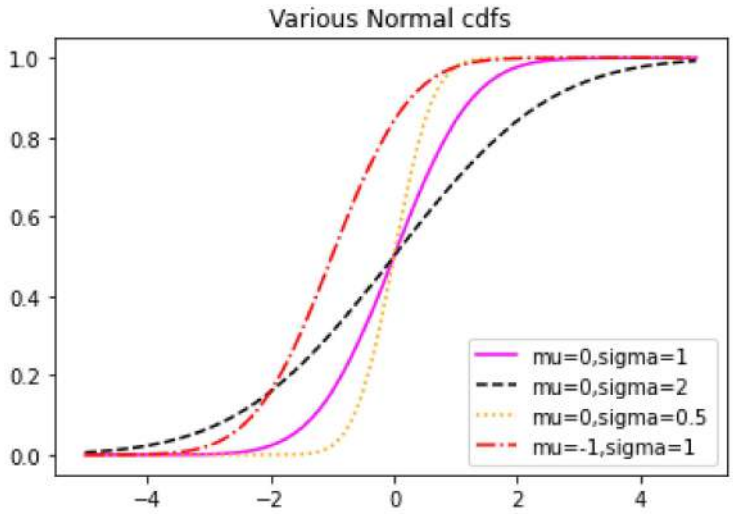
In [2]:

```
def uniform_cdf(x: float) -> float:
    """Returns the probability that a uniform random variable is <= x"""
    if x < 0:   return 0    # uniform random is never less than 0
    elif x < 1: return x    # e.g. P(X <= 0.4) = 0.4
    else:       return 1    # uniform random is always less than 1

import math
SQRT_TWO_PI = math.sqrt(2 * math.pi)

def normal_pdf(x: float, mu: float = 0, sigma: float = 1) -> float:
    return (math.exp(-(x-mu) ** 2 / 2 / sigma ** 2) / (SQRT_TWO_PI * sigma))

import matplotlib.pyplot as plt
xs = [x / 10.0 for x in range(-50, 50)]
plt.plot(xs,[normal_pdf(x,sigma=1) for x in xs],'-',label='mu=0,sigma=1', li
plt.plot(xs,[normal_pdf(x,sigma=2) for x in xs],'--',label='mu=0,sigma=2', l
plt.plot(xs,[normal_pdf(x,sigma=0.5) for x in xs],':',label='mu=0,sigma=0.5'
plt.plot(xs,[normal_pdf(x,mu=-1)   for x in xs],'-.',label='mu=-1,sigma=1',
plt.legend()
plt.title("Various Normal pdfs")
plt.show() #


plt.savefig('D:/Downloads/PDFs-of-Three-Normal-Distributions-Identical-Means
plt.gca().clear()
plt.close()
plt.clf()

def normal_cdf(x: float, mu: float = 0, sigma: float = 1) -> float:
    return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2

xs = [x / 10.0 for x in range(-50, 50)]
plt.plot(xs,[normal_cdf(x,sigma=1) for x in xs],'-',label='mu=0,sigma=1', co
plt.plot(xs,[normal_cdf(x,sigma=2) for x in xs],'--',label='mu=0,sigma=2', c
plt.plot(xs,[normal_cdf(x,sigma=0.5) for x in xs],':',label='mu=0,sigma=0.5'
plt.plot(xs,[normal_cdf(x,mu=-1) for x in xs],'-.',label='mu=-1,sigma=1', co
plt.legend(loc=4) # bottom right
plt.title("Various Normal cdfs")
plt.show() #


plt.close()
plt.gca().clear()
plt.clf()

def inverse_normal_cdf(p: float,
                       mu: float = 0,
                       sigma: float = 1,
                       tolerance: float = 0.00001) -> float:
    """Find approximate inverse using binary search"""

    # if not standard, compute standard and rescale
    if mu != 0 or sigma != 1:
        return mu + sigma * inverse_normal_cdf(p, tolerance=tolerance)

    low_z = -10.0                       # normal_cdf(-10) is (very close to)
```

```
    hi_z  =  10.0                       # normal_cdf(10)  is (very close to)
    while hi_z - low_z > tolerance:
        mid_z = (low_z + hi_z) / 2      # Consider the midpoint
        mid_p = normal_cdf(mid_z)       # and the cdf's value there
        if mid_p < p:
            low_z = mid_z               # Midpoint too low, search above it
        else:
            hi_z = mid_z                # Midpoint too high, search below it

    return mid_z


import random

def bernoulli_trial(p: float) -> int:
    """Returns 1 with probability p and 0 with probability 1-p"""
    return 1 if random.random() < p else 0

def binomial(n: int, p: float) -> int:
    """Returns the sum of n bernoulli(p) trials"""
    return sum(bernoulli_trial(p) for _ in range(n))

from collections import Counter

def binomial_histogram(p: float, n: int, num_points: int) -> None:
    """Picks points from a Binomial(n, p) and plots their histogram"""
    data = [binomial(n, p) for _ in range(num_points)]

    # use a bar chart to show the actual binomial samples
    histogram = Counter(data)
    plt.bar([x - 0.4 for x in histogram.keys()],
            [v / num_points for v in histogram.values()],
            0.8,
            color='0.75')

    mu = p * n
    sigma = math.sqrt(n * p * (1 - p))

    # use a line chart to show the normal approximation
    xs = range(min(data), max(data) + 1)
    ys = [normal_cdf(i + 0.5, mu, sigma) - normal_cdf(i - 0.5, mu, sigma)
          for i in xs]
    plt.plot(xs,ys)
    plt.title("Binomial Distribution vs. Normal Approximation")
    plt.show() #

def main():
    import enum, random

    # An Enum is a typed set of enumerated values. We can use them
    # to make our code more descriptive and readable.
    class Kid(enum.Enum):
        BOY = 0
        GIRL = 1

    def random_kid() -> Kid:
        return random.choice([Kid.BOY, Kid.GIRL])
```

```
114
115     both_girls = 0
116     older_girl = 0
117     either_girl = 0
118
119     random.seed(0)
120
121     for _ in range(10000):
122         younger = random_kid()
123         older = random_kid()
124         if older == Kid.GIRL:
125             older_girl += 1
126         if older == Kid.GIRL and younger == Kid.GIRL:
127             both_girls += 1
128         if older == Kid.GIRL or younger == Kid.GIRL:
129             either_girl += 1
130
131     print("P(both | older):", both_girls / older_girl)      # 0.514 ~ 1/2
132     print("P(both | either): ", both_girls / either_girl)  # 0.342 ~ 1/3
133
134
135
136     assert 0.48 < both_girls / older_girl < 0.52
137     assert 0.30 < both_girls / either_girl < 0.35
138
139     def uniform_pdf(x: float) -> float:
140         return 1 if 0 <= x < 1 else 0
141
142 if __name__ == "__main__": main()
```






```
P(both | older): 0.5007089325501317
P(both | either):  0.3311897106109325
```

```
<Figure size 432x288 with 0 Axes>
```

Linear Algebra

In [3]:

```python
from typing import List

Vector = List[float]

height_weight_age = [70,  # inches,
                     170, # pounds,
                     40 ] # years

grades = [95,   # exam1
          80,   # exam2
          75,   # exam3
          62 ]  # exam4

def add(v: Vector, w: Vector) -> Vector:
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"

    return [v_i + w_i for v_i, w_i in zip(v, w)]

assert add([1, 2, 3], [4, 5, 6]) == [5, 7, 9]

def subtract(v: Vector, w: Vector) -> Vector:
    """Subtracts corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"

    return [v_i - w_i for v_i, w_i in zip(v, w)]

assert subtract([5, 7, 9], [4, 5, 6]) == [1, 2, 3]

def vector_sum(vectors: List[Vector]) -> Vector:
    """Sums all corresponding elements"""
    # Check that vectors is not empty
    assert vectors, "no vectors provided!"

    # Check the vectors are all the same size
    num_elements = len(vectors[0])
    assert all(len(v) == num_elements for v in vectors), "different sizes!"

    # the i-th element of the result is the sum of every vector[i]
    return [sum(vector[i] for vector in vectors)
            for i in range(num_elements)]

assert vector_sum([[1, 2], [3, 4], [5, 6], [7, 8]]) == [16, 20]

def scalar_multiply(c: float, v: Vector) -> Vector:
    """Multiplies every element by c"""
    return [c * v_i for v_i in v]

assert scalar_multiply(2, [1, 2, 3]) == [2, 4, 6]

def vector_mean(vectors: List[Vector]) -> Vector:
    """Computes the element-wise average"""
    n = len(vectors)
    return scalar_multiply(1/n, vector_sum(vectors))

assert vector_mean([[1, 2], [3, 4], [5, 6]]) == [3, 4]
```

```python

def dot(v: Vector, w: Vector) -> float:
    """Computes v_1 * w_1 + ... + v_n * w_n"""
    assert len(v) == len(w), "vectors must be same length"

    return sum(v_i * w_i for v_i, w_i in zip(v, w))

assert dot([1, 2, 3], [4, 5, 6]) == 32  # 1 * 4 + 2 * 5 + 3 * 6

def sum_of_squares(v: Vector) -> float:
    """Returns v_1 * v_1 + ... + v_n * v_n"""
    return dot(v, v)

assert sum_of_squares([1, 2, 3]) == 14  # 1 * 1 + 2 * 2 + 3 * 3

import math

def magnitude(v: Vector) -> float:
    """Returns the magnitude (or length) of v"""
    return math.sqrt(sum_of_squares(v))   # math.sqrt is square root functic

assert magnitude([3, 4]) == 5

def squared_distance(v: Vector, w: Vector) -> float:
    """Computes (v_1 - w_1) ** 2 + ... + (v_n - w_n) ** 2"""
    return sum_of_squares(subtract(v, w))

def distance(v: Vector, w: Vector) -> float:
    """Computes the distance between v and w"""
    return math.sqrt(squared_distance(v, w))


def distance(v: Vector, w: Vector) -> float:  # type: ignore
    return magnitude(subtract(v, w))

# Another type alias
Matrix = List[List[float]]

A = [[1, 2, 3],  # A has 2 rows and 3 columns
     [4, 5, 6]]

B = [[1, 2],     # B has 3 rows and 2 columns
     [3, 4],
     [5, 6]]

from typing import Tuple

def shape(A: Matrix) -> Tuple[int, int]:
    """Returns (# of rows of A, # of columns of A)"""
    num_rows = len(A)
    num_cols = len(A[0]) if A else 0   # number of elements in first row
    return num_rows, num_cols

assert shape([[1, 2, 3], [4, 5, 6]]) == (2, 3)  # 2 rows, 3 columns

def get_row(A: Matrix, i: int) -> Vector:
    """Returns the i-th row of A (as a Vector)"""

```

```
    return A[i]             # A[i] is already the ith row

def get_column(A: Matrix, j: int) -> Vector:
    """Returns the j-th column of A (as a Vector)"""
    return [A_i[j]          # jth element of row A_i
            for A_i in A]   # for each row A_i

from typing import Callable

def make_matrix(num_rows: int,
                num_cols: int,
                entry_fn: Callable[[int, int], float]) -> Matrix:
    """
    Returns a num_rows x num_cols matrix
    whose (i,j)-th entry is entry_fn(i, j)
    """
    return [[entry_fn(i, j)             # given i, create a list
             for j in range(num_cols)]  #   [entry_fn(i, 0), ... ]
            for i in range(num_rows)]   # create one list for each i

def identity_matrix(n: int) -> Matrix:
    """Returns the n x n identity matrix"""
    return make_matrix(n, n, lambda i, j: 1 if i == j else 0)

assert identity_matrix(5) == [[1, 0, 0, 0, 0],
                              [0, 1, 0, 0, 0],
                              [0, 0, 1, 0, 0],
                              [0, 0, 0, 1, 0],
                              [0, 0, 0, 0, 1]]

data = [[70, 170, 40],
        [65, 120, 26],
        [77, 250, 19],
        # ....
       ]

friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
               (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]

#            user 0  1  2  3  4  5  6  7  8  9
#
friend_matrix = [[0, 1, 1, 0, 0, 0, 0, 0, 0, 0],  # user 0
                 [1, 0, 1, 1, 0, 0, 0, 0, 0, 0],  # user 1
                 [1, 1, 0, 1, 0, 0, 0, 0, 0, 0],  # user 2
                 [0, 1, 1, 0, 1, 0, 0, 0, 0, 0],  # user 3
                 [0, 0, 0, 1, 0, 1, 0, 0, 0, 0],  # user 4
                 [0, 0, 0, 0, 1, 0, 1, 1, 0, 0],  # user 5
                 [0, 0, 0, 0, 0, 1, 0, 0, 1, 0],  # user 6
                 [0, 0, 0, 0, 0, 1, 0, 0, 1, 0],  # user 7
                 [0, 0, 0, 0, 0, 0, 1, 1, 0, 1],  # user 8
                 [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]]  # user 9

assert friend_matrix[0][2] == 1, "0 and 2 are friends"
assert friend_matrix[0][8] == 0, "0 and 8 are not friends"

# only need to look at one row
friends_of_five = [i
```

```
                    for i, is_friend in enumerate(friend_matrix[5])
                    if is_friend]

friends_of_five
```

Out[3]: [4, 6, 7]

Type *Markdown* and LaTeX: α^2

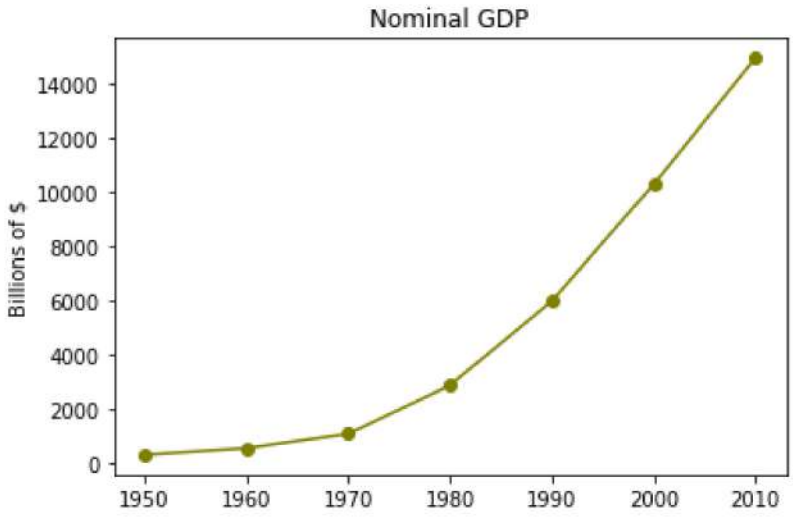
Vizualization Data

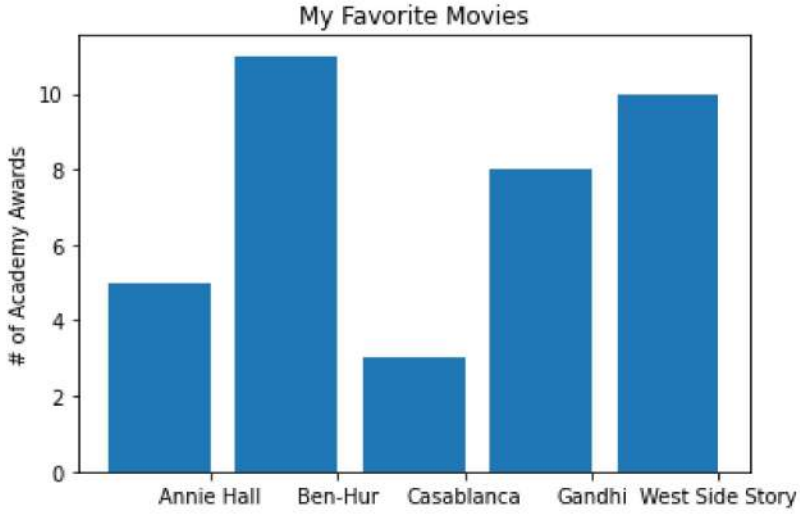
In [13]:

