

# Vishwakarma Government Engineering College

## Computer Engineering

### INDEX

**Subject:** Python for Data Science (3150713)

**Enrollment No:** 180170107030

**Name of the Student:** Gabu Siddharth Merambhai

Sr. No	Aim	CO addressed	Date	Page No	Marks (out of 10)	Faculty Signature
1	<b>Practical Set 1:</b> Write python script for following <ol style="list-style-type: none"><li>1. Given a two integer numbers return their product and if the product is greater than 1000, then return their sum</li><li>2. Given a range of first 10 numbers, Iterate from start number to the end number and print the sum of the current number and previous number</li><li>3. Given a string, display only those characters which are present at an even index number.</li><li>4. Given a string and an integer number n, remove characters from a string starting from zero up to n and return a new string</li><li>5. Given a list of numbers, return True if first and last number of a list is same</li><li>6. Given a list of numbers, Iterate it and print only those numbers which are divisible of 5</li><li>7. Return the total count of string "Emma" appears in the given string</li><li>8. Reverse a given number and return true if it is the same as the original number</li><li>9. Given a two list of numbers create a new list such that new list should contain only odd numbers from the first list and even numbers from the second list</li><li>10. Accept two numbers from the user and calculate multiplication</li><li>11. Display "My Name Is James" as "My**Name**Is**James" using output formatting of a print() function</li><li>12. Convert decimal number to octal using print() output formatting</li><li>13. Display float number with 2 decimal places using print()</li><li>14. Accept list of 5 float numbers as a input from user</li><li>15. write all file content into new file by skiping line 5 from following file</li></ol>	CO1	8/7	6		

# Vishwakarma Government Engineering College

## Computer Engineering

	<p>16. Accept any three string from one input() call</p> <p>17. You have the following data. totalMoney = 1000 ,quantity = 3 price = 450 display above data using string.format() method</p> <p>18. How to check file is empty or not</p> <p>19. Read line number 4 from the given file</p>					
2	<p><b>Practical Set 2:</b> Write python script for following</p> <ol style="list-style-type: none"> <li>Print the following pattern 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5</li> <li>Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]</li> <li>Write a function func1() such that it can accept a variable length of argument and print all arguments value func1(20, 40, 60) func1(80, 100)</li> <li>Write a function calculation () such that it can accept two variables and calculate the addition and subtraction of it. And also it must return both addition and subtraction in a single return call</li> <li>Write a recursive function to calculate the sum of numbers from 0 to 10</li> </ol>	CO1	15/7	12		
3	<p><b>Practical Set 3:</b> Write python script for following</p> <ol style="list-style-type: none"> <li>Given a string of odd length greater 7, return a string made of the middle three chars of a given String</li> <li>Given 2 strings, s1 and s2, create a new string by appending s2 in the middle of s1</li> <li>Given 2 strings, s1, and s2 return a new string made of the first, middle and last char each input string</li> <li>Find all occurrences of "USA" in given string ignoring the case</li> <li>Given a two list. Create a third list by picking an odd-index element from the first list and even index elements from second.</li> <li>Given a list iterate it and count the occurrence of each element and create a</li> </ol>	CO1	22/7	14		

# Vishwakarma Government Engineering College

## Computer Engineering

	<p>dictionary to show the count of each element</p> <p>7. Given a two list of equal size create a set such that it shows the element from both lists in the pair</p> <p>8. Iterate a given list and Check if a given element already exists in a dictionary as a key's value if not delete it from the list</p>					
4	<p><b>Practical Set 4:</b></p> <p>Write python script for following</p> <ol style="list-style-type: none"> <li>1. Reverse the tuple.</li> <li>2. Unpack the tuple having 4 items into 4 variables</li> <li>3. Check if all items in the tuple are the same</li> <li>4. convert two lists into the dictionary</li> <li>5. Merge two Python dictionaries into one</li> <li>6. Delete set of keys from Python Dictionary</li> <li>7. Check if a value 200 exists in a dictionary</li> </ol>	CO1	29/7	17		
5	<p><b>Practical Set 5 :</b></p> <ol style="list-style-type: none"> <li>1. Write a NumPy program to compute the multiplication of two given matrixes</li> <li>2. Write a NumPy program to compute the cross product of two given vectors</li> <li>3. Write a NumPy program to compute the determinant of a given square array</li> <li>4. Write a NumPy program to compute the inverse of a given matrix</li> <li>5. Write a NumPy program to compute the eigenvalues and right eigenvectors of a given square array.</li> </ol>	CO2	5/8	20		
6	<p><b>Practical Set 6 :</b></p> <ol style="list-style-type: none"> <li>1. Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array</li> <li>2. Write a NumPy program to create a structured array from given student name, height, class and their data types. Now sort by class, then height if class are equal.</li> <li>3. Write a NumPy program to get the floor, ceiling and truncated values of the elements of a numpy array.</li> <li>4. Write a NumPy program to multiply a matrix by another matrix of complex numbers and create a new matrix of complex numbers</li> <li>5. Write a NumPy program to find the roots of the following polynomials.               <ol style="list-style-type: none"> <li>a) <math>x^2 - 4x + 7</math>.</li> <li>b) <math>x^4 - 11x^3 + 9x^2 + 11x - 10</math></li> </ol> </li> </ol>	CO2	12/8	22		

# Vishwakarma Government Engineering College

## Computer Engineering

7	<b>Practical Set 7:</b> <ol style="list-style-type: none"> <li>1. Write a Pandas program to write a DataFrame to CSV file using tab separator</li> <li>2. Write a Pandas program to convert a given list of lists into a Dataframe.</li> <li>3. Write a Pandas program to get the first 3 rows of a given DataFrame.</li> <li>4. Write a Pandas program to display a summary of the basic information about a specified DataFrame and its data.</li> </ol>	CO2	2/9	25		
8	<b>Practical Set 8 :</b> <ol style="list-style-type: none"> <li>1. Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.</li> <li>2. Write a Pandas program to select the rows where the score is missing, i.e. is NaN.</li> <li>3. Write a Pandas program to sort the DataFrame first by 'name' in descending order, then by 'score' in ascending order.</li> <li>4. Write a Pandas program to replace the 'qualify' column contains the values 'yes' and 'no' with True and False</li> </ol>	CO2	9/9	27		
9	<b>Practical Set 9</b> <ol style="list-style-type: none"> <li>1. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data                             <ol style="list-style-type: none"> <li>a. Read Total profit of all months and show it using a line plot</li> <li>b. Read all product sales data and show it using a multiline plot</li> <li>c. Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).</li> </ol> </li> <li>2. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data                             <ol style="list-style-type: none"> <li>a. Read toothpaste sales data of each month and show it using a scatter plot</li> <li>b. Read face cream and facewash product sales data and show it using the bar chart</li> <li>c. Calculate total sale data for last year for each product and show it using a Pie chart Note: In Pie chart display Number of units sold per year for each product in percentage.</li> </ol> </li> </ol>	CO3	16/9	29		
10	<b>Practical 10:</b> Create a python program using Scikit-learn to implement the following model on dataset	CO4	7/10	35		

# Vishwakarma Government Engineering College

## Computer Engineering

	1. Logistic Regression 2. Support Vector Machines 3. Naive Bayes 4. Random Forest 5. AdaBoost					
--	---	--	--	--	--	--

### Sample Data for Practical 7 and 8

*Sample DataFrame:*

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',  
                      'Matthew', 'Laura', 'Kevin', 'Jonas'],  
             'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],  
             'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],  
             'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}  
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

### Table for Practical Set 9

month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	total_profit
1	2500	1500	5200	9200	1200	1500	21100	211000
2	2630	1200	5100	6100	2100	1200	18330	183300
3	2140	1340	4550	9550	3550	1340	22470	224700
4	3400	1130	5870	8870	1870	1130	22270	222700
5	3600	1740	4560	7760	1560	1740	20960	209600
6	2760	1555	4890	7490	1890	1555	20140	201400
7	2980	1120	4780	8980	1780	1120	29550	295500
8	3700	1400	5860	9960	2860	1400	36140	361400
9	3540	1780	6100	8100	2100	1780	23400	234000
10	1990	1890	8300	10300	2300	1890	26670	266700
11	2340	2100	7300	13300	2400	2100	41280	412800
12	2900	1760	7400	14400	1800	1760	30020	300200

