

# Python Experiment 1

DATE:

Name : Kaireena Shah

Sapid : 60004210243

Div : CSE 3rd year (A)

Batch : CSE 3rd year (A)

Subject : Python (2)

Date of Submission : 04/12/2023

Scientia (2)

Aim : To implement & study datatypes & operations

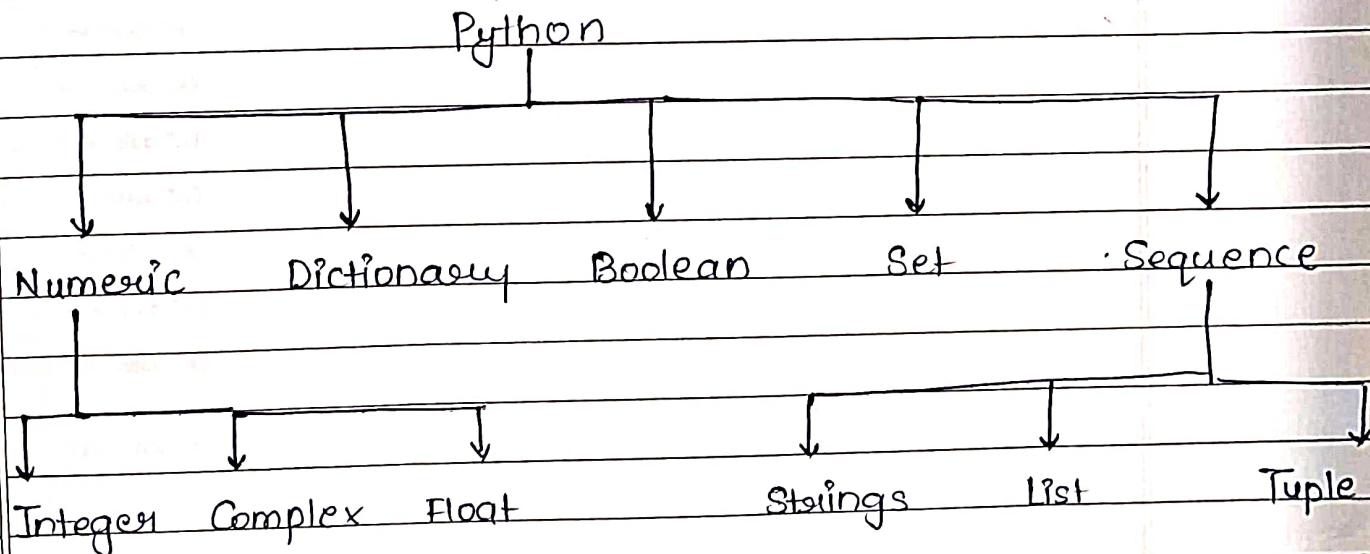
Theory : What are Datatypes & Operations?

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



## OPERATOR OVERLOADING

• It is needed for reuse.

Operator Overloading

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

Conclusion : Thus, we studied & implemented various built-in data types & operator overloading.

Today's effort took care to make our application more user friendly by minimizing its functionality & making it more reliable & maintainable.

Method

a

Program P.

for

String s1

String s2

String s3

int i

int j

int k

int l

### 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 = ('qwerty',)
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:
```

```
PS C:\Users\7400ma\OneDrive\Documents\Python\Workshop> python -m pydoc
Python 3.10.3 (tags/v3.10.3:758dbdd, Jun  6 2022, 16:17:47) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> print(FEAT)
--> FEAT(C:\Users\7400ma\OneDrive\Documents\Python\Workshop\ex1.py)
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'list'>
0
1
2
3
4
5
6
<class 'range'>
<class 'tuple'>
<class 'dict'>
{'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13, 'o': 14, 'p': 15, 'q': 16}
<class 'set'>
<class 'bool'>
<class 'NoneType'>
None
<class 'str'>
<class 'bytes'>
157046111864
*****  