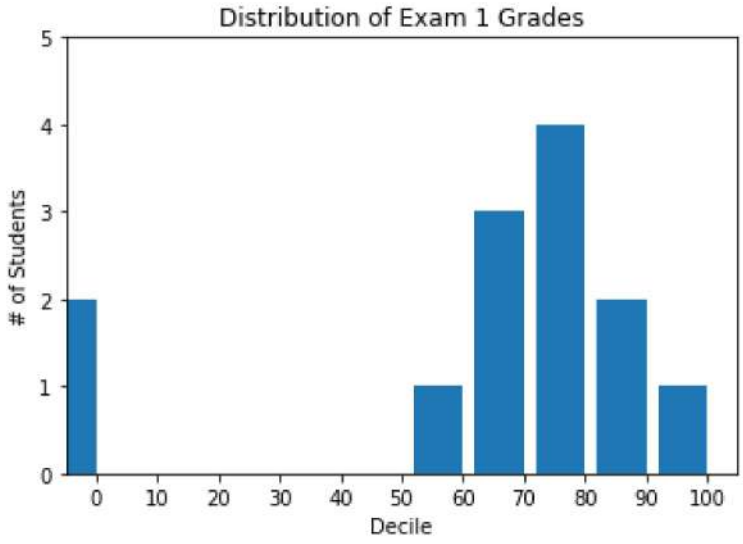
```
import matplotlib.pyplot as plt
from collections import Counter

def make_chart_simple_line_chart():

    years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
    gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]

    # create a line chart, years on x-axis, gdp on y-axis
    plt.plot(years, gdp, color='olive', marker='o', linestyle='solid')

    # add a title
    plt.title("Nominal GDP")

    # add a label to the y-axis
    plt.ylabel("Billions of $")
    plt.show()


def make_chart_simple_bar_chart():

    movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side St
    num_oscars = [5, 11, 3, 8, 10]

    # bars are by default width 0.8, so we'll add 0.1 to the left coordinate
    # so that each bar is centered
    xs = [i + 0.1 for i, _ in enumerate(movies)]

    # plot bars with left x-coordinates [xs], heights [num_oscars]
    plt.bar(xs, num_oscars)
    plt.ylabel("# of Academy Awards")
    plt.title("My Favorite Movies")

    # label x-axis with movie names at bar centers
    plt.xticks([i + 0.5 for i, _ in enumerate(movies)], movies)

    plt.show()

def make_chart_histogram():
    grades = [83,95,91,87,70,0,85,82,100,67,73,77,0]
    decile = lambda grade: grade // 10 * 10
    histogram = Counter(decile(grade) for grade in grades)

    plt.bar([x - 4 for x in histogram.keys()], # shift each bar to the left
            histogram.values(),                # give each bar its correct h
            8)                                 # give each bar a width of 8
    plt.axis([-5, 105, 0, 5])                  # x-axis from -5 to 105,
                                               # y-axis from 0 to 5
    plt.xticks([10 * i for i in range(11)])    # x-axis labels at 0, 10, ...
    plt.xlabel("Decile")
    plt.ylabel("# of Students")
    plt.title("Distribution of Exam 1 Grades")
    plt.show()

def make_chart_misleading_y_axis(mislead=True):

```

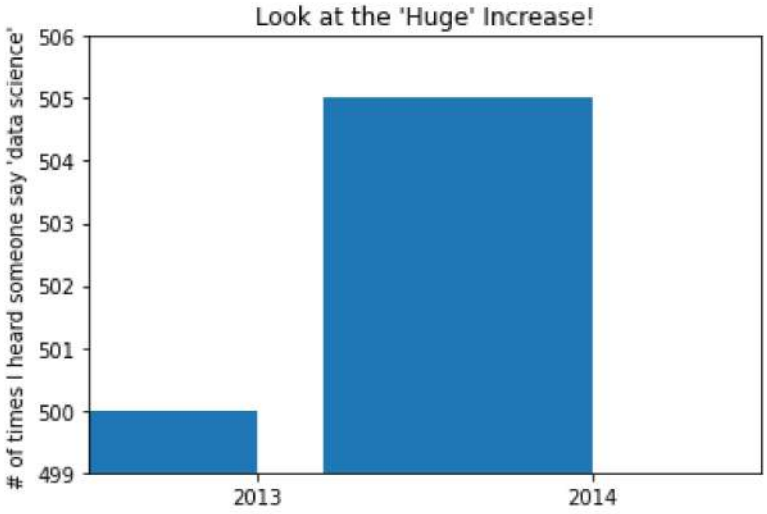
```
    mentions = [500, 505]
    years = [2013, 2014]

    plt.bar([2012.6, 2013.6], mentions, 0.8)
    plt.xticks(years)
    plt.ylabel("# of times I heard someone say 'data science'")

    # if you don't do this, matplotlib will label the x-axis 0, 1
    # and then add a +2.013e3 off in the corner (bad matplotlib!)
    plt.ticklabel_format(useOffset=False)

    if mislead:
        # misleading y-axis only shows the part above 500
        plt.axis([2012.5,2014.5,499,506])
        plt.title("Look at the 'Huge' Increase!")
    else:
        plt.axis([2012.5,2014.5,0,550])
        plt.title("Not So Huge Anymore.")
    plt.show()

def make_chart_several_line_charts():

    variance     = [1,2,4,8,16,32,64,128,256]
    bias_squared = [256,128,64,32,16,8,4,2,1]
    total_error  = [x + y for x, y in zip(variance, bias_squared)]

    xs = range(len(variance))

    # we can make multiple calls to plt.plot
    # to show multiple series on the same chart
    plt.plot(xs, variance,     'g-',  label='variance')    # green solid lir
    plt.plot(xs, bias_squared, 'r-.', label='bias^2')      # red dot-dashed
    plt.plot(xs, total_error,  'b:',  label='total error') # blue dotted lir

    # because we've assigned labels to each series
    # we can get a legend for free
    # loc=9 means "top center"
    plt.legend(loc=9)
    plt.xlabel("model complexity")
    plt.title("The Bias-Variance Tradeoff")
    plt.show()

def make_chart_scatter_plot():

    friends = [ 70, 65, 72, 63, 71, 64, 60, 64, 67]
    minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
    labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

    plt.scatter(friends, minutes)

    # label each point
    for label, friend_count, minute_count in zip(labels, friends, minutes):
        plt.annotate(label,
                     xy=(friend_count, minute_count), # put the label with i
                     xytext=(5, -5), # but slightly offset
                     textcoords='offset points')

```

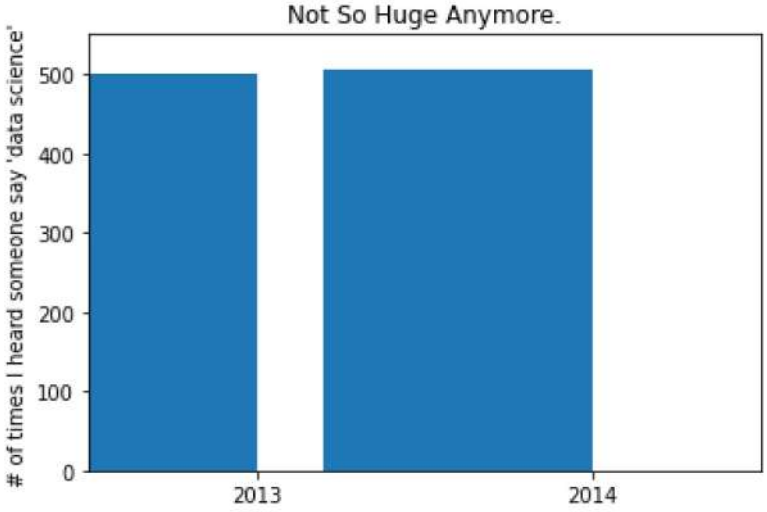
```
    plt.title("Daily Minutes vs. Number of Friends")
    plt.xlabel("# of friends")
    plt.ylabel("daily minutes spent on the site")
    plt.show()

def make_chart_scatterplot_axes(equal_axes=False):

    test_1_grades = [ 99, 90, 85, 97, 80]
    test_2_grades = [100, 85, 60, 90, 70]

    plt.scatter(test_1_grades, test_2_grades)
    plt.xlabel("test 1 grade")
    plt.ylabel("test 2 grade")

    if equal_axes:
        plt.title("Axes Are Comparable")
        plt.axis("equal")
    else:
        plt.title("Axes Aren't Comparable")

    plt.show()

def make_chart_pie_chart():

    plt.pie([0.95, 0.05], labels=["Uses pie charts", "Knows better"])

    # make sure pie is a circle and not an oval
    plt.axis("equal")
    plt.show()


if __name__ == "__main__":

    make_chart_simple_line_chart()

    make_chart_simple_bar_chart()

    make_chart_histogram()

    make_chart_misleading_y_axis(mislead=True)

    make_chart_misleading_y_axis(mislead=False)

    make_chart_several_line_charts()

    make_chart_scatterplot_axes(equal_axes=False)

    make_chart_scatterplot_axes(equal_axes=True)

    make_chart_pie_chart()
```

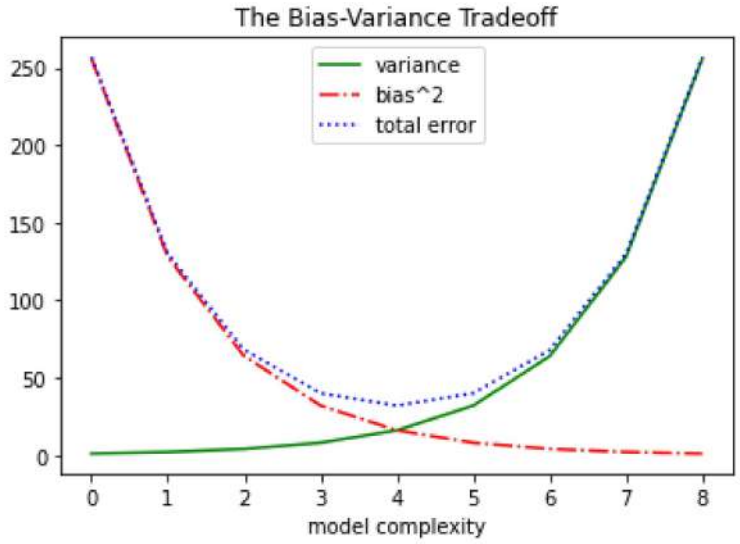

Nominal GDP
Billions of $
14000
12000
10000
8000
6000
4000
2000
0
1950
1960
1970
1980
1990
2000
2010

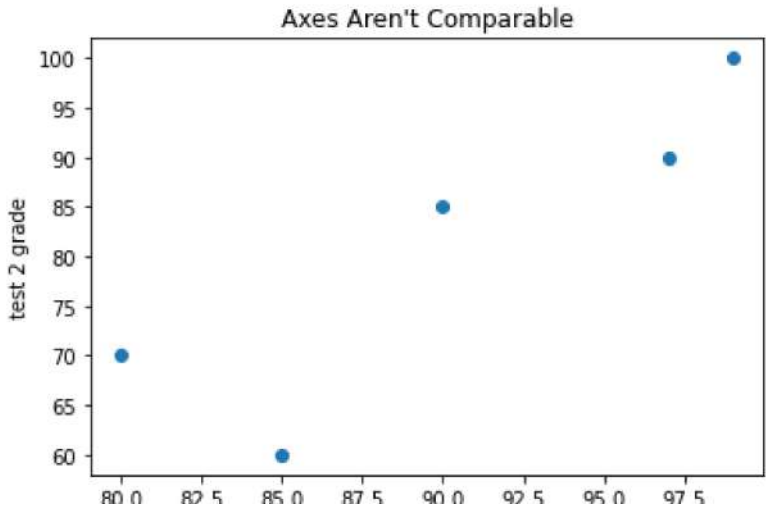

My Favorite Movies
# of Academy Awards
10
8
6
4
2
0
Annie Hall
Ben-Hur
Casablanca
Gandhi
West Side Story

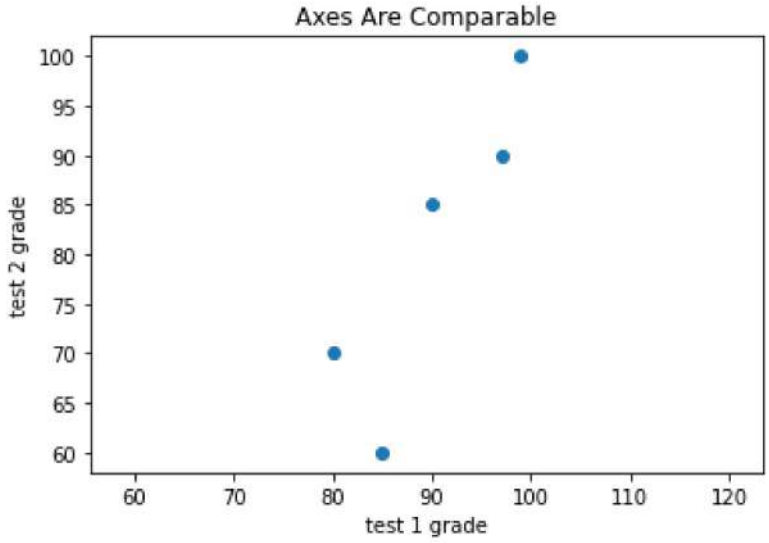

Distribution of Exam 1 Grades
# of Students
5
4
3
2
1
0
0
10
20
30
40
50
60
70
80
90
100
Decile

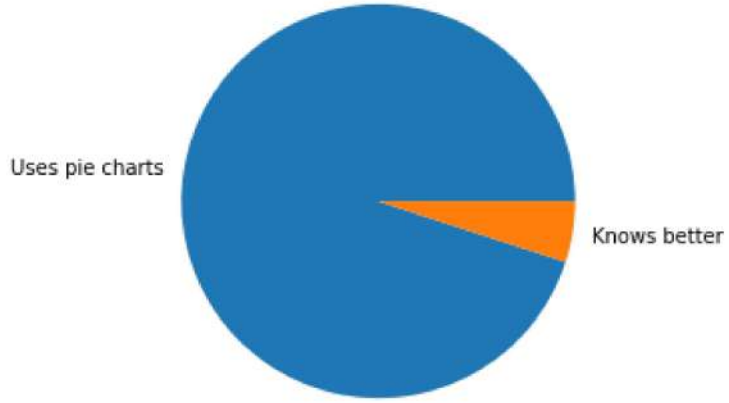

Look at the 'Huge' Increase!
# of times I heard someone say 'data science'
506
505
504
503
502
501
500
499
2013
2014


Not So Huge Anymore.
# of times I heard someone say 'data science'
500
400
300
200
100
0
2013
2014


The Bias-Variance Tradeoff
variance
bias^2
total error
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
model complexity







