PYTHON PROGRAMMING PRACTICAL LIST NO. 01

Akshay Mahadev Yadav
akshaymahadevyadav@gmail.com

PRACTICAL LIST NO – 01

Que 01 :- Python installation and configuration with windows and Linux.

Que 02 :- Programs for understanding the data types, control flow statements, blocks and loops.

Que 03 :- Programs for understanding functions, use of built in functions, user defined functions.

Que 04 :- Programs to use existing modules, packages and creating modules, packages.

Que 05 :- Programs for implementations of all object-oriented concepts like class, method, inheritance, polymorphism etc. (Real life examples must be covered for the implementation of objectoriented concepts).

Que 06 :- Programs for parsing of data, validations like Password, email, URL, etc.

Que 07 :- Programs for Pattern finding should be covered.

Que 08 :- Programs covering all the aspects of Exception handling, user defined exception, Multithreading should be covered.

Que 09 :- Programs demonstrating the IO operations like reading from file, writing into file from different file types like data file, binary file, etc.

Que 10 :- Programs to perform searching, adding, updating the content from the file.

Que 11 :- Program for performing CRUD operation with MongoDB and Python.

Que 12 :- Basic programs with NumPy as Array, Searching and Sorting, date & time and String handling.

Que 13 :- Programs for series and data frames should be covered.

Que 14 :- Programs to demonstrate data pre-processing and data handling with data frame.

Que 15 :- Program for data visualization should be covered.

Que 01 :- Python installation and configuration with windows and Linux.

How to install Python on Windows?

Prerequisite: Python Language Introduction

Before we start with how to install Python3 on Windows, let's first go through the basic introduction to Python. Python is a widely-used general-purpose, high-level programming language. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions- Python 2 and Python 3. Both are quite different.

- **Getting started with Python**

Python is a lot easier to code and learn. Python programs can be written on any plain text editor like **notepad**, **notepad++,** or anything of that sort. One can also use an **online IDE for writing Python codes** or can even install one on their system to make it more feasible to write these codes because IDEs provide a lot of features like intuitive code editor, debugger, compiler, etc.

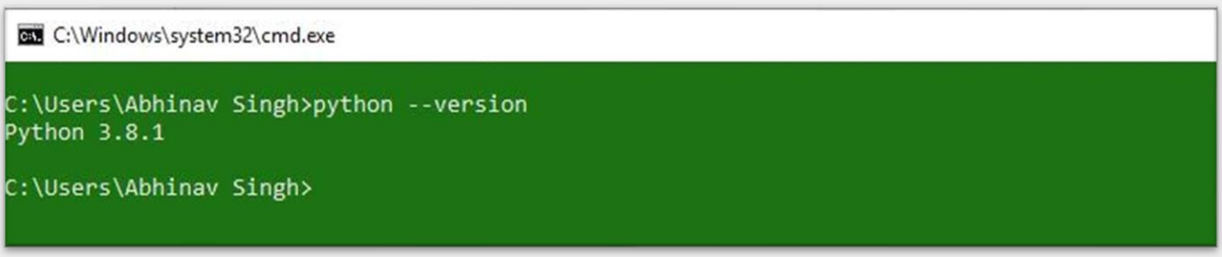
To begin with, writing Python Codes and performing various intriguing and useful operations, one must have Python installed on their System. This can be done by following the step by step instructions provided below:

- What if Python already exists?

Let's check To check if your device is pre-installed with Python or not, just go to the Command line(search for cmd in the Run dialog(Start + R). Now run the following command:

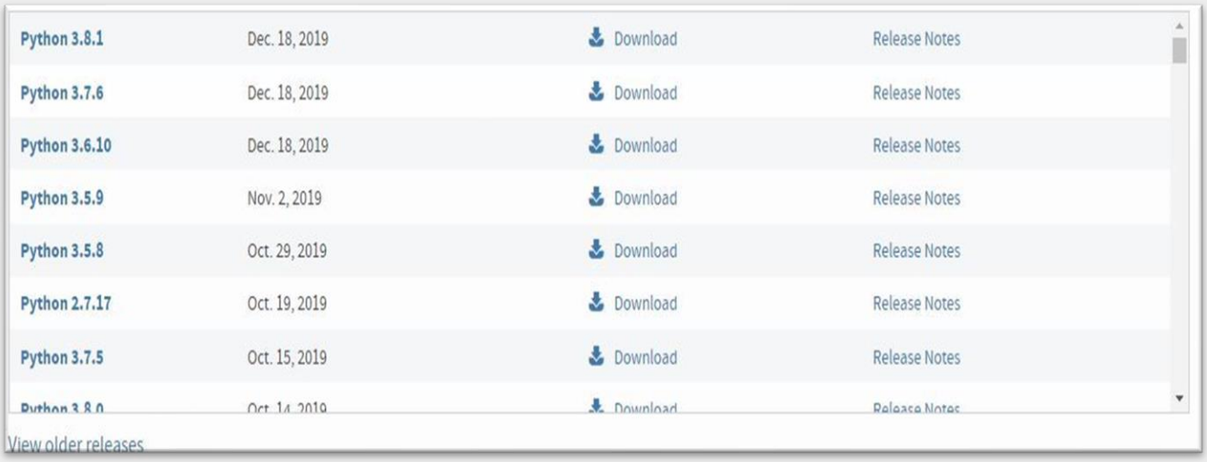
python --version

If Python is already installed, it will generate a message with the Python version



- Download and Install Python:

Before starting with the installation process, you need to download it. For that all versions of Python for Windows are available on python.org.



Python 3.8.1	Dec. 18, 2019	Download	Release Notes
Python 3.7.6	Dec. 18, 2019	Download	Release Notes
Python 3.6.10	Dec. 18, 2019	Download	Release Notes
Python 3.5.9	Nov. 2, 2019	Download	Release Notes
Python 3.5.8	Oct. 29, 2019	Download	Release Notes
Python 2.7.17	Oct. 19, 2019	Download	Release Notes
Python 3.7.5	Oct. 15, 2019	Download	Release Notes
Python 3.8.0	Oct. 14, 2019	Download	Release Notes

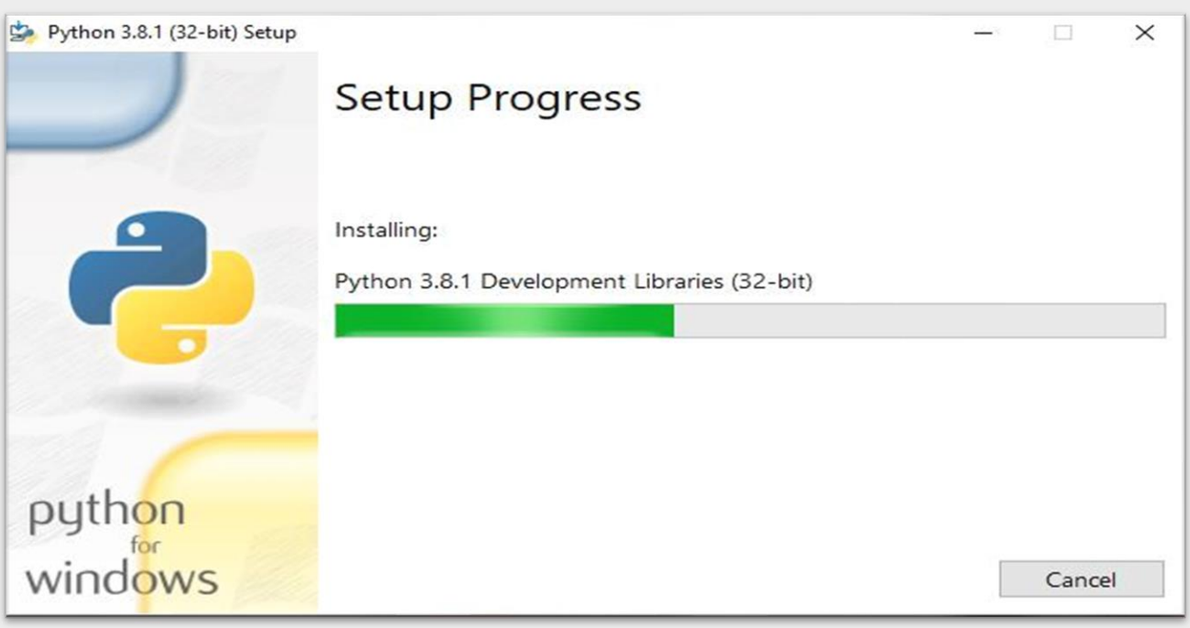
View older releases

Download the required version and follow the further instructions for the installation process.