initial value: 10
New value: 17
New value: 10
New value: 10
New value: 10.0
New value: 1.0
New value: 0.0
New value: 0.0
New value: 127
```

## Python Experiment 2

DATE:

Name : Koleena 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  
various different looping & conditional statements

Programs

(Triangle)  
(Diamond)

(Triangle)  
(Diamond)  
(Square)  
(Circle)  
(Flag)

2021

Astrology  
Roots  
Spiral pattern

Fractals

21

### 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 width 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
2.23606757749979
>>> ===== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-2.py =====
Enter the length of the rectangle:2
Enter the width of the rectangle:3
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', 'sony', 'lenovo', 'hp', 'samsung', 'dell', 'mi']
['apple', 'oneplus', 'lenovo', 'hp', 'samsung', 'dell', 'DELL']
>>> ===== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-6.py =====
Enter the number whose factorial has to be found
The factorial is:
>>> ===== RESTART: C:\Users\djsce.student\Desktop\python1\Expt2-7.py =====
Enter the number of elements in the Fibonacci Series:3
The series is:
0

```

```
File Edit View Insert Options Window Help
>>> ===== 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
```

## Python Experiment 3

4

DATE:

Name : Kereena Shah

Sapid : 60004210243

Div : C

Batch : ITC'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 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: int

Mathematical Function

abs(): to find?

div(): to find?

map()

filter(): function to filter out elements of list

help()

- User defined functions

- Anonymous functions to add a new element A

- Recursive Functions

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

: (min) & (max) for calculating min & max values

**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:**

```

3
[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/Pthon1/exp3-1.py
1
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/Pthon1/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')
sort()
([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'])
()

[2] 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/Pthon1/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
>>>

[3] 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/Pthon1/exp3-1.py
Enter a number: 6
6 is a perfect number
>>>

```

## Python Experiment - 4

Name : Kareena Shah

Sapid : 60004210243

Div : C1B

Batch : C'22

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 share 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(list(filter_object))  
print("Items in fruit starting with 'A' are: ", list(filter_object))  
Output:
```

```
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/CLRS/python/exp 4.py
Welcome Fatema Dolaria to the Comp 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 C
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 C
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 C
Move disk 2 from source B to destination A
Move disk 1 from source A to destination C
Move disk 3 from source D 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 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
```

```
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 D to destination C
Move disk 1 from source A to destination C
Move disk 3 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 B to destination C
Move disk 1 from source A to destination C
Move disk 2 from source A 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 : Kareena Shah

Sapid : 60004210243

Div : C

Batch : C'32

Subject : Python

Date of Submission : 04/12/2022

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 block

Raising an exception

As a developer, one can choose to throw an exception

## Exception Handling

↳ Some important built-in exception

IndexError

ImportError

KeyError

RuntimeError

SyntaxError

ValueError

NameError

**Conclusion :** Thus, we have implemented exception handling successfully.

↳ Just self-practise handling all exceptions.

↳ Should practice & not

overcome just when in doubt of how to handle

just do practice with standard exception handling code

↳ Should practice skipping over errors

↳ Should act

↳ pointing out if except needed when catching errors if

↳ ellipsis

↳ Should practice if what to do with erroneous but necessary step if

↳ handling in pipeline

↳ even in case of standard error can catch with a try

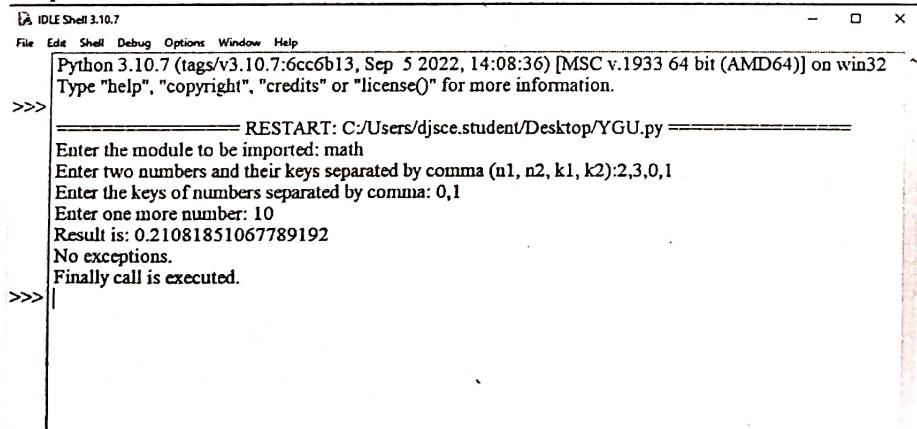
```

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:**



The screenshot shows a terminal window titled 'IDLE Shell 3.10.7'. The window contains a command-line interface for Python 3.10.7. The user has entered several commands, including importing the 'math' module, defining variables, performing calculations, and printing results. The output is displayed in a monospaced font.