In [ ]:
```
1 pip install linear_algebra.py
```

Type *Markdown* and LaTeX: α^2

In [14]:

```
from collections import Counter, defaultdict
from functools import partial, reduce
from linear_algebra import shape, get_row, get_column, make_matrix, \
    vector_mean, vector_sum, dot, magnitude, vector_subtract, scalar_multipl
from stats import correlation, standard_deviation, mean
from probability import inverse_normal_cdf
from gradient_descent import maximize_batch
import math, random, csv
import matplotlib.pyplot as plt
import dateutil.parser

def bucketize(point, bucket_size):
    """floor the point to the next lower multiple of bucket_size"""
    return bucket_size * math.floor(point / bucket_size)

def make_histogram(points, bucket_size):
    """buckets the points and counts how many in each bucket"""
    return Counter(bucketize(point, bucket_size) for point in points)

def plot_histogram(points, bucket_size, title=""):
    histogram = make_histogram(points, bucket_size)
    plt.bar(histogram.keys(), histogram.values(), width=bucket_size)
    plt.title(title)
    plt.show()

def compare_two_distributions():

    random.seed(0)

    uniform = [random.randrange(-100,101) for _ in range(200)]
    normal = [57 * inverse_normal_cdf(random.random())
              for _ in range(200)]

    plot_histogram(uniform, 10, "Uniform Histogram")
    plot_histogram(normal, 10, "Normal Histogram")

def random_normal():
    """returns a random draw from a standard normal distribution"""
    return inverse_normal_cdf(random.random())

xs = [random_normal() for _ in range(1000)]
ys1 = [ x + random_normal() / 2 for x in xs]
ys2 = [-x + random_normal() / 2 for x in xs]


def scatter():
    plt.scatter(xs, ys1, marker='.', color='black', label='ys1')
    plt.scatter(xs, ys2, marker='.', color='gray',  label='ys2')
    plt.xlabel('xs')
    plt.ylabel('ys')
    plt.legend(loc=9)
    plt.show()

def correlation_matrix(data):
    """returns the num_columns x num_columns matrix whose (i, j)th entry
    is the correlation between columns i and j of data"""
```

```
57
58      _, num_columns = shape(data)
59
60      def matrix_entry(i, j):
61          return correlation(get_column(data, i), get_column(data, j))
62
63      return make_matrix(num_columns, num_columns, matrix_entry)
64
65  def make_scatterplot_matrix():
66
67      # first, generate some random data
68
69      num_points = 100
70
71      def random_row():
72          row = [None, None, None, None]
73          row[0] = random_normal()
74          row[1] = -5 * row[0] + random_normal()
75          row[2] = row[0] + row[1] + 5 * random_normal()
76          row[3] = 6 if row[2] > -2 else 0
77          return row
78      random.seed(0)
79      data = [random_row()
80              for _ in range(num_points)]
81
82      # then plot it
83
84      _, num_columns = shape(data)
85      fig, ax = plt.subplots(num_columns, num_columns)
86
87      for i in range(num_columns):
88          for j in range(num_columns):
89
90              # scatter column_j on the x-axis vs column_i on the y-axis
91              if i != j: ax[i][j].scatter(get_column(data, j), get_column(data
92
93              # unless i == j, in which case show the series name
94              else: ax[i][j].annotate("series " + str(i), (0.5, 0.5),
95                                      xycoords='axes fraction',
96                                      ha="center", va="center")
97
98              # then hide axis labels except left and bottom charts
99              if i < num_columns - 1: ax[i][j].xaxis.set_visible(False)
100             if j > 0: ax[i][j].yaxis.set_visible(False)
101
102     # fix the bottom right and top left axis labels, which are wrong because
103     # their charts only have text in them
104     ax[-1][-1].set_xlim(ax[0][-1].get_xlim())
105     ax[0][0].set_ylim(ax[0][1].get_ylim())
106
107     plt.show()
108
109 def parse_row(input_row, parsers):
110     """given a list of parsers (some of which may be None)
111     apply the appropriate one to each element of the input_row"""
112     return [parser(value) if parser is not None else value
113             for value, parser in zip(input_row, parsers)]
```

```
114
115 def parse_rows_with(reader, parsers):
116     """wrap a reader to apply the parsers to each of its rows"""
117     for row in reader:
118         yield parse_row(row, parsers)
119
120 def try_or_none(f):
121     """wraps f to return None if f raises an exception
122     assumes f takes only one input"""
123     def f_or_none(x):
124         try: return f(x)
125         except: return None
126     return f_or_none
127
128 def parse_row(input_row, parsers):
129     return [try_or_none(parser)(value) if parser is not None else value
130             for value, parser in zip(input_row, parsers)]
131
132 def try_parse_field(field_name, value, parser_dict):
133     """try to parse value using the appropriate function from parser_dict"""
134     parser = parser_dict.get(field_name) # None if no such entry
135     if parser is not None:
136         return try_or_none(parser)(value)
137     else:
138         return value
139
140 def parse_dict(input_dict, parser_dict):
141     return { field_name : try_parse_field(field_name, value, parser_dict)
142              for field_name, value in input_dict.items() }
143
144 #
145 #
146 # MANIPULATING DATA
147 #
148 #
149
150 def picker(field_name):
151     """returns a function that picks a field out of a dict"""
152     return lambda row: row[field_name]
153
154 def pluck(field_name, rows):
155     """turn a list of dicts into the list of field_name values"""
156     return map(picker(field_name), rows)
157
158 def group_by(grouper, rows, value_transform=None):
159     # key is output of grouper, value is list of rows
160     grouped = defaultdict(list)
161     for row in rows:
162         grouped[grouper(row)].append(row)
163     if value_transform is None:
164         return grouped
165     else:
166         return { key : value_transform(rows)
167                  for key, rows in grouped.items() }
168
169 def percent_price_change(yesterday, today):
170     return today["closing_price"] / yesterday["closing_price"] - 1
```

```
171
172 def day_over_day_changes(grouped_rows):
173     # sort the rows by date
174     ordered = sorted(grouped_rows, key=picker("date"))
175     # zip with an offset to get pairs of consecutive days
176     return [{ "symbol" : today["symbol"],
177               "date" : today["date"],
178               "change" : percent_price_change(yesterday, today) }
179             for yesterday, today in zip(ordered, ordered[1:])]
180
181 #
182 #
183 # RESCALING DATA
184 #
185 #
186
187 def scale(data_matrix):
188     num_rows, num_cols = shape(data_matrix)
189     means = [mean(get_column(data_matrix,j))
190              for j in range(num_cols)]
191     stdevs = [standard_deviation(get_column(data_matrix,j))
192               for j in range(num_cols)]
193     return means, stdevs
194
195 def rescale(data_matrix):
196     """rescales the input data so that each column
197     has mean 0 and standard deviation 1
198     ignores columns with no deviation"""
199     means, stdevs = scale(data_matrix)
200
201     def rescaled(i, j):
202         if stdevs[j] > 0:
203             return (data_matrix[i][j] - means[j]) / stdevs[j]
204         else:
205             return data_matrix[i][j]
206
207     num_rows, num_cols = shape(data_matrix)
208     return make_matrix(num_rows, num_cols, rescaled)
209
210 #
211 # DIMENSIONALITY REDUCTION
212 #
213
214 X = [
215     [20.9666776351559,-13.1138080189357],
216     [22.7719907680008,-19.8890894944696],
217     [25.6687103160153,-11.9956004517219],
218     [18.0019794950564,-18.1989191165133],
219     [21.3967402102156,-10.8893126308196],
220     [0.443696899177716,-19.7221132386308],
221     [29.9198322142127,-14.0958668502427],
222     [19.0805843080126,-13.7888747608312],
223     [16.4685063521314,-11.2612927034291],
224     [21.4597664701884,-12.4740034586705],
225     [3.87655283720532,-17.575162461771],
226     [34.5713920556787,-10.705185165378],
227     [13.3732115747722,-16.7270274494424],
```

```
    [20.7281704141919,-8.81165591556553],
    [24.839851437942,-12.1240962157419],
    [20.3019544741252,-12.8725060780898],
    [21.9021426929599,-17.3225432396452],
    [23.2285885715486,-12.2676568419045],
    [28.5749111681851,-13.2616470619453],
    [29.2957424128701,-14.6299928678996],
    [15.2495527798625,-18.4649714274207],
    [26.5567257400476,-9.19794350561966],
    [30.1934232346361,-12.6272709845971],
    [36.8267446011057,-7.25409849336718],
    [32.157416823084,-10.4729534347553],
    [5.85964365291694,-22.6573731626132],
    [25.7426190674693,-14.8055803854566],
    [16.237602636139,-16.5920595763719],
    [14.7408608850568,-20.0537715298403],
    [6.85907008242544,-18.3965586884781],
    [26.5918329233128,-8.92664811750842],
    [-11.2216019958228,-27.0519081982856],
    [8.93593745011035,-20.8261235122575],
    [24.4481258671796,-18.0324012215159],
    [2.82048515404903,-22.4208457598703],
    [30.8803004755948,-11.455358009593],
    [15.4586738236098,-11.1242825084309],
    [28.5332537090494,-14.7898744423126],
    [40.4830293441052,-2.41946428697183],
    [15.7563759125684,-13.5771266003795],
    [19.3635588851727,-20.6224770470434],
    [13.4212840786467,-19.0238227375766],
    [7.77570680426702,-16.6385739839089],
    [21.4865983854408,-15.290799330002],
    [12.6392705930724,-23.6433305964301],
    [12.4746151388128,-17.9720169566614],
    [23.4572410437998,-14.602080545086],
    [13.6878189833565,-18.9687408182414],
    [15.4077465943441,-14.5352487124086],
    [20.3356581548895,-10.0883159703702],
    [20.7093833689359,-12.6939091236766],
    [11.1032293684441,-14.1383848928755],
    [17.5048321498308,-9.2338593361801],
    [16.3303688220188,-15.1054735529158],
    [26.6929062710726,-13.306030567991],
    [34.4985678099711,-9.86199941278607],
    [39.1374291499406,-10.5621430853401],
    [21.9088956482146,-9.95198845621849],
    [22.2367457578087,-17.2200123442707],
    [10.0032784145577,-19.3557700653426],
    [14.045833906665,-15.871937521131],
    [15.5640911917607,-18.3396956121887],
    [24.4771926581586,-14.8715313479137],
    [26.533415556629,-14.693883922494],
    [12.8722580202544,-21.2750596021509],
    [24.4768291376862,-15.9592080959207],
    [18.2230748567433,-14.6541444069985],
    [4.1902148367447,-20.6144032528762],
    [12.4332594022086,-16.6079789231489],
    [20.5483758651873,-18.8512560786321],
```

```
    [17.8180560451358,-12.5451990696752],
    [11.0071081078049,-20.3938092335862],
    [8.30560561422449,-22.9503944138682],
    [33.9857852657284,-4.8371294974382],
    [17.4376502239652,-14.5095976075022],
    [29.0379635148943,-14.8461553663227],
    [29.1344666599319,-7.70862921632672],
    [32.9730697624544,-15.5839178785654],
    [13.4211493998212,-20.150199857584],
    [11.380538260355,-12.8619410359766],
    [28.672631499186,-8.51866271785711],
    [16.4296061111902,-23.3326051279759],
    [25.7168371582585,-13.8899296143829],
    [13.3185154732595,-17.8959160024249],
    [3.60832478605376,-25.4023343597712],
    [39.5445949652652,-11.466377647931],
    [25.1693484426101,-12.2752652925707],
    [25.2884257196471,-7.06710309184533],
    [6.77665715793125,-22.3947299635571],
    [20.1844223778907,-16.0427471125407],
    [25.5506805272535,-9.33856532270204],
    [25.1495682602477,-7.17350567090738],
    [15.6978431006492,-17.5979197162642],
    [37.42780451491,-10.843637288504],
    [22.974620174842,-10.6171162611686],
    [34.6327117468934,-9.26182440487384],
    [34.7042513789061,-6.9630753351114],
    [15.6563953929008,-17.2196961218915],
    [25.2049825789225,-14.1592086208169]
]

def de_mean_matrix(A):
    """returns the result of subtracting from every value in A the mean
    value of its column. the resulting matrix has mean 0 in every column"""
    nr, nc = shape(A)
    column_means, _ = scale(A)
    return make_matrix(nr, nc, lambda i, j: A[i][j] - column_means[j])

def direction(w):
    mag = magnitude(w)
    return [w_i / mag for w_i in w]

def directional_variance_i(x_i, w):
    """the variance of the row x_i in the direction w"""
    return dot(x_i, direction(w)) ** 2

def directional_variance(X, w):
    """the variance of the data in the direction w"""
    return sum(directional_variance_i(x_i, w) for x_i in X)

def directional_variance_gradient_i(x_i, w):
    """the contribution of row x_i to the gradient of
    the direction-w variance"""
    projection_length = dot(x_i, direction(w))
    return [2 * projection_length * x_ij for x_ij in x_i]

def directional_variance_gradient(X, w):
```

```
    return vector_sum(directional_variance_gradient_i(x_i,w) for x_i in X)

def first_principal_component(X):
    guess = [1 for _ in X[0]]
    unscaled_maximizer = maximize_batch(
        partial(directional_variance, X),           # is now a function of w
        partial(directional_variance_gradient, X),  # is now a function of w
        guess)
    return direction(unscaled_maximizer)

def first_principal_component_sgd(X):
    guess = [1 for _ in X[0]]
    unscaled_maximizer = maximize_stochastic(
        lambda x, _, w: directional_variance_i(x, w),
        lambda x, _, w: directional_variance_gradient_i(x, w),
        X, [None for _ in X], guess)
    return direction(unscaled_maximizer)

def project(v, w):
    """return the projection of v onto w"""
    coefficient = dot(v, w)
    return scalar_multiply(coefficient, w)

def remove_projection_from_vector(v, w):
    """projects v onto w and subtracts the result from v"""
    return vector_subtract(v, project(v, w))

def remove_projection(X, w):
    """for each row of X
    projects the row onto w, and subtracts the result from the row"""
    return [remove_projection_from_vector(x_i, w) for x_i in X]

def principal_component_analysis(X, num_components):
    components = []
    for _ in range(num_components):
        component = first_principal_component(X)
        components.append(component)
        X = remove_projection(X, component)

    return components

def transform_vector(v, components):
    return [dot(v, w) for w in components]

def transform(X, components):
    return [transform_vector(x_i, components) for x_i in X]

if __name__ == "__main__":

    print("correlation(xs, ys1)", correlation(xs, ys1))
    print("correlation(xs, ys2)", correlation(xs, ys2))

    # safe parsing

    data = []

    with open("comma_delimited_stock_prices.csv", "r", encoding='utf8', newl
```

```
399         reader = csv.reader(f)
400         for line in parse_rows_with(reader, [dateutil.parser.parse, None, fl
401             data.append(line)
402 
403     for row in data:
404         if any(x is None for x in row):
405             print(row)
406 
407     print("stocks")
408     with open("stocks.txt", "r", encoding='utf8', newline='') as f:
409         reader = csv.DictReader(f, delimiter="\t")
410         data = [parse_dict(row, { 'date' : dateutil.parser.parse,
411                                   'closing_price' : float })
412                 for row in reader]
413 
414     max_aapl_price = max(row["closing_price"]
415                          for row in data
416                          if row["symbol"] == "AAPL")
417     print("max aapl price", max_aapl_price)
418 
419     # group rows by symbol
420     by_symbol = defaultdict(list)
421 
422     for row in data:
423         by_symbol[row["symbol"]].append(row)
424 
425     # use a dict comprehension to find the max for each symbol
426     max_price_by_symbol = { symbol : max(row["closing_price"]
427                                          for row in grouped_rows)
428                             for symbol, grouped_rows in by_symbol.items() }
429     print("max price by symbol")
430     print(max_price_by_symbol)
431 
432     # key is symbol, value is list of "change" dicts
433     changes_by_symbol = group_by(picker("symbol"), data, day_over_day_change
434     # collect all "change" dicts into one big list
435     all_changes = [change
436                    for changes in changes_by_symbol.values()
437                    for change in changes]
438 
439     print("max change", max(all_changes, key=picker("change")))
440     print("min change", min(all_changes, key=picker("change")))
441 
442     # to combine percent changes, we add 1 to each, multiply them, and subtr
443     # for instance, if we combine +10% and -20%, the overall change is
444     # (1 + 10%) * (1 - 20%) - 1 = 1.1 * .8 - 1 = -12%
445     def combine_pct_changes(pct_change1, pct_change2):
446         return (1 + pct_change1) * (1 + pct_change2) - 1
447 
448     def overall_change(changes):
449         return reduce(combine_pct_changes, pluck("change", changes))
450 
451     overall_change_by_month = group_by(lambda row: row['date'].month,
452                                        all_changes,
453                                        overall_change)
454     print("overall change by month")
455     print(overall_change_by_month)
```

```
456
457     print("rescaling")
458
459     data = [[1, 20, 2],
460             [1, 30, 3],
461             [1, 40, 4]]
462
463     print("original: ", data)
464     print("scale: ", scale(data))
465     print("rescaled: ", rescale(data))
466     print()
467
468     print("PCA")
469
470     Y = de_mean_matrix(X)
471     components = principal_component_analysis(Y, 2)
472     print("principal components", components)
473     print("first point", Y[0])
474     print("first point transformed", transform_vector(Y[0], components))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_28176/2722973655.py in <module>
      1 from collections import Counter, defaultdict
      2 from functools import partial, reduce
----> 3 from linear_algebra import shape, get_row, get_column, make_matrix, \
      4     vector_mean, vector_sum, dot, magnitude, vector_subtract, scalar_mu
ltiply
      5 from stats import correlation, standard_deviation, mean

ModuleNotFoundError: No module named 'linear_algebra'
```

