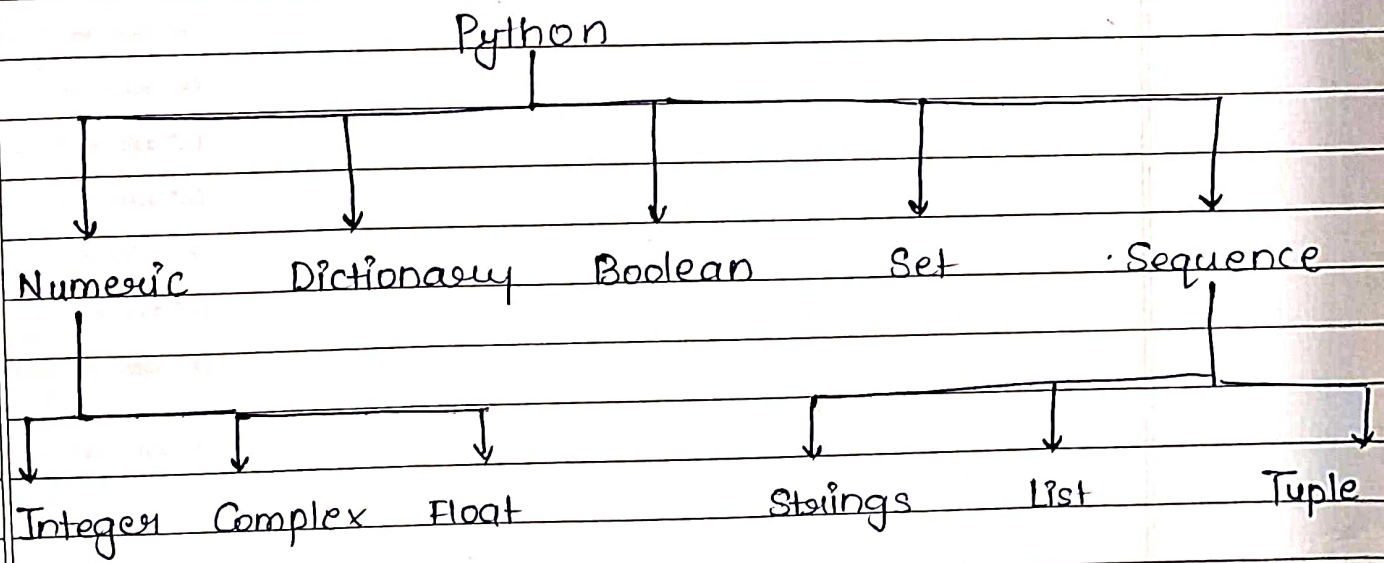
Python Experiment 1

DATE:

Name : Kareena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2023

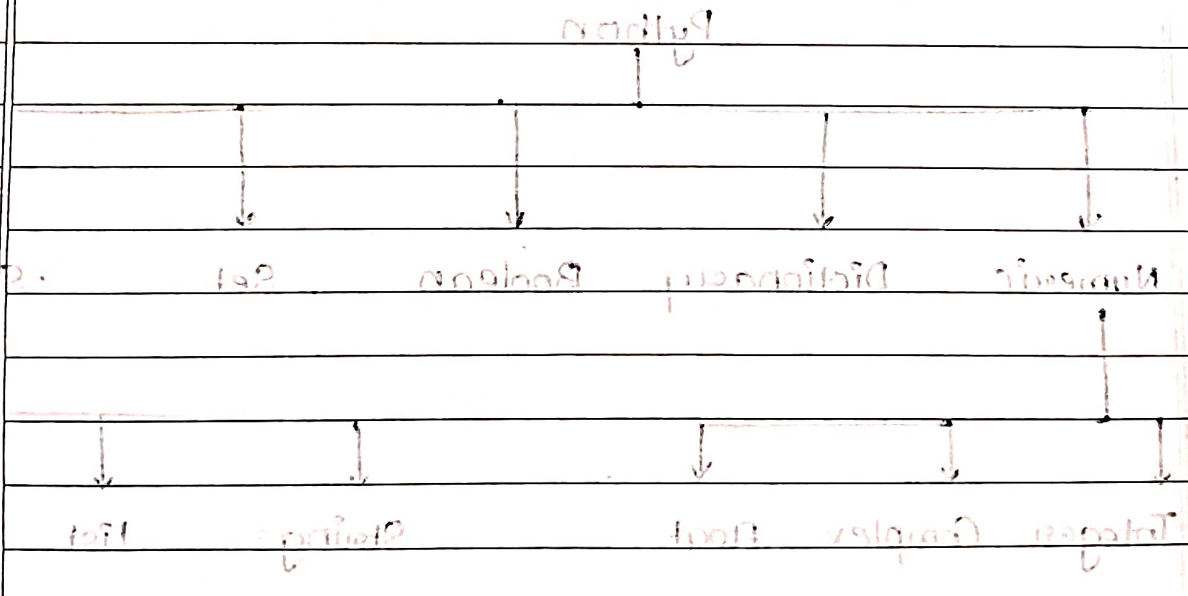
Aim : To implement & study datatypes & operators

Theory :
Datatypes are the classification or categorization of data items.
It represents the kind of value that tells what operations can be performed on a particular data
Datatypes are actually classes
Variables are instances of class



Operators
(1) Arithmetic
(2) Comparison
(3) Logical
(4) Assignment
(5) Bitwise
(6) Identity

Conclusion : Thus, we studied & implemented various datatypes & operators.



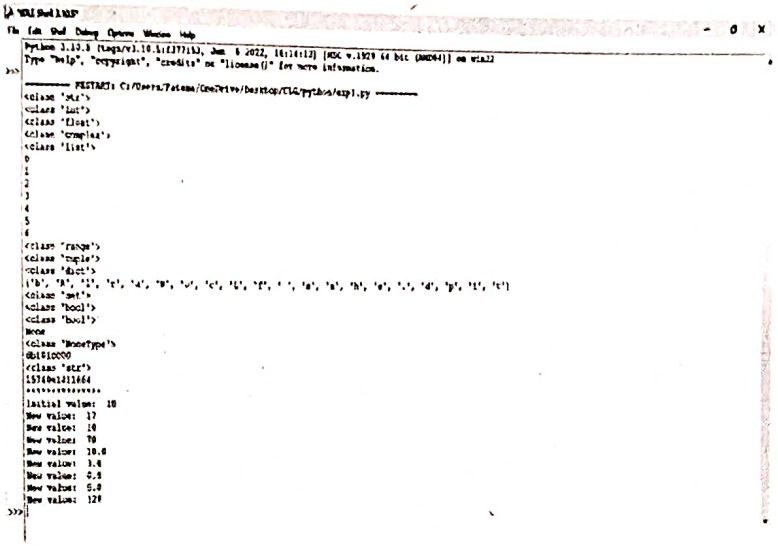
Experiment No: 1

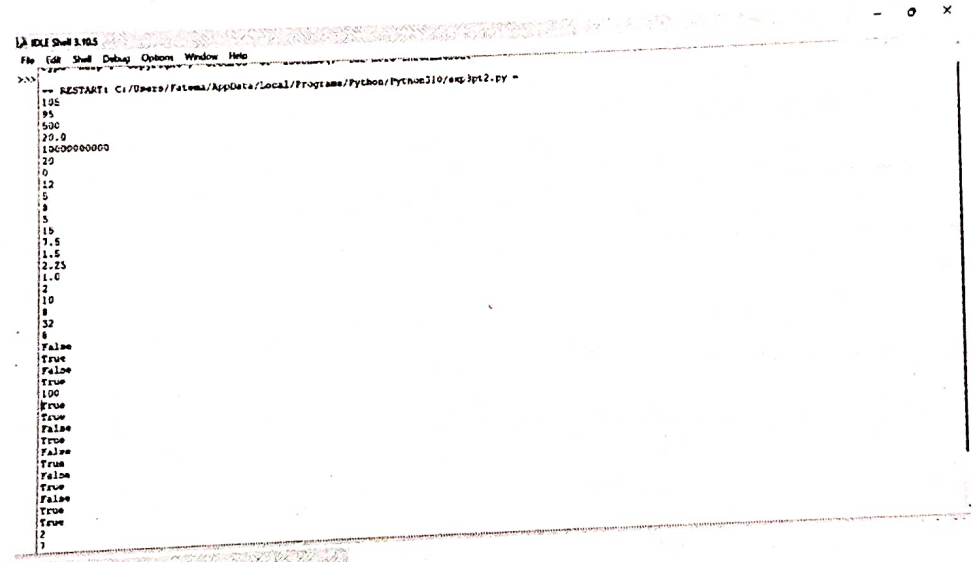
```
x = hello'
print(type(x))
p = 786
print(type(p))
f = 12.3
print(type(f))
c = 2+4j
print(type(c))
l1 = [1, 'were', 76]
print(type(l1))
a = range(7)
for i in a:
    print(i)
print(type(a))
tp = ('qwert',)
print(type(tp))
Dict = dict()
Dict = {'Name': hello', 'SAP':60004210081}
print(type(Dict))
set1 = set("the World is A beautiful pLace.")
print(set1)
print(type(set1))
print(type(True))
print(type(False))
n = None
print(n)
print(type(n))
b = bin(80)
print(b)
print(type(b))
j = 100
print(id(j))
print("**********")
x = 10
print("Initial value: ",x)
x += 7
print("New value: ",x)
x -= 7
print("New value: ",x)
x *= 7
print("New value: ",x)
x /= 7
print("New value: ",x)
x %= 7
print("New value: ",x)
x //= 7
print("New value: ",x)
x **= 7
print("New value: ",x)
x = 2
x **= 7
print("New value: ",x)
x = 100
y = 5
opt = x+y
print(opt)
opt = x-y
print(opt)
opt = x*y
print(opt)
opt = x/y
```