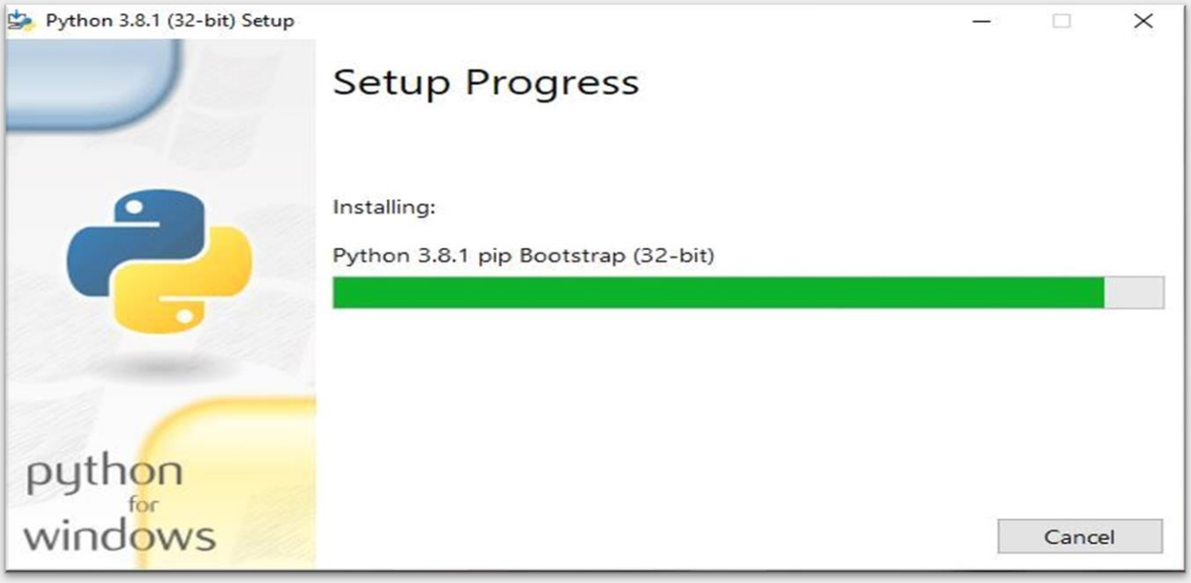
Beginning the installation.

- Getting Started:

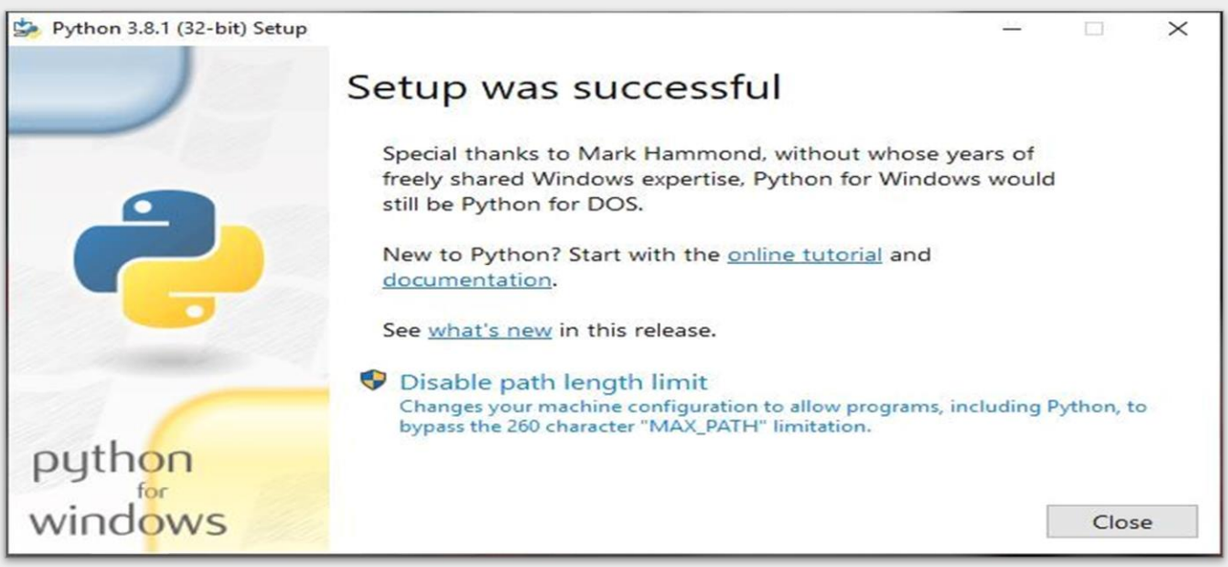
- Installing Libraries :



- Installing pip and other features :

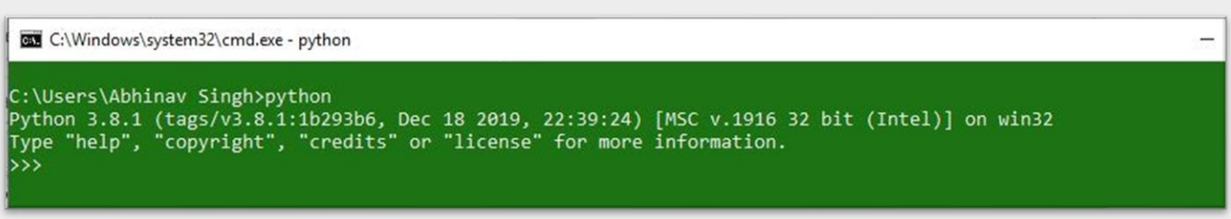


- Finishing Installation:



To verify the installation enter the following commands in your Terminal.

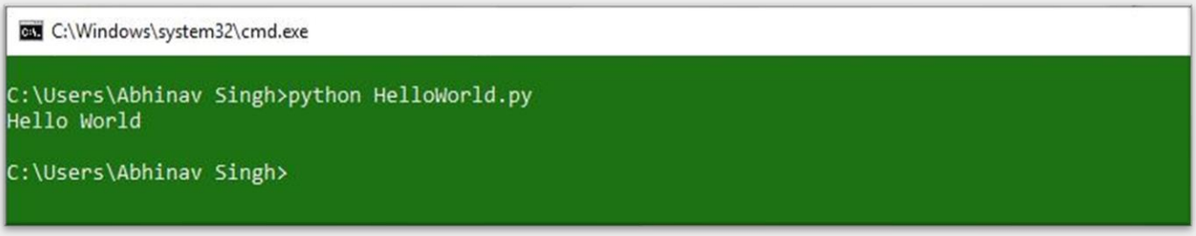
python



Let's consider a simple Hello World Program.

```
# Python program to print
 # Hello World
print("Hello World")
```

Output :



- How to install Python on Linux?

Prerequisite: Python Language Introduction

Before we start with how to install Python3 on Linux, let's first go through the basic introduction to Python. Python is a widely-used general-purpose, high-level programming language. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions- Python 2 and Python 3. Both are quite different.

