

INDEX

Sl. No.	Exercise title	Date of performance	Page number	Sign
1	Programs demonstrating use of Data types, Operators.	23/9/2021	5	
2	Programs demonstrating use of Control structures.	28/9/2021	55	
3	Programs demonstrating use of Lists and Tuples.	1/10/2021	71	
4	Programs demonstrating use of Sets and Dictionaries.	26/9/2021	1141	
5	Programs demonstrating implementation of Functions.	15/10/2021	55	
6	Programs demonstrating use of modules.	20/09/2021	163	
7	Programs demonstrating File Handling and I/O system.	16/10/2021	166	
8	Programs demonstrating use of Randomization in Python.	17/10/2021	175	
9	Programs demonstrating usage of regular expressions and biological problem solving.	17/10/2021	180	
10	Programs demonstrating Object Oriented Programming in Python.	17/10/2021	188	

Sl. No.	Exercise title	Date of performance	Page number	Sign
11	Programs demonstrating usage of CGI Python.	17/10/2021	198	
12	Programs demonstrating usage of DBI Python.	17/10/2021	207	
13	Programs demonstrating usage of BIOPYTHON Modules.	12/10/2021	208	

Python for Bioinformatics

Exercise 1

I-Title: Programs demonstrating use of Data types, Operators

1. W.a.p for demonstrating the usage of built-in number functions in Python

Answer: Some of the Built-in Number Functions in python are-

Algorithm:-

Step1→ start

Step2→ usage of built-in number functions in Python

Step3→stop

- i. max = The largest of its arguments: the value closest to positive infinity

```
max(174,55,22,456,432,563)
456
```

- ii. min = The smallest of its arguments: the value closest to negative infinity

```
min(174,55,22,456,432,563)
22
```

- iii. round = x rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0.

```
[a] IDLE Shell 3.9.5
File Edit Shell Debug Option Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> max(34,53,22,456,432,54)
456
>>> min(34,53,22,456,432,54)
22
>>> round(5.7)
6
>>>
```

```
[a] IDLE Shell 3.9.5
File Edit Shell Debug Option Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> max(34,53,22,456,432,54)
456
>>> min(34,53,22,456,432,54)
22
>>> round(5.7)
6
>>> round(5.3)
5
>>>
```

2. W.a.p for demonstrating the usage of escape characters in strings.

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Answer: To insert characters that are illegal in a string, use an escape character. An escape character is a backslash \ followed by the character you want to insert. So basically, placing a ‘\’ in front of one of these special or reserved characters will tell Python to just treat it as text, ensuring your code is valid and behaves as you’d expect.

Code	Result
\'	Single Quote
\\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

i. Code = \n

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output: \n is used to print word on next line.

The screenshot shows a Windows Command Prompt window titled "Windows PowerShell". The command typed is "python C:\Users\BID1900\Desktop\newline.py". The output shows the text "Hello python" on the first line and "world" on the second line, separated by a new line character (\n).

```
PS C:\Users\BID1900> python C:\Users\BID1900\Desktop\newline.py
Hello python
world
```

ii. Code = \\

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output: \\ is used to print a single backslash.

The screenshot shows the Python IDLE Shell 3.9.5 window. The code printed is:

```
print ("\\hello python\\world")
```

The output in the shell is:

```
>>> hello python
world
>>>
```

The window title is "IDLE Shell 3.9.5". The status bar at the bottom right shows "Ln 1 Col 4" and "Ln 1 Col 22".

iii. Code = \

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output:

The screenshot shows the Python IDLE Shell 3.9.5 window. The code printed is:

```
print ("\\"hello python\\")
```

The output in the shell is:

```
>>> hello python
world
>>>
```

The window title is "IDLE Shell 3.9.5". The status bar at the bottom right shows "Ln 1 Col 4" and "Ln 2 Col 0".

iv. Code = \t

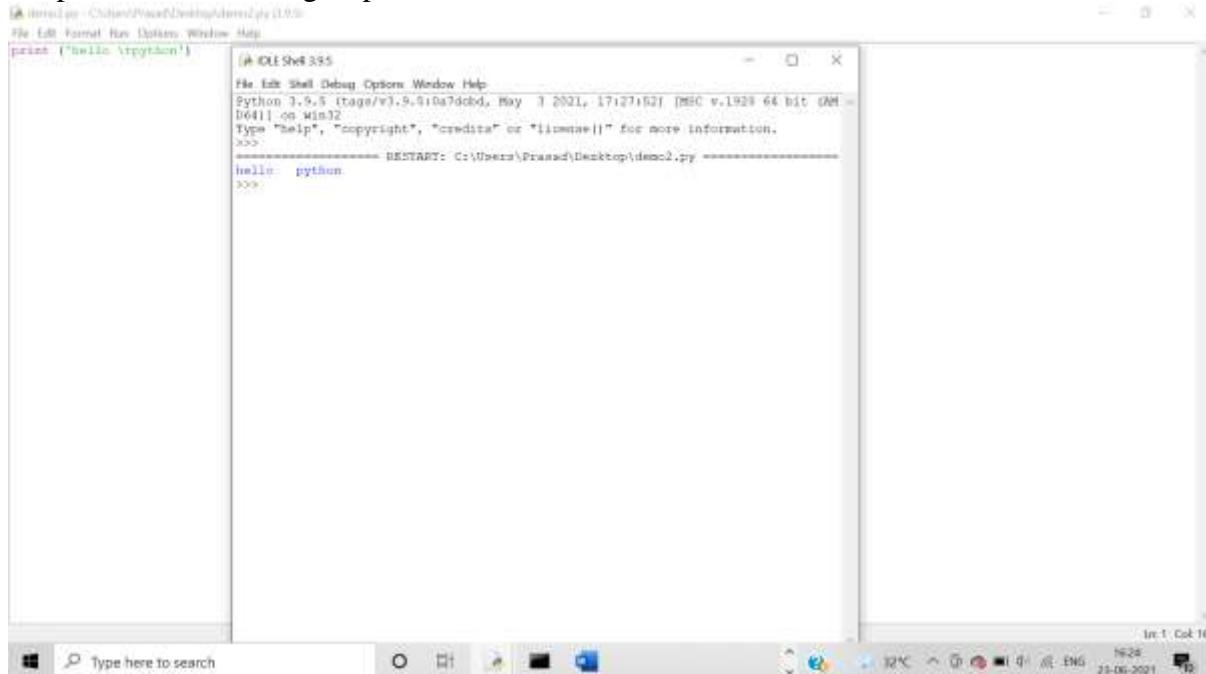
Algorithm:-

Step1→ start

Step2→ Using escape characters before special characters

Step3→ stop

Output: “\t” is used to get space between the words.



```
# C:\Users\Praasad\Desktop>python demo1.py
# IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbf, May  3 2021, 17:37:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praasad\Desktop\demo1.py
hello  python
>>>
```