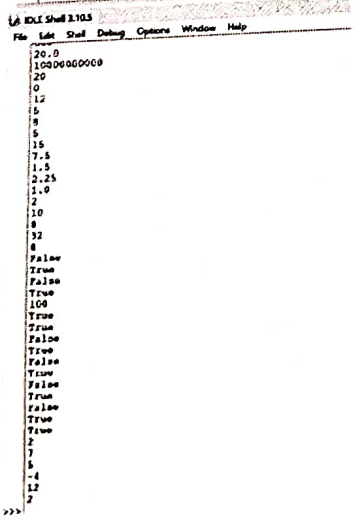
```
print(opt)
opt = x**y
print(opt)
opt = x//y
print(opt)
opt = x%y
print(opt)
t = 12
print(t)
t = y
print(t)
t+=3
print(t)
t-=3
print(t)
t*=3
print(t)
t/=2
print(t)
t%=2
print(t)
t**=2
print(t)
t//=2
print(t)
t=10
t &= 2
print(t)
t=10
t|=2
print(t)
t^=2
print(t)
t<<=2
print(t)
t>>=2
print(t)
print(t==y)
print(t!=x)
print(t>=x)
print(t<=x)
#Logical
print(x)
print(x > 50 and x < 112)
print(x > 3 or x < 4)
print(not(x > 50 and x < 112))
#Identity
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x
print(x is z)
print(x is y)
print(x == y)
print(x is not z)
print(x is not y)
print(x != y)
#Membership
print("banana" in x)
print("orange" not in x)
#Bitwise
print(6 & 3)
print(6 | 3)
```

```
print(6 ^ 3)
print(~3)
print(3 << 2)
print(8 >> 2)
```

Output:







Python Experiment 2

DATE:

Name : Kareena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2023

Aim : To study & implement different input/output & control statements, loops, conditions, etc

Theory :

Input / Output Statements

input()
print()
append()
add()
map()
split()

Loops

for loops
while loops
nested loops

Conditional statements

if

else

elif

break

continue

pass

Conclusion : Thus, we have studied & implemented different looping & conditional statements

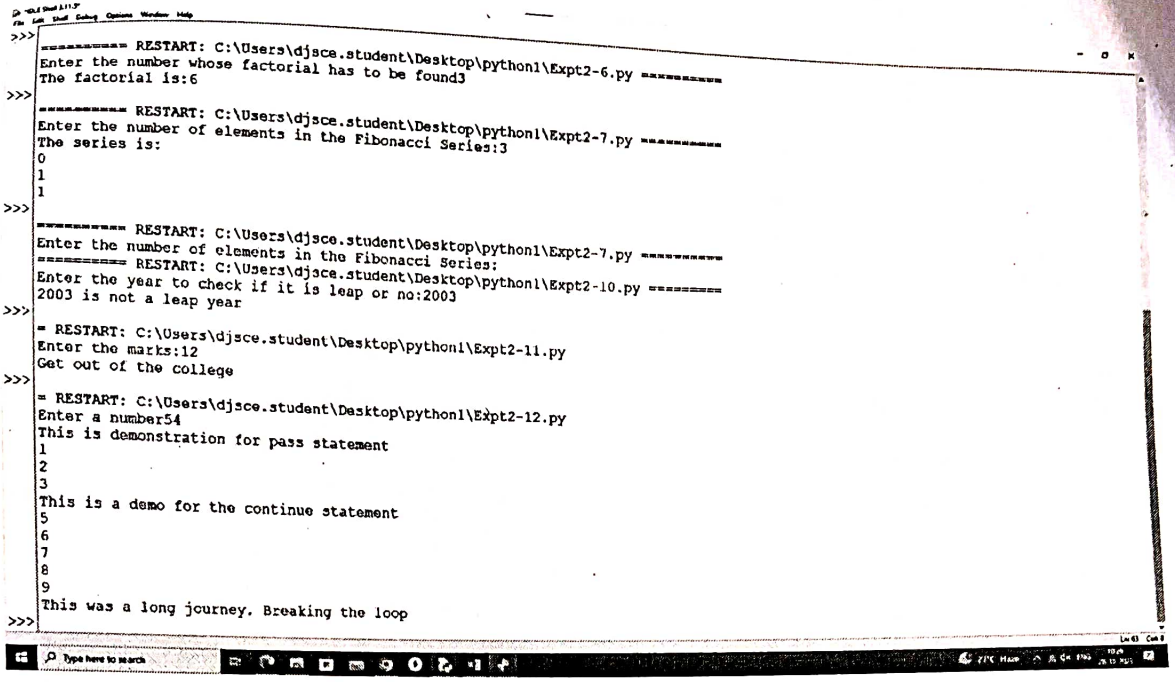
Experiment No: 2

```
import math
x=int(input("Enter the number whose sqrt has to be found:"))
print(math.sqrt(x))
x=int(input("Enter the length of the rectangle"))
y=int(input("Enter the length of the rectangle"))
perimeter=2*(x+y)
area=x*y
print(f"The perimeter is:{perimeter}")
print(f"The area is:{area}")
a=int(input("Enter the first number:"))
b=int(input("Enter the second number:"))
temp=a
a=b
b=temp
print(f"The new a after swap is:{a}")
print(f"The new a is:{b}")
#part 2
sum1=a+b
sum2=a-b
a=int((sum1-sum2)/2)
b=int((sum1+sum2)/2)
print(f"The new a after swap is:{a}")
print(f"The new a is:{b}")
x=["apple","oneplus","lenovo","hp","samsung"]
z=["mi","pastonji"]
y=("apple","oneplus","lenovo","hp","samsung")
x.append("dell")
print(x)
x=x+z
print(x)
x.pop()
print(x)
x.insert(2,"sony")
print(x)
z=("DELL",)#"sony")
a=y+z
#print(a)
#print(a.index("DELL"))
#y.add("DELL")
print(a)
a=int(input("Enter the number whose factorial has to be found"))
#i=1
fact=1
for i in range(a+1):
    if i==0:
        pass
    else:
        fact=fact*i
print(f"The factorial is:{fact}")
n=int(input("Enter the number of elements in the Fibonacci Series:"))
a=0
b=1
c=0
print("The series is:")
print(f"{a}")
print(f"{b}")
for i in range(n-2):
    c=a+b
    print(f"{c}")
    a=b
```

```
  b=c
a=int(input("Enter the number whose factorial has to be found"))
i=1
fact=1
while a!=0:
   fact=fact*a
   a=a-1
print(f"The factorial is:{fact}")
n=int(input("Enter the year to check if it is leap or no:"))
if n%400==0:
   print(f"{n} is a leap year")
elif n%100==0:
   print(f"{n} is not a leap year")
elif n%4==0:
   print(f"{n} is a leap year")
else:
   print(f"{n} is not a leap year")
n=int(input("Enter the marks:"))
if n>90 and n<=100:
   print("You have secured distinction")
elif n>70 and n<=90:
   print("Your score is respectable")
elif n>40 and n<=70:
   print("You need to improve")
else:
   print("Get out of the college")
n=int(input("Enter a number"))
for i in range(n):
   if i==0:
      print("This is demonstration for pass statement")
      pass
   elif i==4:
      print("This is a demo for the continue statement")
      continue
   elif i==10:
      print("This was a long journey. Breaking the loop")
      break
   else:
      print(i)
```

Output:

```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-1.py ==========
Enter the number whose sqrt has to be found:5
2.236067977499979
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-2.py ==========
Enter the length of the rectangle2
Enter the length of the rectangle3
The perimeter is:10
The area is:6
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-3.py ==========
Enter the first number:2
Enter the second number:33
The new a after swap is:33
The new a is:2
The new a after swap is:2
The new a is:33
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-4.py ==========
['apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'dell']
['apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'dell', 'mi', 'pastonji']
['apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'dell', 'mi']
['apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'dell', 'ml']
['apple', 'oneplus', 'sony', 'lenovo', 'hp', 'samsung', 'dell', 'ml']
('apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'DELL')
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-6.py ==========
Enter the number whose factorial has to be found3
The factorial is:6
>>> ========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-7.py ==========
Enter the number of elements in the Fibonacci Series:3
The series is:
0
```


>>>
========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-6.py ==========
Enter the number whose factorial has to be found3
The factorial is:6
>>>
========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-7.py ==========
Enter the number of elements in the Fibonacci Series:3
The series is:
0
1
1
>>>
========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-7.py ==========
Enter the number of elements in the Fibonacci Series:
========== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-10.py ==========
Enter the year to check if it is leap or no:2003
2003 is not a leap year
>>>
= RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-11.py
Enter the marks:12
Get out of the college
>>>
= RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-12.py
Enter a number54
This is demonstration for pass statement
1
2
3
This is a demo for the continue statement
5
6
7
8
9
This was a long journey. Breaking the loop
>>>
Type here to search

Python Experiment 3

DATE:

Name : Kreena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2023

Aim : To study & implement different functions in python

Theory :

A function is a block of code which only runs when it is called

Parameters can be passed into a function

Syntax :

```
def function_name :
    Instruction 1,
    :
    return
```

With parameters

```
def parameter_name (num) :
    Instruction 1
    :
    return
```

Type Conversion

Type Coersion

Mathematical Function

dir()

map()

filter()

help()

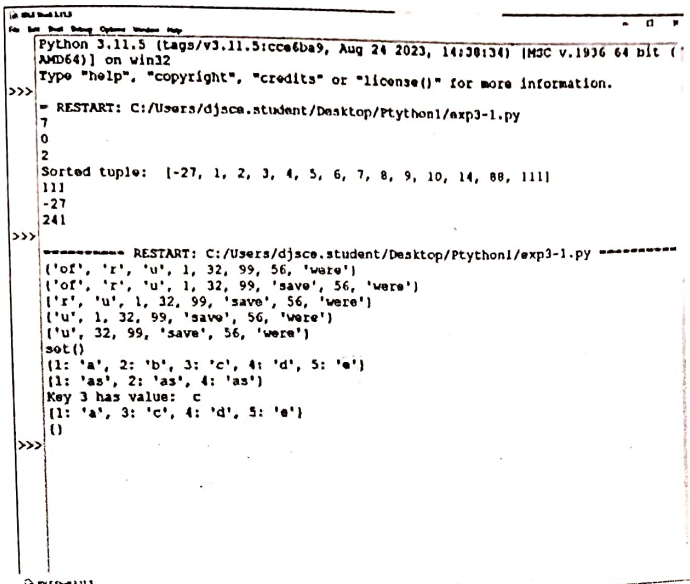
- User defined functions
- Anonymous functions
- Recursive Functions

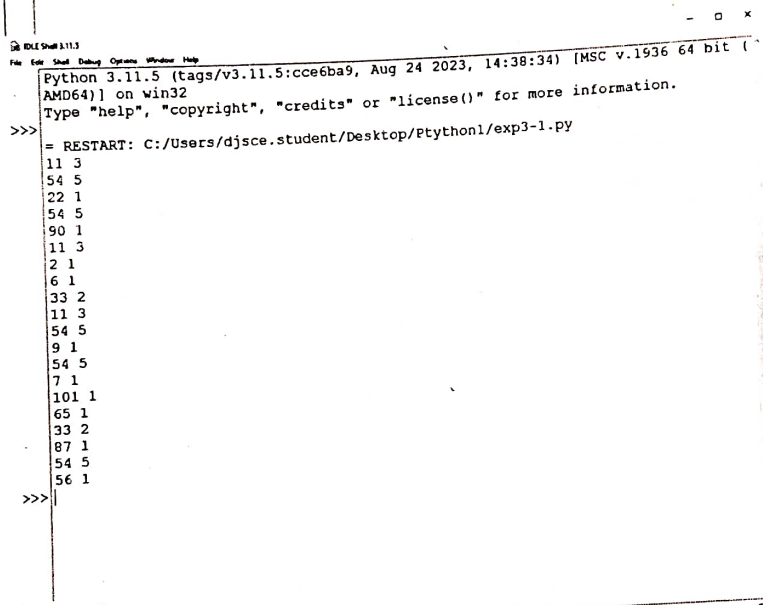
Conclusion : Thus, we implemented & successfully studied various functions.