- **Getting started with Python**

Python is a lot easier to code and learn. Python programs can be written on any plain text editor like notepad, notepad++, or anything of that sort. One can also use an online IDE for writing Python codes or can even install one on their system to make it more feasible to write these codes because IDEs provide a lot of features like intuitive code editor, debugger, compiler, etc.

To begin with, writing Python Codes and performing various intriguing and useful operations, one must have Python installed on their System. This can be done by following the step-by-step instructions provided below:

- **What if Python already exists? Let's check**

Most of the Linux OS has Python pre-installed. To check if your device is pre-installed with Python or not, just go to terminal using Ctrl+Alt+T

- Now run the following command:

For Python2

python --version

For Python3.x

python3.x --version

If Python is already installed, it will generate a message with the Python version available.

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3.8 --version
Python 3.8.0
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3 --version
Python 3.6.9
nikhil@nikhil-Lenovo-V130-15IKB:~$ python --version
Python 2.7.17
nikhil@nikhil-Lenovo-V130-15IKB:~$
```

- **Download and Install Python:**

Before starting with the installation process, you need to download it. For that all versions of Python for Linux are available on python.org.

Python 3.8.1	Dec. 18, 2019	Download	Release Notes
Python 3.7.6	Dec. 18, 2019	Download	Release Notes
Python 3.6.10	Dec. 18, 2019	Download	Release Notes
Python 3.5.9	Nov. 2, 2019	Download	Release Notes
Python 3.5.8	Oct. 29, 2019	Download	Release Notes
Python 2.7.17	Oct. 19, 2019	Download	Release Notes
Python 3.7.5	Oct. 15, 2019	Download	Release Notes
Python 3.8.0	Oct. 14, 2019	Download	Release Notes

View older releases

Download the required version and follow the further instructions for the installation process.

- **Beginning the installation.**

For almost every Linux system, the following command could be used to install Python directly:

$ sudo apt-get install python3.8

- **Getting Started**

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
nikhil@nikhil-Lenovo-V130-15IKB:~$ sudo apt-get install python3.8
[sudo] password for nikhil:
```

- Assigning DiskSpace :

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
nikhil@nikhil-Lenovo-V130-15IKB:~$ sudo apt-get install python3.8
[sudo] password for nikhil:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8-minimal
Suggested packages:
  python3.8-venv python3.8-doc binfmt-support
The following NEW packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8 python3.8-minimal
0 upgraded, 4 newly installed, 0 to remove and 9 not upgraded.
Need to get 4,551 kB of archives.
After this operation, 18.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

- Fetching and Installing Packages :

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
[sudo] password for nikhil:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8-minimal
Suggested packages:
  python3.8-venv python3.8-doc binfmt-support
The following NEW packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8 python3.8-minimal
0 upgraded, 4 newly installed, 0 to remove and 9 not upgraded.
Need to get 4,551 kB of archives.
After this operation, 18.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 libpython3.8-minimal
amd64 3.8.0-3~18.04 [704 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3.8-minimal amd
64 3.8.0-3~18.04 [1,816 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 libpython3.8-stdlib a
md64 3.8.0-3~18.04 [1,677 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3.8 amd64 3.8.0
-3~18.04 [355 kB]
Fetched 4,551 kB in 3s (1,427 kB/s)
```

- Getting through the installation process :

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
-3~18.04 [355 kB]
Fetched 4,551 kB in 3s (1,427 kB/s)
Selecting previously unselected package libpython3.8-minimal:amd64.
(Reading database ... 258857 files and directories currently installed.)
Preparing to unpack .../libpython3.8-minimal_3.8.0-3~18.04_amd64.deb ...
Unpacking libpython3.8-minimal:amd64 (3.8.0-3~18.04) ...
Selecting previously unselected package python3.8-minimal.
Preparing to unpack .../python3.8-minimal_3.8.0-3~18.04_amd64.deb ...
Unpacking python3.8-minimal (3.8.0-3~18.04) ...
Selecting previously unselected package libpython3.8-stdlib:amd64.
Preparing to unpack .../libpython3.8-stdlib_3.8.0-3~18.04_amd64.deb ...
Unpacking libpython3.8-stdlib:amd64 (3.8.0-3~18.04) ...
Selecting previously unselected package python3.8.
Preparing to unpack .../python3.8_3.8.0-3~18.04_amd64.deb ...
Unpacking python3.8 (3.8.0-3~18.04) ...
Setting up libpython3.8-minimal:amd64 (3.8.0-3~18.04) ...
Setting up python3.8-minimal (3.8.0-3~18.04) ...
Setting up libpython3.8-stdlib:amd64 (3.8.0-3~18.04) ...
Setting up python3.8 (3.8.0-3~18.04) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

- **Finished Installation :**

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
-3~18.04 [355 kB]
Fetched 4,551 kB in 3s (1,427 kB/s)
Selecting previously unselected package libpython3.8-minimal:amd64.
(Reading database ... 258857 files and directories currently installed.)
Preparing to unpack .../libpython3.8-minimal_3.8.0-3~18.04_amd64.deb ...
Unpacking libpython3.8-minimal:amd64 (3.8.0-3~18.04) ...
Selecting previously unselected package python3.8-minimal.
Preparing to unpack .../python3.8-minimal_3.8.0-3~18.04_amd64.deb ...
Unpacking python3.8-minimal (3.8.0-3~18.04) ...
Selecting previously unselected package libpython3.8-stdlib:amd64.
Preparing to unpack .../libpython3.8-stdlib_3.8.0-3~18.04_amd64.deb ...
Unpacking libpython3.8-stdlib:amd64 (3.8.0-3~18.04) ...
Selecting previously unselected package python3.8.
Preparing to unpack .../python3.8_3.8.0-3~18.04_amd64.deb ...
Unpacking python3.8 (3.8.0-3~18.04) ...
Setting up libpython3.8-minimal:amd64 (3.8.0-3~18.04) ...
Setting up python3.8-minimal (3.8.0-3~18.04) ...
Setting up libpython3.8-stdlib:amd64 (3.8.0-3~18.04) ...
Setting up python3.8 (3.8.0-3~18.04) ...
Processing triggers for gnome-menus (3.13.3-11ubuntu1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
nikhil@nikhil-Lenovo-V130-15IKB:~$
```

To verify the installation enter the following commands in your Terminal.

python3.8

```
nikhil@nikhil-Lenovo-V130-15IKB: ~
File  Edit  View  Search  Terminal  Help
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3.8
Python 3.8.0 (default, Oct 28 2019, 16:14:01)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Let's consider a simple Hello World Program.

Python program to print

Hello World

print("Hello World")

Output :

```
nikhil@nikhil-Lenovo-V130-15IKB: ~/Desktop/gfg
File  Edit  View  Search  Terminal  Help
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$ python3.8 HelloWorld.py
Hello World
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$
```

Que 02 :- Programs for understanding the data types, control flow statements, blocks and loops.

- Data Types :-
  - Binary Data Type :-
    - Bytes
    - Memory View
    - Byte Array
  - Mapping Data Type :-
    - Dict
  - Numeric Data Type :-
    - Int
    - Float
    - Complex
  - Text Data Type :-
    - Str
  - Boolean Data Type :-
    - Bool
  - Set Data Type :-
    - Set
    - Frozen set
  - Sequence Data Type :-
    - List
    - Range
    - Tuple
- Flow Control Statement :-
  - Sequential.
  - Selection :
    - If
    - If…else
    - Switch
  - Repetition :
    - While Loop
    - Do…while Loop
    - For Loop
- Block / Indentation.

- **Data Types Programs :-**

1. **Bytes Data Type Program :-**