v. Code = \b

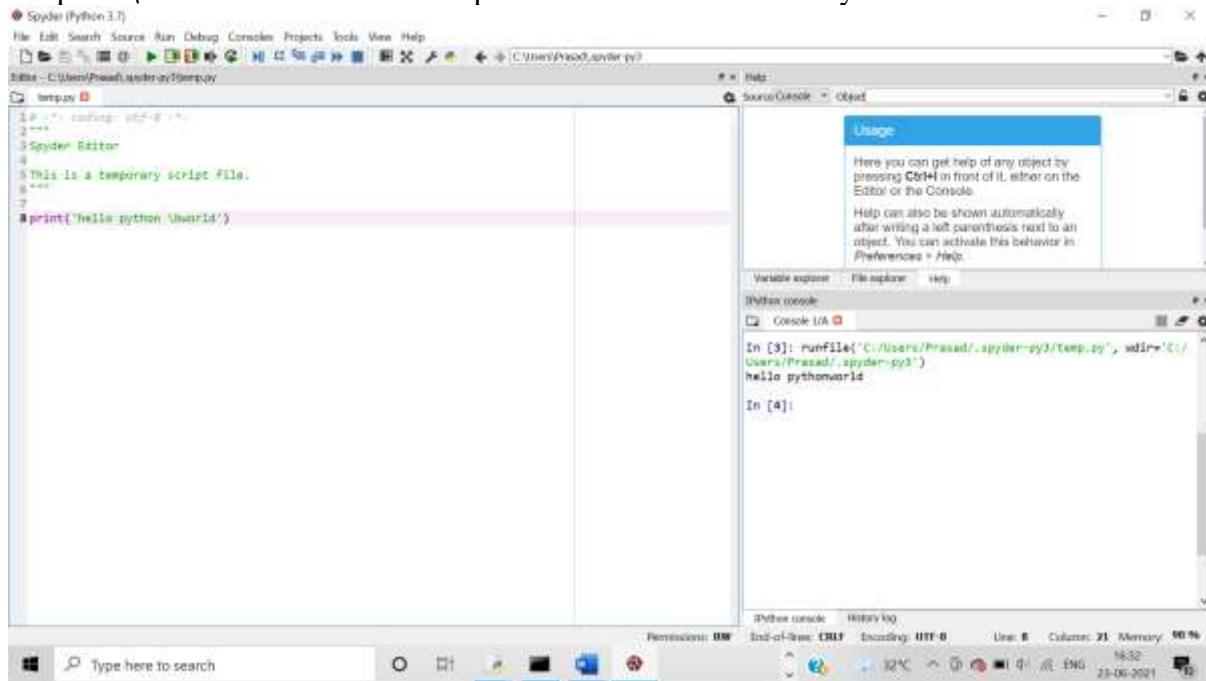
Algorithm:-

Step1→ start

Step2→ Using escape characters before special characters

Step3→ stop

Output: “\b” is used to remove the space between the words in Python.



```
Spyder (Python 3.7)
File Edit Search Source Run Debug Console Projects Tools View Help
C:\Users\Praasad\Desktop>spyder
temp.py
1> # coding: utf-8
2> #
3> # Spyder Editor
4> #
5> # This is a temporary script file.
6> #
7>
8> print('Hello python\bWorld')
Usage
Here you can get help of any object by
pressing Ctrl-H in front of it, either on the
Editor or the Console.
Help can also be shown automatically
after writing a left parenthesis next to an
object. You can activate this behavior in
Preferences > Help.

Variable explorer File explorer Help
Python console
In [3]: runfile('C:/Users/Praasad/.spyder-py3/temp.py', wdir='C:/Users/Praasad/.spyder-py3')
Hello pythonWorld

In [4]:
```

vi. Code = \r Carriage return

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output:

The screenshot shows the Spyder Python 3.7 IDE interface. The code editor window contains the following Python script:

```
1 #!/usr/bin/python3
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 print('Hello python world\rall')
```

The output console window shows the command runfile('C:/Users/Praash/.spyder-py3/temp.py', wdir='C:/Users/Praash/spyder-py3') and the resulting output:

```
In [4]: runfile('C:/Users/Praash/spyder-py3/temp.py', wdir='C:/Users/Praash/spyder-py3')
Hello
all
```

A tooltip titled "Usage" is visible in the top right corner of the code editor, explaining how to get help for objects.

vii. Code = \ooo Octal Values (\110\145\154\154\157)

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output:

The screenshot shows the Spyder Python 3.7 IDE interface. The code editor window contains the following Python script:

```
1 #!/usr/bin/python3
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 print('\110\145\154\154\157')
```

The output console window shows the command runfile('C:/Users/Praash/spyder-py3/temp.py', wdir='C:/Users/Praash/spyder-py3') and the resulting output:

```
In [5]: runfile('C:/Users/Praash/spyder-py3/temp.py', wdir='C:/Users/Praash/spyder-py3')
Hello
```

A tooltip titled "Usage" is visible in the top right corner of the code editor, explaining how to get help for objects.

viii. Code = \xhh Hex value (\x48\x65\x6c\x6c\x6f)

Algorithm:-

Step1→ start

Step2→Using escape characters before special characters

Step3→stop

Output:

The screenshot shows the Spyder Python 3.7 IDE interface. In the top-left corner, there's a status bar with the text "Windows 10". The main window has a title bar "Spyder (Python 3.7)". The menu bar includes File, Edit, Search, Source, Run, Debug, Console, Projects, Tools, View, Help. Below the menu is a toolbar with icons for file operations like Open, Save, Run, and Help. The central area contains a code editor with the following content:

```
1 #!/usr/bin/python3
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 print('\x48\x65\x6c\x6c\x6f')
```

To the right of the code editor is a "Usage" help panel with instructions on how to get help for objects. Below the code editor is a "Python console" tab with the following output:

```
In [8]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
Hello

In [7]:
```

At the bottom of the screen, there's a taskbar with a search bar containing "Type here to search" and several pinned application icons. The system tray shows the date and time as "23-06-2021".

3. W.a.p for demonstrating the usage of string function and string operators

Algorithm

Step1→ start

Step2→ initialize a=string,b=string

Step3→ print(a+b),print(a*b),print(a[2]),print(b[1:3])

Step4→ print i,j,a

Step5→ stop

Output:-

```
# idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0.0)
File Edit Shell Options Window Help
# Python is a interpreted language
print(a+b)
print(a*b)
print(a[2])
print(b[1:3])
print(i,j,a)
print(file.readlines())
# a file named "myfile", will be opened with r
#file = open ("C:/Users/Prasad/Documents/myfile")
#This will print the content
#for each in file:
#    print(each)
#print ("-----")
#print (file.readlines())

#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
boygirl
boyboy
y
it
>>>

# a file named "myfile", will be opened with r
#file = open ("C:/Users/Prasad/Documents/myfile")
#This will print the content
#for each in file:
#    print(each)
#print ("-----")
#print (file.readlines())
#>>>
```

```
# idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0.0)
File Edit Shell Options Window Help
a = "boy"
b = "girl"

print(a+b) #concatenation
print(a*2) #repetition
print(a[1:3]) #slice
print(b[1:3]) #slice

#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
boygirl
boyboy
y
it
>>>

# a file named "myfile", will be opened with r
#file = open ("C:/Users/Prasad/Documents/myfile")
#This will print the content
#for each in file:
#    print(each)
#print ("-----")
#print (file.readlines())
#>>>
```

4. W.a.p for demonstrating the usage of format method

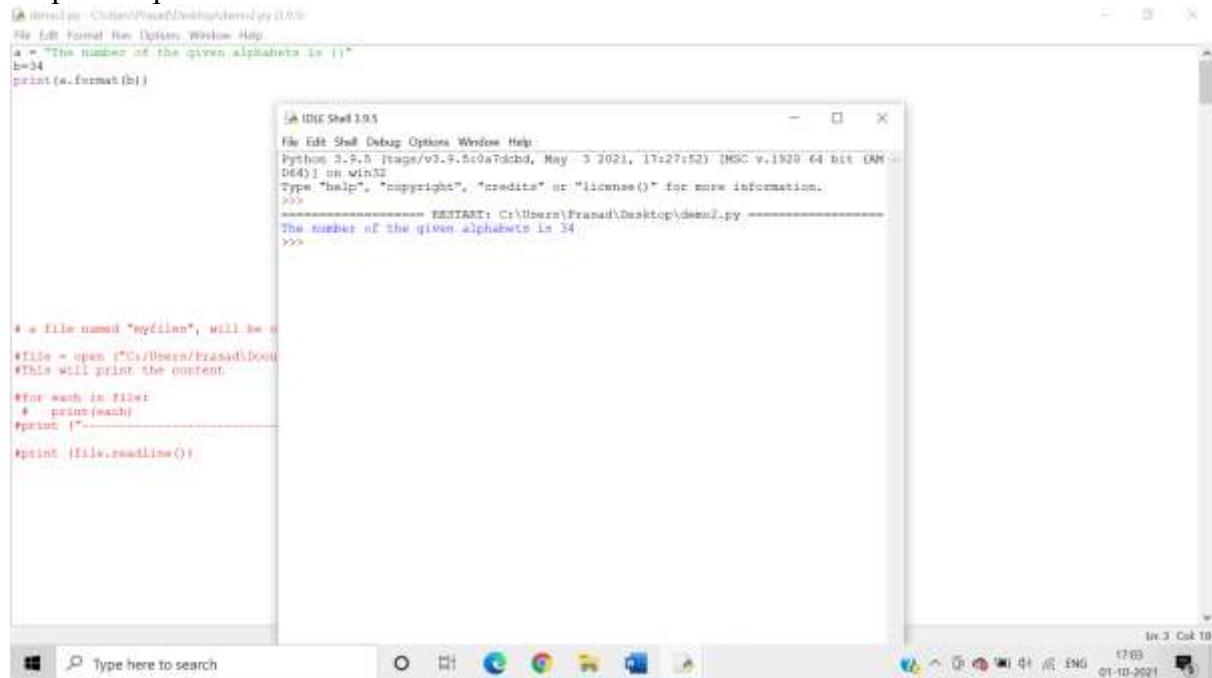
Algorithm

Step1→ start

Step2→ initialize a ="string{}",b=integer

Step3→ print(a.format(b))

Step4→ stop



The screenshot shows an IDE Shell window titled "IDE Shell 3.9.5" running Python 3.9.5. The code in the editor pane is:

```
a = "The number of the given alphabets is {}"
b=34
print(a.format(b))
```

The output pane shows the result of running the code:

```
The number of the given alphabets is 34
```

Below the IDE window, the Windows taskbar is visible with icons for Start, Task View, File Explorer, Edge, Google Chrome, File Manager, and Taskbar settings. The system tray shows the date as 01-10-2021 and the time as 17:03.