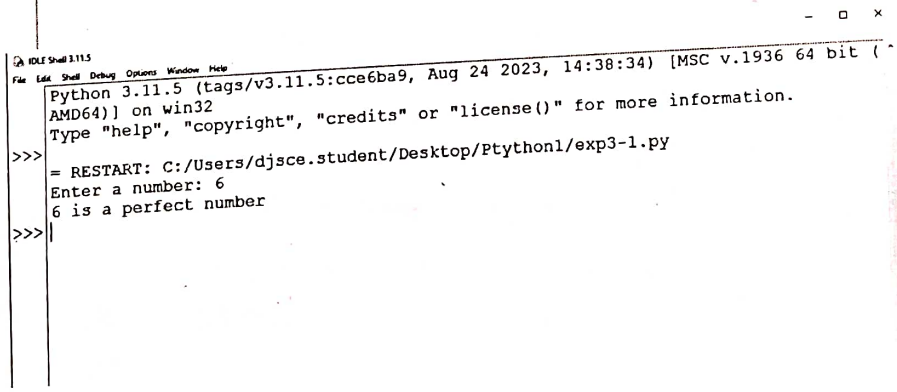
Code: **Experiment No: 3**

```
#tuple
tup = ('apple', 'banana', 'cherry', 24, 656, True, 'apple')
print(len(tup))
print(tup.index('apple'))
print(tup.count('apple'))
st = (2,5,9,3,88,14,4,1,10,111,6,8,7,-27)
aft = sorted(st)
print('Sorted tuple: ', aft)
print(max(st))
print(min(st))
print(sum(st))

#set
myset = {1, 56, 99, 32, 'were', 'of', 'r', 'u' }
print(myset)
myset.add('save')
print(myset)
myset.discard('of')
myset.discard(6)
print(myset)
myset.pop()
print(myset)
#myset.remove(2)
myset.remove(1)
print(myset)
myset.clear()
print(myset)

#dictionary
dict = {1:'a', 2:'b', 3:'c', 4:'d', 5:'e'}
dicta = dict.copy()
print(dicta)
dict3 = (1,2,4)
print(dict.fromkeys(dict3,'as'))
print('Key 3 has value: ', dict.get(3))
dict.pop(2)
print(dict)
dict.clear()
print(dict)

#function
l = [11,54,22,54,90,11,2,6,33,11,54,9,54,7,101,65,33,87,54,56]
def histogram(l):
    count = 0
    for i in l:
        for j in range(0,len(l)):
            if i==l[j]:
                count = count+1
        print(i,count)
        count = 0
histogram(l)
#perfect num
x = int(input('Enter a number: '))
fac = []
for i in range(1,x):
    if x%i==0 :
        fac.append(i)
sum = 0
for i in fac:
    sum = sum+i
if sum == x:
    print(x, 'is a perfect number')
```

Output:


Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/djsce.student/Desktop/Ptython1/exp3-1.py
7
0
2
Sorted tuple: [-27, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 88, 111]
111
-27
241
>>>
========== RESTART: C:/Users/djsce.student/Desktop/Ptython1/exp3-1.py ==========
('of', 'r', 'u', 1, 32, 99, 56, 'were')
('of', 'r', 'u', 1, 32, 99, 'save', 56, 'were')
('r', 'u', 1, 32, 99, 'save', 56, 'were')
('u', 1, 32, 99, 'save', 56, 'were')
('u', 32, 99, 'save', 56, 'were')
set()
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
{1: 'as', 2: 'as', 4: 'as'}
Key 3 has value: c
{1: 'a', 3: 'c', 4: 'd', 5: 'e'}
{}
>>>


IDLE Shell 3.11.5
File Edit Shell Debug Options Window Help
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/djsce.student/Desktop/Ptython1/exp3-1.py
11 3
54 5
22 1
54 5
90 1
11 3
2 1
6 1
33 2
11 3
54 5
9 1
54 5
7 1
101 1
65 1
33 2
87 1
54 5
56 1
>>>


IDLE Shell 3.11.5
File Edit Shell Debug Options Window Help
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/djsce.student/Desktop/Ptython1/exp3-1.py
Enter a number: 6
6 is a perfect number
>>>

Python Experiment 4

DATE:

Name : Koreena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2023

Aim : To implement objects, classes & inheritance

Theory :

Classes

A class is a user defined blueprint or prototype from which objects are created
They provide a means of binding data & functionality together.

Object

An object is an instance of a class
A class is like a blueprint while an instance is a copy of the class with actual values
An object consists of
State : properties
Behaviour : response
Identity : unique name

Inheritance

Core OOPS concept
It helps you create a hierarchy of classes that store a set of properties & methods by designing a class

from another class

Types of inheritance

(1) Single

(2) Multiple

(3) Multilevel

(4) Hierarchical

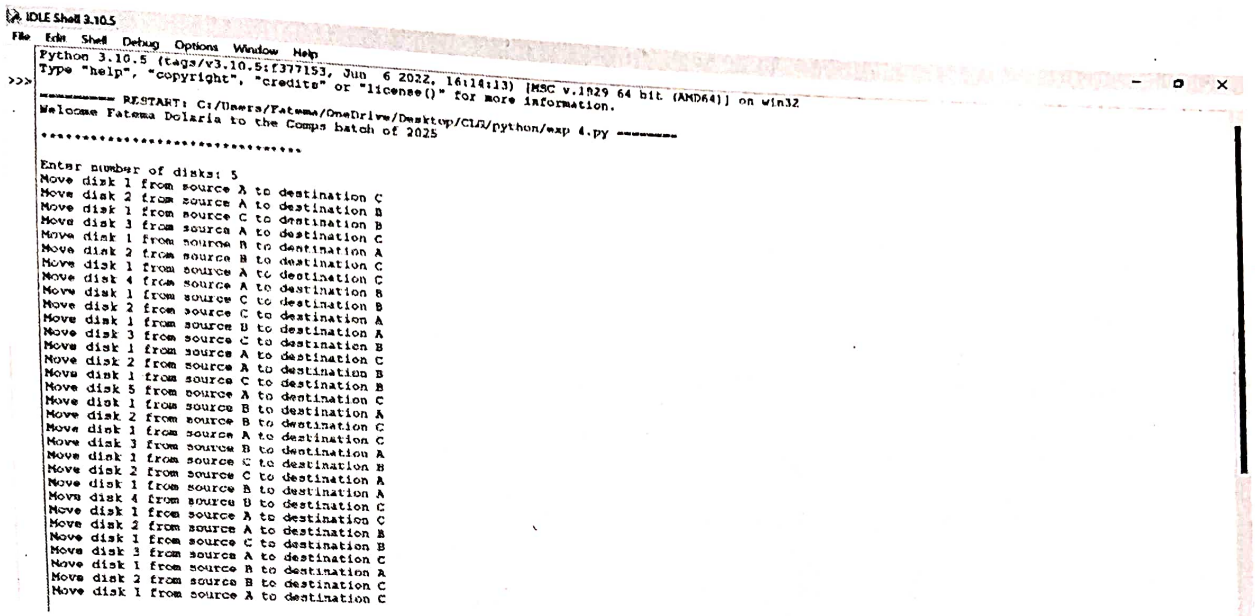
Conclusion : Thus, we have studied objects, classes & inheritance in python

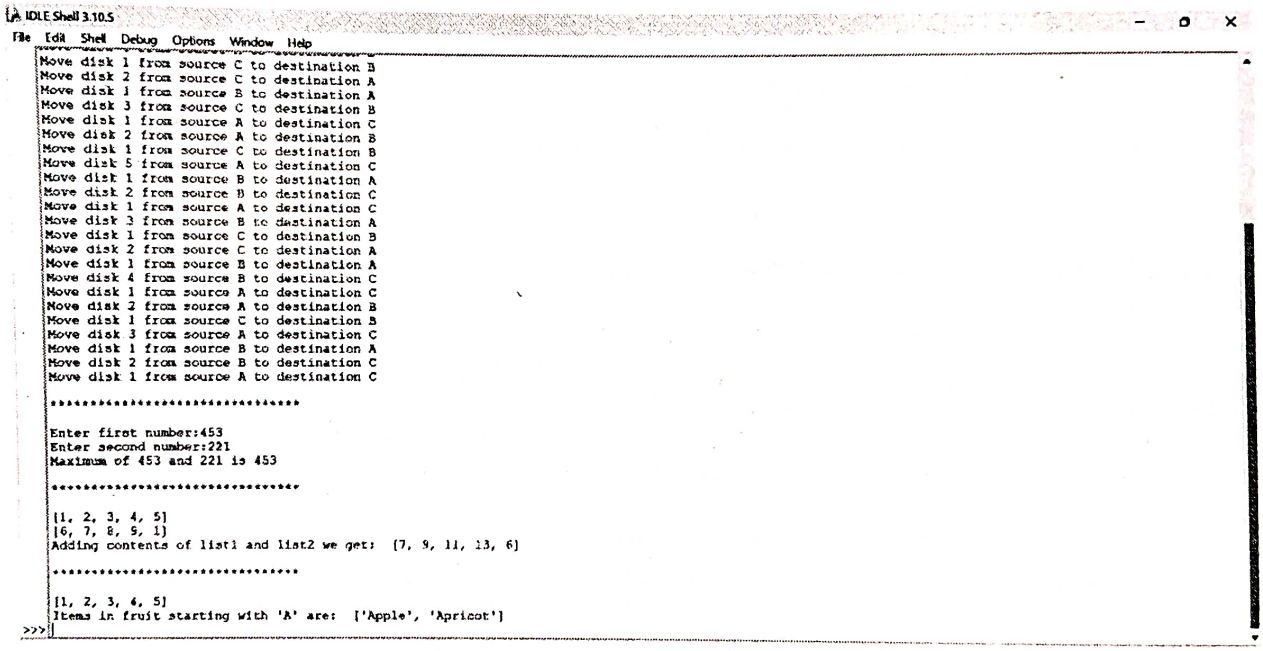
Experiment No: 4

Code:

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname
  def printname(self):
    print(self.firstname, self.lastname)
class Student(Person):
  def __init__(self, fname, lname, year):
    Person.__init__(self, fname, lname)
    self.graduationyear = year
  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the Comps batch of", self.graduationyear)
# class Student(Person):
#  def __init__(self, fname, lname):
#    super().__init__(fname, lname)
x = Student(hello', 'Dolaria', 2025)
x.welcome()
print("\n**********\n")
#tower of hanoi
def TowerOfHanoi(n , source, destination, auxiliary):
        if n==1:
                print ("Move disk 1 from source",source,"to destination",destination)
                return
        TowerOfHanoi(n-1, source, auxiliary, destination)
        print ("Move disk",n,"from source",source,"to destination",destination)
        TowerOfHanoi(n-1, auxiliary, destination, source)
n = int(input('Enter number of disks: '))
TowerOfHanoi(n,'A','C','B')
print("\n**********\n")
#lambda
a = int(input('Enter first number:'))
b = int(input('Enter second number:'))
x = lambda a,b: max(a,b)
print('Maximum of',a,'and',b,'is',x(a,b))
print("\n**********\n")
#map & filter
list1 = [1,2,3,4,5]
list2 = [6,7,8,9,1]
print(list1)
print(list2)
map_obj = map(lambda x,y: x+y, list1, list2)
print("Adding contents of list1 and list2 we get: ",list(map_obj))
print("\n**********\n")
fruit = ["Apple", "Banana", "Pear", "Apricot", "Orange"]
filter_object = filter(lambda s: s[0] == "A", fruit)
print(list1)
print("Items in fruit starting with 'A' are: ", list(filter_object))
```

Output:


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ========== RESTART: C:/Users/Fatema/OneDrive/Desktop/CLG/python/exp 4.py ==========
Welcome Fatema Dolaria to the Comps batch of 2025
****
Enter number of disks: 5
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Move disk 4 from source A to destination B
Move disk 1 from source C to destination B
Move disk 2 from source C to destination A
Move disk 1 from source B to destination A
Move disk 3 from source C to destination B
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 5 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Move disk 3 from source B to destination A
Move disk 1 from source C to destination B
Move disk 2 from source C to destination A
Move disk 1 from source B to destination A
Move disk 4 from source B to destination C
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Move disk 1 from source C to destination B
Move disk 2 from source C to destination A
Move disk 1 from source B to destination A
Move disk 3 from source C to destination B
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 5 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
Move disk 3 from source B to destination A
Move disk 1 from source C to destination B
Move disk 2 from source C to destination A
Move disk 1 from source B to destination A
Move disk 4 from source B to destination C
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B
Move disk 3 from source A to destination C
Move disk 1 from source B to destination A
Move disk 2 from source B to destination C
Move disk 1 from source A to destination C
****
Enter first number:453
Enter second number:221
Maximum of 453 and 221 is 453
****
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 1]
Adding contents of list1 and list2 we get: [7, 9, 11, 13, 6]
****
[1, 2, 3, 4, 5]
Items in fruit starting with 'A' are: ['Apple', 'Apricot']
>>>

Python Experiment 5

DATE: 4

Name : Kreena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2021

Aim : To study & implement exception Handling in python

Theory :-
When an error/exception occurs, python normally stops execution & generates an error message.
These exceptions can be handled using the try statement

Try & except block
It lets you to block a code for reason
The except block lets you handle the error on try block & can have multiple except blocks

Else block
It executes the code where there is no error.

Finally
It gets executed regardless of try & except blocks

Raising an exception
As a developer, one can choose to throw an exception

DATE:

Some important built-in exception

Index

Import

Key

Runtime

Syntax

Value

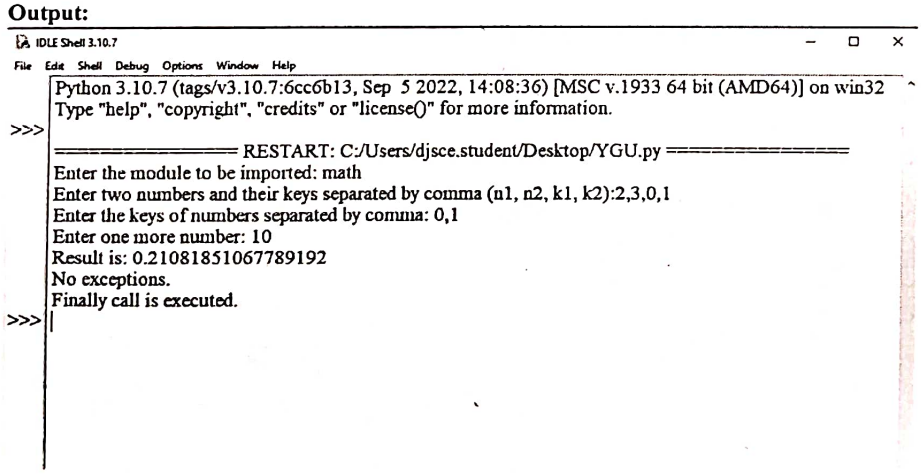
Name

Conclusion : Thus, we have implemented exception handling successfully.

Experiment No: 5

```
import math
try:
   imp = "import "+ input("Enter the module to be imported: ")
   exec(imp)
   n = {}
   n1, n2, k1, k2 = eval(input("Enter two numbers and their keys separated by comma (n1, n2, k1, k2):"))
   n[k1] = n1
   n[k2] = n2
   k1, k2 = eval(input("Enter the keys of numbers separated by comma: "))
   result = n[k1] / n[k2]
   m = int(input("Enter one more number: "))
   print("Result is:", result / math.sqrt(m))
   if (n[k1] == 0):
      raise RuntimeError

except ValueError:
   print("Invalid literal.")
except ImportError:
   print("Module not found.")
except ZeroDivisionError:
   print("Division by zero.")
except SyntaxError:
   print("Comma missing.")
except RuntimeError:
   print("May be meaningless.")
except KeyboardInterrupt:
   print()
   print("Program was interrupted.")
except KeyError:
   print("The requested key wasn't found.")
except:
   print("Something wrong in input.")
else:
   print("No exceptions.")
finally:
   print("Finally call is executed.")
```

Output:



Python Experiment 6

DATE:

Name : Kreena Shah
Sapid : 60004210243
Div : C
Batch : C'32
Subject : Python
Date of Submission : 04/12/2023

Aim : To study file handling

Theory :
Python supports file handling & allows user to handle files in read & write
The concept of file handling has stretched over various languages but the implementation is complicated

File open() method
Key function for working with files
It takes 2 parameters (filename & mode)
'r' - Read
'a' - append
'w' - write
'x' - Update

close - to close file

os.remove - to delete file

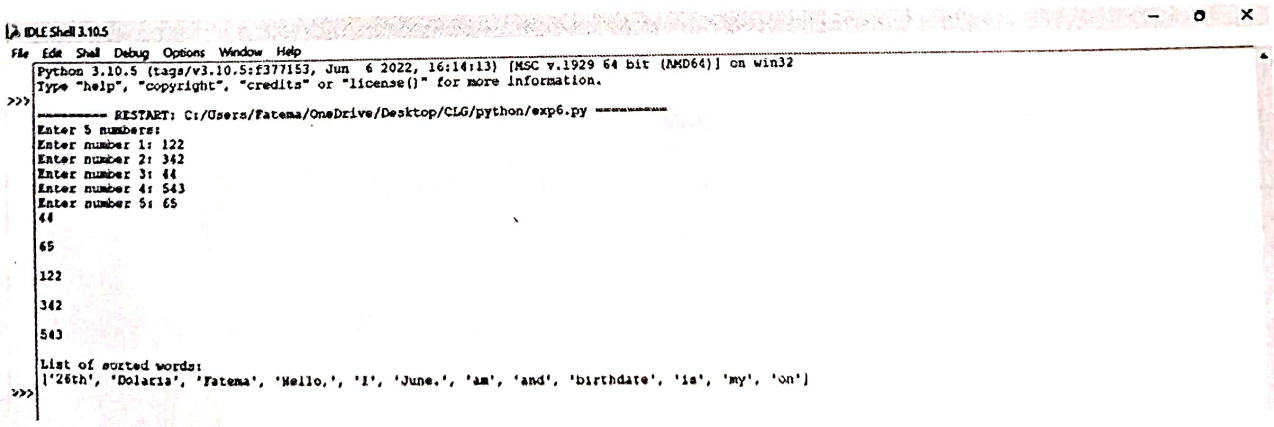
os.rmdir - to delete folder

Conclusion : Thus, we implemented file handling.

Code **Experiment No: 6**