## Practical Set - 1

1. Given a two integer numbers return their product and if the product is greater than 1000, then return their sum.

```
a = int(input("Enter number 1: "))
b = int(input("Enter number 2: "))
c = a*b
if c > 1000:
    print("number is greater than 1000, sum = ",a+b)
else:
    print("numberr is less than 1000, multiplication = ",c)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_1.py
Enter number 1: 67
Enter number 2: 123
multiplication is greator than 1000, sum = 190
```

2. Given a range of first 10 numbers, Iterate from start number to the end number and print the sum of the current number and previous number

```
x = list()
for i in range(10):
    x.append(int(input("Enter number: ")))
print("Whole list ",x)
for i in range(9):
    print("sum of ",x[i+1]," and ",x[i]," is ",x[i]+x[i+1])
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_2.py
Enter number: 1234
Enter number: 123
Enter number: 123
Enter number: 12
Enter number: 3
Enter number: 4
Enter number: 5
Enter number: 6
Enter number: 45
Enter number: 56
Whole list [1234, 123, 123, 12, 3, 4, 5, 6, 45, 56]
sum of 123 and 1234 is 1357
sum of 123 and 123 is 246
sum of 12 and 123 is 135
sum of 3 and 12 is 15
sum of 4 and 3 is 7
sum of 5 and 4 is 9
sum of 6 and 5 is 11
sum of 45 and 6 is 51
sum of 56 and 45 is 101
```

3. Given a string, display only those characters which are present at an even index number.

```
x = input("Enter String Here: ")
print("even index string: ")
for i in range(len(x)):
    if i%2==0:
        print(x[i],end="")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_3.py
Enter String Here: lomujumogabusid heheheh
even index string:
lmjmgbsdhhhh
```

4. Given a string and an integer number n, remove characters from a string starting from zero up to n and return a new string.

```
x = input("Enter String Here: ")
n = int(input("Enter number Here: "))
y = x[:n]
print("substring to n ",y)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_4.py
Enter String Here: siddharth gabu
Enter number Here: 9
substring to n  siddharth
```

5. Given a list of numbers, return True if first and last number of a list is same

```
x = [int(x) for x in input("Enter multipule number: ").split()]
print("last number and first number is same: ")
if x[0]==x[len(x)-1]:
    print("True")
else:
    print("False")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_5.py
Enter multipule number: 45 56 67 78 99 90 1 23 34 45 5 6 67 78 89 33 44 55 60
last number and first number is same:
False
```

6. Given a list of numbers, Iterate it and print only those numbers which are divisible of 5.

```
x = [int(x) for x in input("Enter multipule number: ").split()]
print("numbers are divisiable by 5:")
for i in range(len(x)):
    if x[i]%5==0:
        print(x[i],end=" ")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_6.py
Enter multipule number: 34 45 56 67 78 89 90 15 35
numbers are divisiable by 5:
45 90 15 35
```

7. Return the total count of string “Emma” appears in the given string

```
x = input("Enter Your String Here: ")
print("In Your given string Emma appears ",x.count("Emma")," times")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_7.py
Enter Your String Here: Emma wattson with Emma stone
In Your given string Emma appears 2 times
```

8. Reverse a given number and return true if it is the same as the original number.

```
x = input("Enter number : ")
y = x[::-1]
print("Is number is palindrom?")
if x == y:
    print("True")
else:
    print("False")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_8.py
Enter number : 1234321
Is number is palindrom?
True
```

9. Given a two list of numbers create a new list such that new list should contain only odd numbers from the first list and even numbers from the second list.

```
x = [int(x) for x in input("Enter multipule value1: ").split()]
y = [int(y) for y in input("Enter multipule value2: ").split()]
z = list()
for i in range(len(x)):
    if x[i]%2!=0:
        z.append(x[i])
for i in range(len(y)):
    if y[i]%2==0:
        z.append(y[i])
print(z)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_9.py
Enter multipule value1: 23 34 45 56 67 78 89
Enter multipule value2: 12 32 43 54 65 76 87 98
[23, 45, 67, 89, 12, 32, 54, 76, 98]
```



10. Accept two numbers from the user and calculate multiplication

```
num1 = int(input("Enter number 1: "))  
num2 = int(input("Enter number 2: "))  
print("Multification of two numbers is ", num1*num2)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_10.py  
Enter number 1: 45  
Enter number 2: 54  
Multification of two numbers is 2430
```

11. Display “My Name Is James” as “My\*\*Name\*\*Is\*\*James” using output formatting of a print() function.

```
print("formatting string using print:")  
print("My", "Name", "Is", "James", sep="**")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_11.py  
formatting string using print:  
My**Name**Is**James
```

12. Convert decimal number to octal using print() output formatting

```
x = int(input("Enter decimel number: "))  
print("octal number: ")  
print('%o' % (x))
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_12.py  
Enter decimel number: 456  
octal number:  
710
```

13. Display float number with 2 decimal places using print()

```
x = float(input("Enter float number: "))  
print("precision to two point is :")  
print("%.2f" % x)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_13.py  
Enter float number: 345.98744  
precision to two point is :  
345.99
```

14. Accept list of 5 float numbers as a input from user

```
numbersl = []
```

```
n = int(input("Enter size of inputs: "))
print("\n")
for i in range(0,n):
    print("Enter float number ",i+1," :")
    x = float(input())
    numbersl.append(x)
print("list : ",numbersl)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_14.py
Enter size of inputs: 5

Enter float number 1 :
12.89
Enter float number 2 :
12
Enter float number 3 :
23.9
Enter float number 4 :
23.9008
Enter float number 5 :
134
list : [12.89, 12.0, 23.9, 23.9008, 134.0]
```

15. Write all file content into new file by skipping line 5 from following file.

```
f1 = open("text1.txt","r")
f2 = open("text2.txt","w")
lines = f1.readlines()
for i in range(len(lines)):
    if i == 4:
        continue
    f2.write(lines[i])
print("successfully done content of text1 > text2 except line 5:")
f1.close()
f2.close()
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_15.py
successfully done content of text1 > text2 except line 5:

F:\work of pro\SEM5\pythonS\Practical_Set_1>more text2.txt
line1
line2
line3
line4
line6
line7
line8
line9
line10
```

16. Accept any three string from one input() call.

```
str1,str2,str3 = input("Enter Three String: ").split()
print(str1," ",str2," ",str3)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_16.py
Enter Three String: sid dj cj
sid  dj  cj
```

17. You have the following data. totalMoney = 1000, quantity = 3, price = 450 display above data using string.format() method

```
txt = "i have {totalMoney} Ruppes so i can't buy {quantity} laptop for {price} Ruppees."
print(txt.format(totalMoney = 1000,price = 450,quantity = 3))
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_17.py
i have 1000 Ruppes so i can't buy 3 laptop for 450 Ruppees.
```

18. How to check file is empty or not

```
import os
print("file is empty?")
print(os.stat("text1.txt").st_size == 0)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_18.py
file is empty?
False
```

19. Read line number 4 from the following file

```
f = open("text2.txt","r")
line = f.readlines()
print(line[3])
print("successfully done")
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_1>python P1_19.py
line4

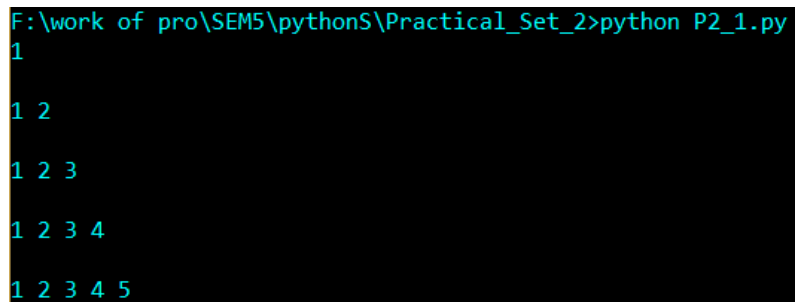
successfully done
```