```

IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:/Users/djsce.student/Desktop/YGU.py =====
Enter the module to be imported: math
Enter two numbers and their keys separated by comma (n1, n2, k1, k2):2,3,0,1
Enter the keys of numbers separated by comma: 0,1
Enter one more number: 10
Result is: 0.21081851067789192
No exceptions.
Finally call is executed.
>>> |

```

## Python Experiment 6

DATE: 04/12/2023

Name : Kareena Shah

Sapid ID : 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' - rewrite

'x' - Update

close - to close file

os.remove - to delete file

os.rmdir - to delete folder

Conclusion : Thus, we implemented file handling

in C++

using file

streaming

and found to be

filehandle object of main

function

return value of filehandle after reading and

writing to file

above handle and filehandle file is opened with

for filehandle opening with the command

handle (file) open

with file handle and filehandle

(above file handle) returning handle

handle = handle

handle = handle

handle = handle

filehandle = handle

filehandle file = handle

cout << endl << handle << endl;

**Code**

```
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")
```

**Experiment No: 6****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/exp6.py -----
Enter 5 numbers:
Enter number 1: 122
Enter number 2: 342
Enter number 3: 44
Enter number 4: 543
Enter number 5: 65
44
65
122
342
543
List of sorted words:
['26th', 'Dolaria', 'Fatema', 'Hello,', 'I', 'June.', 'am', 'and', 'birthdate', 'is', 'my', 'on']
```

## Python Experiment 7

Name : Kireena 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 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.

Its 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 : Kileena 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.

Its primary function 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

## E:\ Computer Application

1. re module

2. sub()

Replaces one or many matches within a string

3. Meta-Characters

Characters with special meaning

- Set of characters

\ - Signals a special sequence

. - Any character except a new line character is

^ - Starts with

\$ - Ends with

\* - Zero or more occurrences

+ - One or more occurrences

? - Zero or one occurrence

{ } - Exactly the specified no. of occurrences

| - 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",  
"rolfmoa@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:**

```
[A] 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.1932 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:/Users/Fatema/OneDrive/Desktop/CLG/python/exp7.py -----  
Name of the user: Anderson  
Name of the user: Thareja  
Name of the user: Morri  
Name of the user: Roy  
Name of the user: Gandhi  
Name of the user: Modi  
Website name: google  
Website name: udemy  
Website name: stackoverflow  
Website name: djsce  
Website name: google  
Email id: rishit.grover@gmail.com  
Email id: kapeesh.grover@yahoo.co.in  
Email id: abhishek.shah@gmail.com  
Email id: shahp98@gmail.com  
Email id: demo_user@gmail.com  
Email id: rolfmoa@yahoo.co.in  
Phone number: 27777647  
Phone number: 233*333*88  
Phone number: 455-78-888  
Phone number: 022-240-93836  
Phone number: 02642*221*381  
>>>
```

## 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 : 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

At  
this  
time

### Creating Database

CREATE DATABASE db.name

## 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

## 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.

# Database Connectivity

1. Creating Table

**Creating Table**

`CREATE TABLE tablename`

2. Selecting from Table  
**'SELECT'**

3. Deleting a record

**'DELETE FROM'** statement

Deletes all the rows matching

4. Updating table  
**'UPDATE'** statement

Updates the rows matching

5. Deleting table

**'DROP TABLE'** statement

6. Deleting a database

**'DROP DATABASE'** statement

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

```

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', 'Sindhwan', 'Borivali')")
cur.execute("INSERT INTO Student VALUES ('117', 'Dhrushi', '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', 'Sindhwan', 'Borivali')
('117', 'Dhrushi', 'Tank', 'SoBo')
('115', 'Mihir', 'Vora', 'Kandivali')
('Viraj', 'Borivali')
('Dhrushi', 'SoBo')
('Mihir', 'Kandivali')
None
('120', 'Viraj', 'Sindhwan', 'Borivali')|
<sqlite3.Cursor object at 0x000002608FDC7E40>
('90', 'Husain', 'Rehmanji', 'Thane', 'gmail.com')

```

Experiment No: 8

## Python Experiment 9

Name : Kireena 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 protocol - Transmission Control Protocol (TCP)