```
print("Enter 5 numbers:")
list1=list()
fileWrite=open("file1.txt","w")
for i in range(5):
    x=int(input(f"Enter number {i+1}: "))
    list1.append(x)
    fileWrite.write(str(list1[i]))
    fileWrite.write("\n")
fileWrite.close()
fileRead=open("file1.txt","r")
list2=list()
for j in fileRead:
    list2.append(int(j))
list2.sort()
fileWrite2=open("file2.txt","w")
for k in list2:
    fileWrite2.write(str(k))
    fileWrite2.write("\n")
fileWrite2.close()
fileRead2=open("file2.txt","r")
for x in fileRead2:
    print(x)
filewrite=open("file1.txt","w")
filewrite.write("Hello, I am Fatema Dolaria and my birthdate is on 26th June.")
filewrite.close()
fileread=open("file1.txt","r")
li=list()
f=fileread.read()
f=f.split()
for line in f:
    li.append(str(line))
    li.sort(key=lambda item: (item,len(item)))
print("List of sorted words:")
print(li)
filewrite2=open("file2.txt","w")
for i in range(len(li)):
    filewrite2.write(str(li[i]))
    filewrite2.write("\n")
```

Output:



Page No.
Date

Python Experiment 7

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement regular expression's in python.

Theory :
A regular expression (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings
It can direct the presence / absence of a text by matching it with a particular pattern & also can split a pattern into subpatterns
Python provides a 're' module that supports the use of regex in python.
're' has several inbuilt functions.
It's primary functions is to offer search.

RegEx Functions

findall()
Returns a list containing all matches

search()
Returns a match object, if there is a match

split()
Returns a list where a string has been split at each match

Page No.
Date

Python Experiment 7

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement regular expressions in python.

Theory :

A regular expression (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings

It can detect the presence / absence of a text by matching it with a particular pattern & also can split a pattern into subpatterns

Python provides a 're' module that supports the use of regex in python.

're' has several inbuilt functions.

It's primary functions is to offer search.

RegEx Functions

A+

Ath 8/12/23

findall()

Returns a list containing all matches

search()

Returns a match object, if there is a match

split()

Returns a list where a string has been split at each match

Date

sub()
Replaces one or many matches within a string

Meta-Characters
Characters with special meaning

[] - Set of characters
\ - Signals a special sequence
. - Any character
^ - Starts with
$ - Ends with
* - Zero or more occurences
+ - One or more occurences
? - Zero or one occurence
{} - Exactly the specified no. of occurence
| - Either or
( ) - Capture & group

Special Sequences
Followed by one of the characters

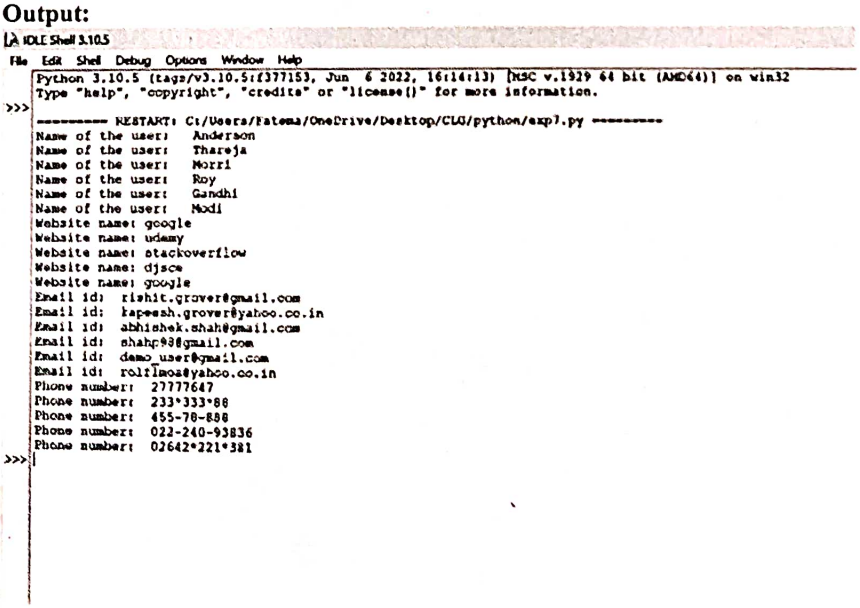
Sets
Set of characters inside a pair of square brackets

Conclusion : Thus, we have studied & implemented regular expressions in python.

Code: **Experiment No: 7**

```
import re

list = ["Mr. Anderson", "Ms. Thareja","Mrs. Morri","Mr. Roy","Ms. Gandhi","Mrs.
Modi","https://www.google.co","http://www.udemy.co",
      "www.udacity.com", "https://www.stackoverflow.com", "http://www.djsce.ac.in", "https://plus.google.com",
"rishit.grover@gmail.com",
      "kapeesh.grover@yahoo.co.in", "abhishek.shah@gmail.com", "shahp98@gmail.com", "demo_user@gmail.com",
"rolflmoa@yahoo.co.in", "27777647", "233*333*88", "455-78-888",
      "022-240-93836", "02642*221*381"]

for val in list:

   if re.search("\AM[rs].", val ):

      x = val.split(".")
      print("Name of the user: ", x[1])

   elif re.search("\Ahttp.", val ):

      x = val.split(".")
      print("Website name:", x[1])

   elif re.search("@.*[.]co.*\Z", val ):

      print("Email id: ", val)


   elif  re.search("(^([0-9]+))([-*].*)?(([0-9]+)$)", val ) :

      print("Phone number: ", val)
```

Output:



Page No.
Date

Python Experiment 8

Name : Koleena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement database connectivity in python.

Theory :

Python can be used in database application in one of the most popular databases in MySQL

Using the MySQL connectors in python, here are the following steps to connect a python application to respective databases.

(1) Import MySQL connector module
(2) Create the connector module object
(3) Create the cursor object
(4) Execute the query.

Creating Connection

To create a connection between the MySQL database & the python application

The connect() method of MySQL connection module is used.

Pass the database details like hostname, username & the database password in the method call

It returns the connection object

Creating Database

```
CREATE DATABASE db_name
```

Page No.
Date

Python Experiment 8

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement database connectivity in python.

Theory :

Python can be used in database application in one of the most popular databases in MySQL

Using the MySQL connectors in python, here are the following steps to connect a python application to respective databases

(1) Import MySQL connector module
(2) Create the connector module object
(3) Create the cursor object
(4) Execute the query.

Creating Connection

To create a connection between the MySQL database & the python application

The connect() method of MySQL connection module is used. Pass the database details like hostname, username & the database password in the method call

It returns the connection object

A+
Asth
7/11/23

Creating Database

```
CREATE DATABASE db_name
```

Page No.
Date

Python Experiment 8

Name : Kareena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement database connectivity in python.

Theory :

Python can be used in database application in one of the most popular databases in MySQL

Using the MySQL connectors in python, here are the following steps to connect a python application to respective databases.

(1) Import MySQL connector module
(2) Create the connector module object
(3) Create the cursor object
(4) Execute the query.

Creating Connection

To create a connection between the MySQL database & the python application

The connect() method of MySQL connection module is used.

Pass the database details like hostname, username & the database password in the method call

It returns the connection object

Creating Database

```
CREATE DATABASE db_name
```

Creating Table
CREATE TABLE tablename

Selecting from Table
'SELECT'

Deleting a record
'DELETE FROM' statement

Updating table
use 'UPDATE' statement

Deleting table
'DROP TABLE' statement

Deleting a database
'DROP DATABASE' statement

Conclusion : Thus, we have successfully studied & implemented database connectivity in python.

Date

Creating Table
CREATE TABLE table_name

Selecting from Table
'SELECT'

Deleting a record
'DELETE FROM' statement

Updating table
Use 'UPDATE' statement

Deleting table
'DROP TABLE' statement

Deleting a database
'DROP DATABASE' statement

Conclusion : Thus, we have successfully studied & implemented database connectivity in python.

Experiment No: 8

```
import sqlite3
c = sqlite3.connect("prac.db")
cur = c.cursor()
cur.execute("DROP TABLE IF EXISTS Student")
table = """ CREATE TABLE Student ( SAP_ID VARCHAR(255) NOT NULL, First_Name
CHAR(25) NOT NULL, Last_Name CHAR(25), Address VARCHAR(255) ); """
cur.execute(table)
cur.execute('"INSERT INTO Student VALUES ('120', 'Viraj', 'Sindhwad', 'Borivali')"')
cur.execute('"INSERT INTO Student VALUES ('117', 'Dhruvi', 'Tank', 'SoBo')"')
cur.execute('"INSERT INTO Student VALUES ('115', 'Mihir', 'Vora', 'Kandivali')"')
print("Data Inserted in the table: ")
data=cur.execute('"SELECT * FROM Student"')
for row in data:
print(row)
data2=cur.execute('"SELECT First_Name, Address FROM Student"')
for row in data2:
print(row)
data3 = cur.fetchone()
print(data3)
data4=cur.execute('"SELECT * FROM Student WHERE SAP_ID="93""')
for row in data4:
print(row)
data5=cur.execute('"SELECT * FROM Student WHERE Last_Name LIKE "G%""')
for row in data5:
print(row)
cur.execute('"DELETE FROM Student WHERE SAP_ID="85""')
data6=cur.execute('"SELECT * FROM Student"')
for row in data6:
print(row)
cur.execute('"UPDATE Student SET Address="Village" WHERE SAP_ID='116';"')
print(cur.execute('"SELECT * FROM Student WHERE SAP_ID="116""'))
cur.execute('"""ALTER TABLE Student ADD Email varchar(255);"""')
cur.execute("INSERT INTO Student VALUES ('90', 'Husain', 'Rehmanji', 'Thane','gmail.com')")
data7 = cur.execute('"SELECT * FROM Student WHERE SAP_ID="90""')
for row in data7:
print(row)
c.commit()
c.close()
```

Output:

```
Data Inserted in the table:
('120', 'Viraj', 'Sindhwad', 'Borivali')
('117', 'Dhruvi', 'Tank', 'SoBo')
('115', 'Mihir', 'Vora', 'Kandivali')
('Viraj', 'Borivali')
('Dhruvi', 'SoBo')
('Mihir', 'Kandivali')
None
('120', 'Viraj', 'Sindhwad', 'Borivali')
<sqlite3.Cursor object at 0x000002608FDC7E40>
('90', 'Husain', 'Rehmanji', 'Thane', 'gmail.com')
```

Page No.
Date

Python Experiment 9

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement socket programming in python.

Theory :

Socket Programming is a way of connecting two nodes on a network to communicate with each other.
One socket listens on a particular port on an IP, while the other socket reaches out to the other to form a connection
The server forms the listener socket while the client reaches out to the server

Socket API overview :

It provides an interface to the Berkeley socket API.
Primary socket API functions :

- socket()
- bind()
- listen()
- accept()
- connect()
- connect-ex()
- send()
- recv()
- close()

Default protol - Transmission Control Protocol (TCP)
It is reliable & has in-order data delivery which makes

Page No.
Date

Python Experiment 9

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement socket programming in python.

Theory :

Socket Programming is a way of connecting two nodes on a network to communicate with each other.
One socket listens on a particular port on an IP, while the other socket reaches out to the other to form a connection
The server forms the listener socket while the client reaches out to the server

Socket API overview :

It provides an interface to the Berkeley socket API.
Primary socket API functions :

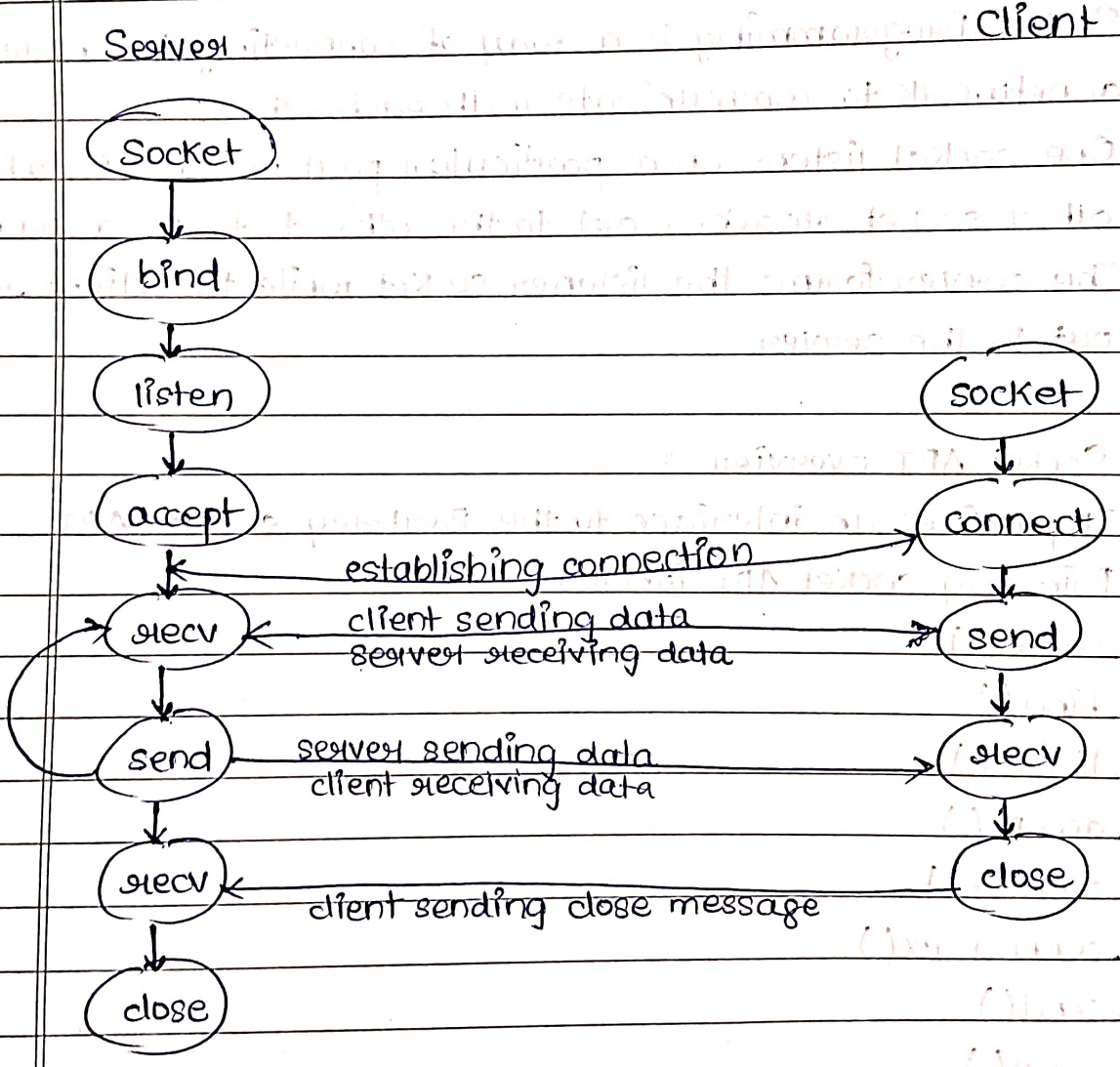
- socket()
- bind()
- listen()
- accept()
- connect()
- connect-ex()
- send()
- recv()
- close()

Default protol - Transmission Control Protocol (TCP)
It is reliable & has in-order data delivery which makes

it very useful

In contrast, UDP sockets are not reliable.

Sequence of socket API calls & data flow for TCP



Conclusion : Thus, we have studied & implemented socket programming in python.

Code: **Experiment No: 9**

```
import socket
def mpm():
    host = '127.0.0.1'
    port = 6000
    s = socket.socket()
    s.bind((host, port))
    s.listen(1)
    print("Waiting for connection...")
    c, addr = s.accept()
    print("Connection Established!")
    print("Client Address:", addr)
    while True:
        try:
            print()
            data = c.recv(1024)
            d = data.decode('ascii')
            print("Client:", d)
            print()
            x = input("Enter New Message: ")
            y = x.encode('ascii')
            c.send(y)

        except KeyboardInterrupt:
            print()
            print("Connection Terminated!")
            break

mpm()
```

```
import socket
def mpm():
    host = '127.0.0.1'
    port = 6000
    s = socket.socket()
    s.connect((host, port))
    print("Connection Established!")
    while True:
        try:
            print()
            x = input("Enter New Message: ")
            y = x.encode('ascii')
            s.send(y)
            print()
            data = s.recv(1024)
            d = data.decode('ascii')
            print('Server:', d)

        except KeyboardInterrupt:
            print()
            print("Connection Terminated!")
            s.send("Connection from Client terminated!".encode('ascii'))
            break

mpm()
```

Output:

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\djsce.student\Desktop\fatemapy>1
'1' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\djsce.student\Desktop\fatemapy>python 1.py
Waiting for connection...
Connection Established!
Client Address: ('127.0.0.1', 49170)

Client: Hi, this is Fatema n my code runs!!

Enter New Message: Cool! U go girl!

Client: Connection from Client terminated!

Enter New Message:
Connection Terminated!

C:\Users\djsce.student\Desktop\fatemapy>
```

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\djsce.student\Desktop\fatemapy>python 2.py
Connection Established!

Enter New Message: Hi, this is Fatema n my code runs!!

Server: Cool! U go girl!

Enter New Message:
Connection Terminated!

C:\Users\djsce.student\Desktop\fatemapy>
```

Page No.
Date

Python Experiment 10

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement GUI using Tkinter in python

Theory :
Python offers multiple options for developing Graphical User Interface
Out of all GUI method, tkinter is the most commonly used method.
It is the standard python interface to the TK GUI toolkit skipped with python
Python with tkinter is the fastest & easiest way to create GUI applications.

Steps to create a tkinter app :
(a) Import the module - tkinter
(b) Create the main window (container)
(c) Add any no. of widgets to the main window
(d) Apply the event trigger on the widget

There are 2 main methods which the user needs to remember while creating the python application with GUI
(1) TK (screenName = None,
baseName = None,
className = 'TK',
useTk = 1)
(2) main loop()

Page No.
Date

Python Experiment 10

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & implement GUI using Tkinter in python

Theory :
Python offers multiple options for developing Graphical User Interface
Out of all GUI method, tkinter is the most commonly used method.
It is the standard python interface to the TK GUI toolkit skipped with python
Python with tkinter is the fastest & easiest way to create GUI applications.

Steps to create a tkinter app :
(a) Import the module - tkinter
(b) Create the main window (container)
(c) Add any no. of widgets to the main window
(d) Apply the event trigger on the widget

There are 2 main methods which the user needs to remember while creating the python application with GUI.
(1) TK (screenName = None,
baseName = None,
className = 'TK',
useTk = 1)
(2) main loop()

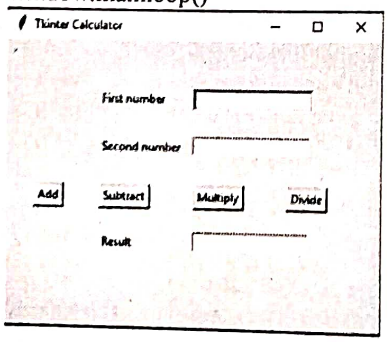
Page No.
Date

Three main geometry manager classes
pack()
grid()
place()

There are a no. of widgets which can be put in a tkinter application
Button
Canvas
CheckButton
Entry
Frame
Label
Listbox
MenuButton
Menu
RadioButton
Scale
Scrollbox

Conclusion : Thus, we have studied & implemented GUI using Tkinter in python.