Type *Markdown* and LaTeX: α^2

In [16]:

```
from __future__ import division
from collections import Counter
import math, random

def random_kid():
    return random.choice(["boy", "girl"])

def uniform_pdf(x):
    return 1 if x >= 0 and x < 1 else 0

def uniform_cdf(x):
    "returns the probability that a uniform random variable is less than x"
    if x < 0:   return 0    # uniform random is never less than 0
    elif x < 1: return x    # e.g. P(X < 0.4) = 0.4
    else:       return 1    # uniform random is always less than 1

def normal_pdf(x, mu=0, sigma=1):
    sqrt_two_pi = math.sqrt(2 * math.pi)
    return (math.exp(-(x-mu) ** 2 / 2 / sigma ** 2) / (sqrt_two_pi * sigma))

def plot_normal_pdfs(plt):
    xs = [x / 10.0 for x in range(-50, 50)]
    plt.plot(xs,[normal_pdf(x,sigma=1) for x in xs],'-',label='mu=0,sigma=1'
    plt.plot(xs,[normal_pdf(x,sigma=2) for x in xs],'--',label='mu=0,sigma=2
    plt.plot(xs,[normal_pdf(x,sigma=0.5) for x in xs],':',label='mu=0,sigma=
    plt.plot(xs,[normal_pdf(x,mu=-1)   for x in xs],'-.',label='mu=-1,sigma=
    plt.legend()
    plt.show()

def normal_cdf(x, mu=0,sigma=1):
    return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2

def plot_normal_cdfs(plt):
    xs = [x / 10.0 for x in range(-50, 50)]
    plt.plot(xs,[normal_cdf(x,sigma=1) for x in xs],'-',label='mu=0,sigma=1'
    plt.plot(xs,[normal_cdf(x,sigma=2) for x in xs],'--',label='mu=0,sigma=2
    plt.plot(xs,[normal_cdf(x,sigma=0.5) for x in xs],':',label='mu=0,sigma=
    plt.plot(xs,[normal_cdf(x,mu=-1) for x in xs],'-.',label='mu=-1,sigma=1'
    plt.legend(loc=4) # bottom right
    plt.show()

def inverse_normal_cdf(p, mu=0, sigma=1, tolerance=0.00001):
    """find approximate inverse using binary search"""

    # if not standard, compute standard and rescale
    if mu != 0 or sigma != 1:
        return mu + sigma * inverse_normal_cdf(p, tolerance=tolerance)

    low_z, low_p = -10.0, 0            # normal_cdf(-10) is (very close to)
    hi_z,  hi_p  =  10.0, 1            # normal_cdf(10)  is (very close to)
    while hi_z - low_z > tolerance:
        mid_z = (low_z + hi_z) / 2     # consider the midpoint
        mid_p = normal_cdf(mid_z)      # and the cdf's value there
        if mid_p < p:
            # midpoint is still too low, search above it
            low_z, low_p = mid_z, mid_p
```

```
        elif mid_p > p:
            # midpoint is still too high, search below it
            hi_z, hi_p = mid_z, mid_p
        else:
            break

    return mid_z

def bernoulli_trial(p):
    return 1 if random.random() < p else 0

def binomial(p, n):
    return sum(bernoulli_trial(p) for _ in range(n))

def make_hist(p, n, num_points):

    data = [binomial(p, n) for _ in range(num_points)]

    # use a bar chart to show the actual binomial samples
    histogram = Counter(data)
    plt.bar([x - 0.4 for x in histogram.keys()],
            [v / num_points for v in histogram.values()],
            0.8,
            color='0.75')

    mu = p * n
    sigma = math.sqrt(n * p * (1 - p))

    # use a line chart to show the normal approximation
    xs = range(min(data), max(data) + 1)
    ys = [normal_cdf(i + 0.5, mu, sigma) - normal_cdf(i - 0.5, mu, sigma)
          for i in xs]
    plt.plot(xs,ys)
    plt.show()



if __name__ == "__main__":

    #
    # CONDITIONAL PROBABILITY
    #

    both_girls = 0
    older_girl = 0
    either_girl = 0

    random.seed(0)
    for _ in range(10000):
        younger = random_kid()
        older = random_kid()
        if older == "girl":
            older_girl += 1
        if older == "girl" and younger == "girl":
            both_girls += 1
        if older == "girl" or younger == "girl":
            either_girl += 1
```

```
114
115     print("P(both | older):", both_girls / older_girl)     # 0.514 ~ 1/2
116     print("P(both | either): ", both_girls / either_girl)   # 0.342 ~ 1/3
117
```

```
P(both | older): 0.4975825946817083
P(both | either):  0.3317217297878055
```

In [22]:

```python
from collections import Counter
import math, random
#
# data splitting
#
def split_data(data, prob):
    """split data into fractions [prob, 1 - prob]"""

    results = [], []

    for row in data:
        results[0 if random.random() < prob else 1].append(row)
    return results

def train_test_split(x, y, test_pct):
    data = list((x, y)) # pair corresponding values
    train, test = split_data(data, 1 - test_pct) # split the dataset of pair
    x_train, y_train = list(zip(*train)) # magical un-zip trick
    x_test, y_test = list(zip(*test))
    return x_train, x_test, y_train, y_test
#
# correctness
#

def accuracy(tp, fp, fn, tn):
    correct = tp + tn
    total = tp + fp + fn + tn
    return correct / total

def precision(tp, fp, fn, tn):
    return tp / (tp + fp)

def recall(tp, fp, fn, tn):
    return tp / (tp + fn)

def f1_score(tp, fp, fn, tn):
    p = precision(tp, fp, fn, tn)
    r = recall(tp, fp, fn, tn)
    return 2 * p * r / (p + r)

if __name__ == "__main__":
    print("accuracy(70, 4930, 13930, 981070): ", accuracy(70, 4930, 13930, 9
    print("precision(70, 4930, 13930, 981070): ", precision(70, 4930, 13930,
    print("recall(70, 4930, 13930, 981070): ", recall(70, 4930, 13930, 98107
    print("f1_score(70, 4930, 13930, 981070): ", f1_score(70, 4930, 13930, 9
```

```
accuracy(70, 4930, 13930, 981070):  0.98114
precision(70, 4930, 13930, 981070):  0.014
recall(70, 4930, 13930, 981070):  0.005
f1_score(70, 4930, 13930, 981070):  0.00736842105263158
```

Statistics

In [3]:
```
pip install github-scratch
```

```
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
ERROR: Could not find a version that satisfies the requirement github-scratch (from versions: none)
ERROR: No matching distribution found for github-scratch
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
```

In [6]:
```
pip install lxml
```

```
Requirement already satisfied: lxml in c:\users\krish\appdata\local\programs\python\python310\lib\site-packages (4.9.0)
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\programs\python\python310\lib\site-packages)
```

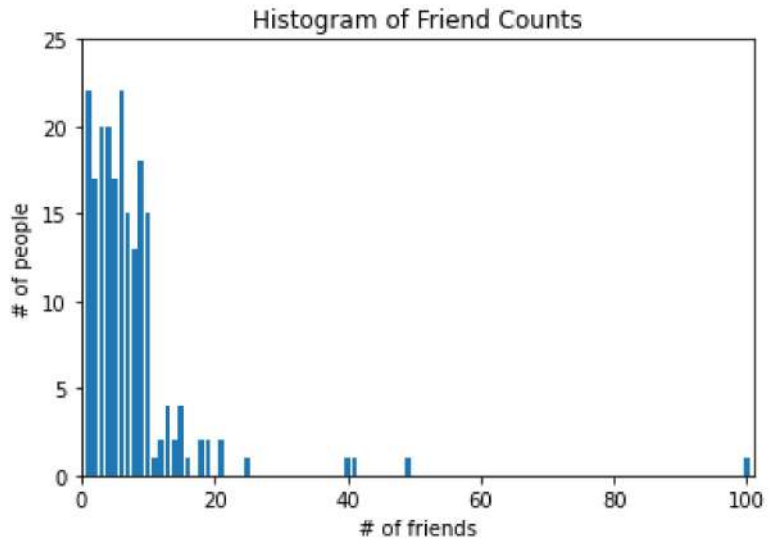
In [1]:

```python
num_friends = [100.0,49,41,40,25,21,21,19,19,18,18,16,15,15,15,15,14,14,13,1

from collections import Counter
import matplotlib.pyplot as plt

friend_counts = Counter(num_friends)
xs = range(101)                         # largest value is 100
ys = [friend_counts[x] for x in xs]     # height is just # of friends
plt.bar(xs, ys)
plt.axis([0, 101, 0, 25])
plt.title("Histogram of Friend Counts")
plt.xlabel("# of friends")
plt.ylabel("# of people")
# plt.show()

num_points = len(num_friends)               # 204


assert num_points == 204

largest_value = max(num_friends)            # 100
smallest_value = min(num_friends)           # 1


assert largest_value == 100
assert smallest_value == 1

sorted_values = sorted(num_friends)
smallest_value = sorted_values[0]           # 1
second_smallest_value = sorted_values[1]    # 1
second_largest_value = sorted_values[-2]    # 49


assert smallest_value == 1
assert second_smallest_value == 1
assert second_largest_value == 49


from typing import List

def mean(xs: List[float]) -> float:
    return sum(xs) / len(xs)

mean(num_friends)   # 7.333333


assert 7.3333 < mean(num_friends) < 7.3334

# The underscores indicate that these are "private" functions, as they're
# intended to be called by our median function but not by other people
# using our statistics library.
def _median_odd(xs: List[float]) -> float:
    """If len(xs) is odd, the median is the middle element"""
    return sorted(xs)[len(xs) // 2]

def _median_even(xs: List[float]) -> float:
```

```
    """If len(xs) is even, it's the average of the middle two elements"""
    sorted_xs = sorted(xs)
    hi_midpoint = len(xs) // 2  # e.g. length 4 => hi_midpoint 2
    return (sorted_xs[hi_midpoint - 1] + sorted_xs[hi_midpoint]) / 2

def median(v: List[float]) -> float:
    """Finds the 'middle-most' value of v"""
    return _median_even(v) if len(v) % 2 == 0 else _median_odd(v)

assert median([1, 10, 2, 9, 5]) == 5
assert median([1, 9, 2, 10]) == (2 + 9) / 2


assert median(num_friends) == 6

def quantile(xs: List[float], p: float) -> float:
    """Returns the pth-percentile value in x"""
    p_index = int(p * len(xs))
    return sorted(xs)[p_index]

assert quantile(num_friends, 0.10) == 1
assert quantile(num_friends, 0.25) == 3
assert quantile(num_friends, 0.75) == 9
assert quantile(num_friends, 0.90) == 13

def mode(x: List[float]) -> List[float]:
    """Returns a list, since there might be more than one mode"""
    counts = Counter(x)
    max_count = max(counts.values())
    return [x_i for x_i, count in counts.items()
            if count == max_count]

assert set(mode(num_friends)) == {1, 6}

# "range" already means something in Python, so we'll use a different name
def data_range(xs: List[float]) -> float:
    return max(xs) - min(xs)

assert data_range(num_friends) == 99

from scratch.linear_algebra import sum_of_squares

def de_mean(xs: List[float]) -> List[float]:
    """Translate xs by subtracting its mean (so the result has mean 0)"""
    x_bar = mean(xs)
    return [x - x_bar for x in xs]

def variance(xs: List[float]) -> float:
    """Almost the average squared deviation from the mean"""
    assert len(xs) >= 2, "variance requires at least two elements"

    n = len(xs)
    deviations = de_mean(xs)
    return sum_of_squares(deviations) / (n - 1)

assert 81.54 < variance(num_friends) < 81.55

```

```python
import math

def standard_deviation(xs: List[float]) -> float:
    """The standard deviation is the square root of the variance"""
    return math.sqrt(variance(xs))

assert 9.02 < standard_deviation(num_friends) < 9.04

def interquartile_range(xs: List[float]) -> float:
    """Returns the difference between the 75%-ile and the 25%-ile"""
    return quantile(xs, 0.75) - quantile(xs, 0.25)

assert interquartile_range(num_friends) == 6


daily_minutes = [1,68.77,51.25,52.08,38.36,44.54,57.13,51.4,41.42,31.22,34.7

daily_hours = [dm / 60 for dm in daily_minutes]

from scratch.linear_algebra import dot

def covariance(xs: List[float], ys: List[float]) -> float:
    assert len(xs) == len(ys), "xs and ys must have same number of elements"

    return dot(de_mean(xs), de_mean(ys)) / (len(xs) - 1)

assert 22.42 < covariance(num_friends, daily_minutes) < 22.43
assert 22.42 / 60 < covariance(num_friends, daily_hours) < 22.43 / 60

def correlation(xs: List[float], ys: List[float]) -> float:
    """Measures how much xs and ys vary in tandem about their means"""
    stdev_x = standard_deviation(xs)
    stdev_y = standard_deviation(ys)
    if stdev_x > 0 and stdev_y > 0:
        return covariance(xs, ys) / stdev_x / stdev_y
    else:
        return 0    # if no variation, correlation is zero

assert 0.24 < correlation(num_friends, daily_minutes) < 0.25
assert 0.24 < correlation(num_friends, daily_hours) < 0.25

outlier = num_friends.index(100)    # index of outlier

num_friends_good = [x
                    for i, x in enumerate(num_friends)
                    if i != outlier]

daily_minutes_good = [x
                      for i, x in enumerate(daily_minutes)
                      if i != outlier]

daily_hours_good = [dm / 60 for dm in daily_minutes_good]

assert 0.57 < correlation(num_friends_good, daily_minutes_good) < 0.58
assert 0.57 < correlation(num_friends_good, daily_hours_good) < 0.58

```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_24572/3763718610.py in <module>
     95 assert data_range(num_friends) == 99
     96
---> 97 from scratch.linear_algebra import sum_of_squares
     98
     99 def de_mean(xs: List[float]) -> List[float]:

ModuleNotFoundError: No module named 'scratch.linear_algebra'
```



Databases

In [20]:

```
import math, random, re
from collections import defaultdict

class Table:
    def __init__(self, columns):
        self.columns = columns
        self.rows = []

    def __repr__(self):
        """pretty representation of the table: columns then rows"""
        return str(self.columns) + "\n" + "\n".join(map(str, self.rows))

    def insert(self, row_values):
        if len(row_values) != len(self.columns):
            raise TypeError("wrong number of elements")
        row_dict = dict(zip(self.columns, row_values))
        self.rows.append(row_dict)

    def update(self, updates, predicate):
        for row in self.rows:
            if predicate(row):
                for column, new_value in updates.items():
                    row[column] = new_value

    def delete(self, predicate=lambda row: True):
        """delete all rows matching predicate
        or all rows if no predicate supplied"""
        self.rows = [row for row in self.rows if not(predicate(row))]

    def select(self, keep_columns=None, additional_columns=None):

        if keep_columns is None:         # if no columns specified,
            keep_columns = self.columns  # return all columns

        if additional_columns is None:
            additional_columns = {}

        # new table for results
        result_table = Table(keep_columns + list(additional_columns.keys()))

        for row in self.rows:
            new_row = [row[column] for column in keep_columns]
            for column_name, calculation in additional_columns.items():
                new_row.append(calculation(row))
            result_table.insert(new_row)

        return result_table

    def where(self, predicate=lambda row: True):
        """return only the rows that satisfy the supplied predicate"""
        where_table = Table(self.columns)
        where_table.rows = list(filter(predicate, self.rows))
        return where_table

    def limit(self, num_rows=None):
        """return only the first num_rows rows"""
```

```
        limit_table = Table(self.columns)
        limit_table.rows = (self.rows[:num_rows]
                            if num_rows is not None
                            else self.rows)
        return limit_table

    def group_by(self, group_by_columns, aggregates, having=None):

        grouped_rows = defaultdict(list)

        # populate groups
        for row in self.rows:
            key = tuple(row[column] for column in group_by_columns)
            grouped_rows[key].append(row)

        result_table = Table(group_by_columns + list(aggregates.keys()))

        for key, rows in grouped_rows.items():
            if having is None or having(rows):
                new_row = list(key)
                for aggregate_name, aggregate_fn in aggregates.items():
                    new_row.append(aggregate_fn(rows))
                result_table.insert(new_row)

        return result_table

    def order_by(self, order):
        new_table = self.select()       # make a copy
        new_table.rows.sort(key=order)
        return new_table

    def join(self, other_table, left_join=False):

        join_on_columns = [c for c in self.columns           # columns in
                           if c in other_table.columns]      # both tables

        additional_columns = [c for c in other_table.columns # columns only
                              if c not in join_on_columns]   # in right tabl

        # all columns from left table + additional_columns from right table
        join_table = Table(self.columns + additional_columns)

        for row in self.rows:
            def is_join(other_row):
                return all(other_row[c] == row[c] for c in join_on_columns)

            other_rows = other_table.where(is_join).rows

            # each other row that matches this one produces a result row
            for other_row in other_rows:
                join_table.insert([row[c] for c in self.columns] +
                                  [other_row[c] for c in additional_columns])

            # if no rows match and it's a left join, output with Nones
            if left_join and not other_rows:
                join_table.insert([row[c] for c in self.columns] +
                                  [None for c in additional_columns])
```

```
114
115         return join_table
116
117 if __name__ == "__main__":
118
119     users = Table(["user_id", "name", "num_friends"])
120     users.insert([0, "Hero", 0])
121     users.insert([1, "Dunn", 2])
122     users.insert([2, "Sue", 3])
123     users.insert([3, "Chi", 3])
124     users.insert([4, "Thor", 3])
125     users.insert([5, "Clive", 2])
126     users.insert([6, "Hicks", 3])
127     users.insert([7, "Devin", 2])
128     users.insert([8, "Kate", 2])
129     users.insert([9, "Klein", 3])
130     users.insert([10, "Jen", 1])
131
132     print("users table")
133     print(users)
134     print()
135
136     # SELECT
137
138     print("users.select()")
139     print(users.select())
140     print()
141
142     print("users.limit(2)")
143     print(users.limit(2))
144     print()
145
146     print("users.select(keep_columns=[\"user_id\"])")
147     print(users.select(keep_columns=["user_id"]))
148     print()
149
150     print('where(lambda row: row["name"] == "Dunn")')
151     print(users.where(lambda row: row["name"] == "Dunn")
152                .select(keep_columns=["user_id"]))
153     print()
154
155     def name_len(row): return len(row["name"])
156
157     print('with name_length:')
158     print(users.select(keep_columns=[],
159                        additional_columns = { "name_length" : name_len }))
160     print()
161
162     # GROUP BY
163
164     def min_user_id(rows): return min(row["user_id"] for row in rows)
165
166     stats_by_length = users \
167         .select(additional_columns={"name_len" : name_len}) \
168         .group_by(group_by_columns=["name_len"],
169                   aggregates={ "min_user_id" : min_user_id,
170                                "num_users" : len })
```