```
x = b"Hello"

print( x, "And")

print("Data Type is", type(x))
```

Output :-

```
b'Hello' And

Data Type is <class 'bytes'>
```

2. **Memory View Data Type Program :-**

```
x = memoryview(bytes(5))

print( x, "And")

print("Data Type is", type(x))
```

Output :-

```
<memory at 0x000001B1B5609D00> And

Data Type is <class 'memoryview'>
```

3. **Byte Array Data Type Program :-**

```
x = bytearray(5)

print( x, "And")

print("Data Type is", type(x))
```

Output :-

```
bytearray(b'\x00\x00\x00\x00\x00') And

Data Type is <class 'bytearray'>
```

4. **Dict Data Type Program :-**

```
x = {"name" : "John", "age" : 36}

print("The Dictionary", x, "And")

print("Data Type is", type(x))
```

Output :-

The Dictionary {'name': 'John', 'age': 36} And

Data Type is <class 'dict'>

5. **Int Data Type Program :-**

```
x = 20
print("The Number", x, "And")
print("Data Type is", type(x))
```

Output :-

The Number 20 And

Data Type is <class 'int'>

6. **Float Data Type Program :-**

```
x = 20.20
print("The Number", x, "And")
print("Data Type is", type(x))
```

Output :-

The Number 20.2 And

Data Type is <class 'float'>

7. **Complex Data Type Program :-**

```
x = 20 + 20j
print("The Number", x, "And")
print("Data Type is", type(x))
```

Output :-

The Number (20+20j) And

Data Type is <class 'complex'>

8. **String Data Type Program :-**

```
x = "Akshay"

print("The String", x, "And")

print("Data Type is", type(x))
```

Output :-

```
The String Akshay And

Data Type is <class 'str'>
```

9. **Boolean Data Type Program :-**

```
x = True

print( x, "And")

print("Data Type is", type(x))
```

Output :-

```
True And

Data Type is <class 'bool'>
```

10. **Set Data Type Program :-**

```
x = {"apple", "banana", "cherry"}

print("The Set", x, "And")

print("Data Type is", type(x))
```

Output :-

```
The Set {'apple', 'cherry', 'banana'} And

Data Type is <class 'set'>
```

11. **Frozen set Data Type Program :-**

```
x = frozenset({"apple", "banana", "cherry"})

print("The Frozen Set", x, "And")

print("Data Type is", type(x))
```

Output :-

The Frozen Set frozenset({'banana', 'apple', 'cherry'}) And

Data Type is <class 'frozenset'>

12. **List Data Type Program :-**

```
x = ["apple", "banana", "cherry"]
print("The List", x, "And")
print("Data Type is", type(x))
```

Output :-

The List ['apple', 'banana', 'cherry'] And

Data Type is <class 'list'>

13. **Range Data Type Program :-**

```
x = range(6)
print("The Range", x, "And")
print("Data Type is", type(x))
```

Output :-

The Range range(0, 6) And

Data Type is <class 'range'>

14. **Tuple Data Type Program :-**

```
x = ("apple", "banana", "cherry")
print("The Tuple", x, "And")
print("Data Type is", type(x))
```

Output :-

The Tuple('apple', 'banana', 'cherry') And

Data Type is <class 'tuple'>

- **Flow Control Statements Programs :-**

1. **Sequential Program :-**

```
print("Enter the First Number")
Num1=int(input())
print("Enter the Second Number")
Num2=int(input())
print("Addition Of Two Numbers is ",(Num1 + Num2 ) )
```

Output :-

```
Enter the First Number
2
Enter the Second Number
2
Addition Of Two Numbers is  4
```

2. **Simple If Program :-**

```
print("Enter the First Number")
Num1=int(input())
if(Num1%2==0):
    print("Given Number Is Even Number")
```

Output :-

```
Enter the First Number
10
Given Number Is Even Number
Enter the First Number
9
```

3. **If…else Program :-**

```
print("Enter the First Number")
Num1=int(input())
if(Num1%2==0):
```

```
    print("Given Number Is Even Number")
else:
    print("Given Number Is Odd Number")
```

Output :-

Enter the First Number

10

Given Number Is Even Number

Enter the First Number

9

Given Number Is Odd Number

4. **Switch Program :-**

```
def numbers_to_strings(argument):
        switcher = {
                0: "Zero",
                1: "One",
                2: "Two",
        3: "Three",
        4: "Four",
        5: "Five",
        }
        return switcher.get(argument, "nothing")
if __name__ == "__main__":
        argument=int(input("Enter The Number "))
        print (numbers_to_strings(argument))
```

Output :-

Enter The Number 4

Four

5. **While Loop Program :-**

```
print("Enter the Number")
A = int(input())
count=0
while (count < A):
    count = count + 1
    print("Python Programming")
else:
    print("In Else Block")
```

Output :-

```
Enter the Number
2
Python Programming
Python Programming
In Else Block
```

6. **Do…while Program :-**

```
number = 1
while number <= 5:
    print(f"Number is {number}!")
    number = number + 1
```

Output :-

```
Number is 1!
Number is 2!
Number is 3!
Number is 4!
Number is 5!
```

7. **For Loop Program :-**

```
print("Enter the First Number")
n=int(input(""))
for i in range(0, n):
    print("\t",i)
```

Output :-

```
Enter the First Number
5
        0
        1
        2
        3
        4
```

Program :-

```
print("String Iteration\n")
s = "Akshay"
for i in s :
    print(i)
```

Output :-

```
String Iteration
A
k
s
h
a
y
```

- **Indentation / Block Program :-**

```
from math import sqrt

n = input("Maximum Number? ")

n = int(n)+1

for a in range(1,n):

    for b in range(a,n):

        c_square = a**2 + b**2

        c = int(sqrt(c_square))

        if ((c_square - c**2) == 0):

            print(a, b, c)
```

Output :-

```
Maximum Number? 12

3 4 5

5 12 13

6 8 10

9 12 15
```

Program :-

```
site = 'gfg'

if site == 'gfg':

        print('Logging on to System...')

else:

        print('retype the URL.')

print('All Set !')
```

Output :-

```
Logging on to System...

All Set !
```

Que 03 :- Programs for understanding functions, use of built in functions, user defined functions.

- **User Defined Function Program**

```
def addnumbers(x,y):
   sum = x + y
   return sum
num1 = int(input("Enter the First Number "))
num2 = int(input("Enter the Second Number "))
print("The sum is", addnumbers(num1, num2))
```

Output :-

Enter the First Number 10

Enter the Second Number 5

The sum is 15

- **Built-in Functions :-**

abs()	delattr()	hash()	memoryview()
set()	all()	dict()	help()
min()	setattr()	any()	dir()
hex()	next()	slice()	ascii()
divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()
staticmethod()	bool()	eval()	int()
open()	str()	breakpoint()	exec()
isinstance()	ord()	sum()	bytearray()
filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()
tuple()	callable()	format()	len()
property()	type()	chr()	frozenset()
list()	range()	classmethod()	vars()
getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()

__import__()	complex()	hasattr()	max()
round()			

1. **Python abs() Function Example :-**

```
integer = 20

print('Absolute value of 40 is:', abs(integer))
```

Output :-

```
Absolute value of 40 is: 20
```

2. **Python all() Function Example :-**

```
k = [1, 3, 4, 6]

print(all(k))

k = [0, False]

print(all(k))

k = [1, 3, 7, 0]

print(all(k))

k = [0, False, 5]

print(all(k))

k = []

print(all(k))
```

Output :-

```
True

False

False

False

True
```

3. **Python bin() Function Example :-**

```
x = 150

y = bin(x)
```

```
print(y)
```

Output :-

```
0b10010110
```

4. **Python bool() Function Example :-**

```
test1 = False
print(test1,'is',bool(test1))
test1 = 'Easy string'
print(test1,'is',bool(test1))
```

Output :-

```
False is False
Easy string is True
```

5. **Python callable() Function Example :-**