```
        self.t1.place(x=200, y=50)
        self.lbl2.place(x=100, y=100)
        self.t2.place(x=200, y=100)
        self.b1=Button(win, text='Add')
        self.b1.bind('<Button-1>', self.add)
        self.b2=Button(win, text='Subtract')
        self.b2.bind('<Button-1>', self.sub)
        self.b3=Button(win, text='Multiply')
        self.b3.bind('<Button-1>', self.mul)
        self.b4=Button(win, text='Divide')
        self.b4.bind('<Button-1>', self.div)
        self.b1.place(x=30, y=150)
        self.b2.place(x=100, y=150)
        self.b3.place(x=200, y=150)
        self.b4.place(x=300, y=150)
        self.lbl3.place(x=100, y=200)
        self.t3.place(x=200, y=200)
    def add(self,event):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1+num2
        self.t3.insert(END, str(result))
    def sub(self, event):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1-num2
        self.t3.insert(END, str(result))
    def mul(self, event):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1*num2
        self.t3.insert(END, str(result))
    def div(self, event):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1/num2
        self.t3.insert(END, str(result))
window=Tk()
mywin=MyWindow(window)
window.title('Tkinter Calculator')
window.geometry("400x300+10+10")
window.mainloop()
```



Page No.
Date

Python Experiment 11

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To study & demonstrate the use of pandas & matplotlib modules in python.

Theory :-

Pandas

It is an open source library in python that is mainly made for working with relational or labelled data, both easily & intutively.
It provides various data structures & operations for manipulating numerical data & time series.
This library is built on top of the Numpy library of python.
It is fast & it has high performance & productivity for users

List of things that could be achieved via Pandas
(a) Dataset cleaning, merging & joining
(b) Easy handling of missing data
(c) Columns can be inserted & deleted from df & higher dimensional objects.
(d) Powerful group by functionality for performing split-apply-combine operations.
(e) Data Visualization

Page No.
Date

Python Experiment 11

Name : Kreena Shah
SapId : 60004210243
Batch : C'32

Aim : To study & demonstrate the use of pandas & matplotlib modules in python.

Theory :

Pandas

It is an open source library in python that is mainly made for working with relational or labelled data, both easily & intutively.
It provides various data structures & operations for manipulating numerical data & time series.
This library is built on top of the Numpy library of python.
It is fast & it has high performance & productivity for users

List of things that could be achieved via Pandas
(a) Dataset cleaning, merging & joining
(b) Easy handling of missing data
(c) Columns can be inserted & deleted from df & higher dimensional objects.
(d) Powerful group by functionality for performing split-apply-combine operations.
(e) Data Visualization

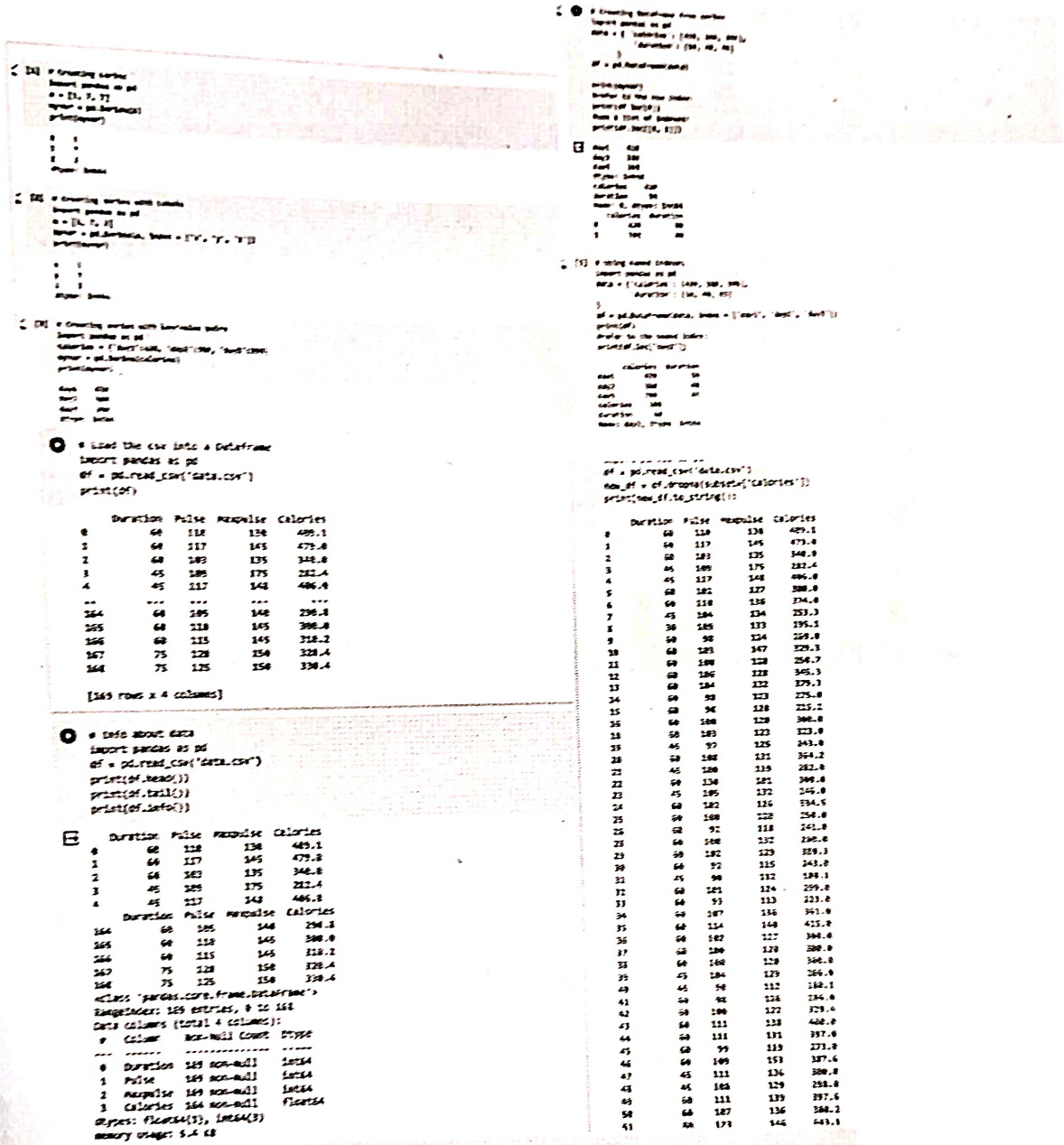
Page No.
Date

Matplotlib

It is cross platform data visualization & graphical plotting library for python & it's numerical extension - Numpy.
As such, it offers open source alternatives to MATLAB.
Developers can also use matplotlib's APIs to embed plots in GUI applications
A python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a dataplot.

Conclusion : Thus, we have studied & implemented succesfull demonstrated the use of Pandas & Matplotlib modules in python.

Experiment No: 11



```
[10] # Replacing with random value for one column
     import pandas as pd
     df = pd.read_csv('data.csv')
     df['Calories'].fillna(130, inplace = True)
     print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

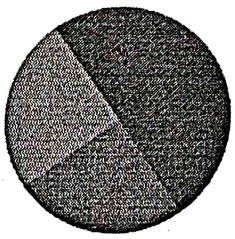
```
[11] # With Mean
     import pandas as pd
     df = pd.read_csv('data.csv')
     x = df['Calories'].mean()
     df['Calories'].fillna(x, inplace = True)
     print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

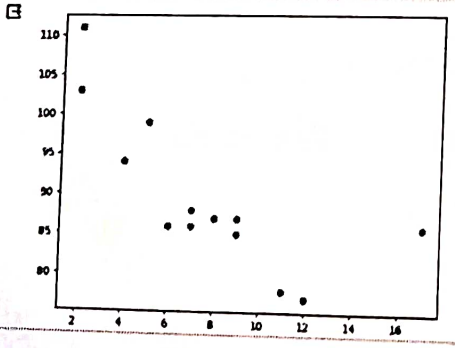
```
[12] # With Median
     import pandas as pd
     df = pd.read_csv('data.csv')
     x = df['Calories'].median()
     df['Calories'].fillna(x, inplace = True)
     print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
```

```
[16] import numpy as np
     y = np.array([35, 25, 25, 15])
     plt.pie(y)
     plt.show()
```



```
# Scatter Plot
import matplotlib.pyplot as plt
import numpy as np
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```



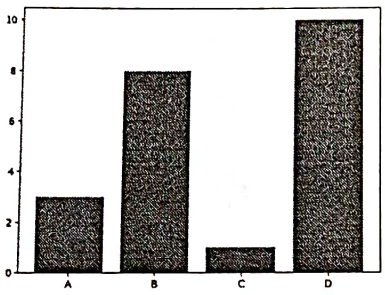

```
[13] # Statistical Information
     import pandas as pd
     df = pd.read_csv('data.csv')
     print(df.describe())
```

```
          Duration       Pulse    Maxpulse    Calories
count   169.000000  169.000000  169.000000  164.000000
mean     63.846154  107.461538  134.047337  375.790244
std      42.299949   14.510259   16.450434  266.379919
min      15.000000   80.000000  100.000000   50.300000
25%      45.000000  100.000000  124.000000  250.925000
50%      60.000000  105.000000  131.000000  318.600000
75%      60.000000  111.000000  141.000000  387.600000
max     300.000000  159.000000  184.000000  1860.400000
```

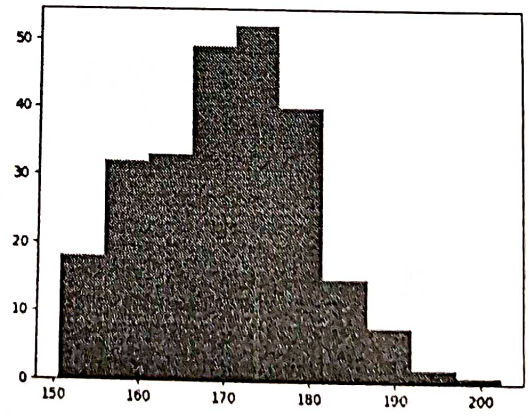
```
[14] # Establish Relationship
     import pandas as pd
     df = pd.read_csv('data.csv')
     print(df.corr())
```

```
             Duration     Pulse  Maxpulse  Calories
Duration     1.000000 -0.155408  0.009403  0.922717
Pulse       -0.155408  1.000000  0.786535  0.025121
Maxpulse     0.009403  0.786535  1.000000  0.203813
Calories     0.922717  0.025121  0.203813  1.000000
```

```
# Barchart
import matplotlib.pyplot as plt
import numpy as np
x = np.array(['A', 'B', 'C', 'D'])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
```




```
# Histogram
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```



Page No.
Date

Python Experiment 12

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To implement a web application using Django Framework & understand CRUD functionality in databases

Theory :

Django is a python-based web framework that allows you to quickly create efficient web applications.
It is also called batteries included framework because it provides built-in features for everything.
It includes
- Django admin interface
- default database SQLite3

Features of Django Framework

(1) Excellent documentation
(2) High Scalability
(3) Easy to learn
(4) Free, open source
(5) Backend server side web framework.

Django Views

It takes HTTP requests & returns HTTP response

Django Templates

It is a text document or a python string marked up using

Page No.
Date

Python Experiment 12

Name : Kreena Shah
Sapid : 60004210243
Batch : C'32

Aim : To implement a web application using Django Framework & understand CRUD functionality in databases

Theory :

Django is a python-based web framework that allows you to quickly create efficient web applications.
It is also called batteries included framework because it provides built-in features for everything.
It includes
- Django admin interface
- default database SQLite3

Features of Django Framework
(1) Excellent documentation
(2) High scalability
(3) Easy to learn
(4) Free, open source
(5) Backend server side web framework.

A+
Akk 7/4/23

Django Views
It takes HTTP requests & returns HTTP response

Django Templates
It is a text document or a python string marked up using

the django template language

- Variables
- Tags
- Filters
- Comments

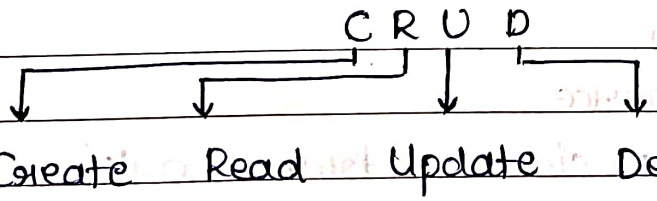
Django Models

It is django's single, definitive, source of information about your data.

Django Migrations

Django's way of propogating changes you can make to your models

- migrate
- make migrations
- sql migrate
- show migrations



Conclusion : Thus, we have implemented a web application using Django & understood the CRUD functionality.