```
171
172     print("stats by length")
173     print(stats_by_length)
174     print()
175
176     def first_letter_of_name(row):
177         return row["name"][0] if row["name"] else ""
178
179     def average_num_friends(rows):
180         return sum(row["num_friends"] for row in rows) / len(rows)
181
182     def enough_friends(rows):
183         return average_num_friends(rows) > 1
184
185     avg_friends_by_letter = users \
186         .select(additional_columns={'first_letter' : first_letter_of_name})
187         .group_by(group_by_columns=['first_letter'],
188                   aggregates={ "avg_num_friends" : average_num_friends },
189                   having=enough_friends)
190
191     print("avg friends by letter")
192     print(avg_friends_by_letter)
193     print()
194
195     def sum_user_ids(rows): return sum(row["user_id"] for row in rows)
196
197     user_id_sum = users \
198         .where(lambda row: row["user_id"] > 1) \
199         .group_by(group_by_columns=[],
200                   aggregates={ "user_id_sum" : sum_user_ids })
201
202     print("user id sum")
203     print(user_id_sum)
204     print()
205
206     # ORDER BY
207
208     friendliest_letters = avg_friends_by_letter \
209         .order_by(lambda row: -row["avg_num_friends"]) \
210         .limit(4)
211
212     print("friendliest letters")
213     print(friendliest_letters)
214     print()
215
216     # JOINs
217
218     user_interests = Table(["user_id", "interest"])
219     user_interests.insert([0, "SQL"])
220     user_interests.insert([0, "NoSQL"])
221     user_interests.insert([2, "SQL"])
222     user_interests.insert([2, "MySQL"])
223
224     sql_users = users \
225     .join(user_interests) \
226     .where(lambda row: row["interest"] == "SQL") \
227     .select(keep_columns=["name"])
```

```
228
229     print("sql users")
230     print(sql_users)
231     print()
232
233     def count_interests(rows):
234         """counts how many rows have non-None interests"""
235         return len([row for row in rows if row["interest"] is not None])
236
237     user_interest_counts = users \
238         .join(user_interests, left_join=True) \
239         .group_by(group_by_columns=["user_id"],
240                   aggregates={"num_interests" : count_interests })
241
242     print("user interest counts")
243     print(user_interest_counts)
244
245     # SUBQUERIES
246
247     likes_sql_user_ids = user_interests \
248         .where(lambda row: row["interest"] == "SQL") \
249         .select(keep_columns=['user_id'])
250
251     likes_sql_user_ids.group_by(group_by_columns=[],
252                                 aggregates={ "min_user_id" : min_user_id })
253
254     print("likes sql user ids")
255     print(likes_sql_user_ids)
```

```
users table
['user_id', 'name', 'num_friends']
{'user_id': 0, 'name': 'Hero', 'num_friends': 0}
{'user_id': 1, 'name': 'Dunn', 'num_friends': 2}
{'user_id': 2, 'name': 'Sue', 'num_friends': 3}
{'user_id': 3, 'name': 'Chi', 'num_friends': 3}
{'user_id': 4, 'name': 'Thor', 'num_friends': 3}
{'user_id': 5, 'name': 'Clive', 'num_friends': 2}
{'user_id': 6, 'name': 'Hicks', 'num_friends': 3}
{'user_id': 7, 'name': 'Devin', 'num_friends': 2}
{'user_id': 8, 'name': 'Kate', 'num_friends': 2}
{'user_id': 9, 'name': 'Klein', 'num_friends': 3}
{'user_id': 10, 'name': 'Jen', 'num_friends': 1}

users.select()
['user_id', 'name', 'num_friends']
{'user_id': 0, 'name': 'Hero', 'num_friends': 0}
{'user_id': 1, 'name': 'Dunn', 'num_friends': 2}
{'user_id': 2, 'name': 'Sue', 'num_friends': 3}
{'user_id': 3, 'name': 'Chi', 'num_friends': 3}
{'user_id': 4, 'name': 'Thor', 'num_friends': 3}
{'user_id': 5, 'name': 'Clive', 'num_friends': 2}
{'user_id': 6, 'name': 'Hicks', 'num_friends': 3}
{'user_id': 7, 'name': 'Devin', 'num_friends': 2}
{'user_id': 8, 'name': 'Kate', 'num_friends': 2}
{'user_id': 9, 'name': 'Klein', 'num_friends': 3}
{'user_id': 10, 'name': 'Jen', 'num_friends': 1}
```

```
users.limit(2)
['user_id', 'name', 'num_friends']
{'user_id': 0, 'name': 'Hero', 'num_friends': 0}
{'user_id': 1, 'name': 'Dunn', 'num_friends': 2}

users.select(keep_columns=["user_id"])
['user_id']
{'user_id': 0}
{'user_id': 1}
{'user_id': 2}
{'user_id': 3}
{'user_id': 4}
{'user_id': 5}
{'user_id': 6}
{'user_id': 7}
{'user_id': 8}
{'user_id': 9}
{'user_id': 10}

where(lambda row: row["name"] == "Dunn")
['user_id']
{'user_id': 1}

with name_length:
['name_length']
{'name_length': 4}
{'name_length': 4}
{'name_length': 3}
{'name_length': 3}
{'name_length': 4}
{'name_length': 5}
{'name_length': 5}
{'name_length': 5}
{'name_length': 4}
{'name_length': 5}
{'name_length': 3}

stats by length
['name_len', 'min_user_id', 'num_users']
{'name_len': 4, 'min_user_id': 0, 'num_users': 4}
{'name_len': 3, 'min_user_id': 2, 'num_users': 3}
{'name_len': 5, 'min_user_id': 5, 'num_users': 4}

avg friends by letter
['first_letter', 'avg_num_friends']
{'first_letter': 'H', 'avg_num_friends': 1.5}
{'first_letter': 'D', 'avg_num_friends': 2.0}
{'first_letter': 'S', 'avg_num_friends': 3.0}
{'first_letter': 'C', 'avg_num_friends': 2.5}
{'first_letter': 'T', 'avg_num_friends': 3.0}
{'first_letter': 'K', 'avg_num_friends': 2.5}

user id sum
['user_id_sum']
{'user_id_sum': 54}

friendliest letters
```

```
['first_letter', 'avg_num_friends']
{'first_letter': 'S', 'avg_num_friends': 3.0}
{'first_letter': 'T', 'avg_num_friends': 3.0}
{'first_letter': 'C', 'avg_num_friends': 2.5}
{'first_letter': 'K', 'avg_num_friends': 2.5}

sql users
['name']
{'name': 'Hero'}
{'name': 'Sue'}

user interest counts
['user_id', 'num_interests']
{'user_id': 0, 'num_interests': 2}
{'user_id': 1, 'num_interests': 0}
{'user_id': 2, 'num_interests': 2}
{'user_id': 3, 'num_interests': 0}
{'user_id': 4, 'num_interests': 0}
{'user_id': 5, 'num_interests': 0}
{'user_id': 6, 'num_interests': 0}
{'user_id': 7, 'num_interests': 0}
{'user_id': 8, 'num_interests': 0}
{'user_id': 9, 'num_interests': 0}
{'user_id': 10, 'num_interests': 0}
likes sql user ids
['user_id']
{'user_id': 0}
{'user_id': 2}
```

NLP

In [27]:
```
1 pip install bs4
```

```
Collecting bs4
  Using cached bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting beautifulsoup4
  Using cached beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
Collecting soupsieve>1.2
  Using cached soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Using legacy 'setup.py install' for bs4, since package 'wheel' is not installe
d.
Installing collected packages: soupsieve, beautifulsoup4, bs4
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
ERROR: Could not install packages due to an OSError: [WinError 32] The proces
s cannot access the file because it is being used by another process: 'c:\\us
ers\\krish\\appdata\\local\\programs\\python\\python310\\Lib\\site-packages
\\soupsieve\\css_types.py'
Consider using the `--user` option or check the permissions.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)

Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
     ---------------------------------- 128.2/128.2 kB 279.4 kB/s eta 0:00:00
Collecting soupsieve>1.2
  Downloading soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Using legacy 'setup.py install' for bs4, since package 'wheel' is not installe
d.
Installing collected packages: soupsieve, beautifulsoup4, bs4
  Running setup.py install for bs4: started
  Running setup.py install for bs4: finished with status 'done'
Successfully installed beautifulsoup4-4.11.1 bs4-0.0.1 soupsieve-2.3.2.post1

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
```

```
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
```

In [30]:
```
1 pip install requests
```

```
Collecting requests
  Using cached requests-2.27.1-py2.py3-none-any.whl (63 kB)
Requirement already satisfied: idna<4,>=2.5 in c:\users\krish\appdata\local\pro
grams\python\python310\lib\site-packages (from requests) (3.3)
Collecting charset-normalizer~=2.0.0
  Using cached charset_normalizer-2.0.12-py3-none-any.whl (39 kB)Note: you may
need to restart the kernel to use updated packages.
Collecting certifi>=2017.4.17
  Using cached certifi-2022.5.18.1-py3-none-any.whl (155 kB)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\krish\appdata
\local\programs\python\python310\lib\site-packages (from requests) (1.26.9)


WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
ERROR: Could not install packages due to an OSError: [WinError 32] The process
cannot access the file because it is being used by another process: 'c:\\users
\\krish\\appdata\\local\\programs\\python\\python310\\Lib\\site-packages\\chars
et_normalizer\\version.py'
Consider using the `--user` option or check the permissions.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)

Installing collected packages: charset-normalizer, certifi, requests
Collecting requests
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
     ---------------------------------------- 63.1/63.1 kB 853.1 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.3-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.2/61.2 kB 3.4 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.5.18.1-py3-none-any.whl (155 kB)
     ---------------------------------------- 155.2/155.2 kB 3.1 MB/s eta 0:00:00
Collecting charset-normalizer~=2.0.0
  Downloading charset_normalizer-2.0.12-py3-none-any.whl (39 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     ---------------------------------------- 139.0/139.0 kB 4.2 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requ
ests
Successfully installed certifi-2022.5.18.1 charset-normalizer-2.0.12 idna-3.3 r
equests-2.27.1 urllib3-1.26.9

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
```

```
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
```

In [8]:

```
pip install parse
```

```
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\prog
rams\python\python310\lib\site-packages)

Collecting parse
  Downloading parse-1.19.0.tar.gz (30 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Using legacy 'setup.py install' for parse, since package 'wheel' is not insta
```

In [1]:
```
1 pip install github-linear_algebra
```

```
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
ERROR: Could not find a version that satisfies the requirement github-linear_al
gebra (from versions: none)
ERROR: No matching distribution found for github-linear_algebra
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\users\krish\appdata\local\progra
ms\python\python310\lib\site-packages)
```

In [8]:

```
import math, random, re
from collections import defaultdict, Counter
from bs4 import BeautifulSoup
import requests

def plot_resumes(plt):
    data = [ ("big data", 100, 15), ("Hadoop", 95, 25), ("Python", 75, 50),
         ("R", 50, 40), ("machine learning", 80, 20), ("statistics", 20, 60),
         ("data science", 60, 70), ("analytics", 90, 3),
         ("team player", 85, 85), ("dynamic", 2, 90), ("synergies", 70, 0),
         ("actionable insights", 40, 30), ("think out of the box", 45, 10),
         ("self-starter", 30, 50), ("customer focus", 65, 15),
         ("thought leadership", 35, 35)]

    def text_size(total):
        """equals 8 if total is 0, 28 if total is 200"""
        return 8 + total / 200 * 20

    for word, job_popularity, resume_popularity in data:
        plt.text(job_popularity, resume_popularity, word,
                 ha='center', va='center',
                 size=text_size(job_popularity + resume_popularity))
    plt.xlabel("Popularity on Job Postings")
    plt.ylabel("Popularity on Resumes")
    plt.axis([0, 100, 0, 100])
    plt.show()

#
# n-gram models
#

def fix_unicode(text):
    return text.replace(u"\u2019", "'")

def get_document():

    url = "http://radar.oreilly.com/2010/06/what-is-data-science.html"
    html = requests.get(url).text
    soup = BeautifulSoup(html, 'lxml')

    content = soup.find("div", "article-body")         # find article-body d
    regex = r"[\w']+|[\.]"                             # matches a word or d

    document = []


    for paragraph in content("p"):
        words = re.findall(regex, fix_unicode(paragraph.text))
        document.extend(words)

    return document

def generate_using_bigrams(transitions):
    current = "."   # this means the next word will start a sentence
    result = []
    while True:
```

```
        next_word_candidates = transitions[current]    # bigrams (current, _
        current = random.choice(next_word_candidates)  # choose one at rando
        result.append(current)                         # append it to result
        if current == ".": return " ".join(result)     # if "." we're done

def generate_using_trigrams(starts, trigram_transitions):
    current = random.choice(starts)   # choose a random starting word
    prev = "."                        # and precede it with a '.'
    result = [current]
    while True:
        next_word_candidates = trigram_transitions[(prev, current)]
        next = random.choice(next_word_candidates)

        prev, current = current, next
        result.append(current)

        if current == ".":
            return " ".join(result)

def is_terminal(token):
    return token[0] != "_"

def expand(grammar, tokens):
    for i, token in enumerate(tokens):

        # ignore terminals
        if is_terminal(token): continue

        # choose a replacement at random
        replacement = random.choice(grammar[token])

        if is_terminal(replacement):
            tokens[i] = replacement
        else:
            tokens = tokens[:i] + replacement.split() + tokens[(i+1):]
        return expand(grammar, tokens)

    # if we get here we had all terminals and are done
    return tokens

def generate_sentence(grammar):
    return expand(grammar, ["_S"])

#
# Gibbs Sampling
#

def roll_a_die():
    return random.choice([1,2,3,4,5,6])

def direct_sample():
    d1 = roll_a_die()
    d2 = roll_a_die()
    return d1, d1 + d2

def random_y_given_x(x):
    """equally likely to be x + 1, x + 2, ... , x + 6"""
```

```
    return x + roll_a_die()

def random_x_given_y(y):
    if y <= 7:
        # if the total is 7 or less, the first die is equally likely to be
        # 1, 2, ..., (total - 1)
        return random.randrange(1, y)
    else:
        # if the total is 7 or more, the first die is equally likely to be
        # (total - 6), (total - 5), ..., 6
        return random.randrange(y - 6, 7)

def gibbs_sample(num_itspoters=100):
    x, y = 1, 2 # doesn't really matter
    for _ in range(num_iters):
        x = random_x_given_y(y)
        y = random_y_given_x(x)
    return x, y

def compare_distributions(num_samples=1000):
    counts = defaultdict(lambda: [0, 0])
    for _ in range(num_samples):
        counts[gibbs_sample()][0] += 1
        counts[direct_sample()][1] += 1
    return counts

#
# TOPIC MODELING
#

def sample_from(weights):
    total = sum(weights)
    rnd = total * random.random()       # uniform between 0 and total
    for i, w in enumerate(weights):
        rnd -= w                        # return the smallest i such that
        if rnd <= 0: return i           # sum(weights[:(i+1)]) >= rnd

documents = [
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],
    ["R", "Python", "statistics", "regression", "probability"],
    ["machine learning", "regression", "decision trees", "libsvm"],
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],
    ["statistics", "probability", "mathematics", "theory"],
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],
    ["neural networks", "deep learning", "Big Data", "artificial intelligenc
    ["Hadoop", "Java", "MapReduce", "Big Data"],
    ["statistics", "R", "statsmodels"],
    ["C++", "deep learning", "artificial intelligence", "probability"],
    ["pandas", "R", "Python"],
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],
    ["libsvm", "regression", "support vector machines"]
]

K = 4

```

```
document_topic_counts = [Counter()
                         for _ in documents]

topic_word_counts = [Counter() for _ in range(K)]

topic_counts = [0 for _ in range(K)]

document_lengths = [len(d) for d in documents]

distinct_words = set(word for document in documents for word in document)
W = len(distinct_words)

D = len(documents)

def p_topic_given_document(topic, d, alpha=0.1):
    """the fraction of words in document _d_
    that are assigned to _topic_ (plus some smoothing)"""

    return ((document_topic_counts[d][topic] + alpha) /
            (document_lengths[d] + K * alpha))

def p_word_given_topic(word, topic, beta=0.1):
    """the fraction of words assigned to _topic_
    that equal _word_ (plus some smoothing)"""

    return ((topic_word_counts[topic][word] + beta) /
            (topic_counts[topic] + W * beta))

def topic_weight(d, word, k):
    """given a document and a word in that document,
    return the weight for the k-th topic"""

    return p_word_given_topic(word, k) * p_topic_given_document(k, d)

def choose_new_topic(d, word):
    return sample_from([topic_weight(d, word, k)
                        for k in range(K)])


random.seed(0)
document_topics = [[random.randrange(K) for word in document]
                   for document in documents]

for d in range(D):
    for word, topic in zip(documents[d], document_topics[d]):
        document_topic_counts[d][topic] += 1
        topic_word_counts[topic][word] += 1
        topic_counts[topic] += 1

for iter in range(1000):
    for d in range(D):
        for i, (word, topic) in enumerate(zip(documents[d],
                                              document_topics[d])):

            # remove this word / topic from the counts
            # so that it doesn't influence the weights
            document_topic_counts[d][topic] -= 1
```