```
x = 150
print(callable(x))
```

Output :-

```
False
```

6. **Python compile() Function Example :-**

```
code_str = 'x=5\ny=10\nprint("sum =",x+y)'
code = compile(code_str, 'sum.py', 'exec')
print(type(code))
exec(code)
```

Output :-

```
<class 'code'>
sum = 15
```

7. **Python exec() Function Example :-**

```
x = 20
```

```
exec('print(x==20)')
exec('print(x+20)')
```

Output :-

```
True
40
```

8. **Python sum() Function Example :-**

```
s = sum([1, 2, 4], 10)
print("Sum = ",s)
```

Output :-

```
Sum =  17
```

9. **Python ascii() Function Example :-**

```
normalText = 'Python is interesting'
print(ascii(normalText))
otherText = 'Pythön is interesting'
print(ascii(otherText))
print('PytAkshayhon is interesting')
```

Output :-

```
'Python is interesting'
'Pyth\xf6n is interesting'
PytAkshayhon is interesting
```

10. **Python eval() Function Example :-**

```
x = 10
print(eval('x + 1'))
```

Output :-

```
11
```

11. **Python getattr() Function Example :-**

```
class Details:
    age = 22
    name = "Akshay"
details = Details()
print('The age is:', getattr(details, "age"))
print('The age is:', details.age)
```

Output :-

```
The age is: 22
The age is: 22
```

12. Python iter() Function Example :-

```
list = [1,2,3,4,5]
listIter = iter(list)
print(next(listIter))
print(next(listIter))
print(next(listIter))
print(next(listIter))
print(next(listIter))
```

Output :-

```
1
2
3
4
5
```

13. Python map() Function Example :-

```
def calculateAddition(n):
  return n+n
numbers = (1, 2, 3, 4)
```

```
result = map(calculateAddition, numbers)

print(result)

numbersAddition = set(result)

print(numbersAddition)
```

Output :-

```
<map object at 0x000001E80490DCD0>

{8, 2, 4, 6}
```

14. Python object() Function Example :-

```
python = object()

print(type(python))

print(dir(python))
```

Output :-

```
<class 'object'>

['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__',
'__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__']
```

15. Python divmod() Function Example :-

```
result = divmod(10,2)

print(result)
```

Output :-

```
(5, 0)
```

Que 04 :- Programs to use existing modules, packages and creating modules, packages.

Module Sum.py :-

```
def Addition(Num1, Num2):
    return Num1 + Num2
```

Module Multiplication.py :-

```
def Multiplication(Num1, Num2):
    return Num1 * Num2
```

Module ASMD.py :-

```
import Question4.Sum as s
import Question4.Multiplication as m
def Calculator():
    print("Press 1 For Addition\nPress 2 for Multiplication")
    c=int(input("Enter the Your Choice\n"))
    if c == 1:
        a=int(input("Enter the First Number\n"))
        b=int(input("Enter the Second Number\n"))
        print("Addition of Two Numbers = ",s.Addition(a,b))
    elif c == 2:
        a=int(input("Enter the First Number\n"))
        b=int(input("Enter the Second Number\n"))
        print("Multiplication of Two Numbers = ",m.Multiplication(a,b))
    else:
        print("Your Choice is Wrong\n")
```

Create Package __init__.py :-

Program.py

```
import Question4.ASMD as a

a.Calculator()
```

Output :-

```
Press 1 For Addition

Press 2 for Multiplication

Enter the Your Choice

2

Enter the First Number

10

Enter the Second Number

10

Multiplication of Two Numbers =  100
```

Que 05 :- Programs for implementations of all object-oriented concepts like class, method, inheritance, polymorphism etc. (Real life examples must be covered for the implementation of objectoriented concepts).

- **Inheritance Program :-**

```
class Dog:
    def __init__(self, name, age, friendliness):
        self.name=name
        self.age=age
        self.friendliness=friendliness
    def LikesWalks(self):
        return True
class Samoyed(Dog):
    def __init__(self, name, age, friendliness):
        super().__init__(name, age, friendliness)
class Poodle(Dog):
    def __init__(self, name, age, friendliness):
        super().__init__(name, age, friendliness)
class GoldenRetriever(Dog):
    def __init__(self, name, age, friendliness):
        super().__init__(name, age, friendliness)
Sammy = Samoyed('Sammy', 2, 10)
print(Sammy.name, Sammy.age, Sammy.friendliness)
print(Sammy.LikesWalks())
```

Output :-

```
Sammy 2 10
True
```

Polymorphism Program :-

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")
    def language(self):
        print("Hindi is the most widely spoken language of India.")
    def type(self):
        print("India is a developing country.")
class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")
    def language(self):
        print("English is the primary language of USA.")
    def type(self):
        print("USA is a developed country.")
obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
    country.type()
```

Output :-

```
New Delhi is the capital of India.
Hindi is the most widely spoken language of India.
India is a developing country.
Washington, D.C. is the capital of USA.
English is the primary language of USA.
USA is a developed country.
```

Que 06 :- Programs for parsing of data, validations like Password, email, URL, etc.

- **Validation For Email Id :-**

```
import re
regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
def check(email):
    if(re.fullmatch(regex, email)):
        print("Valid Email")
    else:
        print("Invalid Email")
if __name__ == '__main__':
    email =str(input("Enter the Email\n"))
    check(email)
    email = str(input("Enter the Email\n"))
    check(email)
```

Output :-

```
Enter the Email
Akshay.com
Invalid Email
Enter the Email
Akshay@gmail.com
Valid Email
```

- **Validation For Password :-**

```
import re
def main():
    Password =str(input("Enter the Password\n"))
    reg = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!#%*?&]{6,20}$"
```

```
    Pat = re.compile(reg)
    mat = re.search(Pat, Password)
    if mat:
        print("Password is Valid.")
    else:
        print("Password is Invalid !!")
if __name__ == '__main__':
    main()
    main()
```

Output :-

Enter the Password

Akshay

Password is Invalid !!

Enter the Password

Akshay@123

Password is Valid.

- **Validation For URL :-**

```
import re
def Find(string):
    regex = r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:'\".,<>?«»“”’’]))"
    url = re.findall(regex, string)
    return [x[0] for x in url]
string = str(input("Enter the URL\n")) #http://www.geeksforgeeks.org
print("URLS: ", Find(string))
```

Output :-

Enter the URL

http://www.geeksforgeeks.org

URLS: ['http://www.geeksforgeeks.org']

Output :-

Enter the URL

ahshsja

URLS: []

Que 07 :- Programs for Pattern finding should be covered.

Program :-

```
print("Print equilateral triangle Pyramid using stars ")

size = 7

m = (2 * size) - 2

for i in range(0, size):

    for j in range(0, m):

        print(end=" ")

    m = m - 1

    for j in range(0, i + 1):

        print("* ", end=" ")

    print(" ")
```

Output :-

```
Print equilateral triangle Pyramid using stars

          *

         * *

        * * *

       * * * *

      * * * * *

     * * * * * *

    * * * * * * *
```

Que 08 :- Programs covering all the aspects of Exception handling, user defined exception, Multithreading should be covered.