```
Django settings for django_form project.
Generated by 'django-admin startproject' using Django 4.1.3.
For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""
from pathlib import Path
import os
MEDIA_ROOT =  os.path.dirname(os.path.abspath(__file__))
MEDIA_URL = '/images/'
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-n_9jw@o-jq@8rtonj^j@wtve+2=ti7(u&$w(+ox+$eq4ou+)bc'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'multistep_form',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'django_form.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'django_form.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
django_form\urls.py:
"""django_form URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
```

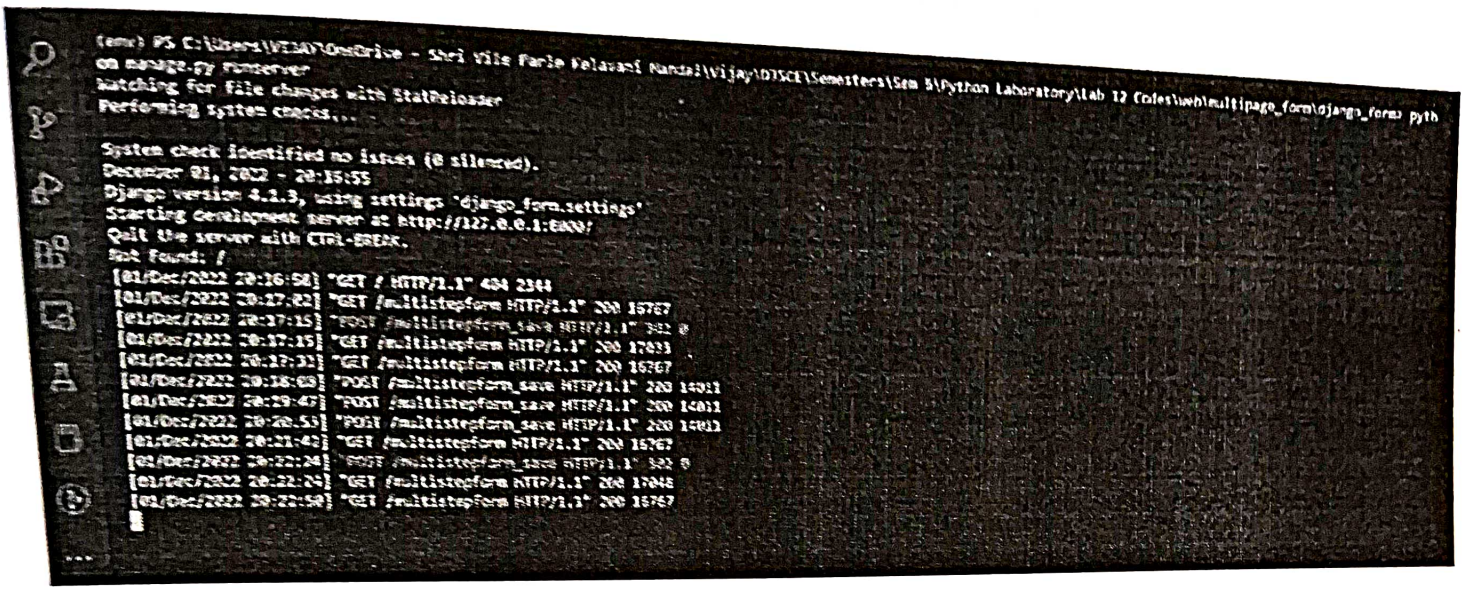
```
from django.contrib import admin
from django.urls import path
from multistep_form import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('multistepform', views.multistepform, name='multistepform'),
    path('multistepform_save', views.multistepform_save, name='multistepform_save'),
]
```

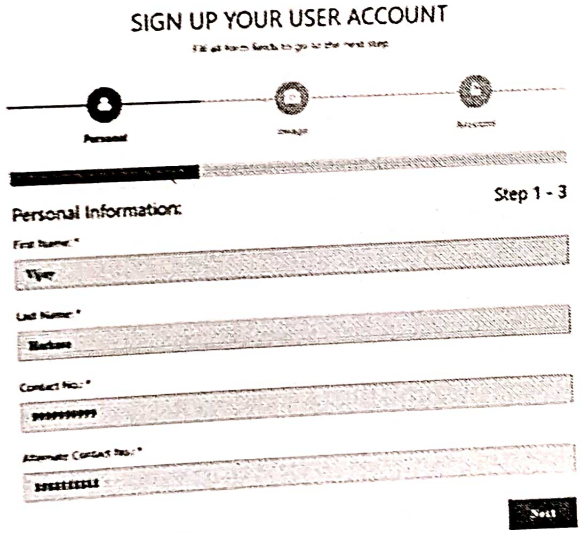
multistep_form\models.py:

```
from django.db import models

class MultiStepForm(models.Model):
    id = models.AutoField(primary_key=True)
    email = models.CharField(max_length=255)
    uname = models.CharField(max_length=255)
    pwd = models.CharField(max_length=255)
    fname = models.CharField(max_length=255)
    lname = models.CharField(max_length=255)
    phno = models.CharField(max_length=255)
    phno_2 = models.CharField(max_length=255)
    photo = models.FileField()
    sign = models.FileField()
    objects = models.Manager()
```

multistep_form\views.py:

```
from django.contrib import messages
from django.shortcuts import render
from django.http import *
from django.urls import *

from multistep_form.models import MultiStepForm

# Create your views here.
def multistepform(request):
    return render(request, "multistepform.html")

def multistepform_save(request):
    if request.method != "POST":
        return HttpResponseRedirect(reverse("multistepform"))
    else:
        email = request.POST.get("email")
        uname = request.POST.get("uname")
        pwd = request.POST.get("pwd")
        cpwd = request.POST.get("cpwd")
        fname = request.POST.get("fname")
        lname = request.POST.get("lname")
        phno = request.POST.get("phno")
        phno_2 = request.POST.get("phno_2")
        photo = request.POST.get("photo")
        sign = request.POST.get("sign")
        if pwd!=cpwd:
            messages.error(request,"Confirm Password Doesn't Match!")
            return HttpResponseRedirect(reverse('multistepform'))
        try:
            multistepform = MultiStepForm(email=email, uname=uname, pwd=pwd, fname=fname, lname=lname, phno=phno,
phno_2=phno_2, photo=photo, sign=sign)
            multistepform.save()
            messages.success(request,"Data Saved Successfully!")
            return render(request, 'success.html')
        except:
            messages.error(request,"Error in Saving Data!")
            return HttpResponseRedirect(reverse('multistepform'))
```


(env) PS C:\Users\VEDA\OneDrive - Shri Vile Parle Kelavani Mandal\Vijay\DJSCE\Semesters\Sem 5\Python Laboratory\Lab 12 Codes\web\multipage_form\django_form> Pyth
on manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
December 01, 2022 - 20:15:55
Django version 4.1.3, using settings 'django_form.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Not Found: /
[01/Dec/2022 20:16:58] "GET / HTTP/1.1" 404 2344
[01/Dec/2022 20:17:02] "GET /multistepform HTTP/1.1" 200 15767
[01/Dec/2022 20:17:15] "POST /multistepform_save HTTP/1.1" 302 0
[01/Dec/2022 20:17:15] "GET /multistepform HTTP/1.1" 200 17033
[01/Dec/2022 20:17:31] "GET /multistepform HTTP/1.1" 200 15767
[01/Dec/2022 20:18:03] "POST /multistepform_save HTTP/1.1" 200 14011
[01/Dec/2022 20:19:47] "POST /multistepform_save HTTP/1.1" 200 14011
[01/Dec/2022 20:20:59] "POST /multistepform_save HTTP/1.1" 200 14011
[01/Dec/2022 20:21:42] "GET /multistepform HTTP/1.1" 200 15767
[01/Dec/2022 20:22:24] "POST /multistepform_save HTTP/1.1" 302 0
[01/Dec/2022 20:22:24] "GET /multistepform HTTP/1.1" 200 17048
[01/Dec/2022 20:23:58] "GET /multistepform HTTP/1.1" 200 15767


SIGN UP YOUR USER ACCOUNT
Personal
Step 1 - 3
Personal Information:
First Name: *
Vijay
Last Name: *
Mahale
Contact No.: *
9999999999
Alternate Contact No.: *
8888888888
Next

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2023-24

Name: Kreena Shah SAP ID: 60004210243

Course: **Python** Course Code: **DJ19CEL504**

Year: Third Year B.Tech. Semester: V Batch: C'32

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	Σ	Avg
Course Outcome	1	1	2	2	3	3	3	4	4	5	—	—
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	4	4	5	5	4	4	4	4	4	4	42	4.2
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	4	4	5	4	4	4	4	4	4	4	41	4.1
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	5	5	4	5	5	5	5	5	5	5	49	4.9
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	5	5	4	4	5	5	5	5	5	5	48	4.8
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	-	-	-	-	-	-	-	-	-	-	—	—
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	5	5	5	5	5	5	5	5	5	5	50	5
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	-	-	-	-	-	-	-	-	-	-	—	—
Total	23	23	23	23	23	23	23	23	23	23	230	23
Signature of the faculty member												

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. = 23	Assignment marks Σ Avg. = —	Total Term-work (25) = 23
Laboratory Scaled to (15) = 14	Assignment Scaled to (10) = —	Sign of the Student: Kreena

Signature of the Faculty member:
Name of the Faculty member: Prof Sindhu Mahm
Prof Ashok Patade 7/12/23

Signature of Head of the Department
Date: 07/12/2023

Bloom's (Revised) Taxonomy