5. Write set of programs for demonstrating the usage of following String Methods along with its variants-

- i. capitalize(), lower(), upper(), title(), casefold(), swapcase()

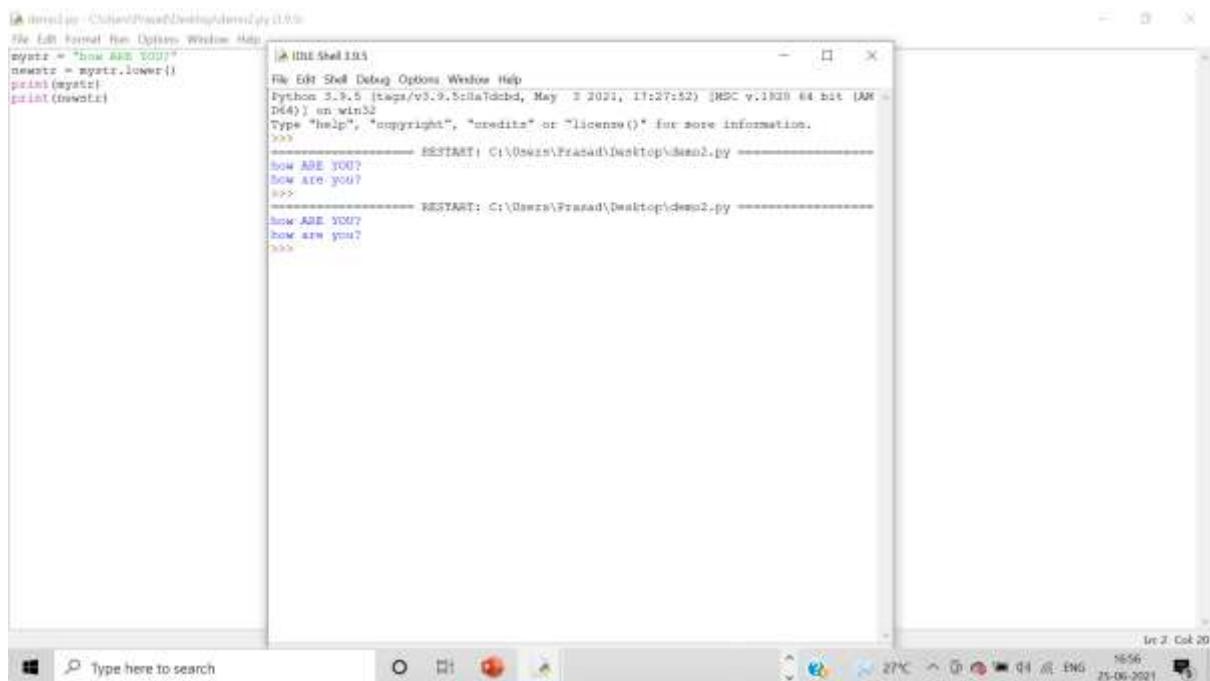
Algorithm:- Use the capitalize(), lower(), upper(), title(), casefold(), swapcase() string methods to alter and modify the given strings.

- capitalize()

The screenshot shows the Python IDLE Shell 3.9.5 interface. In the code editor pane, there is a single line of code: `mystr = "how are you?" newstr = mystr.capitalize(); print(mystr); print(newstr)`. When run, it outputs two lines: "how are you?" and "How are you?". The status bar at the bottom right indicates the current time as 16:53 and date as 25-06-2021.

lower()

The screenshot shows the Python IDLE Shell 3.9.5 interface. In the code editor pane, there is a single line of code: `mystr = "How ARE YOU?" newstr = mystr.casefold(); print(mystr); print(newstr)`. When run, it outputs two lines: "How ARE YOU?" and "how are you?". The status bar at the bottom right indicates the current time as 16:56 and date as 25-06-2021.



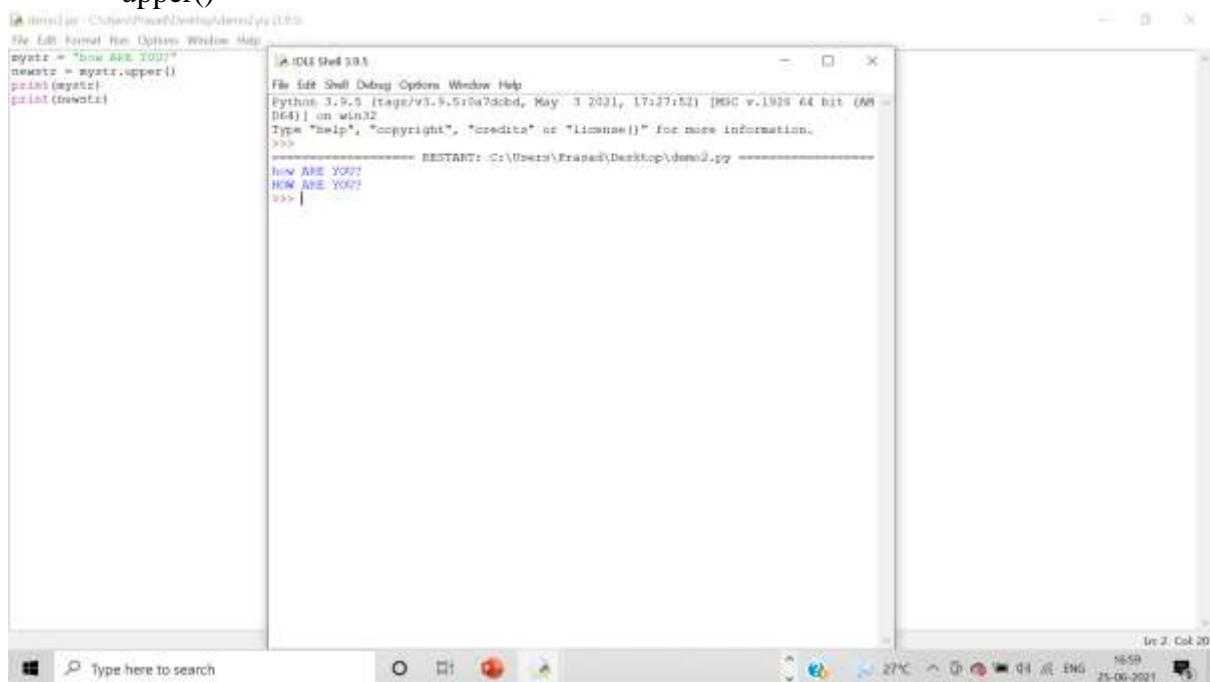
```
mystr = "How ARE YOU"
newstr = mystr.lower()
print(newstr)

How ARE YOU?
how are you?
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
```

```
mystr = "How ARE YOU";
newstr = mystr.upper();
print(newstr)

How ARE YOU?
HOW ARE YOU?
>>>
```

- **upper()**



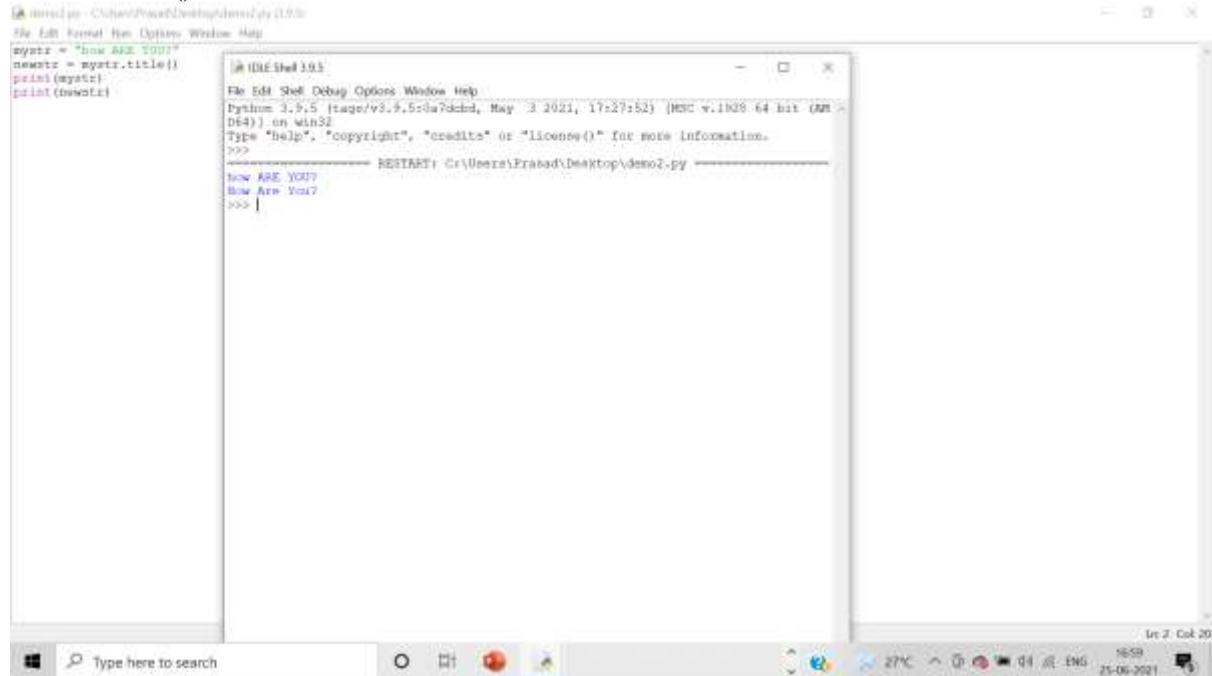
```
mystr = "How ARE YOU";
newstr = mystr.upper();
print(newstr)

HOW ARE YOU?
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
```

```
mystr = "How ARE YOU";
newstr = mystr.upper();
print(newstr)

HOW ARE YOU?
>>>
```