## Practical set – 2

1. Print the following pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
for i in range(1,6):
    for j in range(1,i+1):
        print(j,"",end="")
    print("\n")
```

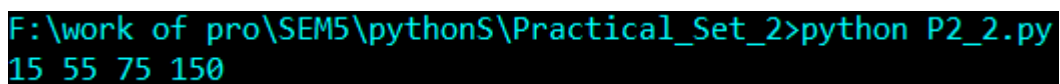


```
F:\work of pro\SEM5\pythonS\Practical_Set_2>python P2_1.py
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

2. Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration

```
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
```

```
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
for i in range(len(list1)):
    if list1[i]>150:
        break
    if list1[i]%5==0:
        print(list1[i],end=" ")
```



```
F:\work of pro\SEM5\pythonS\Practical_Set_2>python P2_2.py
15 55 75 150
```

3. Write a function func1() such that it can accept a variable length of argument and print all arguments value

```
func1(20, 40, 60)
```

```
func1(80, 100)
```

```
def func1(*vargas):
```

```
for i in vargas:
    print(i,end=" ")
print()
func1(20, 40, 60)
func1(80, 100)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_2>python P2_3.py
20 40 60
80 100
```

4. Write a function calculation() such that it can accept two variables and calculate the addition and subtraction of it. And also it must return both addition and subtraction in a single return call.

```
def calculation(a,b):
    return a+b,a-b
suma,sub = calculation(10,5)
print(suma,sub)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_2>python P2_4.py
15 5
```

5. Write a recursive function to calculate the sum of numbers from 0 to 10.

```
def sum_10(n):
    if(n > 0):
        result = n + sum_10(n - 1)
    else:
        result = 0
    return result
print(sum_10(10))
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_2>python P2_5.py
55
```

## Practical set – 3

1. Given a string of odd length greater 7, return a string made of the middle three chars of a given String

```
def getMiddleThreeChars(x):  
    y = int(len(x)/2)  
    z = x[y-1:y+2]  
    return z  
print("Enter odd length string: ")  
x = input()  
z = getMiddleThreeChars(x)  
print("Middle ThreeChars:",z)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_1.py  
Enter odd length string:  
halogarberamava  
Middle ThreeChars: rbe
```

2. Given 2 strings, s1 and s2, create a new string by appending s2 in the middle of s1 Expected Outcome: appendMiddle("Chrisdem", IamNewString) → "ChrIamNewStringisdem"

```
s1 = input("Enter String1 : ")  
s2 = input("Entrr String2 : ")  
l = int(len(s1)/2)  
new = s1[:l]+s2+s1[l:]  
print("appendMiddle(\"Chrisdem\", IamNewString) - > ",new)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_2.py  
Enter String1 : siddharth  
Entrr String2 : gabu  
appendMiddle("Chrisdem", IamNewString) - > siddgabuharth
```

3. Given 2 strings, s1, and s2 return a new string made of the first, middle and last char each input string Expected Outcome: mixString("America", "Japan") = ""AJrpan"

```
s1,s2 = input("Enter string1 : "),input("Enter string2 : ")  
l1,l2 = len(s1),len(s2)  
new = s1[0]+s2[0]+s1[int(l1/2)]+s2[int(l2/2)]+s1[l1-1]+s2[l2-1]  
print("mixString(\"America\", \"Japan\") = ",new)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_3.py  
Enter string1 : siddharth  
Enter string2 : gabu  
mixString("America", "Japan") = sghbhuh
```

4. Find all occurrences of “USA” in given string ignoring the case Expected Outcome: input\_str = "Welcome to USA. usa awesome, isn't it?" The USA count is: 2

```
s = "Welcome to USA. usa awesome, isn't it?"  
ct = s.lower().count("USA".lower())  
print("The USA count is: ",ct)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_4.py  
The USA count is: 2
```

5. Given a two list. Create a third list by picking an odd-index element from the first list and even index elements from second.

```
listOne = [3, 6, 9, 12, 15, 18, 21]  
listTwo = [4, 8, 12, 16, 20, 24, 28]  
newlist = list()  
for i in range(len(listOne)):  
    if i%2!=0:  
        newlist.append(listOne[i])  
for i in range(len(listTwo)):  
    if i%2==0:  
        newlist.append(listTwo[i])  
print("Printing Final third list ",newlist)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_5.py  
Printing Final third list [6, 12, 18, 4, 12, 20, 28]
```

6. Given a list iterate it and count the occurrence of each element and create a dictionary to show the count of each element

```
thislist = [11, 45, 8, 11, 23, 45, 23, 45, 89]  
thisset = set(thislist)  
print(thisset)  
thisdict = dict()  
for i in range(len(thisset)):  
    count = 0  
    y = thisset.pop()  
    for j in range(len(thislist)):  
        if y == thislist[j]:  
            count = count + 1  
    thisdict.update({y:count})  
print("Printing count of each item ",thisdict)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_6.py  
{8, 11, 45, 23, 89}  
Printing count of each item {8: 1, 11: 2, 45: 3, 23: 2, 89: 1}
```

7. Given a two list of equal size create a set such that it shows the element from both lists in the pair.

```
firstList = [2, 3, 4, 5, 6, 7, 8]
secondList = [4, 9, 16, 25, 36, 49, 64]
result = zip(firstList, secondList)
resultSet = set(result)
print("Result is ",resultSet)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_7.py
Result is  {(6, 36), (5, 25), (4, 16), (7, 49), (3, 9), (2, 4), (8, 64)}
```

8. Iterate a given list and Check if a given element already exists in a dictionary as a key's value if not delete it from the list

```
rollNumber = [47, 64, 69, 37, 76, 83, 95, 97]
sampleDict = {'Jhon':47, 'Emma':69, 'Kelly':76, 'Jason':97}
checklist = list(sampleDict.values())
print("before list ",rollNumber)
for i in checklist:
    for j in rollNumber:
        if i == j:
            rollNumber.remove(j)
print("after removing unwanted elements from list ",rollNumber)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_3>python P3_8.py
before list  [47, 64, 69, 37, 76, 83, 95, 97]
after removing unwanted elements from list  [64, 37, 83, 95]
```



## Practical set – 4

1. Reverse the following tuple aTuple = (10, 20, 30, 40, 50)

```
aTuple = (10, 20, 30, 40, 50)
aTuple = aTuple[::-1] #Stride while Slicing Strings
print("Reverse order tuples: ",aTuple)
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_1.py
Reverse order tuples:  (50, 40, 30, 20, 10)
```

2. . Unpack the following tuple into 4 variables aTuple = (10, 20, 30, 40)

```
aTuple = (10, 20, 30, 40)
a,b,c,d = aTuple
print("before unpacking: ",aTuple)
print("After unpacking: ")
print(a)
print(b)
print(c)
print(d)
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_2.py
before unpacking:  (10, 20, 30, 40)
After unpacking:
10
20
30
40
```

3. Check if all items in the following tuple are the same tuple1 = (45, 45, 45, 45).

```
aTuple = (45,45,45,45)
aset = set(aTuple)
print("Tuple contains duplicate: ")
if len(aset) == 1:
    print("True")
else:
    print("False")
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_3.py
Tuple contains duplicate:
True
```

4. Below are the two lists convert it into the dictionary: keys = ['Ten', 'Twenty', 'Thirty'], values = [10, 20, 30]  
keys = ['Ten', 'Twenty', 'Thirty']

```
values = [10, 20, 30]
adict = dict()
for i in range(len(keys)):
    adict.update({keys[i]:values[i]})
print("merging list into dict: ",adict)
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_4.py
merging list into dict: {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

5. Merge following two Python dictionaries into one: dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}, dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
- ```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
adict = dict1.copy()
adict.update(dict2)
print(adict)
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_7.py
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