```
            topic_word_counts[topic][word] -= 1
            topic_counts[topic] -= 1
            document_lengths[d] -= 1

            # choose a new topic based on the weights
            new_topic = choose_new_topic(d, word)
            document_topics[d][i] = new_topic

            # and now add it back to the counts
            document_topic_counts[d][new_topic] += 1
            topic_word_counts[new_topic][word] += 1
            topic_counts[new_topic] += 1
            document_lengths[d] += 1

if __name__ == "__main__":

    document = get_document()

    bigrams = list(zip(document, document[1:]))
    transitions = defaultdict(list)
    for prev, current in bigrams:
        transitions[prev].append(current)

    random.seed(0)
    print("bigram sentences")
    for i in range(10):
        print(i, generate_using_bigrams(transitions))
    print()

    # trigrams

    trigrams = list(zip(document, document[1:], document[2:]))
    trigram_transitions = defaultdict(list)
    starts = []

    for prev, current, next in trigrams:

        if prev == ".":              # if the previous "word" was a period
            starts.append(current)   # then this is a start word

        trigram_transitions[(prev, current)].append(next)

    print("trigram sentences")
    for i in range(10):
        print(i, generate_using_trigrams(starts, trigram_transitions))
    print()

    grammar = {
        "_S"  : ["_NP _VP"],
        "_NP" : ["_N",
                 "_A _NP _P _A _N"],
        "_VP" : ["_V",
                 "_V _NP"],
        "_N"  : ["data science", "Python", "regression"],
        "_A"  : ["big", "linear", "logistic"],
        "_P"  : ["about", "near"],
        "_V"  : ["learns", "trains", "tests", "is"]
```

```
    }

    print("grammar sentences")
    for i in range(10):
        print(i, " ".join(generate_sentence(grammar)))
    print()

    print("gibbs sampling")
    comparison = compare_distributions()
    for roll, (gibbs, direct) in comparison.items():
        print(roll, gibbs, direct)


    # topic MODELING

    for k, word_counts in enumerate(topic_word_counts):
        for word, count in word_counts.most_common():
            if count > 0: print(k, word, count)

    topic_names = ["Big Data and programming languages",
                   "databases",
                   "machine learning",
                   "statistics"]

    for document, topic_counts in zip(documents, document_topic_counts):
        print(document)
        for topic, count in topic_counts.most_common():
            if count > 0:
                print(topic_names[topic], count)
        print()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_27256/2265568515.py in <module>
    242 if __name__ == "__main__":
    243
--> 244     document = get_document()
    245
    246     bigrams = list(zip(document, document[1:]))

~\AppData\Local\Temp/ipykernel_27256/2265568515.py in get_document()
     45
     46
---> 47     for paragraph in content("p"):
     48         words = re.findall(regex, fix_unicode(paragraph.text))
     49         document.extend(words)

TypeError: 'NoneType' object is not callable
```

Statistics

In [12]:

```
from __future__ import division
from collections import Counter
from linear_algebra import sum_of_squares, dot
import math

num_friends = [100,49,41,40,25,21,21,19,19,18,18,16,15,15,15,15,14,14,13,13,

def make_friend_counts_histogram(plt):
    friend_counts = Counter(num_friends)
    xs = range(101)
    ys = [friend_counts[x] for x in xs]
    plt.bar(xs, ys)
    plt.axis([0,101,0,25])
    plt.title("Histogram of Friend Counts")
    plt.xlabel("# of friends")
    plt.ylabel("# of people")
    plt.show()

num_points = len(num_friends)               # 204

largest_value = max(num_friends)            # 100
smallest_value = min(num_friends)           # 1

sorted_values = sorted(num_friends)
smallest_value = sorted_values[0]           # 1
second_smallest_value = sorted_values[1]    # 1
second_largest_value = sorted_values[-2]    # 49

# this isn't right if you don't from __future__ import division
def mean(x):
    return sum(x) / len(x)

def median(v):
    """finds the 'middle-most' value of v"""
    n = len(v)
    sorted_v = sorted(v)
    midpoint = n // 2

    if n % 2 == 1:
        # if odd, return the middle value
        return sorted_v[midpoint]
    else:
        # if even, return the average of the middle values
        lo = midpoint - 1
        hi = midpoint
        return (sorted_v[lo] + sorted_v[hi]) / 2

def quantile(x, p):
    """returns the pth-percentile value in x"""
    p_index = int(p * len(x))
    return sorted(x)[p_index]

def mode(x):
    """returns a list, might be more than one mode"""
    counts = Counter(x)
    max_count = max(counts.values())
```

```
    return [x_i for x_i, count in counts.iteritems()
            if count == max_count]

# "range" already means something in Python, so we'll use a different name
def data_range(x):
    return max(x) - min(x)

def de_mean(x):
    """translate x by subtracting its mean (so the result has mean 0)"""
    x_bar = mean(x)
    return [x_i - x_bar for x_i in x]

def variance(x):
    """assumes x has at least two elements"""
    n = len(x)
    deviations = de_mean(x)
    return sum_of_squares(deviations) / (n - 1)

def standard_deviation(x):
    return math.sqrt(variance(x))

def interquartile_range(x):
    return quantile(x, 0.75) - quantile(x, 0.25)

####
#
# CORRELATION
#
#####

daily_minutes = [1,68.77,51.25,52.08,38.36,44.54,57.13,51.4,41.42,31.22,34.7

def covariance(x, y):
    n = len(x)
    return dot(de_mean(x), de_mean(y)) / (n - 1)

def correlation(x, y):
    stdev_x = standard_deviation(x)
    stdev_y = standard_deviation(y)
    if stdev_x > 0 and stdev_y > 0:
        return covariance(x, y) / stdev_x / stdev_y
    else:
        return 0 # if no variation, correlation is zero

outlier = num_friends.index(100) # index of outlier

num_friends_good = [x
                    for i, x in enumerate(num_friends)
                    if i != outlier]

daily_minutes_good = [x
                      for i, x in enumerate(daily_minutes)
                      if i != outlier]



if __name__ == "__main__":
```

```
114
115     print("num_points"), len(num_friends)
116     print("largest value"), max(num_friends)
117     print("smallest value"), min(num_friends)
118     print("second_smallest_value"), sorted_values[1]
119     print("second_largest_value"), sorted_values[-2]
120     print("mean(num_friends)"), mean(num_friends)
121     print("median(num_friends)"), median(num_friends)
122     print("quantile(num_friends, 0.10)"), quantile(num_friends, 0.10)
123     print("quantile(num_friends, 0.25)"), quantile(num_friends, 0.25)
124     print("quantile(num_friends, 0.75)"), quantile(num_friends, 0.75)
125     print("quantile(num_friends, 0.90)"), quantile(num_friends, 0.90)
126     print("mode(num_friends)"), mode(num_friends)
127     print("data_range(num_friends)"), data_range(num_friends)
128     print("variance(num_friends)"), variance(num_friends)
129     print("standard_deviation(num_friends)"), standard_deviation(num_friends
130     print("interquartile_range(num_friends)"), interquartile_range(num_frier
131
132     print("covariance(num_friends, daily_minutes)"), covariance(num_friends,
133     print("correlation(num_friends, daily_minutes)"), correlation(num_friend
134     print("correlation(num_friends_good, daily_minutes_good)"), correlation(
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_27256/1416293811.py in <module>
      2 from collections import Counter
      3 import scipy.linalg
----> 4 from linear_algebra import sum_of_squares, dot
      5 import math
      6

ModuleNotFoundError: No module named 'linear_algebra'
```

In [ ]:

```
1
```

Name: Krish Agarwal
Registration NUmber: 21112016
Class: 2BSc DS A
Subject: Data Structures Using Python
CIA - I

Random Numbers:	92	47	40	16	82	22	12	13	56	73	24	55	18	66

Question) With the above numbers, perform the below mentioned sorts/searches:

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**
4. **Merge Sort**
5. **Quick Sort**
6. **Linear Search**
7. **Binary Search**

Bubble Sort

Positions	0	1	2	3	4	5	6	7	8	9
First Pass	92	47	40	16	82	22	12	13	56	73
	47	92	40	16	82	22	12	13	56	73
	47	40	92	16	82	22	12	13	56	73
	47	40	16	92	82	22	12	13	56	73
	47	40	16	82	92	22	12	13	56	73
	47	40	16	82	22	92	12	13	56	73
	47	40	16	82	22	12	92	13	56	73
	47	40	16	82	22	12	13	92	56	73
	47	40	16	82	22	12	13	56	92	73
	47	40	16	82	22	12	13	56	73	92
Second Pass	40	16	47	82	22	12	13	56	73	92
Third Pass	16	40	47	82	22	12	13	56	73	92
Fourth Pass	16	40	47	82	22	12	13	56	73	92
Fifth Pass	16	40	47	22	12	13	56	73	82	92
Sixth Pass	16	40	47	12	13	22	56	73	82	92

Unsorted Array

The cell is selected and is checked if it can be swapped

Checking the next cell if it is smaller than the previous cell

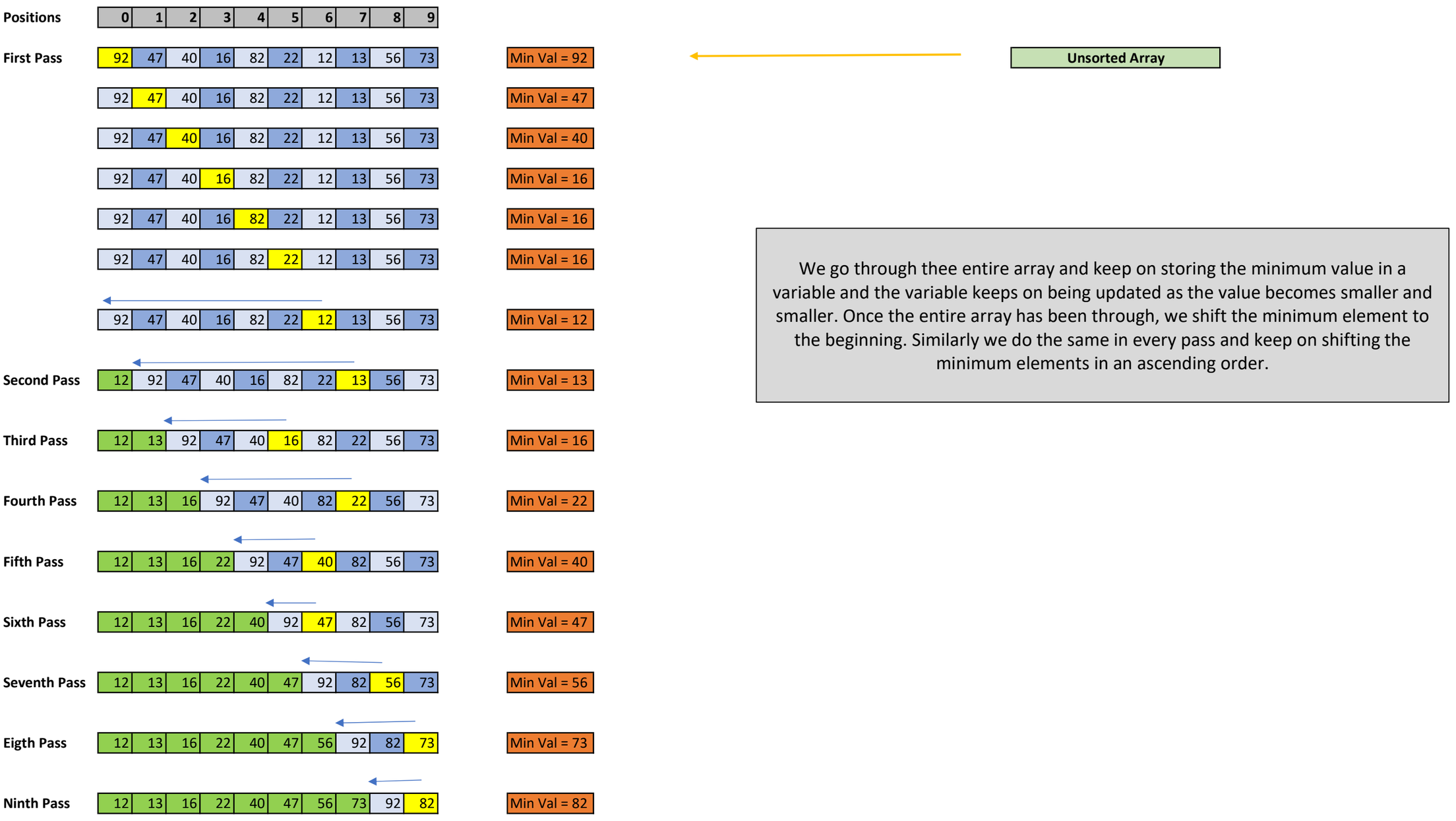
We choose the first element of the array, then we check it with the preceeding element and if the preceeding number is smaller, we swap the elements and so on. And if the preceeding element is greater, we do not swap and stop there. After executing this whole process, we move on to the next element and do the same. We end the entire process once no elements can further be swapped.

The number being swapped

The next number to be checked

Sorted Array

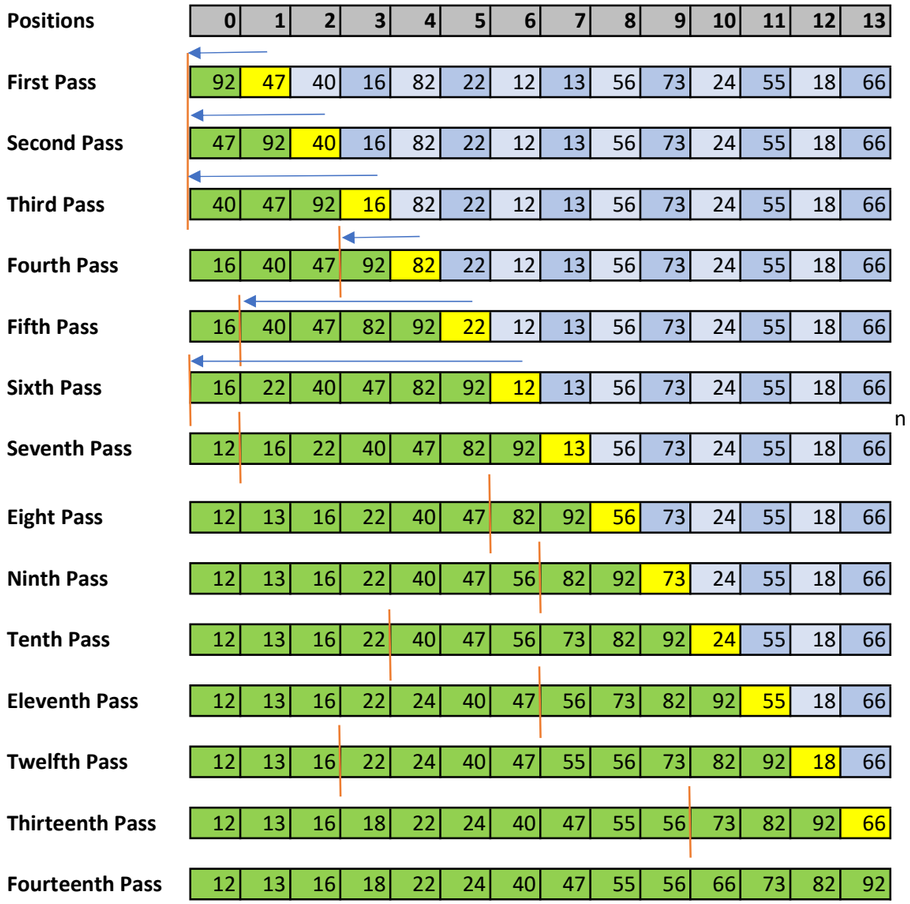
Selection Sort



We go through thee entire array and keep on storing the minimum value in a variable and the variable keeps on being updated as the value becomes smaller and smaller. Once the entire array has been through, we shift the minimum element to the beginning. Similarly we do the same in every pass and keep on shifting the minimum elements in an ascending order.

Tenth Pass

Insertion Sort



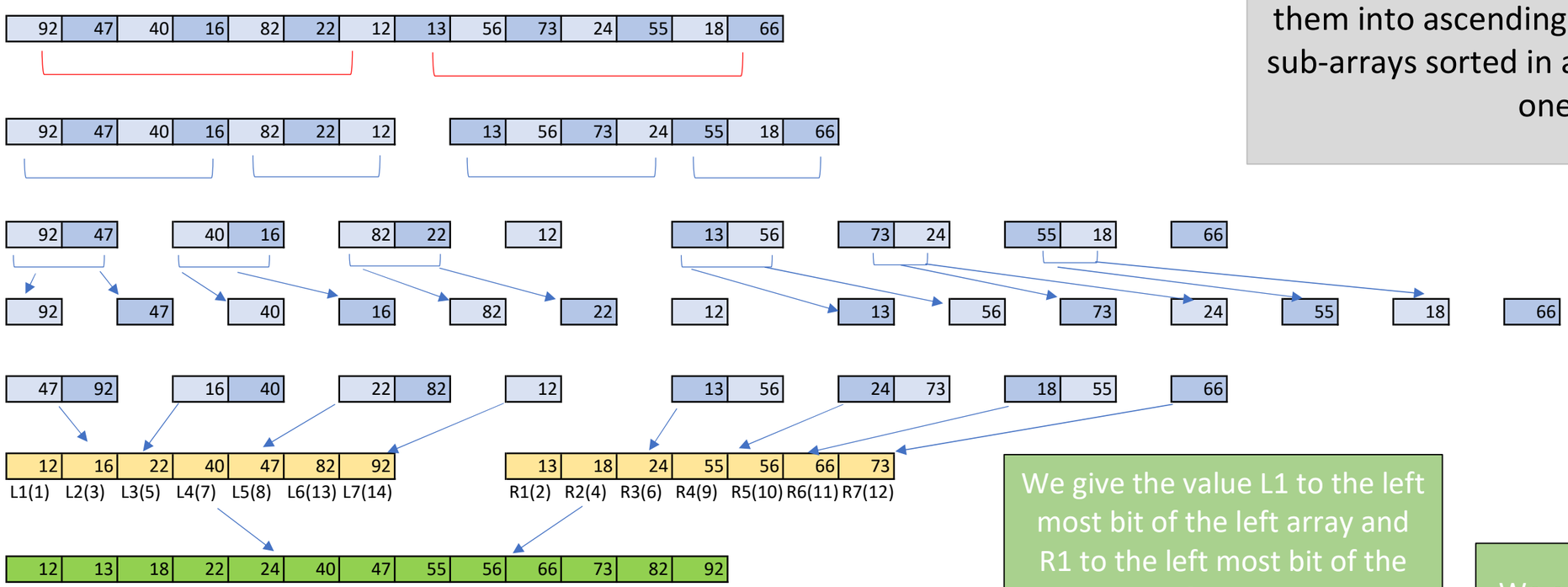
Unsorted Array

Sorted Array

We start with the second element of the array and check whether it is greater than or smaller than the previous element, first element. Accordingly, we place the element to the left or the right of the first element. We then do the same with the next element and so on. In the end, we get a sorted array in ascending order.

The red line indicates where the number is to be inserted

Merge Sort



We divide the array into 2 parts initially. Then we di into 2 more parts and on and on. We individualize them into ascending order and start merging them sub-arrays sorted in ascending order and have to m one single array in ascending orde

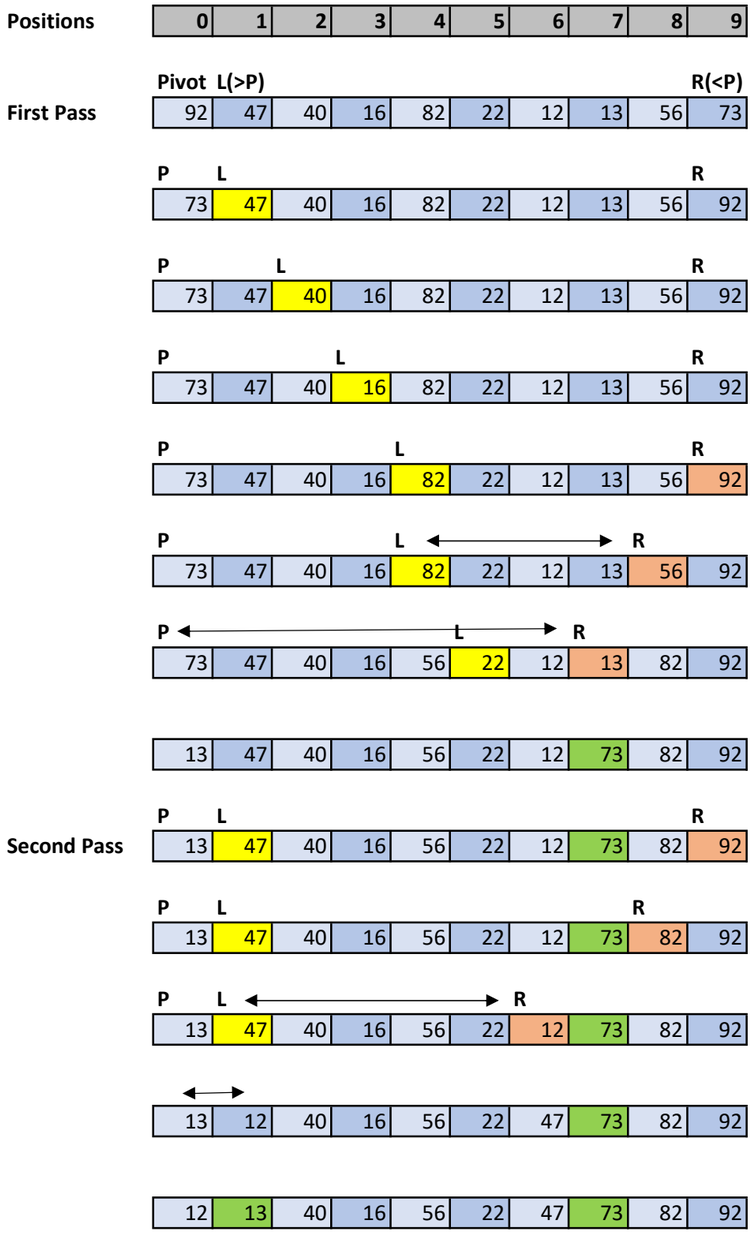
The numbers in the () are the step numbers meaning in which order did we go here.

We give the value L1 to the left most bit of the left array and R1 to the left most bit of the right array. If L1<R1, we put L1 as the first element of the sorted array, then L moves a step forward and thus, L2. We check if L2<R1 but R1<L2 so we put R2 as the second element of the sorted array, then R moves a step forward and thus, R2.

We continue this process (as seen besides) until we reach a sorted array.

ivide the sub-arrays
the elements, sort
. We end up with 2
erge them to make
r.

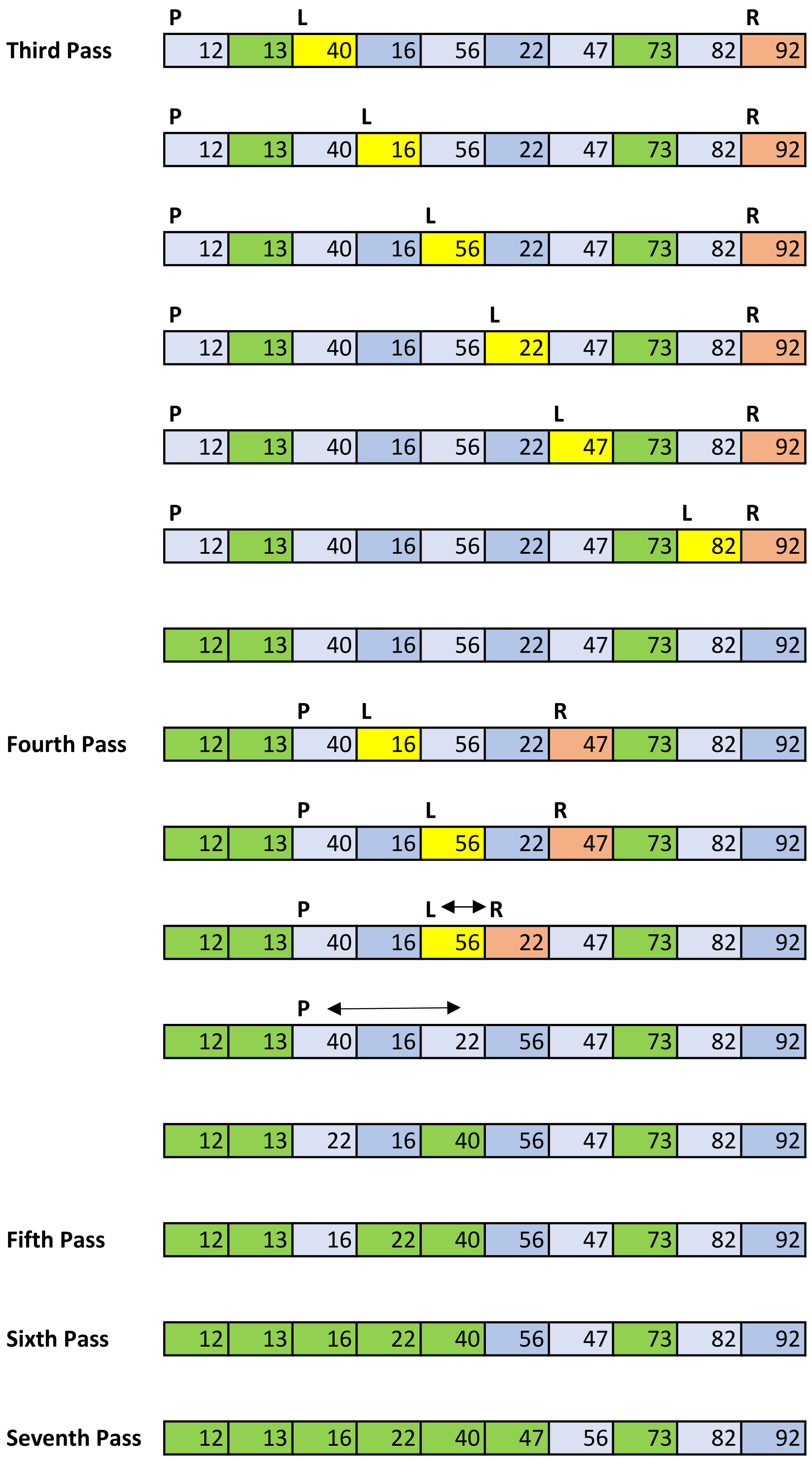
Quick Sort



In this sorting technique, we have a pivot element (P), a left mark (L) and a right mark (R). The first element being the pivot element and the first and the last element after the pivot element as L and R respectively.

We start with L, we check if the element is larger than the pivot number. If not, we move 1 position ahead and do the same. If an element is bigger than the pivot element, we start with R.

From the right side, we check if elements are smaller than the pivot element. If not we move 1 position ahead and do the same. If an element is smaller than the pivot element, we swap the element with the element where we stopped L.



We do this until all the elements have been through it and when all the elements are done, we put the pivot element in the right place (swap it in the right place) and that particular element is sorted.

Now the first element is a new pivot number and we execute the same process again until we have a fully sorted array (ascending order).

Eigth Pass

12	13	16	22	40	47	56	73	82	92

Ninth Pass

12	13	16	22	40	47	56	73	82	92

Linear Search

Ex: Finding 24 in the given array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Positions														
First Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Second Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Third Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Fourth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Fifth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Sixth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Seventh Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Eigth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Ninth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Tenth Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