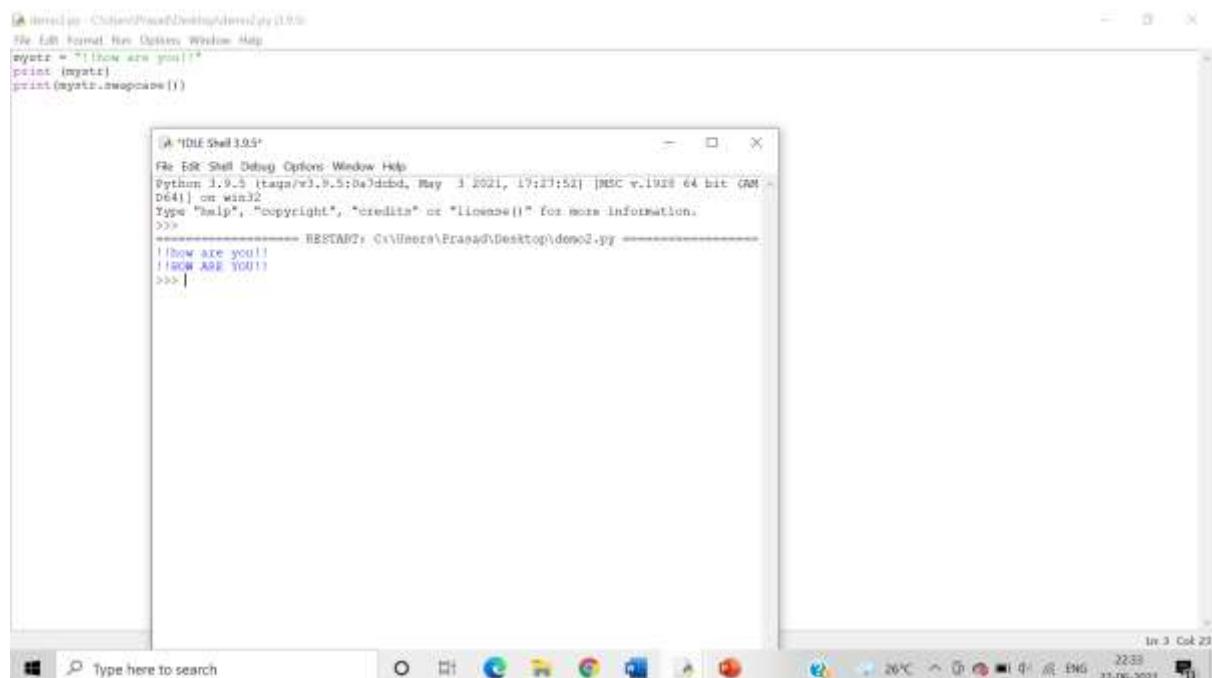
- title()



```
[idle] >>> mystr = "How ARE YOU?"
[idle] >>> newstr = mystr.title()
[idle] >>> print(newstr)
[idle] >>> print(newstr)
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
[idle] >>> How Are You?
[idle] >>>
```

- swapcase()

- uppercase



```
[idle] >>> mystr = "I!how are you!!"
[idle] >>> print(mystr)
[idle] >>> print(mystr.swapcase())
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
[idle] >>> !How are you!!
[idle] >>> !HOW ARE YOU!!
[idle] >>>
```

- lowercase

```

A:\> idle3 - C:\Users\Prasad\Desktop\demo2.py [0:0]
File Edit Shell Debug Options Window Help
mystr = "I HOW ARE YOU!!"
print(mystr)
print(mystr.swapcase())

```

A: IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb4, May 3 2021, 17:21:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo2.py ======
I how are you!!
I HOW ARE YOU!!
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo2.py ======
I HOW ARE YOU!!
I how are you!!
>>>

- mixed

```

A:\> idle3 - C:\Users\Prasad\Desktop\demo2.py [0:0]
File Edit Shell Debug Options Window Help
mystr = "I HOW are you!!"
print(mystr)
print(mystr.swapcase())

```

A: IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb4, May 3 2021, 17:21:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo2.py ======
I HOW are you!!
I how ARE You!!
>>>

ii. center(),ljust(), rjust(), zfill()

Algorithm:- Use the center(),ljust(), rjust(), zfill() string methods to alter and modify the given string.

- center()

```
[1] In [1]: mystr = "How are you?"  
mystr = mystr.center(10)  
print(mystr)  
  
[1] Out[1]: '        how are you?'
```

```
In [2]: RESTART: C:\Users\Prasad\Desktop\demo2.py  
how are you?  
In [3]:
```

- ljust()

```
[1] In [1]: mystr = "Hello python"  
print(mystr.ljust(7))  
  
[1] Out[1]: 'Hello python'
```

```
In [2]: RESTART: C:\Users\Prasad\Desktop\demo2.py  
Hello python  
In [3]:
```

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/1)
File Edit Shell Debug Options Window Help
mystr = "Hello python"
print(mystr.ljust(34, " "))

[1] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
Hello python*****
```

Type here to search 29°C 02:11 27-06-2021

- **rjust()**

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/1)
File Edit Shell Debug Options Window Help
mystr = "Hello python"
print(mystr)
print(mystr.rjust(35,""))

[1] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
Hello python
*****Hello python
>>> 
```

Type here to search 27°C 22:03 27-06-2021

- `zfill()`

```
mystr = "!!BIDM are you!!"
print(mystr)
print(mystr.zfill(25))
```

```
mystr = "!!BIDM are you!!"
print(mystr)
print(mystr.zfill(2))
```

```
mystr = "!!BIDM are you!!"
print(mystr)
print(mystr.zfill(25))
```

```
mystr = "!!BIDM are you!!"
print(mystr)
print(mystr.zfill(2))
```

iii. count(), replace()

Algorithm:- Use the count() and replace() method to alter and modify the given string.

- count()

```
[1] In [1]: mystr = "GangnamStyleisntitjustqueango"
mystr.count("abc")
print(mystr)
print(mycount)

[2] In [2]: RESTART: C:\Users\Prasad\Desktop\demo2.py
mystr.count("abc")
2
>>>
```

The screenshot shows a Windows desktop with two Python IDLE windows. The left window contains Python code: `mystr = "GangnamStyleisntitjustqueango"`, `mystr.count("abc")`, `print(mystr)`, and `print(mycount)`. The right window shows the output: `RESTART: C:\Users\Prasad\Desktop\demo2.py` followed by `mystr.count("abc")` and `2`.

- replace()

```
[1] In [1]: mystr = "Hello BID"
newstr = mystr.replace("BID", "teambeem")
print(mystr)
print(newstr)

[2] In [2]: RESTART: C:\Users\Prasad\Desktop\demo2.py
Hello BID
Hello teambeem
>>>
```

The screenshot shows a Windows desktop with two Python IDLE windows. The left window contains Python code: `mystr = "Hello BID"`, `newstr = mystr.replace("BID", "teambeem")`, `print(mystr)`, and `print(newstr)`. The right window shows the output: `RESTART: C:\Users\Prasad\Desktop\demo2.py` followed by `Hello BID` and `Hello teambeem`.

```
[&] C:\Users\Praaad\Desktop>demo2.py (1.0.0)
File Edit Shell Debug Options Window Help
mystr = "atgcatggcgtacgatacatacgatgcgt"
newstr = mystr.replace("t","u",2)
print(newstr)
print(newstr)

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praaad\Desktop\demo2.py -----
atgcatggcgtacgatacatacgatgcgt
atgcatggcgtacgatacatacgatgcgt
>>> |
```

iv. endswith(), startswith()

Algorithm:- Use the endswith(), startswith() methods to alter and modify the given strings.