It is reliable & has in-order data delivery which makes

## 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()

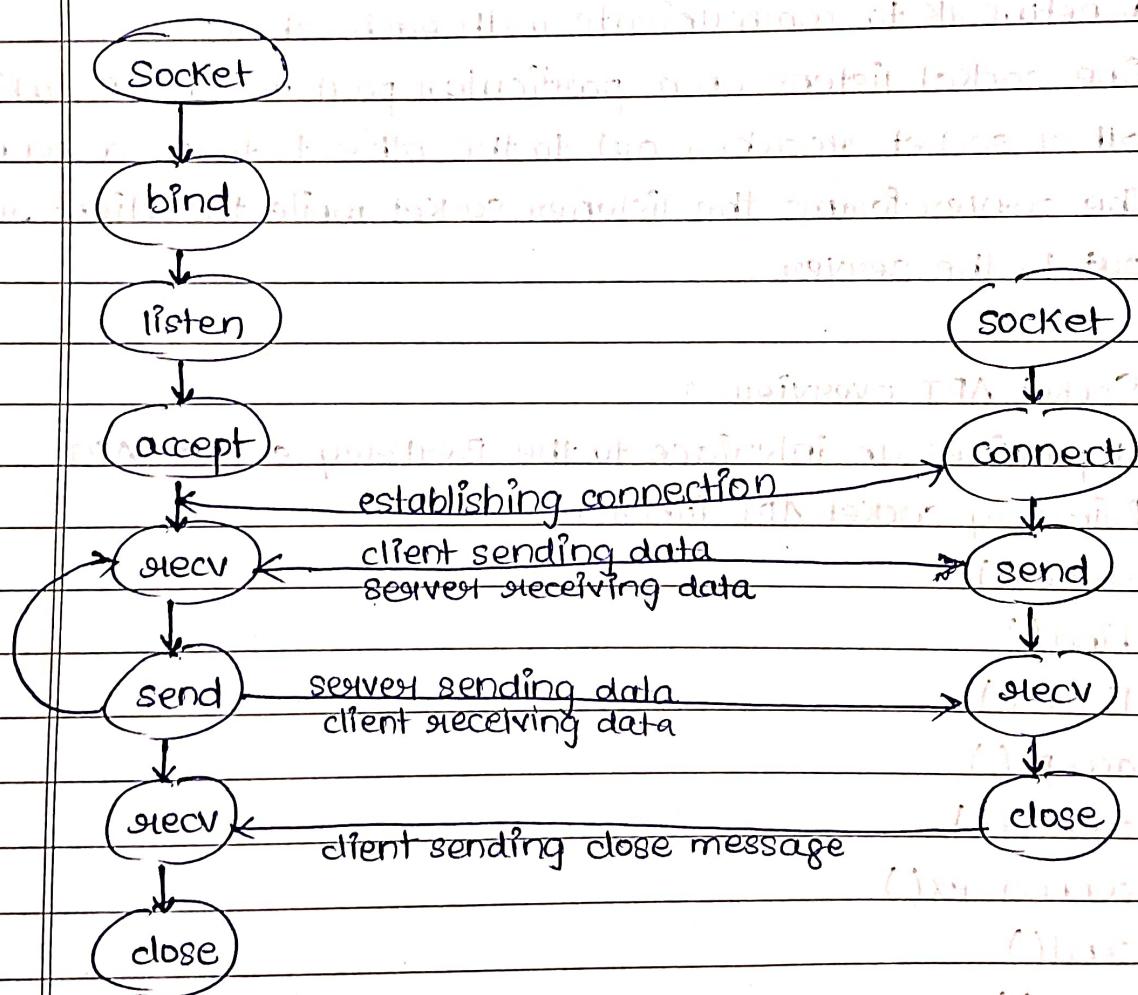
Default protocol - Transmission Control Protocol (TCP)

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:**

```
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()
```

**Experiment No: 9**

```
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 : Kireena 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 shipped with python

Python with Tkinter is the fastest & easiest way to create GUI applications.

Steps to create a Tkinter app :

- Import the module - Tkinter
- Create the main window (container)
- Add any no. of widgets to the main window
- 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) mainloop()

Page No.	
Date	

## Python Experiment 10

Name : Krienna 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 :

- Import the module - Tkinter
- Create the main window (container)
- Add any no. of widgets to the main window
- 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()

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`

`Scrollbar`

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

### Experiment No: 10