- **User Defined Exception :-**

```
class BalanaceException(Exception):
    pass
def CheckBalance():
    Money=20000
    Withdraw=int(input("Enter the Withdrawal Money\n"))
    try:
        Balance=Money-Withdraw
        if(Balance<=2000):
            raise BalanaceException("Insuffient Balance ! Please Try
Again")
        print("Remaing Balance is ",Balance)
    except BalanaceException as be:
        print(be)
CheckBalance()
```

Output :-

```
Enter the Withdrawal Money
155
Remaing Balance is  19845
```

Output :-

```
Enter the Withdrawal Money
19500
Insuffient Balance ! Please Try Again
```

- **Multithreading :-**

```
import threading
```

```
import os

def task1():

        print("Task 1 Assigned to Thread :- {}".format(threading.current_thread().name))

        print("ID of Process Running Task 1 :- {}".format(os.getpid()))

def task2():

        print("Task 2 Assigned to Thread :- {}".format(threading.current_thread().name))

        print("ID of Process Running Task 2 :- {}".format(os.getpid()))

if __name__ == "__main__":

        print("ID of Process Running Main Program :- {}".format(os.getpid()))

        print("Main Thread Name :- {}".format(threading.current_thread().name))

        t1 = threading.Thread(target=task1, name='T1')

        t2 = threading.Thread(target=task2, name='T2')

        t1.start()

        t2.start()

        t1.join()

        t2.join()
```

Output :-

```
ID of Process Running Main Program :- 5160

Main Thread Name :- MainThread

Task 1 Assigned to Thread :- T1

ID of Process Running Task 1 :- 5160

Task 2 Assigned to Thread :- T2

ID of Process Running Task 2 :- 5160
```

Que 09 :- Programs demonstrating the IO operations like reading from file, writing into file from different file types like data file, binary file, etc.

```
fo = open("Akshay.txt", "r+")

str = fo.read(50)

print("Read String is : ", str)

position = fo.tell()

print("Current file Position is : ", position)

position = fo.seek(0, 0)

str = fo.read(15)

print("Again Read String is : ", str)

fo.close()
```

Output :-

```
Read String is :  Akshay Mahadev Yadav

Current file Position is :  20

Again Read String is :  Akshay Mahadev
```

Program :-

```
fo = open("PatternQue7.py", "r+")

str = fo.read(200)

print("Read String is : ", str)

position = fo.tell()

print("Current file Position is : ", position)

position = fo.seek(0, 0)

str = fo.read(100)

print("Again Read String is : ", str)

fo.close()
```

Output :-

Read String is : print("Print equilateral triangle Pyramid using stars ")

size = 7

m = (2 * size) - 2

for i in range(0, size):

    for j in range(0, m):

        print(end=" ")

    m = m - 1

    for j in range(0, i + 1

Current file Position is : 207

Again Read String is : print("Print equilateral triangle Pyramid using stars ")

size = 7

m = (2 * size) - 2

for i in range(

Que 10 :- Programs to perform searching, adding, updating the content from the file.

```
from tkinter import*

from tkinter import ttk, messagebox

from PIL import Image,ImageTk

import pymysql

class Medicine:

    def __init__(self,root):

        self.root = root

        self.root.title("Medicine Store Window")

        self.root.geometry("1500x750+10+40")

        self.root.resizable(False, False)

        LBLTitle = Label(self.root, text="Programs to perform searching, adding, updating the content from the file", font=("Century", 25, "bold"), bd=8, relief=RIDGE,bg="white", fg="red", padx=2, pady=4)

        LBLTitle.pack(side=TOP, fill=X)

        LBLTitle1 = Label(self.root, text="Programs to Perform Searching, Adding, Updating The Content From the File",font=("Century", 16, "bold"), bd=8, relief=RIDGE, bg="white", fg="red", padx=2, pady=4)

        LBLTitle1.pack(side=BOTTOM, fill=X)

        MainFrame = Frame(self.root, bd=8, relief=RIDGE, bg="white", padx=20, width=550, height=635)

        MainFrame.place(x=0, y=65)

        self.MedicineIdVar=StringVar()

        LabelMedicineId = Label(MainFrame, text="Medicine Id :", font=("Century", 15, "bold"), fg="Red",bg="white").place(x=5, y=10)

        txtMedicineId = Entry(MainFrame, textvariable=self.MedicineIdVar, font=("Century", 15), bg="lightgrey")

        txtMedicineId.place(x=5, y=45, width=490, height=35)
```

```
        self.MedicineNameVar=StringVar()

        LabelMedicineName = Label(MainFrame, text="Medicine Name :",
font=("Century", 15, "bold"), fg="Red",bg="white").place(x=5, y=90)

        txtMedicineName = Entry(MainFrame,
textvariable=self.MedicineNameVar, font=("Century", 15), bg="lightgrey")

        txtMedicineName.place(x=5, y=125, height=35, width=490)

        self.MedicineCompanyVar=StringVar()

        LabelMedicineCompay = Label(MainFrame, text="Medicine Company :",
font=("Century", 15, "bold"), fg="Red",bg="white").place(x=5, y=170)

        txtMedicineCompany = Entry(MainFrame,
textvariable=self.MedicineCompanyVar, font=("Century", 15), bg="lightgrey")

        txtMedicineCompany.place(x=5, y=205, height=35, width=490)

        self.QuatitiesVar=StringVar()

        LabelQuanities = Label(MainFrame, text="Medicine Quantity :",
font=("Century", 15, "bold"), fg="Red",bg="white").place(x=5, y=250)

        txtQuanities = Entry(MainFrame, textvariable=self.QuatitiesVar,
font=("Century", 15), bg="lightgrey")

        txtQuanities.place(x=5, y=285, width=490, height=35)

        self.PriceVar=StringVar()

        LabelPrice = Label(MainFrame, text="Medicine Price :", font=("Century",
15, "bold"), fg="Red",bg="white").place(x=5, y=330)

        txtPrice = Entry(MainFrame, textvariable=self.PriceVar, font=("Century",
15), bg="lightgrey")

        txtPrice.place(x=5, y=365, height=35, width=490)

        self.TotalPriceVar=StringVar()

        LabelTotalPrice = Label(MainFrame, text="Medicine Total Price :",
font=("Century", 15, "bold"), fg="Red",bg="white").place(x=5, y=410)

        txtTotalPrice = Entry(MainFrame, textvariable=self.TotalPriceVar,
font=("Century", 15), bg="lightgrey")

        txtTotalPrice.place(x=5, y=445, height=35, width=490)
```

```
        self.SearchCombo = ttk.Combobox(MainFrame, font=("Century", 15),
state='readonly', justify=CENTER)

        self.SearchCombo['values'] = ("Search By", "MedicineId", "MedicineName",
"MedicineCompany")

        self.SearchCombo.place(x=5, y=495, height=37, width=220)

        self.SearchCombo.current(0)

        self.SearchText=StringVar()

        txtSearch = Entry(MainFrame, textvariable=self.SearchText, bd=2,
relief=RIDGE, width=20, font=("Century", 17))

        txtSearch.place(x=227, y=495, height=37)

        btnSave = Button(MainFrame, command=self.SaveMedicine,
text="SAVE", font=("Century", 16, "bold"), width=15,
bg="green",fg="white").place(x=5, y=555, height=45, width=150)

        btnUpdate = Button(MainFrame, command=self.UpdateMedicine,
text="UPDATE", font=("Century", 16, "bold"), width=15,
bg="lime",fg="white").place(x=177, y=555, height=45, width=150)

        btnSearch = Button(MainFrame, command=self.SearchData,
text="SEARCH", font=("Century", 16, "bold"), width=15,
bg="lime",fg="white").place(x=347, y=555, height=45, width=150)

        MainFrame1 = Frame(self.root, bd=8, relief=RIDGE, bg="white")

        MainFrame1.place(x=555, y=65, width=945, height=635)

        TableFrame = Frame(MainFrame1, bd=8, relief=RIDGE, bg="white")

        TableFrame.place(x=0, y=0, width=930, height=620)

        ScrollX = ttk.Scrollbar(TableFrame, orient=HORIZONTAL)

        ScrollX.pack(side=BOTTOM, fill=X)

        ScrollY = ttk.Scrollbar(TableFrame, orient=VERTICAL)

        ScrollY.pack(side=RIGHT, fill=Y)

        self.MedicineTable = ttk.Treeview(TableFrame, column=("Medicine Id",
"Medicine Name", "Medicine Company", "Quantities", "Price", "Total Price"),
xscrollcommand=ScrollX.set,yscrollcommand=ScrollY.set)
```

```
        ScrollX.config(command=self.MedicineTable.xview)
        ScrollY.config(command=self.MedicineTable.yview)
        self.MedicineTable.heading("Medicine Id", text="Medicine Id")
        self.MedicineTable.heading("Medicine Name", text="Medicine Name")
        self.MedicineTable.heading("Medicine Company", text="Medicine
Company")
        self.MedicineTable.heading("Quantities", text="Gender")
        self.MedicineTable.heading("Price", text="Price")
        self.MedicineTable.heading("Total Price", text="Total Price")
        self.MedicineTable["show"] = "headings"
        self.MedicineTable.pack(fill=BOTH, expand=1)
        self.MedicineTable.bind("<ButtonRelease-1>", self.GetCursor)
        self.FatchDataMedicine()
        self.MedicineTable.column("Medicine Id", width=100)
        self.MedicineTable.column("Medicine Name", width=180)
        self.MedicineTable.column("Quantities", width=100)
        self.MedicineTable.column("Price", width=100)
        self.MedicineTable.column("Total Price", width=100)
    #---- Save Button Code ----
    def SaveMedicine(self):
        con = pymysql.connect(host="localhost", user="root", password="",
database="Medicine")
        cur = con.cursor()
        cur.execute("insert into medicinestore(MedicineId, MedicineName,
MedicineCompany, Quantities, Price, TotalPrice) Values(%s, %s, %s, %s, %s,
%s)",
                    (
                        self.MedicineIdVar.get(),
```