- endswith()

```
[&] C:\Users\Praaad\Desktop>demo2.py (1.0.0)
File Edit Shell Debug Options Window Help
mystr = "atgcatggcgtacgatacatacgatgcgt"
print(mystr.endswith("est"))
print(mystr.endswith("est"))

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praaad\Desktop\demo2.py -----
atgcatggcgtacgatacatacgatgcgt
True
>>> |
```

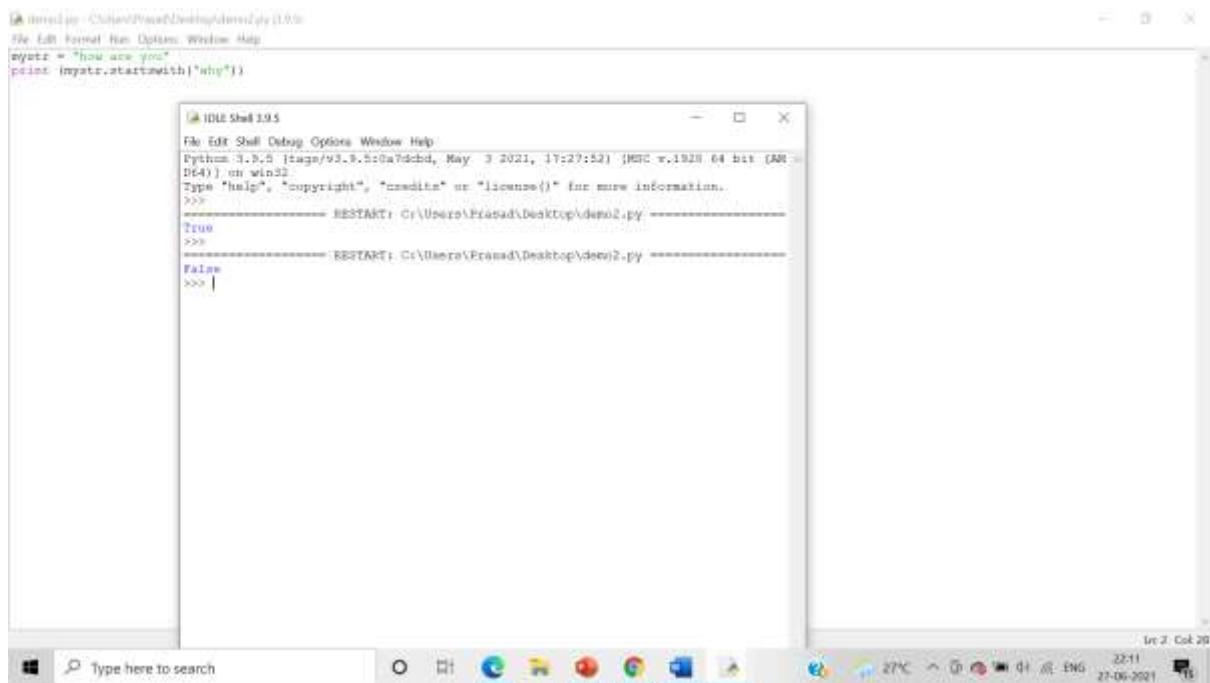
```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
mystr = "aaabbbcccaaaaabbbcccaaaa"
print(mystr)
print(mystr.endswith("ccc"))

idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May 3 2021, 17:27:52) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mystr = "aaabbbcccaaaaabbbcccaaaa"
>>> print(mystr.endswith("ccc"))
False
>>> mystr = "aaabbbcccaaaaabbbcccaaaa"
>>> print(mystr.endswith("ccc"))
True
>>>
```

startswith()

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
mystr = "How are you"
print(mystr.startswith("How"))

idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May 3 2021, 17:27:52) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mystr = "How are you"
>>> print(mystr.startswith("How"))
True
>>>
```

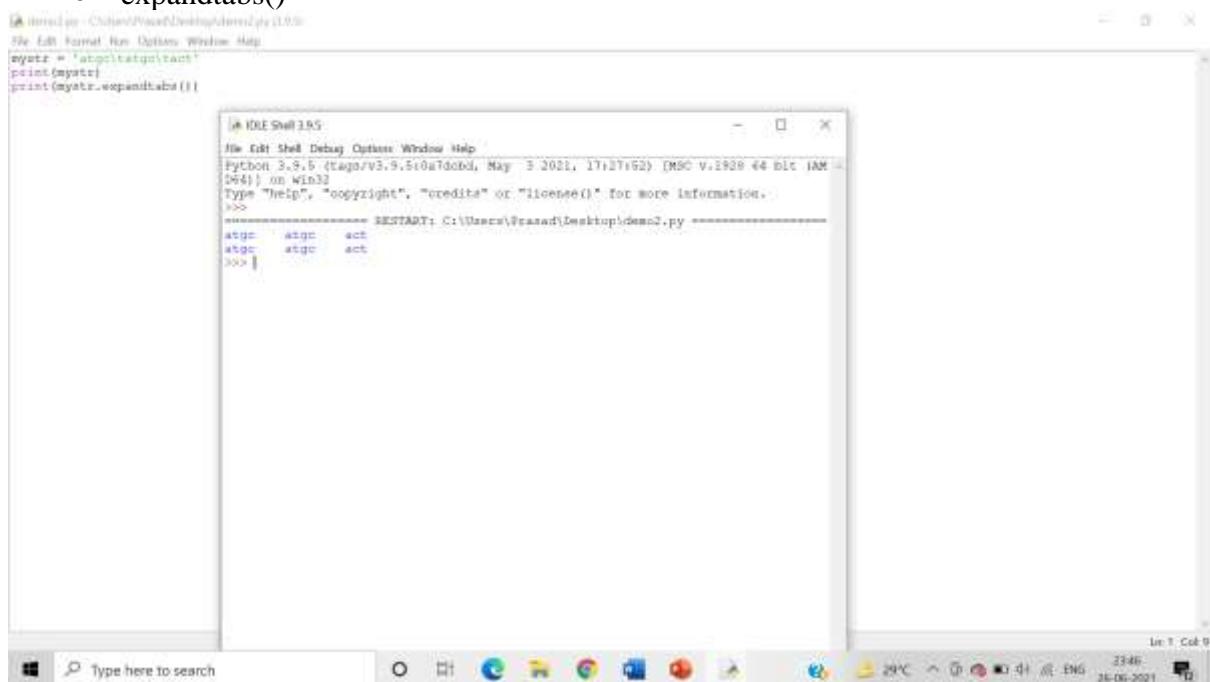


```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
mystr = "How are you"
print(mystr.startswith("why"))
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
True
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
False
>>>
```

v. expandtabs(), format()

Algorithm:- Use the expandtabs(), format() to alter and modify the given string.

- expandtabs()



```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
mystr = "    abc\tabc\tabc"
print(mystr.expandtabs())
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
abc  abc  abc
>>>
```

A screenshot of the Python IDLE shell window. The code in the editor pane is:

```
[1] mystr = "ab\c\tg\ta\tc\t"
[2] print(mystr)
[3] print(mystr.expandtabs(20))
```

The output in the shell pane is:

```
[4] RESTART: C:\Users\Prasad\Desktop\democ2.py
[5] ab      c\tg\ta\tc\t
[6] >>>
```

The status bar at the bottom shows "Line 4 Col 10".

vi. find(), rfind(), index(), rindex()

Algorithm:- Use the find(), rfind(), index(), rindex() methods to modify or alter the given string.

- find()

A screenshot of the Python IDLE shell window. The code in the editor pane is:

```
[1] mystr = "ab\c\tg\ta\tc\t"
[2] print(mystr.find("in"))
```

The output in the shell pane is:

```
[3] RESTART: C:\Users\Prasad\Desktop\democ2.py
[4] -1
[5] >>>
```

The status bar at the bottom shows "Line 3 Col 22".

```
[1] C:\Users\Prasad\Desktop>py demo1.py
File Edit Insert Run Options Window Help
mystr = "Temporarycopyrightattempstoguess"
print(mystr.rfind("int", 2, 9))
[2] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d9d8, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo1.py =====
atgetatgetat
5
>>> |
```

- **rfind()**

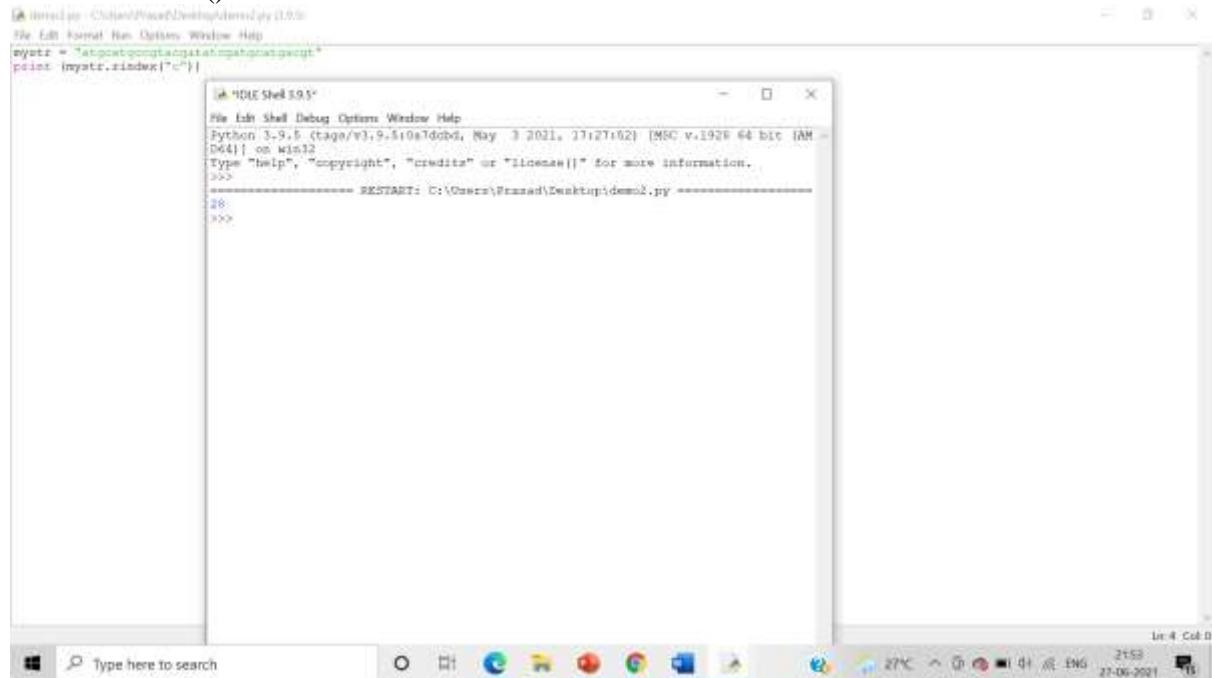
```
[1] C:\Users\Prasad\Desktop>py demo2.py
File Edit Insert Run Options Window Help
mystr = "Temporarycopyrightattempstoguess"
print(mystr.rfind("t"))
[2] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d9d8, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
30
>>> |
```