Eleventh Pass	92	47	40	16	82	22	12	13	56	73	24	55	18	66
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Output: 10

In this searching technique, we go through the array, one element by one element. And as we reach the the element we return its position. In this example, we took 24 and its positions is 10.

Binary Search

Ex: Finding 13 in the given array

Positions	0	1	2	3	4	5	6	7	8	9
	92	47	40	16	82	22	12	13	56	73

Sorting the list in ascending order

First Pass

12	13	16	22	40	47	56	73	82	92
S				M					E

M: 0+9//2 = 4

Second Pass

12	13	16	22	40	47	56	73	82	92
S		M		R					

M: 0+4//2 = 2

Third Pass

12	13	16	22	40	47	56	73	82	92
S	M	E							

M: 0+2//2 = 1

Output: 1

We divide the array into 2 parts; the first value L, the middle value M and the end value E. We divide the array into 2 parts by M = L+E//2 and and check if the number to be found is gretaer than, smaller than or equal to the middle value.

If the number is equal to the middle value then the search is over but if it doesnot match M, we move to either left or right depending upon the number being smaller or greater and terminate the other leftover array.

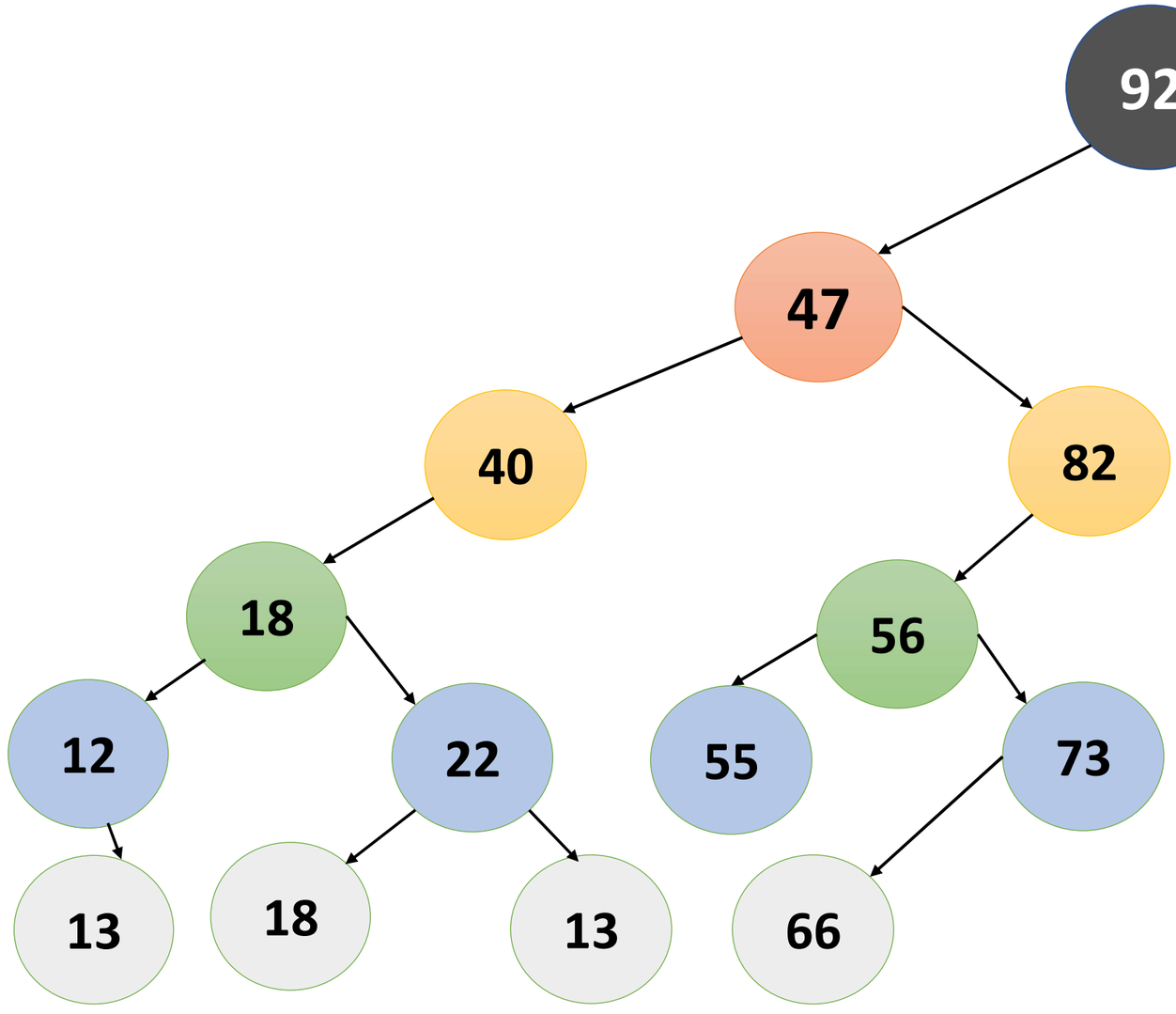
We keep on diving the array into 2 parts (S, M, E) until we find the number.

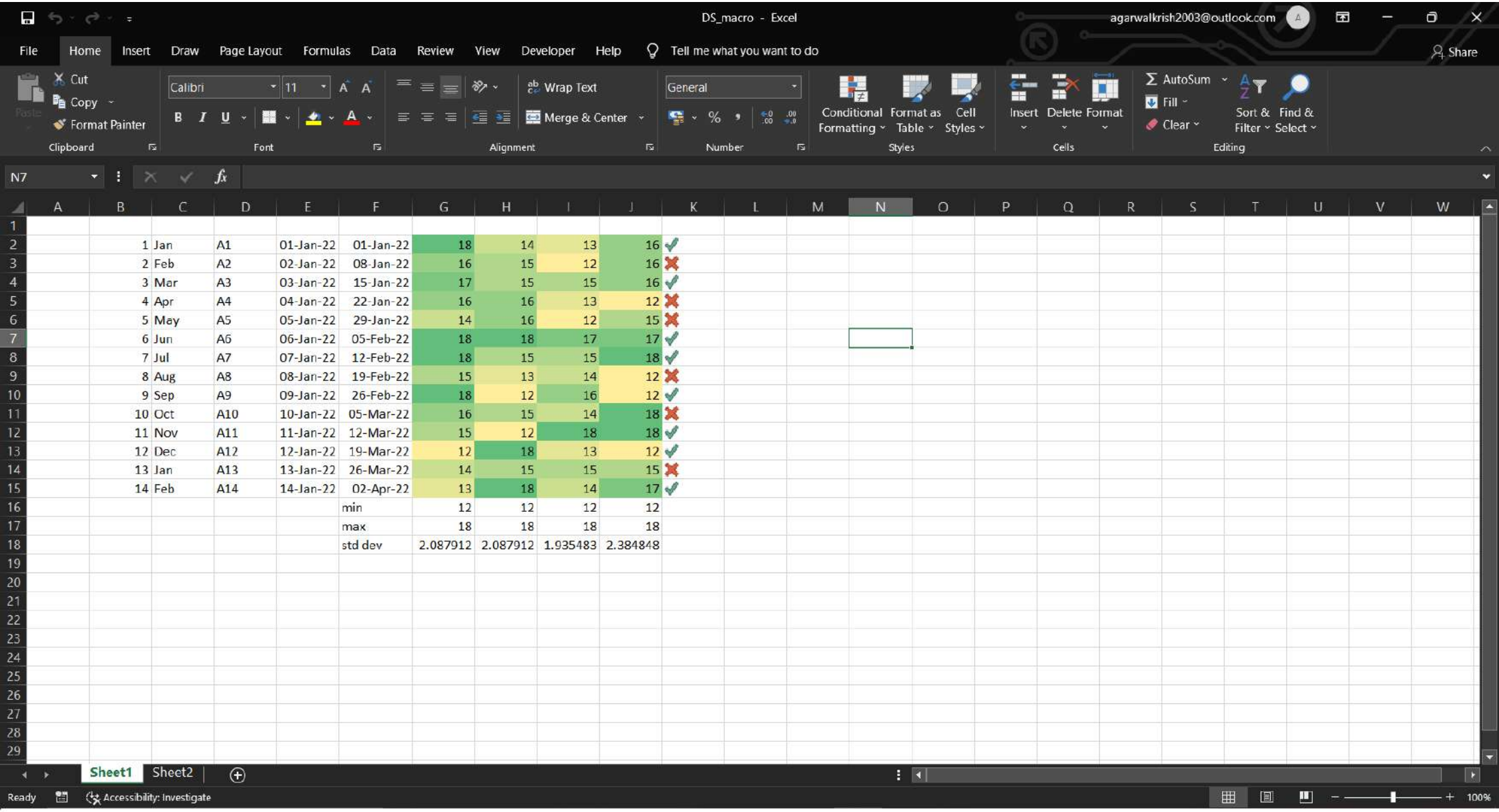
Binary Search Tree

Positions	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	92	47	40	16	82	22	12	13	56	73	24	55	18	66

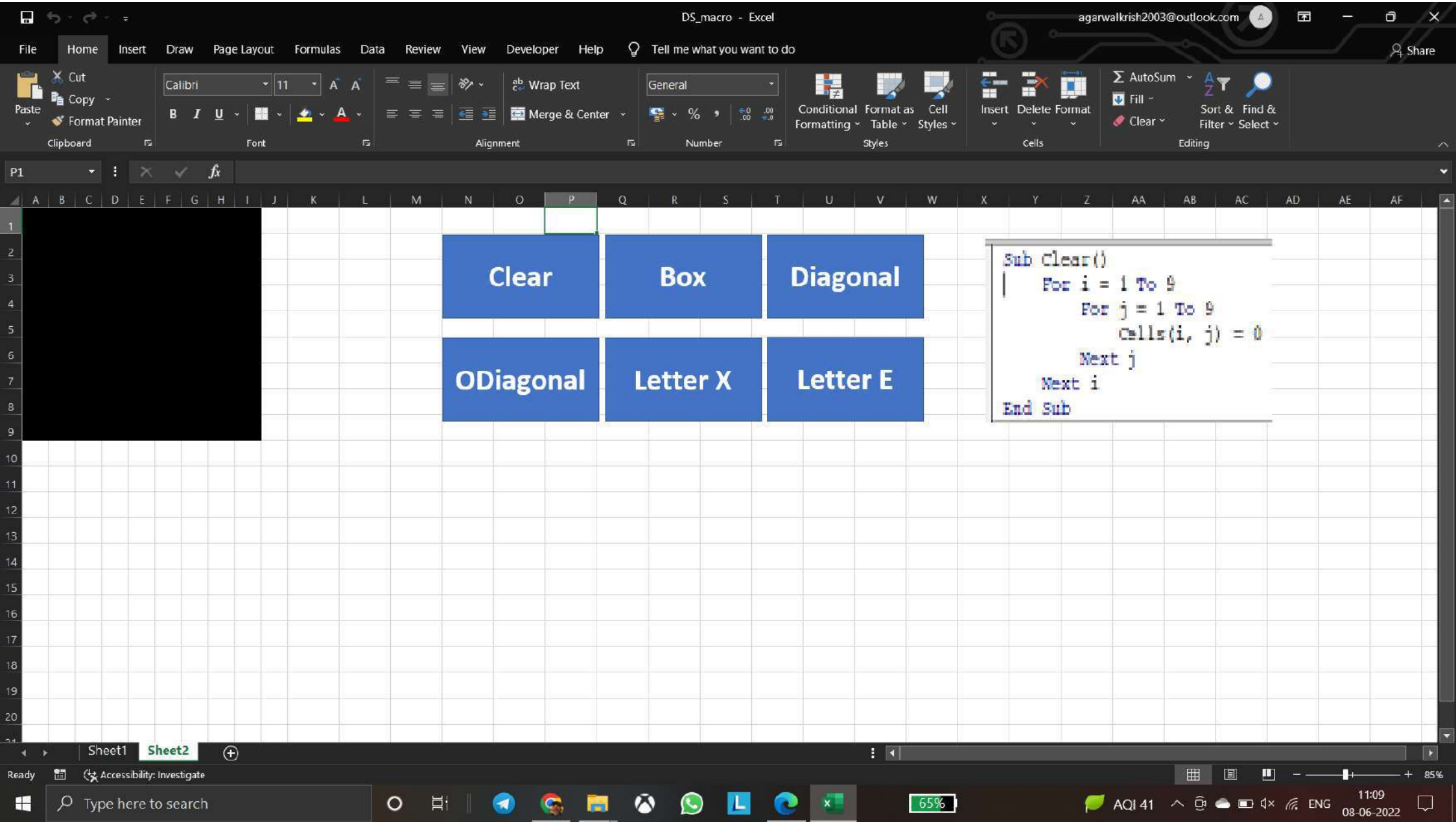
The Pivot element i.e the first element becomes the base of the tree. We then move on to the next element and check if it is greater than or smaller than the base element. If the element is smaller we create a branch to the left and if greater then to the right.

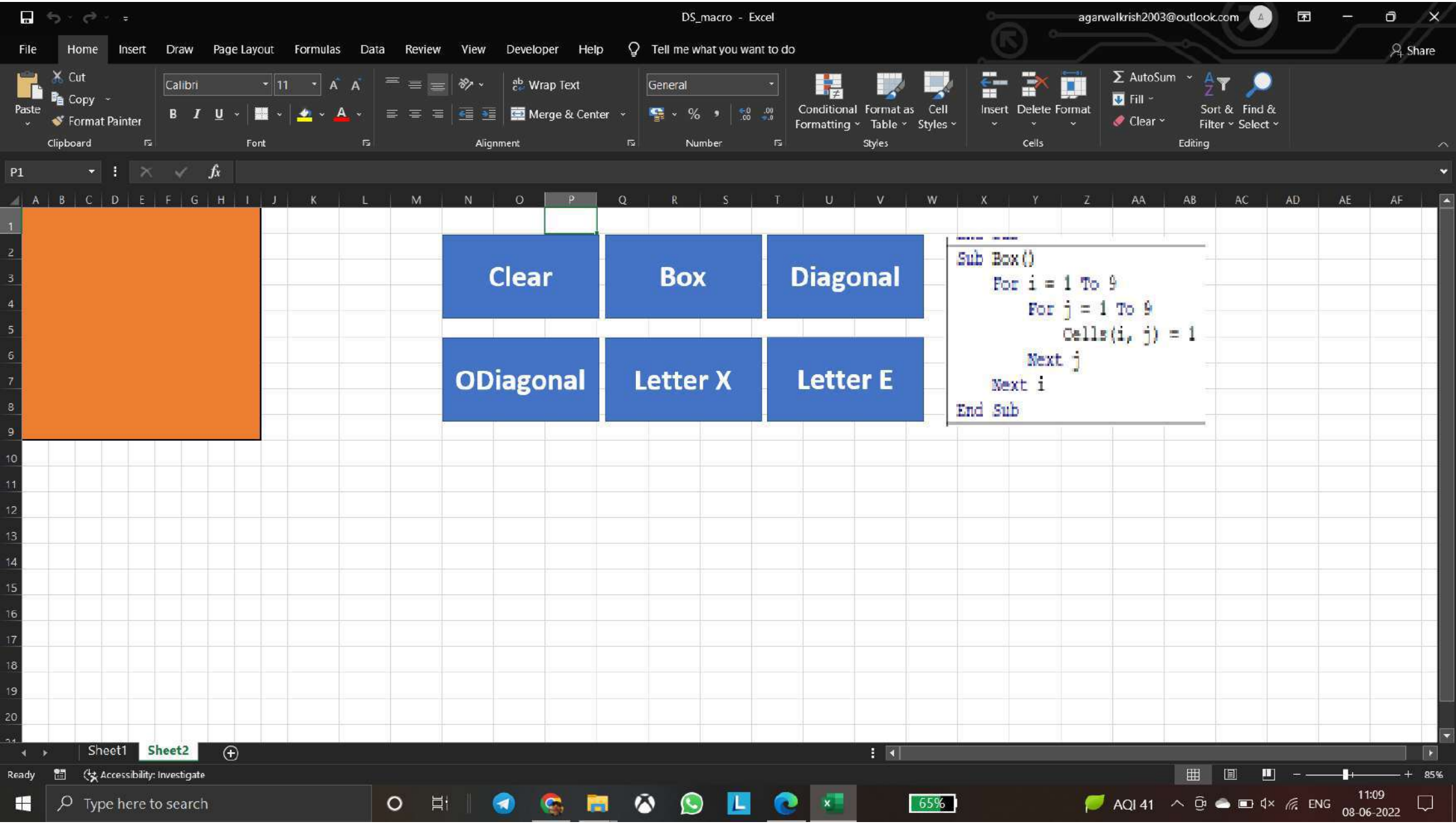
Similarly, we do the same to every element and keep on branching the tree until the array of elements is over.

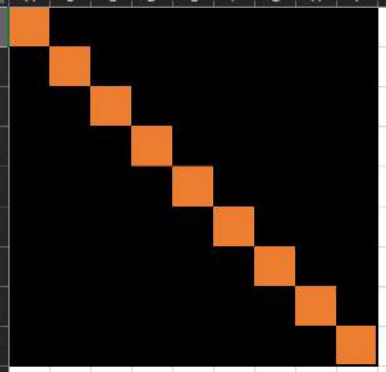




1	Jan	A1	01-Jan-22	01-Jan-22	18	14	13	16	✓	
2	Feb	A2	02-Jan-22	08-Jan-22	16	15	12	16	✗	
3	Mar	A3	03-Jan-22	15-Jan-22	17	15	15	16	✓	
4	Apr	A4	04-Jan-22	22-Jan-22	16	16	13	12	✗	
5	May	A5	05-Jan-22	29-Jan-22	14	16	12	15	✗	
6	Jun	A6	06-Jan-22	05-Feb-22	18	18	17	17	✓	
7	Jul	A7	07-Jan-22	12-Feb-22	18	15	15	18	✓	
8	Aug	A8	08-Jan-22	19-Feb-22	15	13	14	12	✗	
9	Sep	A9	09-Jan-22	26-Feb-22	18	12	16	12	✓	
10	Oct	A10	10-Jan-22	05-Mar-22	16	15	14	18	✗	
11	Nov	A11	11-Jan-22	12-Mar-22	15	12	18	18	✓	
12	Dec	A12	12-Jan-22	19-Mar-22	12	18	13	12	✓	
13	Jan	A13	13-Jan-22	26-Mar-22	14	15	15	15	✗	
14	Feb	A14	14-Jan-22	02-Apr-22	13	18	14	17	✓	
				min	12	12	12	12		
				max	18	18	18	18		
				std dev	2.087912	2.087912	1.935483	2.384848		


DS_macro - Excel
agarwalkrish2003@outlook.com
File
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Developer
Help
Tell me what you want to do
Share
Cut
Copy
Format Painter
Paste
Clipboard
Calibri
11
Font
Wrap Text
Merge & Center
Alignment
General
Number
Conditional Formatting
Format as Table
Cell Styles
Styles
Insert
Delete
Format
Cells
AutoSum
Fill
Clear
Sort & Filter
Find & Select
Editing
P1
Clear
Box
Diagonal
ODiagonal
Letter X
Letter E
Sub Clear()
For i = 1 To 9
For j = 1 To 9
Cells(i, j) = 0
Next j
Next i
End Sub
Sheet1
Sheet2
Ready
Accessibility: Investigate
85%
Type here to search
65%
AQI 41
ENG
11:09
08-06-2022

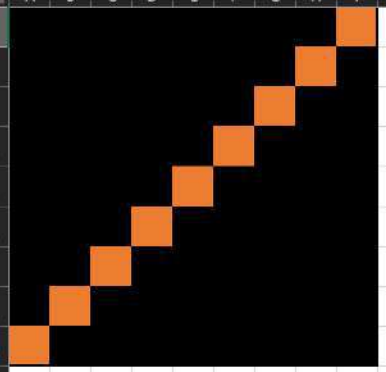

DS_macro - Excel
agarwalkrish2003@outlook.com
File
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Developer
Help
Tell me what you want to do
Share
Cut
Copy
Format Painter
Paste
Clipboard
Calibri
11
Font
Wrap Text
Merge & Center
Alignment
General
Number
Conditional Formatting
Format as Table
Cell Styles
Styles
Insert
Delete
Format
Cells
AutoSum
Fill
Clear
Sort & Filter
Find & Select
Editing
P1
Clear
Box
Diagonal
ODiagonal
Letter X
Letter E
Sub Box()
For i = 1 To 9
For j = 1 To 9
Cells(i, j) = 1
Next j
Next i
End Sub
Sheet1
Sheet2
Ready
Accessibility: Investigate
85%
Type here to search
65%
AQI 41
ENG
11:09
08-06-2022



Clear | Box | Diagonal

ODiagonal | Letter X | Letter E

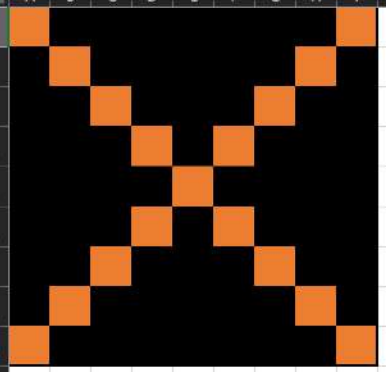
```
Sub Diagonal()
    For i = 1 To 9
        For j = 1 To 9
            If (i = j) Then
                Cells(i, j) = 1
            Else
                Cells(i, j) = 0
            End If
        Next j
    Next i
End Sub
```



Clear | Box | Diagonal

ODiagonal | Letter X | Letter E

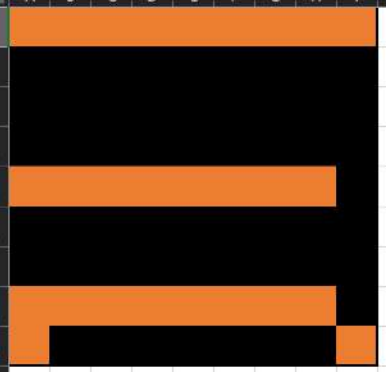
```
Sub oDiagonal()
    For i = 1 To 9
        For j = 1 To 9
            If (i + j = 9 + 1) Then
                Cells(i, j) = 1
            Else
                Cells(i, j) = 0
            End If
        Next j
    Next i
End Sub
```



Clear | Box | Diagonal

ODiagonal | Letter X | Letter E

```
Sub X()
    For i = 1 To 9
        For j = 1 To 9
            If (i + j = 9 + 1) Or (i = j) Then
                Cells(i, j) = 1
            Else
                Cells(i, j) = 0
            End If
        Next j
    Next i
End Sub
```



Clear | Box | Diagonal

ODiagonal | Letter X | Letter E

```
Sub E()
    For i = 1 To 9 - 1
        For j = 1 To 9 - 1
            If (j = 2 And i = 2 And i = 8) Or (i = 1) Or (i = 5) Or (i = 9 - 1) Then
                Cells(i, j) = 1
            Else
                Cells(i, j) = 0
            End If
        Next j
    Next i
End Sub
```