```
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)

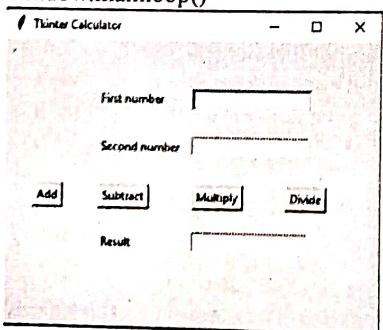
def add(self, event):
    self.t3.delete(0, 'end')
    num1=int(self.t1.get())
    num2=int(self.t2.get())
    result=num1+num2
    self.t3.insert(END, str(result))

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 & intuitively.

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 & intuitively.

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

## Matplotlib

It is cross platform data visualization & graphical plotting library for python & its 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 successfully demonstrated the use of Pandas & Matplotlib modules in python.

## Experiment No: 11

Q 10 • Creating series  
Import pandas as pd  
df = pd.read\_csv('data.csv')  
print(df)

Q 11 • Creating series with labels  
Import pandas as pd  
series = pd.Series([100, 200, 300, 400])  
print(series)

Q 12 • Creating series with labels and index  
Import pandas as pd  
series = pd.Series([100, 200, 300, 400], index=[1, 2, 3, 4])  
print(series)

Q 13 • Load the csv into a DataFrame  
Import pandas as pd  
df = pd.read\_csv('data.csv')  
print(df)

	Duration	Pulse	Maxpulse	Calories
0	60	120	130	400.1
1	60	117	145	473.8
2	60	100	135	346.8
3	45	100	175	221.4
4	45	117	140	406.4
..	..	..	..	..
164	60	100	140	296.4
165	60	110	145	306.0
166	60	110	145	318.2
167	75	120	150	328.4
168	75	120	150	330.4

[169 rows x 4 columns]

Q 14 • Info about data  
Import pandas as pd  
df = pd.read\_csv('data.csv')  
print(df.info())  
print(df.describe())  
print(df.describe(include='all'))