```
[1] m1=py - C:\Users\Prasad\Desktop\demo2.py (1.9.5)
File Edit Format Run Options Window Help
mystr = "atgotatgotatgotatgotatgotatgotat"
print(mystr.rfind("i"))
[2]                               IDE Shell 1.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----
30
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----
-1
>>> |
```

- index()

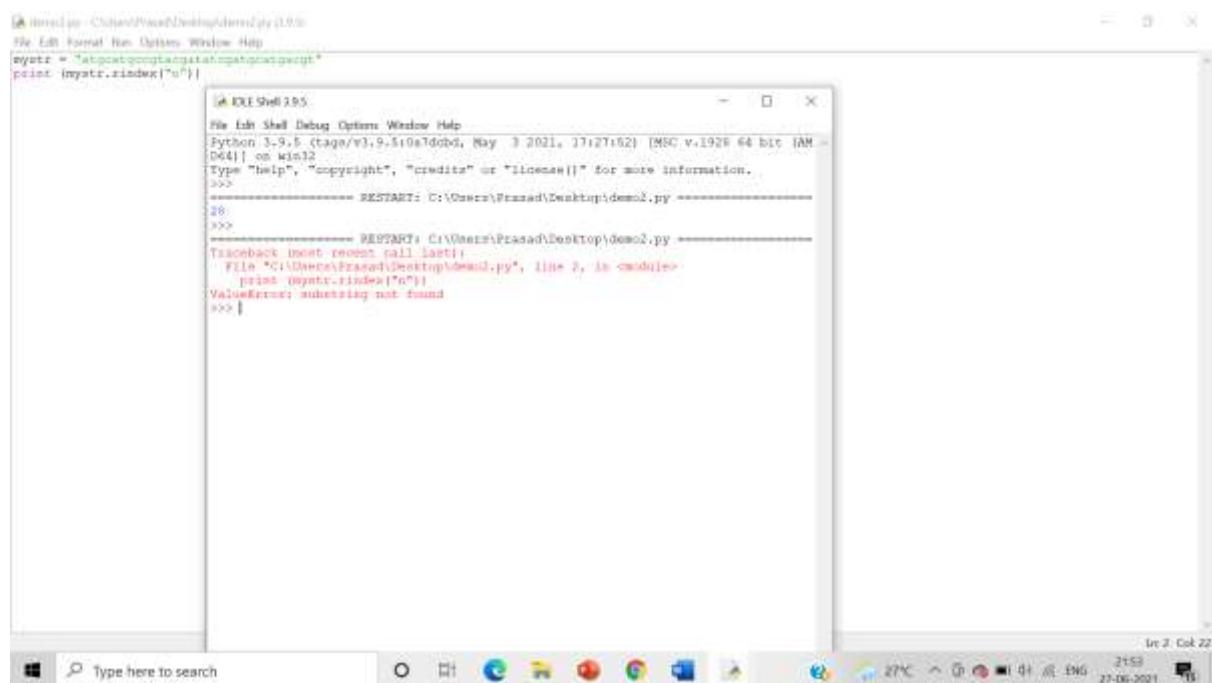
```
[1] m1=py - C:\Users\Prasad\Desktop\demo2.py (1.9.5)
File Edit Format Run Options Window Help
mystr = "atgotatgotat"
print(mystr.index("a",3,9))
[2]                               IDE Shell 1.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----
atgotatgotat
5
>>>
```

- `rindex()`



```
mystr = "Temptingguitarplayerguitar"
print(mystr.rindex("c"))
```

The screenshot shows the Python IDLE Shell 3.9.5 window. The code `mystr = "Temptingguitarplayerguitar"\nprint(mystr.rindex("c"))` is entered. The output shows the command prompt `>>>` followed by the message `RESTART: C:\Users\Prasad\Desktop\demo2.py` and then an empty line, indicating that the function call was successful without raising an error.



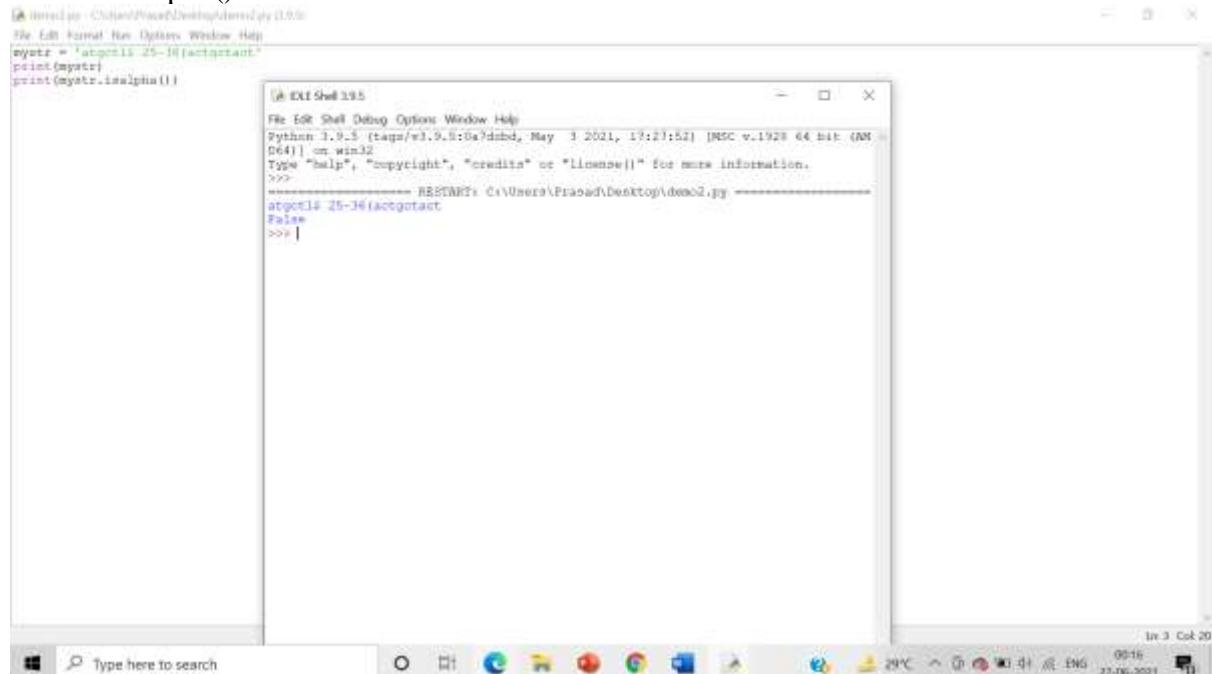
```
mystr = "Temptingguitarplayerguitar"
print(mystr.rindex("o"))
```

The screenshot shows the Python IDLE Shell 3.9.5 window. The code `mystr = "Temptingguitarplayerguitar"\nprint(mystr.rindex("o"))` is entered. The output shows the command prompt `>>>` followed by the message `RESTART: C:\Users\Prasad\Desktop\demo2.py` and then a traceback. The traceback indicates that the function call failed because the character 'o' was not found in the string, resulting in a `ValueError: substring not found`.

vii. isalnum(), isalpha(), isdigit(), isidentifier(), islower(), isnumeric(), isspace, isupper(), istitle()

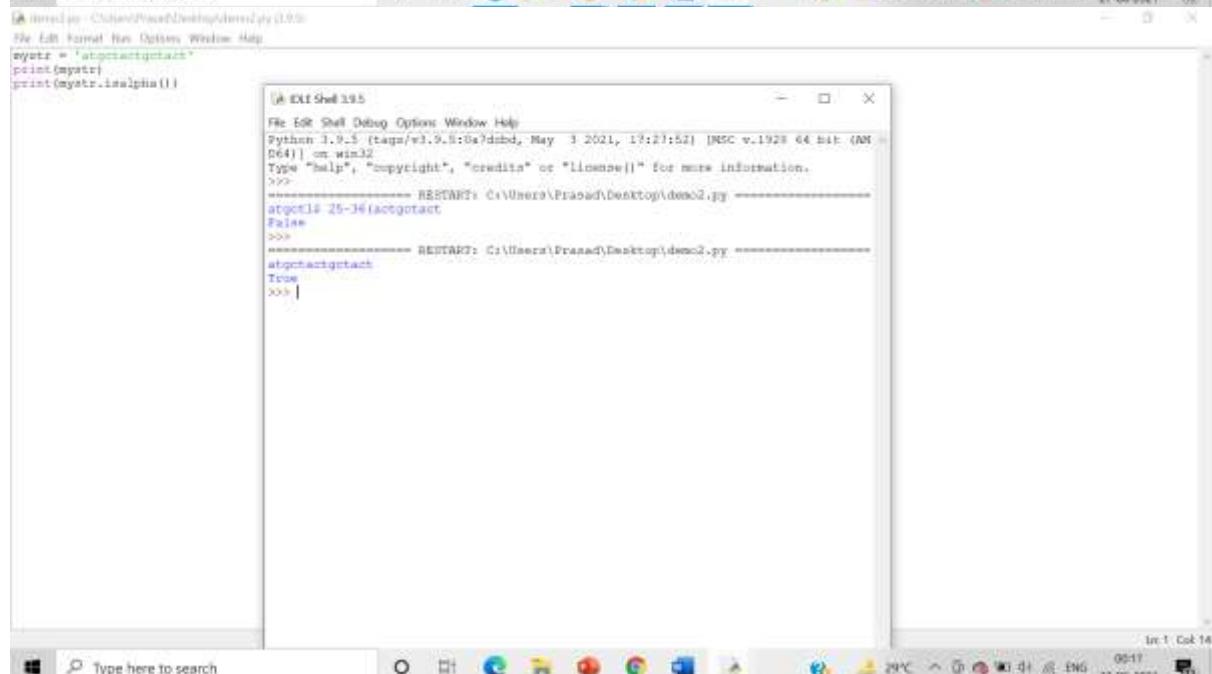
Algorithm: Use the . isalnum(), isalpha(), isdigit(), isidentifier(), islower(), isnumeric(), isspace, isupper(), istitle() string methods to check the class of the given string.

- isalpha()