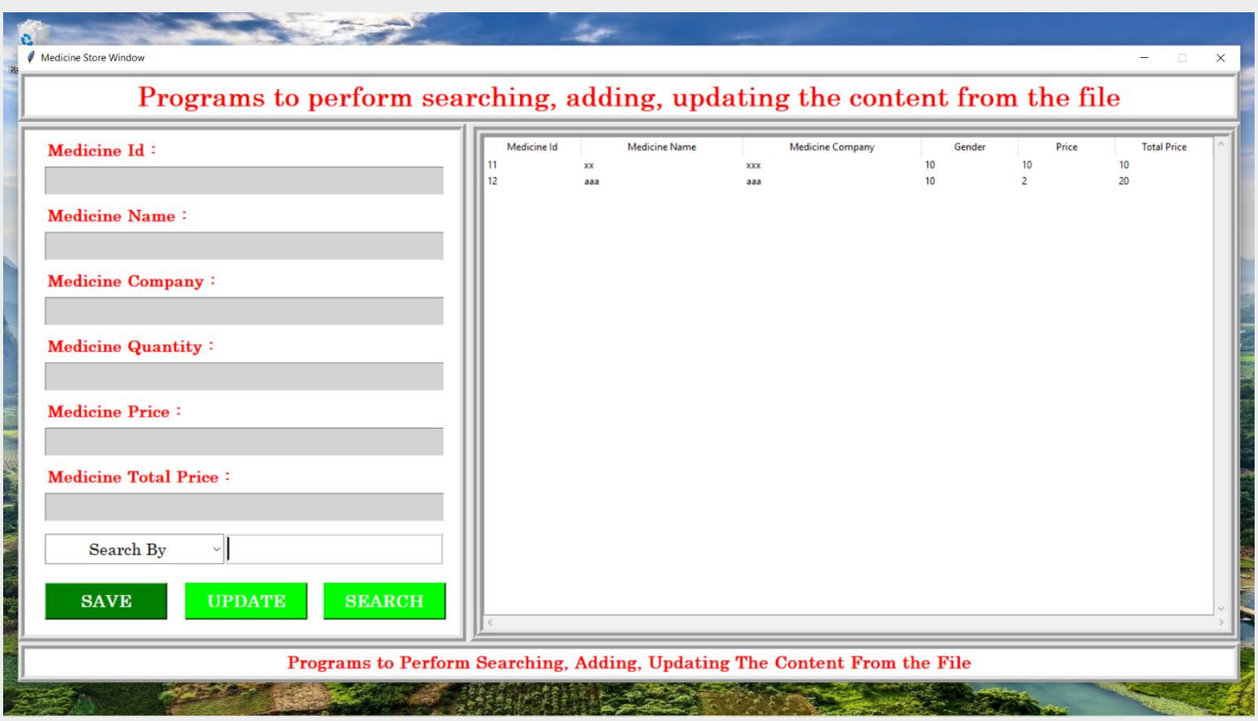
```
                self.MedicineNameVar.get(),
                self.MedicineCompanyVar.get(),
                self.QuatitiesVar.get(),
                self.PriceVar.get(),
                self.TotalPriceVar.get()
            ))
        con.commit()
        self.FatchDataMedicine()
        con.close()
        messagebox.showinfo("Medicine Store", "Medicine Save Successfully")
    def FatchDataMedicine(self):
        con = pymysql.connect(host="localhost", user="root", password="",
database="Medicine")
        cur = con.cursor()
        cur.execute("select * from medicinestore")
        Row = cur.fetchall()
        if len(Row) != 0:
            self.MedicineTable.delete(*self.MedicineTable.get_children())
            for i in Row:
                self.MedicineTable.insert("", END, values=i)
            con.commit()
        con.close()
    def GetCursor(self, Event=""):
        CursorRow = self.MedicineTable.focus()
        Content = self.MedicineTable.item(CursorRow)
        Row = Content['values']
        self.MedicineIdVar.set(Row[0])
```

```
        self.MedicineNameVar.set(Row[1])
        self.MedicineCompanyVar.set(Row[2])
        self.QuatitiesVar.set(Row[3])
        self.PriceVar.set(Row[4])
        self.TotalPriceVar.set(Row[5])
    def UpdateMedicine(self):
        con = pymysql.connect(host="localhost", user="root", password="",
database="Medicine")
        cur = con.cursor()
        cur.execute(
            "update medicinestore set MedicineName=%s, MedicineCompany=%s,
Quantities=%s, Price=%s, TotalPrice=%s where MedicineId=%s",
            (
                self.MedicineNameVar.get(),
                self.MedicineCompanyVar.get(),
                self.QuatitiesVar.get(),
                self.PriceVar.get(),
                self.TotalPriceVar.get(),
                self.MedicineIdVar.get()
            ))
        con.commit()
        messagebox.showinfo("Medicine Store", "Medicine Updated Successfully")
        self.FatchDataMedicine()
        con.close()
    def SearchData(self):
        con = pymysql.connect(host="localhost", user="root", password="",
database="Medicine")
        cur = con.cursor()
```