Q 15 • Duration Pulse Maxpulse Calories

	Duration	Pulse	Maxpulse	Calories
0	60	120	130	400.1
1	60	117	145	473.8
2	60	100	135	346.8
3	45	100	175	221.4
4	45	117	140	406.4
..	..	..	..	..
164	60	120	130	296.4
165	60	117	145	306.0
166	60	110	145	318.2
167	75	120	150	328.4
168	75	120	150	330.4
169	60	120	130	346.4
170	60	117	145	473.8
171	60	100	135	346.8
172	45	100	175	221.4
173	45	117	140	406.4
174	60	120	130	306.0
175	60	117	145	346.8
176	60	110	145	318.2
177	75	120	150	328.4
178	75	120	150	330.4
179	60	120	130	346.4
180	60	117	145	473.8
181	60	110	145	346.8
182	45	100	175	221.4
183	45	117	140	406.4
184	60	120	130	306.0
185	60	117	145	346.8
186	60	110	145	318.2
187	75	120	150	328.4
188	75	120	150	330.4
189	60	120	130	346.4
190	60	117	145	473.8
191	60	110	145	346.8
192	45	100	175	221.4
193	45	117	140	406.4
194	60	120	130	306.0
195	60	117	145	346.8
196	60	110	145	318.2
197	75	120	150	328.4
198	75	120	150	330.4
199	60	120	130	346.4
200	60	117	145	473.8
201	60	110	145	346.8
202	45	100	175	221.4
203	45	117	140	406.4
204	60	120	130	306.0
205	60	117	145	346.8
206	60	110	145	318.2
207	75	120	150	328.4
208	75	120	150	330.4
209	60	120	130	346.4
210	60	117	145	473.8
211	60	110	145	346.8
212	45	100	175	221.4
213	45	117	140	406.4
214	60	120	130	306.0
215	60	117	145	346.8
216	60	110	145	318.2
217	75	120	150	328.4
218	75	120	150	330.4
219	60	120	130	346.4
220	60	117	145	473.8
221	60	110	145	346.8
222	45	100	175	221.4
223	45	117	140	406.4
224	60	120	130	306.0
225	60	117	145	346.8
226	60	110	145	318.2
227	75	120	150	328.4
228	75	120	150	330.4
229	60	120	130	346.4
230	60	117	145	473.8
231	60	110	145	346.8
232	45	100	175	221.4
233	45	117	140	406.4
234	60	120	130	306.0
235	60	117	145	346.8
236	60	110	145	318.2
237	75	120	150	328.4
238	75	120	150	330.4
239	60	120	130	346.4
240	60	117	145	473.8
241	60	110	145	346.8
242	45	100	175	221.4
243	45	117	140	406.4
244	60	120	130	306.0
245	60	117	145	346.8
246	60	110	145	318.2
247	75	120	150	328.4
248	75	120	150	330.4
249	60	120	130	346.4
250	60	117	145	473.8
251	60	110	145	346.8
252	45	100	175	221.4
253	45	117	140	406.4
254	60	120	130	306.0
255	60	117	145	346.8
256	60	110	145	318.2
257	75	120	150	328.4
258	75	120	150	330.4
259	60	120	130	346.4
260	60	117	145	473.8
261	60	110	145	346.8
262	45	100	175	221.4
263	45	117	140	406.4
264	60	120	130	306.0
265	60	117	145	346.8
266	60	110	145	318.2
267	75	120	150	328.4
268	75	120	150	330.4
269	60	120	130	346.4
270	60	117	145	473.8
271	60	110	145	346.8
272	45	100	175	221.4
273	45	117	140	406.4
274	60	120	130	306.0
275	60	117	145	346.8
276	60	110	145	318.2
277	75	120	150	328.4
278	75	120	150	330.4
279	60	120	130	346.4
280	60	117	145	473.8
281	60	110	145	346.8
282	45	100	175	221.4
283	45	117	140	406.4
284	60	120	130	306.0
285	60	117	145	346.8
286	60	110	145	318.2
287	75	120	150	328.4
288	75	120	150	330.4
289	60	120	130	346.4
290	60	117	145	473.8
291	60	110	145	346.8
292	45	100	175	221.4
293	45	117	140	406.4
294	60	120	130	306.0
295	60	117	145	346.8
296	60	110	145	318.2
297	75	120	150	328.4
298	75	120	150	330.4
299	60	120	130	346.4
300	60	117	145	473.8
301	60	110	145	346.8
302	45	100	175	221.4
303	45	117	140	406.4
304	60	120	130	306.0
305	60	117	145	346.8
306	60	110	145	318.2
307	75	120	150	328.4
308	75	120	150	330.4
309	60	120	130	346.4
310	60	117	145	473.8
311	60	110	145	346.8
312	45	100	175	221.4
313	45	117	140	406.4
314	60	120	130	306.0
315	60	117	145	346.8
316	60	110	145	318.2
317	75	120	150	328.4
318	75	120	150	330.4
319	60	120	130	346.4
320	60	117	145	473.8
321	60	110	145	346.8
322	45	100	175	221.4
323	45	117	140	406.4
324	60	120	130	306.0
325	60	117	145	346.8
326	60	110	145	318.2
327	75	120	150	328.4
328	75	120	150	330.4
329	60	120	130	346.4
330	60	117	145	473.8
331	60	110	145	346.8
332	45	100	175	221.4
333	45	117	140	406.4
334	60	120	130	306.0
335	60	117	145	346.8
336	60	110	145	318.2
337	75	120	150	328.4
338	75	120	150	330.4
339	60	120	130	346.4
340	60	117	145	473.8
341	60	110	145	346.8
342	45	100	175	221.4
343	45	117	140	406.4
344	60	120	130	306.0
345	60	117	145	346.8
346	60	110	145	318.2
347	75	120	150	328.4
348	75	120	150	330.4
349	60	120	130	346.4
350	60	117	145	473.8
351	60	110	145	346.8
352	45	100	175	221.4
353	45	117	140	406.4
354	60	120	130	306.0
355	60	117	145	346.8
356	60	110	145	318.2
357	75	120	150	328.4
358	75	120	150	330.4
359	60	120	130	346.4
360	60	117	145	473.8
361	60	110	145	346.8
362	45	100	175	221.4
363	45	117	140	406.4
364	60	120	130	306.0
365	60	117	145	346.8
366	60	110	145	318.2
367	75	120	150	328.4
368	75	120	150	330.4
369	60	120	130	346.4
370	60	117	145	473.8
371	60	110	145	346.8
372	45	100	175	221.4
373	45	117	140	406.4
374	60	120	130	306.0
375	60	117	145	346.8
376	60	110	145	318.2
377	75	120	150	328.4
378	75	120	150	330.4
379	60	120	130	346.4
380	60	117	145	473.8
381	60	110	145	346.8
382	45	100	175	221.4
383	45	117	140	406.4
384	60	120	130	306.0
385	60	117	145	346.8
386	60	110	145	318.2
387	75	120	150	328.4
388	75	120	150	330.4
389	60	120	130	346.4
390	60	117	145	473.8
391	60	110	145	346.8
392	45	100	175	221.4
393	45	117	140	406.4
394	60	120	130	306.0
395	60	117	145	346.8
396	60	110	145	318.2
397	75	120	150	328.4
398	75	120	150	330.4
399	60	120	130	346.4
400	60	117	145	473.8
401	60	110	145	346.8
402	45	100	175	221.4
403	45	117	140	406.4
404	60	120	130	306.0
405	60	117	145	346.8
406	60	110	145	318.2
407	75	120	150	328.4
408	75	120	150	330.4
409	60	120	130	346.4
410	60	117	145	473.8
411	60	110	145	346.8
412	45	100	175	221.4
413	45	117	140	406.4
414	60	1		

```
[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	400.1
1	60	117	145	470.0
2	60	103	135	340.0
3	45	99	175	282.4
4	45	117	148	400.0
..	..	..	..	..
164	60	105	140	290.0
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

[149 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	400.1
1	60	117	145	470.0
2	60	103	135	340.0
3	45	100	175	282.4
4	45	117	148	400.0
..	..	..	..	..
164	60	105	140	290.0
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

[149 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	400.1
1	60	117	145	470.0
2	60	103	135	340.0
3	45	100	175	282.4
4	45	117	148	400.0
..	..	..	..	..
164	60	105	140	290.0
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

[149 rows x 4 columns]

[13] # statistical information

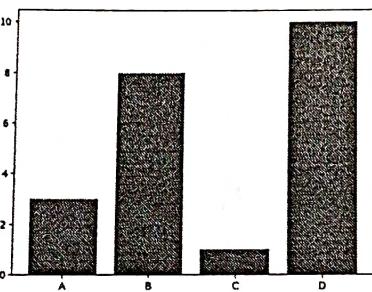
```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.describe())
```