```
[1] In [1]: C:\Users\Prasad\Desktop\demo2.py (1.0%)
File Edit Shell Debug Options Window Help
mystr = 'actgactgact'
print(mystr.isalpha())
>>> 
```

In 3 Col 20



```
[1] In [1]: C:\Users\Prasad\Desktop\demo2.py (1.0%)
File Edit Shell Debug Options Window Help
mystr = 'actgactgact'
print(mystr.isalpha())
>>> 
```

In 1 Col 14

- `isdigit()`

```
[<ipython> - C:\Users\Prasad\Desktop\demol2.py (1.0%)
File Edit Shell Debug Options Window Help
mystr = '123456789'
print(mystr)
print(mystr.isdigit())
>>>
----- RESTART: C:\Users\Prasad\Desktop\demol2.py -----
123456789
True
>>>
```

The screenshot shows the Python IDLE Shell window. The code `mystr = '123456789'; print(mystr); print(mystr.isdigit())` is run, resulting in the output `123456789` and `True`. The window title is "IDLE Shell 3.9.5". The system tray at the bottom right shows the date as 27-06-2021.

```
[<ipython> - C:\Users\Prasad\Desktop\demol2.py (1.0%)
File Edit Shell Debug Options Window Help
mystr = '12345abcde09'
print(mystr)
print(mystr.isdigit())
>>>
----- RESTART: C:\Users\Prasad\Desktop\demol2.py -----
12345abcde09
False
>>>
```

The screenshot shows the Python IDLE Shell window. The code `mystr = '12345abcde09'; print(mystr); print(mystr.isdigit())` is run, resulting in the output `12345abcde09` and `False`. The window title is "IDLE Shell 3.9.5". The system tray at the bottom right shows the date as 27-06-2021.

- isidentifier()

```
File Edit Shell Debug Options Window Help
Python 3.5.5 (tag:b735.51f0a7ddbd, May  3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mystr
'mystr'
True
>>>

RESTART: C:\Users\Prasad\Desktop\demo2.py
```

```
File Edit Shell Debug Options Window Help
Python 3.5.5 (tag:b735.51f0a7ddbd, May  3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mystr
'mystr'
True
>>>
mystr00000
False
>>>
```

```
islower.py - C:\Users\Prasad\Desktop\islower.py (1.0.0)
File Edit Shell Debug Options Window Help
mystr = "MyNameIsPrasad"
print(mystr)
print(mystr.isidentifier())
A IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May  3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\islower.py =====
myNameIsPrasad
True
>>>
===== RESTART: C:\Users\Prasad\Desktop\islower.py =====
MyNameIsPrasad
False
>>>
===== RESTART: C:\Users\Prasad\Desktop\islower.py =====
_myNameIsPrasad
True
>>> |
```

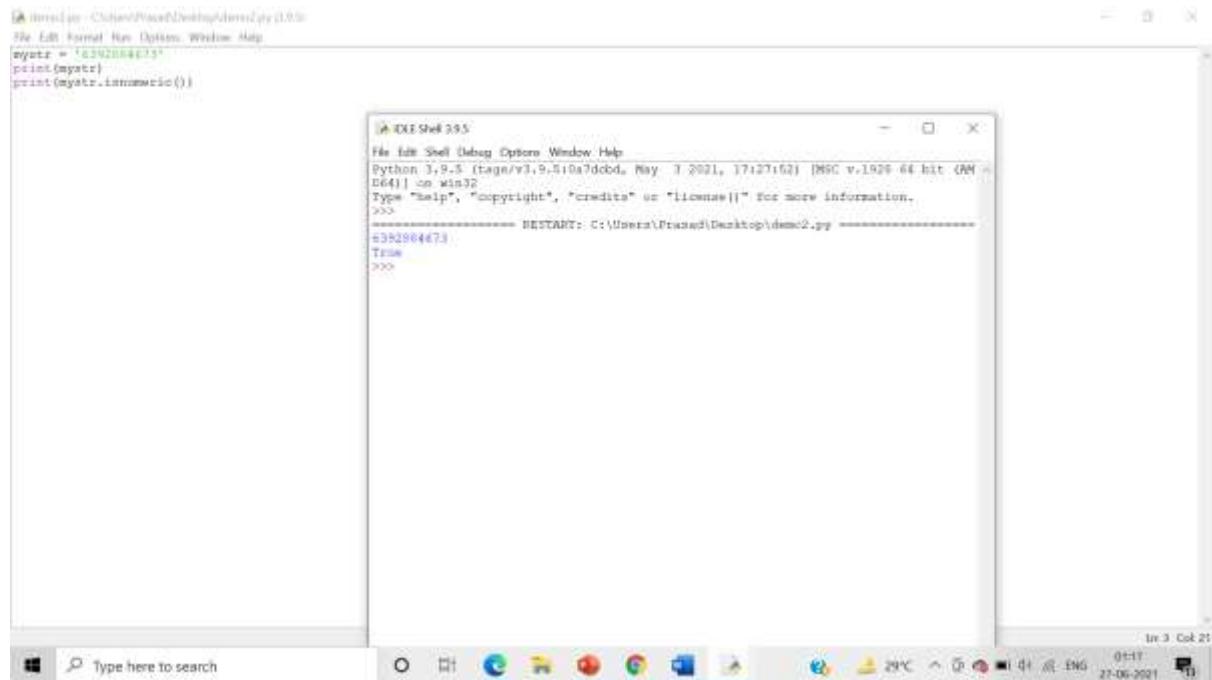
- **islower()**

```
islower.py - C:\Users\Prasad\Desktop\islower.py (1.0.0)
File Edit Shell Debug Options Window Help
mystr = "HOW ARE YOU"
print(mystr)
print(mystr.islower())
A IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May  3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\islower.py =====
HOW ARE YOU
False
>>> |
```

```
laptop-pc - C:\Users\Pranad\Desktop\demo2.py (1/1)
File Edit Insert Run Options Window Help
mystr = "How are4 you ?"
print(mystr)
print(mystr.islower())
# IDE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:37:02) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Pranad\Desktop\demo2.py -----
HOW ARE YOU
False
>>> ----- RESTART: C:\Users\Pranad\Desktop\demo2.py -----
how are4 you ?
True
>>>
```