6. Delete set of keys from Python Dictionary.

```
sampleDict = {
    "name": "Kelly",
    "age":25,
    "salary": 8000,
    "city": "New york"
}
keysToRemove = ["name", "salary"]
print("before removing keys: ",sampleDict)
sampleDict.pop(keysToRemove[0])
sampleDict.pop(keysToRemove[1])
print("after removing keys: ",sampleDict)
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_5.py
before removing keys: {'name': 'Kelly', 'age': 25, 'salary': 8000, 'city': 'New york'}
after removing keys: {'age': 25, 'city': 'New york'}
```

7. Check if a value 200 exists in a dictionary sampleDict = {'a': 100, 'b': 200, 'c': 300}

```
sampleDict = {'a': 100, 'b': 200, 'c': 300}  
print(sampleDict, "This dict contain 200 as value:")  
print(200 in sampleDict.values())
```

```
F:\work of pro\SEM5\pythonS\Practical_set_4>python P4_6.py  
{'a': 100, 'b': 200, 'c': 300} This dict contain 200 as value:  
True
```

## Practical set – 5

1. Write a NumPy program to compute the multiplication of two given matrixes.

```
import numpy as np
x = np.random.rand(3,5)
y = np.random.rand(5,4)
z = np.dot(x,y)
print("Multiplication of ttwo matrix x and y: \n",z)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_5>python P5_1.py
Multiplication of ttwo matrix x and y:
[[1.15067372 0.84175099 0.39143138 0.57352484]
 [1.66863334 1.32162258 0.91163472 1.07818595]
 [0.87126241 0.38054059 0.29050368 0.63059123]]
```

2. Write a NumPy program to compute the cross product of two given vectors.

```
import numpy as np
x = np.random.rand(2,2)
y = np.random.rand(2,2)
z = np.cross(x,y)
print("Cross product of two vector: \n",z)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_5>python P5_2.py
Cross product of two vector:
[-0.19571751 0.25613839]
```

3. Write a NumPy program to compute the determinant of a given square array.

```
import numpy as np
x = np.array([[1,0],[1,2]])
print("determinant of square array ",np.linalg.det(x))
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_5>python P5_3.py
determinant of square array  2.0
```

4. Write a NumPy program to compute the inverse of a given matrix.

```
import numpy as np
x = np.array([[1,2],[3,4]])
print("determinant of array :\n",np.linalg.det(x))
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_5>python P5_4.py
determinant of array :
-2.0000000000000004
```

5. Write a NumPy program to compute the eigenvalues and right eigenvectors of a given square array.

```
import numpy as np
x = np.array([[-1,2],[0,1]])
w, v = np.linalg.eig(x)
print( "Eigenvalues of the matrix",w)
print( "Eigenvectors of the matrix",v)
```

```
F:\work of pro\SEM5\pythonS\Practical_Set_5>python P5_5.py
Eigenvalues of the matrix [-1.  1.]
Eigenvectors of the matrix [[1.         0.70710678]
 [0.         0.70710678]]
```

## Practical – 6

6. Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.

```
import numpy as np
x = np.array([[10,30,20],[20,20,10]])
print("Original array:")
print(x)
print("Sort the array along the first axis:")
print(np.sort(x, axis=0))
print("Sort the array along the last axis:")
print(np.sort(x))
print("Sort the flattened array:")
print(np.sort(x, axis=None))
```

```
Original array:
[[10 30 20]
 [20 20 10]]
Sort the array along the first axis:
[[10 20 10]
 [20 30 20]]
Sort the array along the last axis:
[[10 20 30]
 [10 20 20]]
Sort the flattened array:
[10 10 20 20 20 30]
```

7. Write a NumPy program to create a structured array from given student name, height, class and their data types. Now sort by class, then height if class are equal.

```
import numpy as np
x = np.array([('James', 5, 48.5), ('Nail', 6, 52.5), ('Paul', 5, 42.1), ('Pit', 5, 40.11)], dtype=[('name', 'U10'), ('class', int), ('height', float)])
print("Original array:")
print(x)
print("Sort by height")
print(np.sort(x, order=['class', 'height']))
```

```
Original array:
[('James', 5, 48.5) ('Nail', 6, 52.5) ('Paul', 5, 42.1) ('Pit', 5, 40.11)]
Sort by height
[('Pit', 5, 40.11) ('Paul', 5, 42.1) ('James', 5, 48.5) ('Nail', 6, 52.5)]
```

8. Write a NumPy program to get the floor, ceiling and truncated values of the elements of a numpy array.

```
import numpy as np
```

```
x = np.array([1.7,40,34,23,12.09,-3,-34.56,21])
print("Original array:")
print(x)
print("Floor values of the above array elements:")
print(np.floor(x))
print("Ceil values of the above array elements:")
print(np.ceil(x))
print("Truncated values of the above array elements:")
print(np.trunc(x))
```

```
Original array:
[ 1.7  40.   34.   23.   12.09 -3.  -34.56  21. ]
Floor values of the above array elements:
[ 1.  40.  34.  23.  12.  -3. -35.  21.]
Ceil values of the above array elements:
[ 2.  40.  34.  23.  13.  -3. -34.  21.]
Truncated values of the above array elements:
[ 1.  40.  34.  23.  12.  -3. -34.  21.]
```

9. Write a NumPy program to multiply a matrix by another matrix of complex numbers and create a new matrix of complex numbers.

```
import numpy as np
x = np.array([[11+2j,34+4j],[9+6j,7+12j]])
print("First array:")
print(x)
y = np.array([[9+6j,7+12j],[11+2j,34+4j]])
print("Second array:")
print(y)
z = np.vdot(x, y)
print("Product of above two arrays:")
print(z)
```

```
First array:
[[11. +2.j 34. +4.j]
 [ 9. +6.j  7.+12.j]]
Second array:
[[ 9. +6.j  7.+12.j]
 [11. +2.j 34. +4.j]]
Product of above two arrays:
(794+0j)
```

10. Write a NumPy program to find the roots of the following polynomials.

a)  $x^2 - 4x + 7$ .

b)  $x^4 - 11x^3 + 9x^2 + 11x - 10$

```
import numpy as np
print("Roots of the first polynomial:")
print(np.roots([1, -4, 7]))
print("Roots of the second polynomial:")
```

```
print(np.roots([1, -11, 9, 11, -10]))
```

```
Roots of the first polynomial:  
[2.+1.73205081j 2.-1.73205081j]  
Roots of the second polynomial:  
[10.+0.00000000e+00j -1.+0.00000000e+00j  1.+9.6357437e-09j  
 1.-9.6357437e-09j]
```



## Practical – 7

1. Write a Pandas program to write a DataFrame to CSV file using tab separator.

```
import numpy as np
import pandas as pd
Data = {'Name':['Siddharth','Gabu','SidPro','Ark'],'Age':[27,12,19,45]}
df = pd.DataFrame(data=Data)
print("Original DataFrame")
print(df)
print('Data from new_file.csv file:')
df.to_csv('new_file.csv', sep='\t', index=False)
new_df = pd.read_csv('new_file.csv')
print(new_df)
```

```
Original DataFrame
   Name  Age
0  Siddharth  27
1     Gabu   12
2   SidPro   19
3     Ark   45
Data from new_file.csv file:
   Name\tAge
0  Siddharth\t27
1     Gabu\t12
2   SidPro\t19
3     Ark\t45
```

2. Write a Pandas program to convert a given list of lists into a Dataframe.

```
import numpy as np
import pandas as pd
lit = [[2, 4], [1, 3]]
df = pd.DataFrame(lit, columns=['col1', 'col2'])
print(df)
```

```
   col1  col2
0      2     4
1      1     3
```