```
Duration      Pulse      Maxpulse      Calories
count    149.000000  149.000000  149.000000  149.000000
mean    61.266154  109.999999  134.047337  375.796244
std     42.293949  14.210239  16.458434  264.379913
min    15.000000  64.000000  64.000000  50.000000
25%    45.000000  104.000000  116.000000  254.715000
50%    60.000000  109.000000  121.000000  317.000000
75%    75.000000  111.000000  141.000000  387.000000
max    300.000000  159.000000  184.000000  1266.000000
```

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

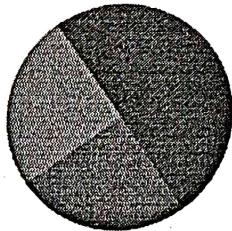
Duration	Pulse	Maxpulse	Calories	
Duration	1.000000	-0.155480	0.022403	0.027717
Pulse	-0.155480	1.000000	0.786535	0.025121
Maxpulse	0.022403	0.786535	1.000000	0.208117
Calories	0.027717	0.025121	0.208117	1.000000

[15] # Bar chart
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()



[16] import numpy as np

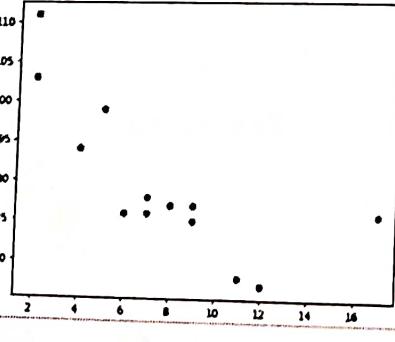
```
y = np.array([35, 25, 25, 15])
plt.pie(y)
plt.show()
```



[17] # Scatter Plot

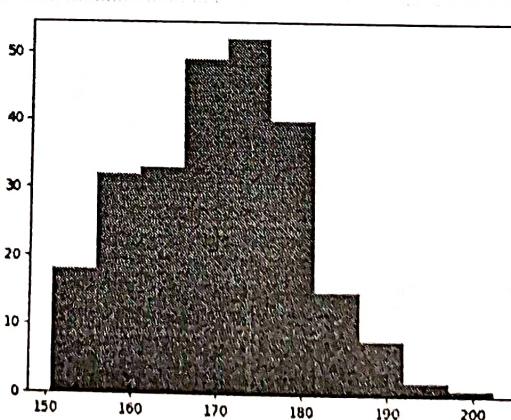
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([5,7,8,7,17,2,9,4,12,12,9,6])
y = np.array([55,85,87,45,111,65,103,87,54,78,77,45,84])
plt.scatter(x, y)
plt.show()
```