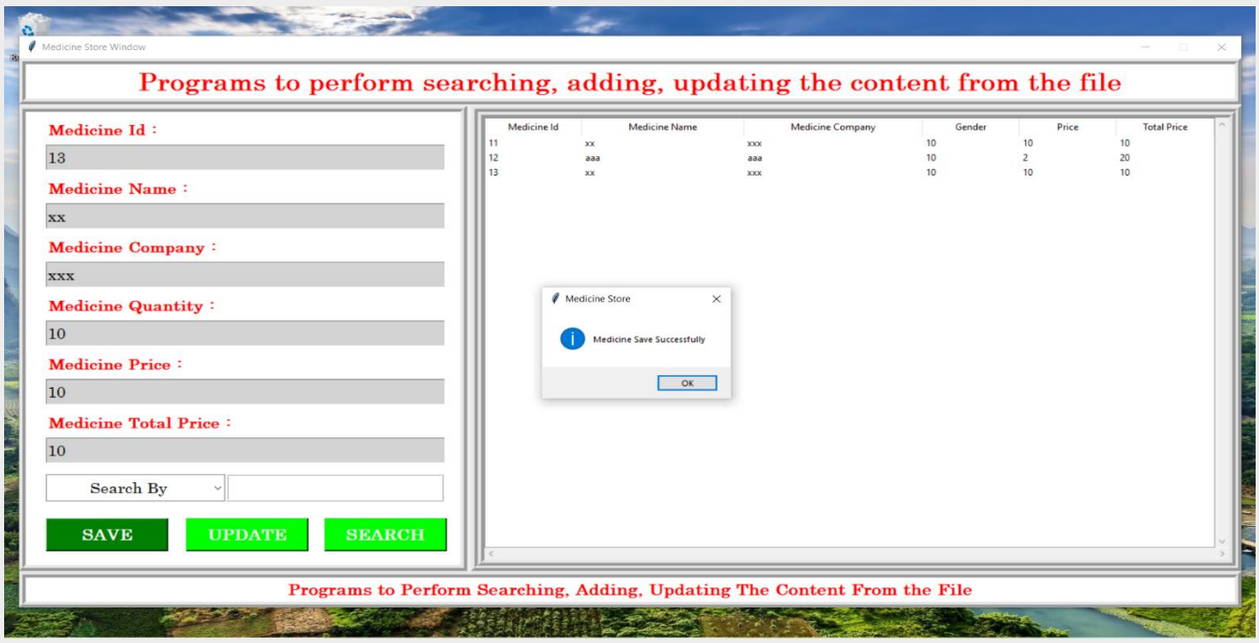
```
        cur.execute("select * from medicinestore where " +
str(self.SearchCombo.get()) + " LIKE '%" + str(self.SearchText.get()) + "%'")
        Row = cur.fetchall()
        if len(Row) != 0:
            self.MedicineTable.delete(*self.MedicineTable.get_children())
            for i in Row:
                self.MedicineTable.insert("", END, values=i)
        con.commit()
        con.close()
if __name__ =="__main__":
    root=Tk()
    obj=Medicine(root)
    root.mainloop()
```

Output :-

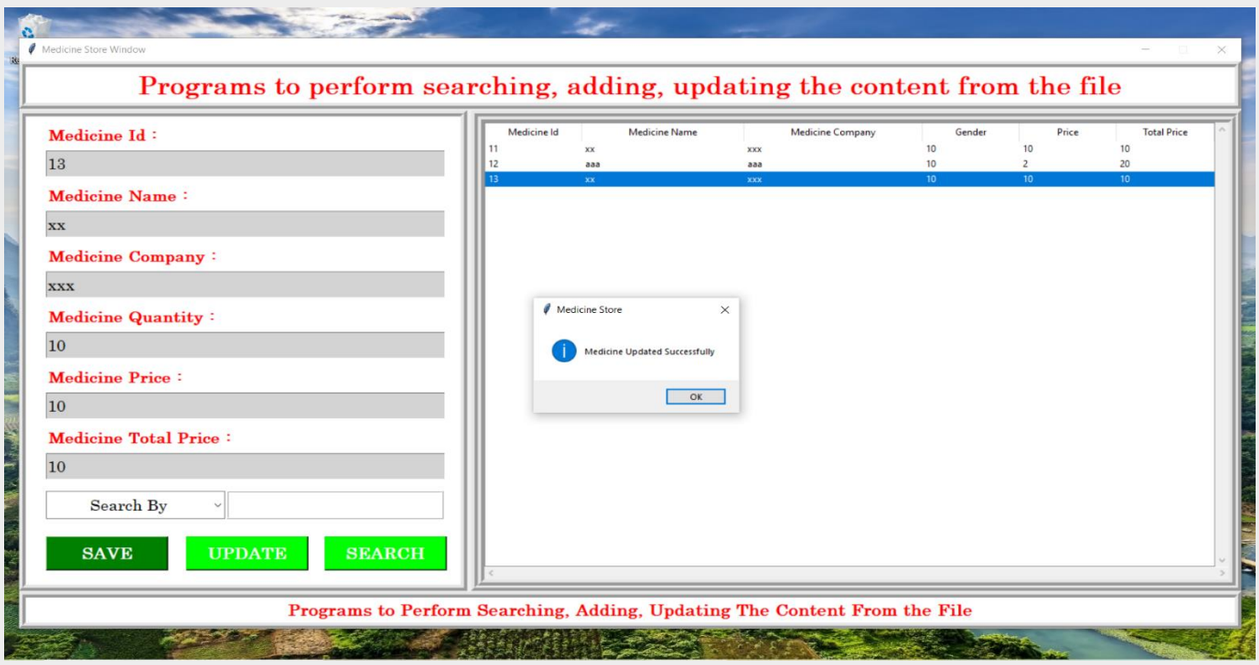
Form Design :-



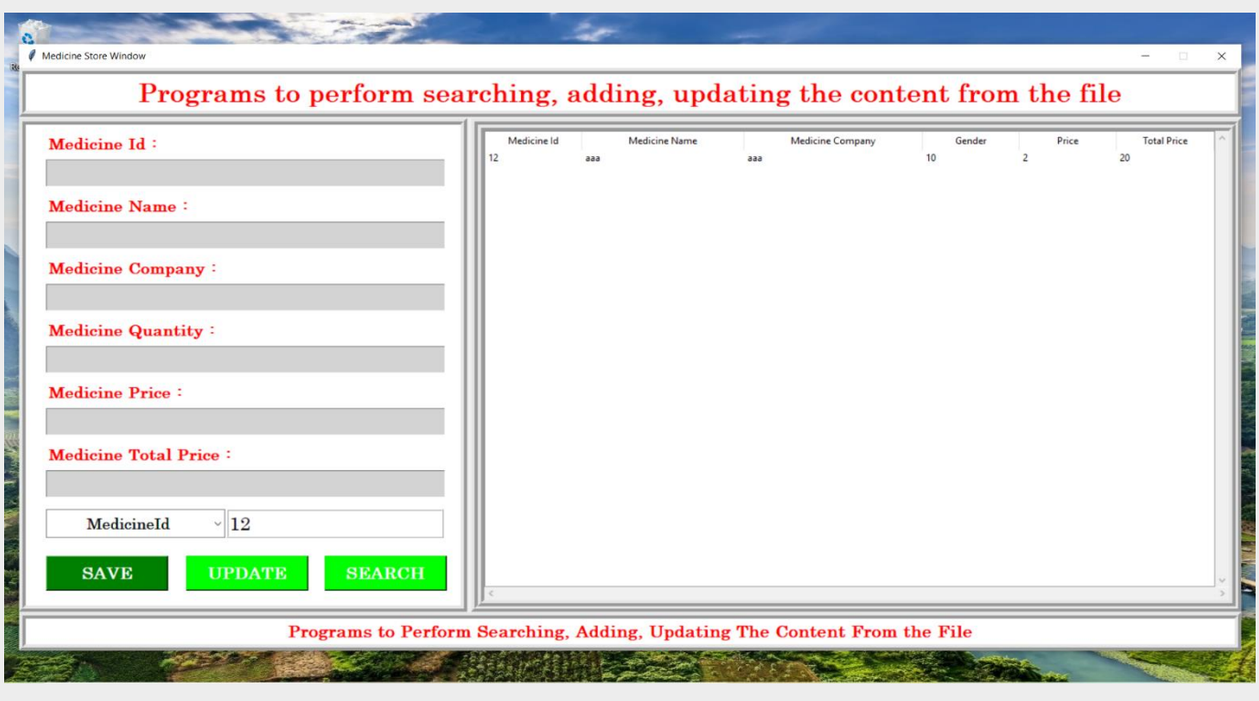
Save Operation :-



Update Operation :-



Search Operation :-



Que 11 :- Program for performing CRUD operation with MongoDB and Python.

Que 12 :- Basic programs with NumPy as Array, Searching and Sorting, date & time and String handling.

Que 13 :- Programs for series and data frames should be covered.

Que 14 :- Programs to demonstrate data pre-processing and data handling with data frame.

Que 15 :- Program for data visualization should be covered.