3. Write a Pandas program to get the first 3 rows of a given DataFrame..

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
```

```
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, labels)
print(df[0:3])
```

|   | name      | score | attempts | qualify |
|---|-----------|-------|----------|---------|
| a | Anastasia | 12.5  | 1        | yes     |
| b | Dima      | 9.0   | 3        | no      |
| c | Katherine | 16.5  | 2        | yes     |

4. Write a Pandas program to display a summary of the basic information about a specified DataFrame and its data.

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, labels)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   name        10 non-null    object
1   score       8 non-null     float64
2   attempts    10 non-null    int64
3   qualify     10 non-null    object
dtypes: float64(1), int64(1), object(2)
memory usage: 280.0+ bytes
None
```

## Practical – 8

1. Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, labels)
print(df[df.attempts>2])
```

|   | name    | score | attempts | qualify |
|---|---------|-------|----------|---------|
| b | Dima    | 9.0   | 3        | no      |
| d | James   | NaN   | 3        | no      |
| f | Michael | 20.0  | 3        | yes     |

2. Write a Pandas program to select the rows where the score is missing, i.e. is NaN.

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, labels)
print(df[df['score'].isnull()])
```

|   | name  | score | attempts | qualify |
|---|-------|-------|----------|---------|
| d | James | NaN   | 3        | no      |
| h | Laura | NaN   | 1        | no      |

3. Write a Pandas program to sort the DataFrame first by 'name' in descending order, then by 'score' in ascending order.

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
df = pd.DataFrame(exam_data, labels)
df.sort_values(["name", "score"], axis=0,
               ascending=[False, True], inplace=True)
print(df)
```

|   | name      | score | attempts | qualify |
|---|-----------|-------|----------|---------|
| f | Michael   | 20.0  | 3        | yes     |
| g | Matthew   | 14.5  | 1        | yes     |
| h | Laura     | NaN   | 1        | no      |
| i | Kevin     | 8.0   | 2        | no      |
| c | Katherine | 16.5  | 2        | yes     |
| j | Jonas     | 19.0  | 1        | yes     |
| d | James     | NaN   | 3        | no      |
| e | Emily     | 9.0   | 2        | no      |
| b | Dima      | 9.0   | 3        | no      |
| a | Anastasia | 12.5  | 1        | yes     |

4. Write a Pandas program to replace the 'qualify' column contains the values 'yes' and 'no' with True and False

```
import numpy as np
import pandas as pd
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
                     'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
             'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
             'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
             'qualify': ['yes', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, labels)
df['qualify'] = df['qualify'].map({'yes': True, 'no': False})
print(df)
```

|   | name      | score | attempts | qualify |
|---|-----------|-------|----------|---------|
| f | Michael   | 20.0  | 3        | True    |
| g | Matthew   | 14.5  | 1        | True    |
| h | Laura     | NaN   | 1        | False   |
| i | Kevin     | 8.0   | 2        | False   |
| c | Katherine | 16.5  | 2        | True    |
| j | Jonas     | 19.0  | 1        | True    |
| d | James     | NaN   | 3        | False   |
| e | Emily     | 9.0   | 2        | False   |
| b | Dima      | 9.0   | 3        | False   |
| a | Anastasia | 12.5  | 1        | True    |

## Practical – 9

1. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data.

| month_number | facecream | facewash | toothpaste | bathings | shampoo | moisturizer | total_units | total_profit |
|--------------|-----------|----------|------------|----------|---------|-------------|-------------|--------------|
| 1            | 2500      | 1500     | 5200       | 9200     | 1200    | 1500        | 21100       | 211000       |
| 2            | 2630      | 1200     | 5100       | 6100     | 2100    | 1200        | 18330       | 183300       |
| 3            | 2140      | 1340     | 4550       | 9550     | 3550    | 1340        | 22470       | 224700       |
| 4            | 3400      | 1130     | 5870       | 8870     | 1870    | 1130        | 22270       | 222700       |
| 5            | 3600      | 1740     | 4560       | 7760     | 1560    | 1740        | 20960       | 209600       |
| 6            | 2760      | 1555     | 4890       | 7490     | 1890    | 1555        | 20140       | 201400       |
| 7            | 2980      | 1120     | 4780       | 8980     | 1780    | 1120        | 29550       | 295500       |
| 8            | 3700      | 1400     | 5860       | 9960     | 2860    | 1400        | 36140       | 361400       |
| 9            | 3540      | 1780     | 6100       | 8100     | 2100    | 1780        | 23400       | 234000       |
| 10           | 1990      | 1890     | 8300       | 10300    | 2300    | 1890        | 26670       | 266700       |
| 11           | 2340      | 2100     | 7300       | 13300    | 2400    | 2100        | 41280       | 412800       |
| 12           | 2900      | 1760     | 7400       | 14400    | 1800    | 1760        | 30020       | 300200       |

- a) Read Total profit of all months and show it using a line plot.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('F:\work of pro\SEM5\pythonS\Practical_Set_9\data.csv')

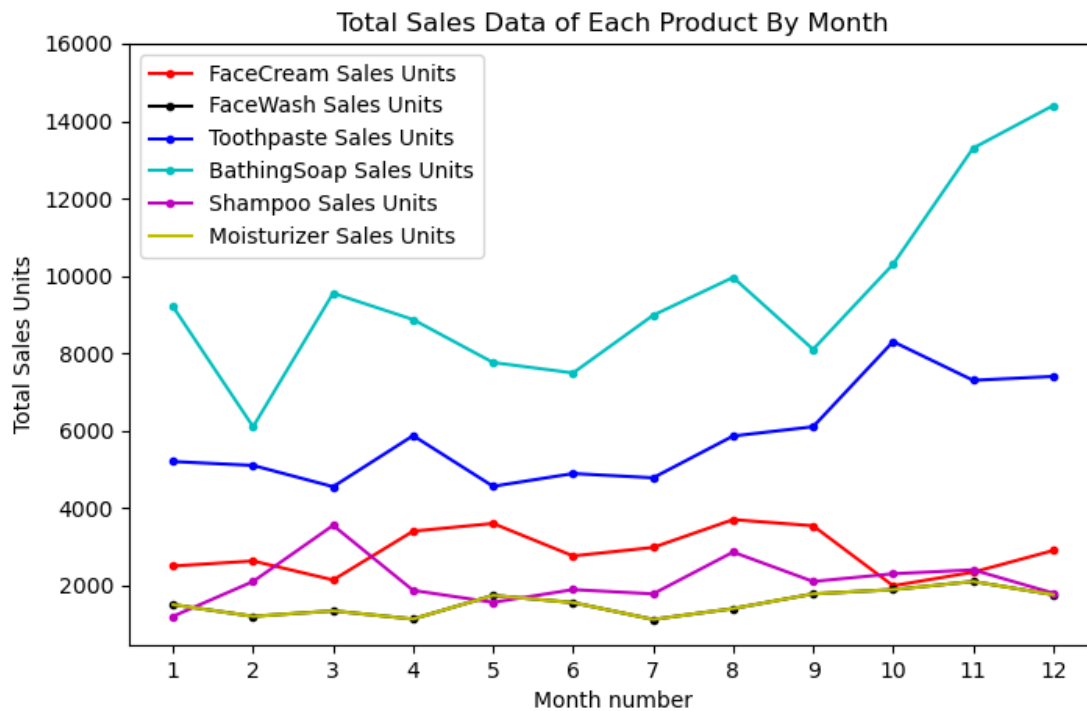
plt.figure(figsize=(8,5))
plt.plot(df.month_number,df.total_profit,'g.-',label="Total Profit")
plt.title("Total profit By months")
plt.xlabel("Month number")
plt.ylabel("Total profit")
plt.yticks([100000,200000,300000,400000,500000])
plt.xticks(df.month_number)
plt.legend()
plt.show()
```



- b) Read all product sales data and show it using a multiline plot Display the number of units sold per month for each product using multiline plots. (i.e., separate Plotline for each product).