[18]



[19] # Histogram

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



## Python Experiment 12

Name : Kreena Shah

SapId : 60004210243

Batch : C'22

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

## 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

## the Django template language

- Variables

- Tags

- Filters

- Comments

## Django Models

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

## Django Migrations

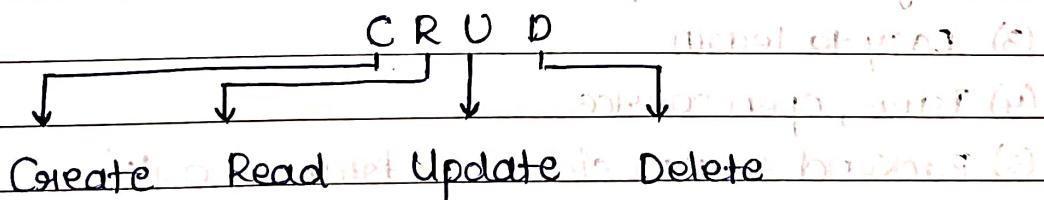
Django's way of propagating changes you can make to your models

- migrate

- make migrations

- sql migrate

- show migrations



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

**Experiment No: 12**

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\Vihaan\OneDrive - Shri Mata Parvati Lalwani\Vihaan\DTSCETSemesters\Sem 3\Python Laboratory\Lab 12 (file)\multipage_form\django_forms.pyth
on manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
December 21, 2022 - 20:35:55
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: /
[21/Dec/2022 20:36:58] "GET / HTTP/1.1" 404 2344
[21/Dec/2022 20:37:02] "GET /multistepform HTTP/1.1" 200 15767
[21/Dec/2022 20:37:15] "POST /multistepform_save HTTP/1.1" 302 0
[21/Dec/2022 20:37:15] "GET /multistepform HTTP/1.1" 200 17033
[21/Dec/2022 20:37:31] "GET /multistepform HTTP/1.1" 200 16267
[21/Dec/2022 20:38:03] "POST /multistepform_save HTTP/1.1" 200 15011
[21/Dec/2022 20:39:47] "POST /multistepform_save HTTP/1.1" 200 14811
[21/Dec/2022 20:40:55] "POST /multistepform_save HTTP/1.1" 200 15811
[21/Dec/2022 20:41:42] "GET /multistepform HTTP/1.1" 200 15767
[21/Dec/2022 20:42:24] "POST /multistepform_save HTTP/1.1" 302 0
[21/Dec/2022 20:42:24] "GET /multistepform HTTP/1.1" 200 17045
[21/Dec/2022 20:42:54] "GET /multistepform HTTP/1.1" 200 15767
```

## SIGN UP YOUR USER ACCOUNT

Click Next Step to go to the next step

Step 1 - 3

Personal Information:

First Name \*

Vivek

Last Name \*

Kumar

Contact No. \*

9899999999

Alternate Contact No. \*

9899999999

Next



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	$\Sigma$	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	5	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)	5	—	—	—	—	—	—	—	—	—	—	—
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 $\Sigma$ Avg. = 23	Assignment marks $\Sigma$ Avg. = —	Total Term-work (25) = 23
Laboratory Scaled to (15) = 14	Assignment Scaled to (10) = —	Sign of the Student: <u>Kreena Shah</u>

Signature of the Faculty member:

Name of the Faculty member: Prof. Sindhu Maben  
Prof. Ashok Patade Ashok Patade 211423

Signature of Head of the Department

Date: 07/12/2023

Bloom's (Revised) Taxonomy