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('F:\work of pro\SEM5\pythonS\Practical_Set_9\data.csv')

plt.figure(figsize=(8,5))
plt.plot(df.month_number,df.facecream,'r.-',label="FaceCream Sales Units")
plt.plot(df.month_number,df.facewash,'k.-',label="FaceWash Sales Units")
plt.plot(df.month_number,df.toothpaste,'b.-',label="Toothpaste Sales Units")
plt.plot(df.month_number,df.bathingssoap,'c.-',label="BathingSoap Sales Units")
plt.plot(df.month_number,df.shampoo,'m.-',label="Shampoo Sales Units")
plt.plot(df.month_number,df.moisturizer,'y',label="Moisturizer Sales Units")
plt.title("Total Sales Data of Each Product By Month")
plt.xlabel("Month number")
plt.ylabel("Total Sales Units")
plt.yticks([2000,4000,6000,8000,10000,12000,14000,16000])
plt.xticks(df.month_number)
plt.legend()
plt.show()
```



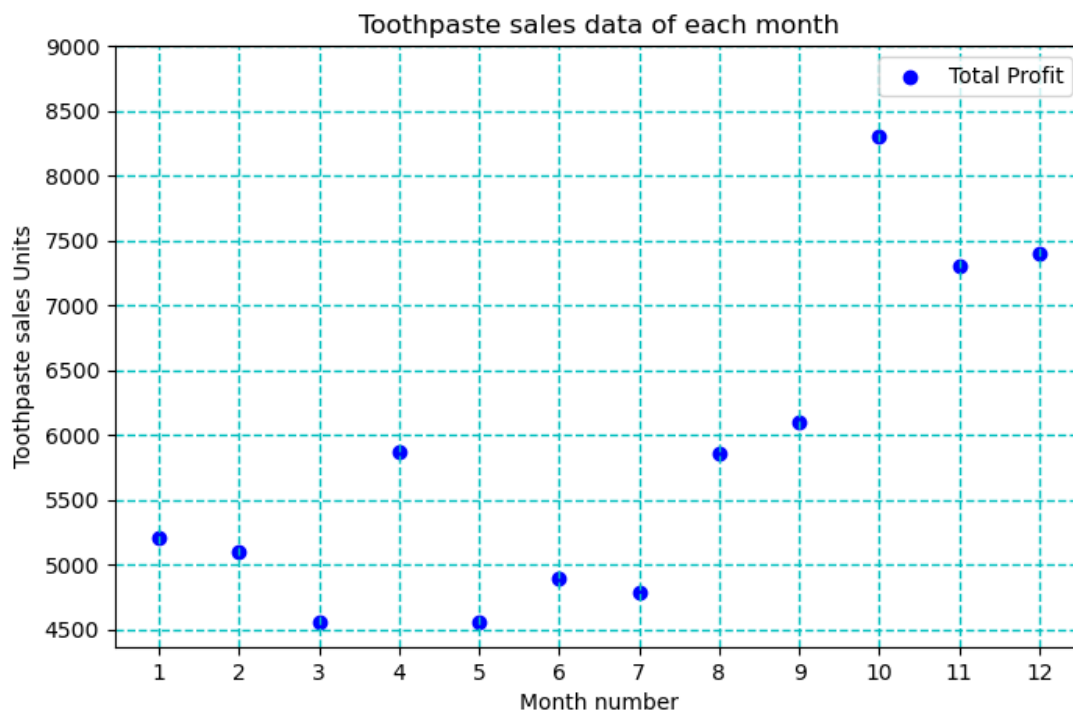
## 2. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data.

| month_n<br>umber | facecrea<br>m | facewa<br>sh | toothp<br>aste | bathings<br>oap | shampo<br>o | moisturiz<br>er | total_unit<br>s | total_pro<br>fit |
|------------------|---------------|--------------|----------------|-----------------|-------------|-----------------|-----------------|------------------|
| 1                | 2500          | 1500         | 5200           | 9200            | 1200        | 1500            | 21100           | 211000           |
| 2                | 2630          | 1200         | 5100           | 6100            | 2100        | 1200            | 18330           | 183300           |
| 3                | 2140          | 1340         | 4550           | 9550            | 3550        | 1340            | 22470           | 224700           |
| 4                | 3400          | 1130         | 5870           | 8870            | 1870        | 1130            | 22270           | 222700           |
| 5                | 3600          | 1740         | 4560           | 7760            | 1560        | 1740            | 20960           | 209600           |
| 6                | 2760          | 1555         | 4890           | 7490            | 1890        | 1555            | 20140           | 201400           |
| 7                | 2980          | 1120         | 4780           | 8980            | 1780        | 1120            | 29550           | 295500           |
| 8                | 3700          | 1400         | 5860           | 9960            | 2860        | 1400            | 36140           | 361400           |
| 9                | 3540          | 1780         | 6100           | 8100            | 2100        | 1780            | 23400           | 234000           |
| 10               | 1990          | 1890         | 8300           | 10300           | 2300        | 1890            | 26670           | 266700           |
| 11               | 2340          | 2100         | 7300           | 13300           | 2400        | 2100            | 41280           | 412800           |
| 12               | 2900          | 1760         | 7400           | 14400           | 1800        | 1760            | 30020           | 300200           |

- a. Read toothpaste sales data of each month and show it using a scatter plot.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('F:\work of pro\SEM5\pythonS\Practical_Set_9\data.csv')

plt.figure(figsize=(8,5))
plt.scatter(df.month_number,df.toothpaste,label="Total Profit",color='b')
plt.title("Toothpaste sales data of each month")
plt.xlabel("Month number")
plt.ylabel("Toothpaste sales Units")
plt.yticks([4500,5000,5500,6000,6500,7000,7500,8000,8500,9000])
plt.xticks(df.month_number)
plt.grid(color='c',linestyle='--',linewidth=1)
plt.legend()
plt.show()
```

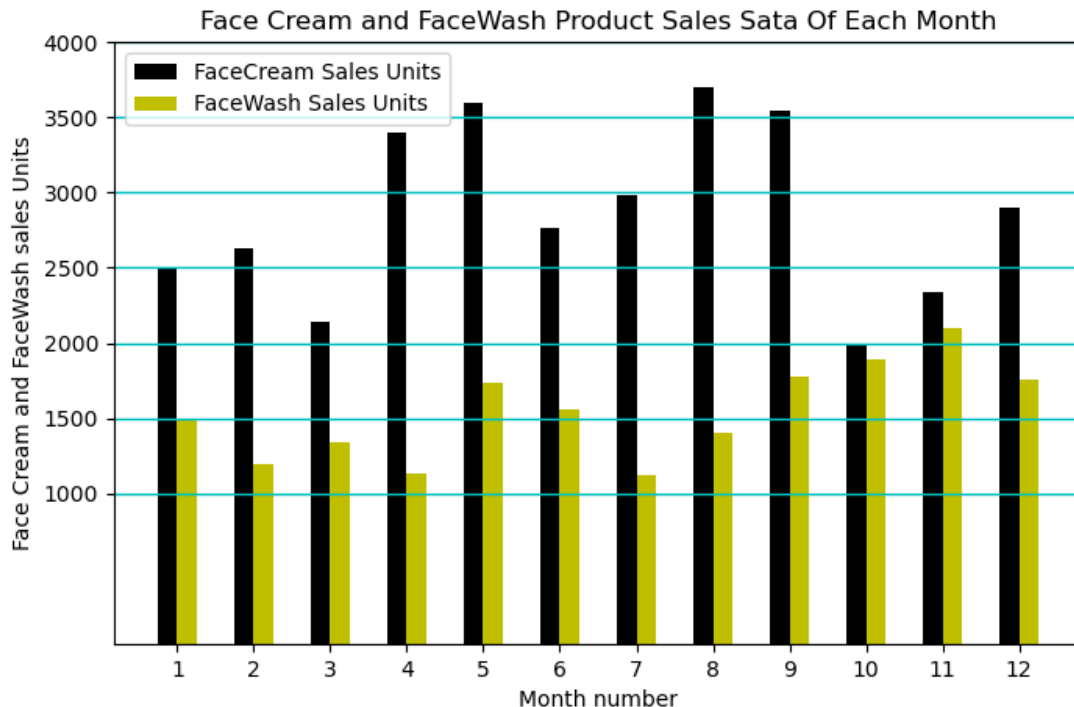




- b. Read face cream and facewash product sales data and show it using the bar chart.

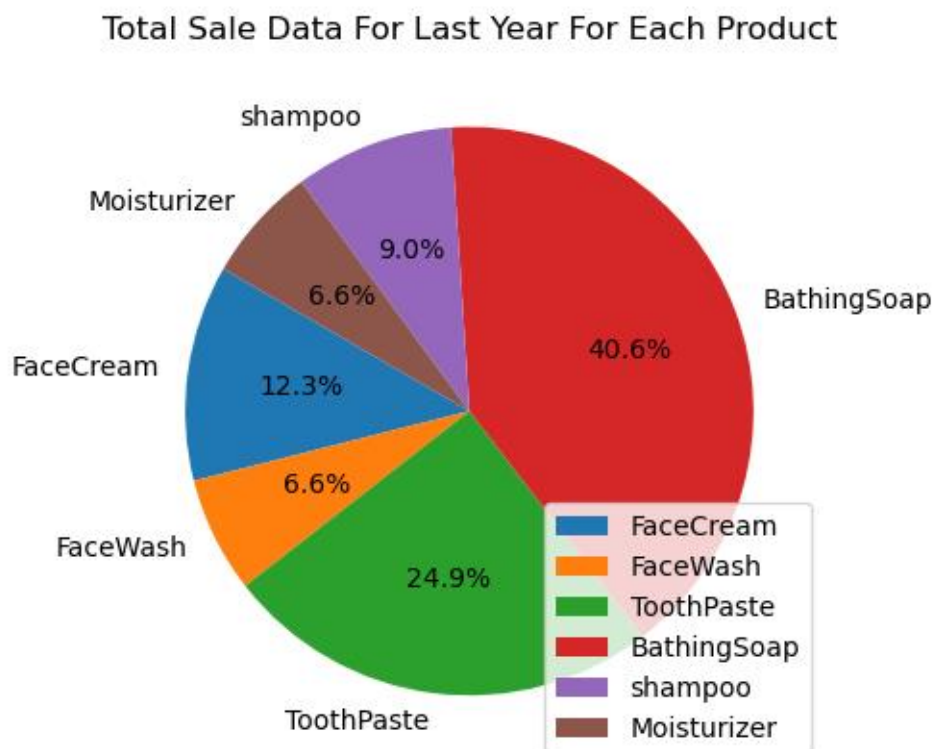
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('F:\work of pro\SEM5\pythonS\Practical_Set_9\data.csv')

plt.figure(figsize=(8,5))
plt.bar([i - 0.25 for i in
df.month_number],df.facecream,width=0.25,label="FaceCream Sales
Units",color='k',align='edge')
plt.bar([i for i in df.month_number],df.facewash,width=0.25,label="FaceWash Sales
Units",color='y',align='edge')
plt.title("Face Cream and FaceWash Product Sales Sata Of Each Month")
plt.xlabel("Month number")
plt.ylabel("Face Cream and FaceWash sales Units")
plt.yticks([1000,1500,2000,2500,3000,3500,4000])
plt.xticks(df.month_number)
plt.grid(axis='y',color='c',linestyle='-',linewidth=1)
plt.legend()
plt.show()
```



- c. Calculate total sale data for last year for each product and show it using a Pie chart Note: In Pie chart display Number of units sold per year for each product in percentage.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('F:\work of pro\SEM5\pythonS\Practical_Set_9\data.csv')
slices =
[df.facecream.sum(),df.facewash.sum(),df.toothpaste.sum(),df.bathingsoap.sum(),df.s
hampoo.sum(),df.moisturizer.sum()]
label =
["FaceCream", "FaceWash", "ToothPaste", "BathingSoap", "shampoo", "Moisturizer"]
plt.pie(slices,
        labels=label,
        startangle=150,
        shadow=False,
        explode=(0,0,0,0,0,0),
        autopct='%1.1f%%')
plt.title("Total Sale Data For Last Year For Each Product")
plt.legend(loc='lower right')
plt.show()
```



## Practical – 10

Create a python program using Scikit-learn to implement the following model on dataset:

### Logistic Regression:

If the desired output consists of one or more continuous variables, then the task is called *regression*. Plot decision surface of multinomial and One-vs-Rest Logistic Regression. The hyperplanes corresponding to the three One-vs-Rest (OVR) classifiers are represented by the dashed lines.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression

# make 3-class dataset for classification
centers = [[-5, 0], [0, 1.5], [5, -1]]
X, y = make_blobs(n_samples=1000, centers=centers, random_state=40)
transformation = [[0.4, 0.2], [-0.4, 1.2]]
X = np.dot(X, transformation)

for multi_class in ('multinomial', 'ovr'):
    clf = LogisticRegression(solver='sag', max_iter=100, random_state=42,
                            multi_class=multi_class).fit(X, y)

    # print the training scores
    print("training score : %.3f (%s)" % (clf.score(X, y), multi_class))

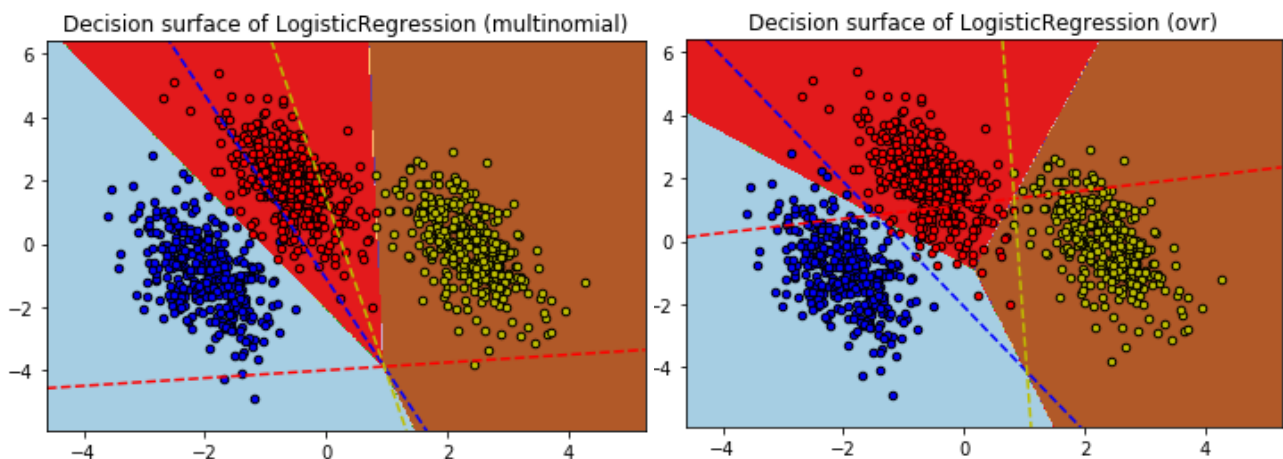
    # create a mesh to plot in
    h = .02 # step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                        np.arange(y_min, y_max, h))

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max][y_min, y_max].
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)
    plt.title("Decision surface of LogisticRegression (%s)" % multi_class)
    plt.axis('tight')
    # Plot also the training points
    colors = "bry"
```

```
for i, color in zip(clf.classes_, colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, cmap=plt.cm.Paired,
                edgecolor='black', s=20)
# Plot the three one-against-all classifiers
xmin, xmax = plt.xlim()
ymin, ymax = plt.ylim()
coef = clf.coef_
intercept = clf.intercept_
def plot_hyperplane(c, color):
    def line(x0):
        return -(x0 * coef[c, 0]) - intercept[c] / coef[c, 1]
    plt.plot([xmin, xmax], [line(xmin), line(xmax)],
            ls="--", color=color)
for i, color in zip(clf.classes_, colors):
    plot_hyperplane(i, color)

plt.show()
```

```
training score : 0.995 (multinomial)
training score : 0.976 (ovr)
```



## Support Vector Machines:

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

Plot decision function of a weighted dataset, where the size of points is proportional to its weight. The sample weighting rescales the C parameter, which means that the classifier puts more emphasis on getting these points right. The effect might often be subtle. To emphasize the effect here, we particularly weight outliers, making the deformation of the decision boundary very visible.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

def plot_decision_function(classifier, sample_weight, axis, title):
    # plot the decision function
    xx, yy = np.meshgrid(np.linspace(-4, 5, 500), np.linspace(-4, 5, 500))

    Z = classifier.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # plot the line, the points, and the nearest vectors to the plane
    axis.contourf(xx, yy, Z, alpha=0.75, cmap=plt.cm.bone)
    axis.scatter(X[:, 0], X[:, 1], c=y, s=100 * sample_weight, alpha=0.9,
                cmap=plt.cm.bone, edgecolors='black')

    axis.axis('off')
    axis.set_title(title)

# we create 20 points
np.random.seed(0)
X = np.r_[np.random.randn(10, 2) + [1, 1], np.random.randn(10, 2)]
y = [1] * 10 + [-1] * 10
sample_weight_last_ten = abs(np.random.randn(len(X)))
sample_weight_constant = np.ones(len(X))
# and bigger weights to some outliers
sample_weight_last_ten[15:] *= 5
sample_weight_last_ten[9] *= 15

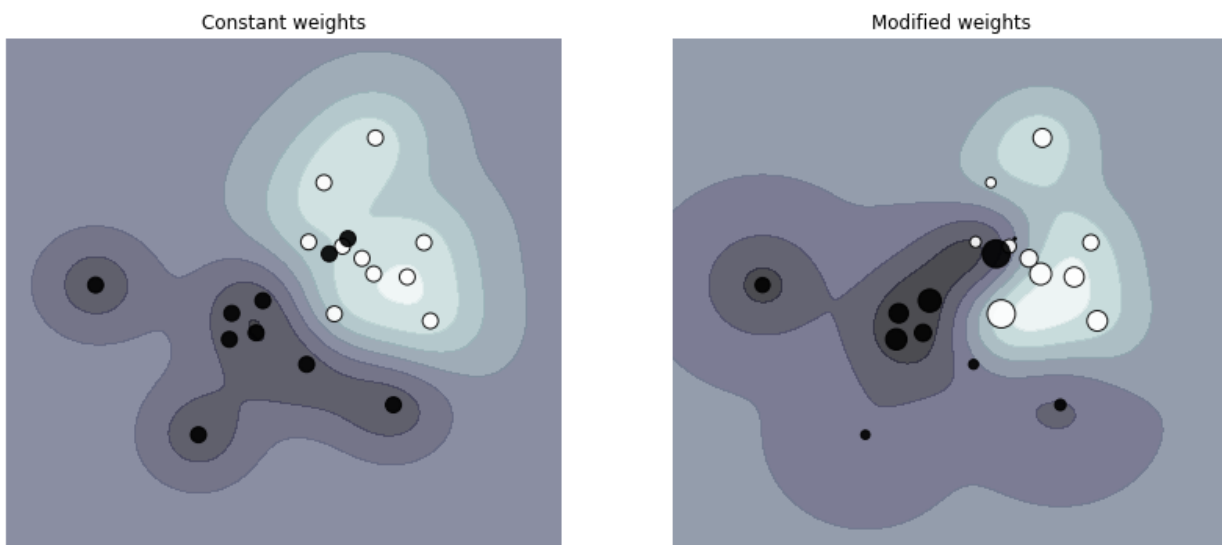
# for reference, first fit without sample weights

# fit the model
clf_weights = svm.SVC(gamma=1)
clf_weights.fit(X, y, sample_weight=sample_weight_last_ten)

clf_no_weights = svm.SVC(gamma=1)
clf_no_weights.fit(X, y)
VGEC_5th_CE_G2
```

```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
plot_decision_function(clf_no_weights, sample_weight_constant, axes[0],
                      "Constant weights")
plot_decision_function(clf_weights, sample_weight_last_ten, axes[1],
                      "Modified weights")

plt.show()
```



## Naive Bayes:

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Implementation of Gaussian Naive Bayes classifier using scikit-learn.

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
VGEC_5th_CE_G2
```

## Python for Data Science (3150713)

### 180170107030

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```

## Random Forest:

The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

IRIS flower datasets to train and test the model. We will build a model to classify the type of flower.

```
# importing required libraries
# importing Scikit-learn library and datasets package
from sklearn import datasets

# Loading the iris plants dataset (classification)
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)
# dividing the datasets into two parts i.e. training datasets and test datasets
X, y = datasets.load_iris( return_X_y = True)

# Splitting arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
# i.e. 80 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.70)
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
# creating dataframe of IRIS dataset
data = pd.DataFrame({'sepalength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1], 'petallength': iris.data[:, 2], 'petalwidth': iris.data[:, 3], 'species': iris.target})
# printing the top 5 datasets in iris dataset
print(data.head())
# creating a RF classifier
clf = RandomForestClassifier(n_estimators = 100)
# Training the model on the training dataset
VGEC_5th_CE_G2
```

## Python for Data Science (3150713)

### 180170107030

```
# fit function is used to train the model using the training sets as parameters
clf.fit(X_train, y_train)
# performing predictions on the test dataset
y_pred = clf.predict(X_test)
# metrics are used to find accuracy or error
from sklearn import metrics
print()
# using metrics module for accuracy calculation
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
# predicting which type of flower it is.
clf.predict([[3, 3, 2, 2]])
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
# Create a Random forest Classifier
clf = RandomForestClassifier(n_estimators = 100)
# Train the model using the training sets
clf.fit(X_train, y_train)
# using the feature importance variable
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_, index = iris.feature_names).sort_values(ascending
= False)
feature_imp
```

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
   sepallength  sepalwidth  petallength  petalwidth  species
0           5.1          3.5           1.4          0.2         0
1           4.9          3.0           1.4          0.2         0
2           4.7          3.2           1.3          0.2         0
3           4.6          3.1           1.5          0.2         0
4           5.0          3.6           1.4          0.2         0
```

```
ACCURACY OF THE MODEL:  0.9619047619047619
```

```
petal length (cm)    0.474336
petal width (cm)     0.397979
sepal length (cm)    0.084210
sepal width (cm)     0.043475
dtype: float64
```

## AdaBoost:

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers.

We'll be using a dataset that categorizes people as attractive or not based on certain features.



## Python for Data Science (3150713)

180170107030

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

breast_cancer = load_breast_cancer()
print(breast_cancer.feature_names)
print(breast_cancer.data)
X = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
y = pd.Categorical.from_codes(breast_cancer.target, breast_cancer.target_names)
encoder = LabelEncoder()
binary_encoded_y = pd.Series(encoder.fit_transform(y))
train_X, test_X, train_y, test_y = train_test_split(X, binary_encoded_y, random_state=1)
classifier = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200)
classifier.fit(train_X, train_y)
predictions = classifier.predict(test_X)
print(predictions)
confusion_matrix(test_y, predictions)

[ 'mean radius' 'mean texture' 'mean perimeter' 'mean area'
  'mean smoothness' 'mean compactness' 'mean concavity'
  'mean concave points' 'mean symmetry' 'mean fractal dimension'
  'radius error' 'texture error' 'perimeter error' 'area error'
  'smoothness error' 'compactness error' 'concavity error'
  'concave points error' 'symmetry error' 'fractal dimension error'
  'worst radius' 'worst texture' 'worst perimeter' 'worst area'
  'worst smoothness' 'worst compactness' 'worst concavity'
  'worst concave points' 'worst symmetry' 'worst fractal dimension']
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]

array([[86,  2],
       [ 3, 52]], dtype=int64)
```