```
laptop-pc - C:\Users\Pranad\Desktop\demo2.py (1/1)
File Edit Insert Run Options Window Help
mystr = "How are4 you ?"
print(mystr)
print(mystr.islower())
# IDE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:37:02) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Pranad\Desktop\demo2.py -----
HOW ARE YOU
False
>>> ----- RESTART: C:\Users\Pranad\Desktop\demo2.py -----
how are4 you ?
True
>>> ----- RESTART: C:\Users\Pranad\Desktop\demo2.py -----
how are you
True
>>> |
```

- isnumeric()

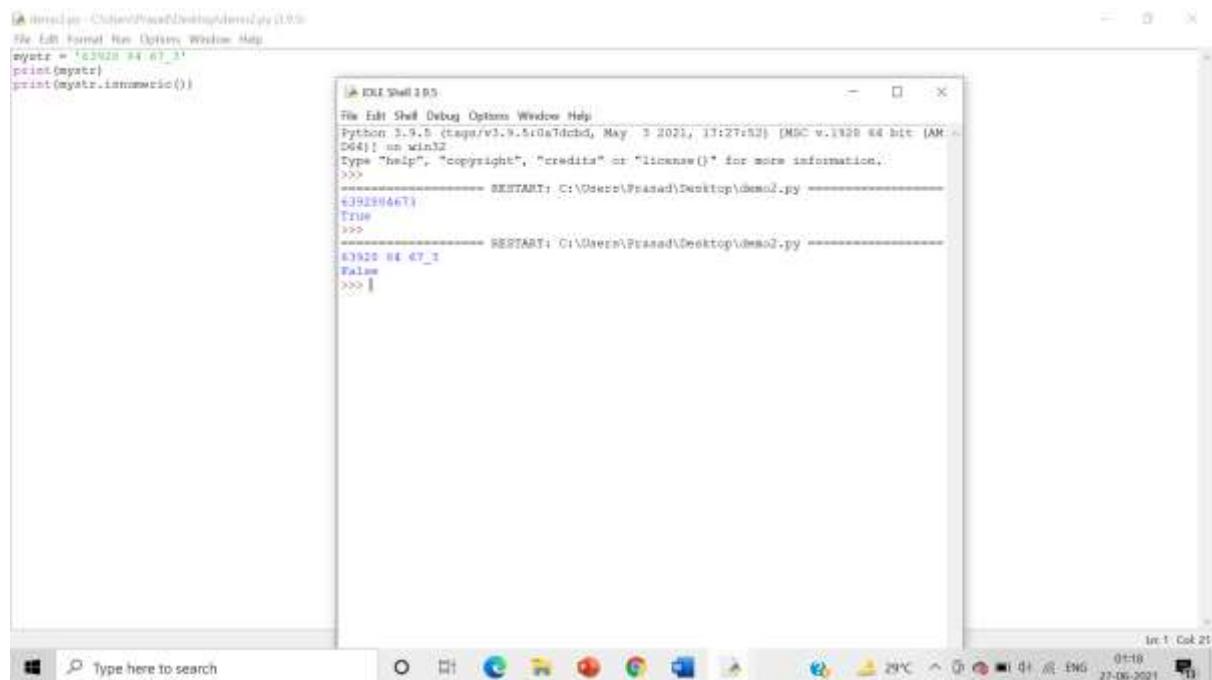


```
mystr = '1234567890'
print(mystr)
print(mystr.isnumeric())

-----
```

 RESTART: C:\Users\Prawad\Desktop\demo2.py
 1234567890
 True
 >>>

 RESTART: C:\Users\Prawad\Desktop\demo2.py
 1234 67_1
 False
 >>>



```
mystr = '1234567890'
print(mystr)
print(mystr.isnumeric())

-----
```

 RESTART: C:\Users\Prawad\Desktop\demo2.py
 1234567890
 True
 >>>

 RESTART: C:\Users\Prawad\Desktop\demo2.py
 1234 67_1
 False
 >>>

- isspace()

A screenshot of the Python IDLE interface. The code editor window contains the following Python script:

```
#isspace.py - Checks if a character is whitespace or not.
mystr = "Hello BID"
print(mystr)
print(mystr.isspace())
```

The IDLE Shell window shows the output of the script:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Hello BID
True
>>>
```

The taskbar at the bottom of the screen shows various application icons.

viii. lstrip, rstrip, strip

Algorithm:- Use the lstrip, rstrip, strip to alter or modify the given string.

- lstrip

A screenshot of the Python IDLE interface. The code editor window contains the following Python script:

```
#lstrip.py - Checks if a character is whitespace or not.
mystr = "Hello BID"
print(mystr.lstrip("He"))
```

The IDLE Shell window shows the output of the script:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Hello BID
o BID
>>>
```

The taskbar at the bottom of the screen shows various application icons.

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/6)
File Edit Shell Debug Options Window Help
mystr = "Hello BID"
print(mystr.lstrip())
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/6)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0a0d64d5, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
Hello BID
>>>
```

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/6)
File Edit Shell Debug Options Window Help
mystr = "Hello BID"
print(mystr)
print(mystr.lstrip())
idle.py - C:\Users\Prasad\Desktop\demo2.py (1/6)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0a0d64d5, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
Hello BID
>>>
```

- strip()

The screenshot shows a Windows desktop environment. In the center is a terminal window titled "IDLE Shell 3.9.5" running Python 3.9.5. The window displays a script with code to print a stripped string. Below the terminal is a taskbar with several icons, including the Start button, File Explorer, and browser icons. The system tray at the bottom right shows the date as 27-06-2021.

```
# IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:caedbe7, May  3 2021, 17:27:52) [GCC v.1020 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> how are you
how are you
>>> RESTART: C:\Users\Straaed\Desktop\demo2.py
```

The screenshot shows a Windows desktop with several open windows. In the foreground, the Python 3.9.5 IDLE Shell window is active. The code in the editor pane is:

```
mystr = "I how are you!!"
print(mystr)
print(mystr.strip("!!"))
```

The output pane shows the following results:

```
I how are you!!
how are you
>>> |
```

The status bar at the bottom right indicates the file is 'In 3. Col 2'.

6. Write set of programs for demonstrating the usage of following Operators

Algorithm: - We use the arithmetic operators (+,-,*,/,**,//) to perform the operations of addition, subtraction, multiplication, division, floor division as well as to find the exponent between 2 variables ‘a’ and ‘b’

i. Arithmetic operators

- Addition (+)



The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The Python version is listed as "Python 3.9.5 (tags/v3.9.5:5f9df53, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32". The shell prompt shows the code: `>>> a = 3
>>> b = 5
>>> a + b
8
>>>`. The output is the sum of a and b, which is 8.

- Subtraction (-)



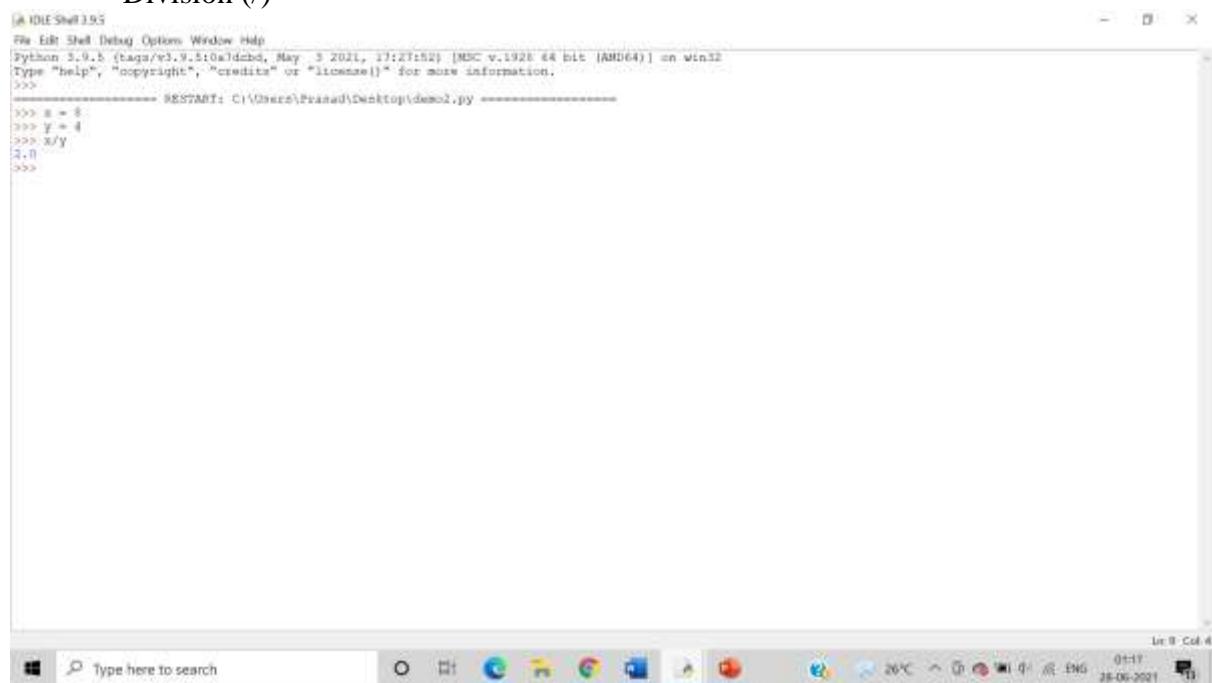
The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The Python version is listed as "Python 3.9.5 (tags/v3.9.5:5f9df53, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32". The shell prompt shows the code: `>>> x = 7
>>> y = 3
>>> x - y
4
>>>`. The output is the result of subtracting y from x, which is 4.

- Multiplication (*)



A screenshot of the Python IDLE Shell window. The title bar says "IDLE Shell 3.9.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, Help. The Python version is listed as "Python 3.9.5 (tags/v3.9.5:0a0dcbdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32". The shell prompt shows the following code and output:
```python  
>>> x = 3  
>>> y = 4  
>>> x\*y  
12  
>>>```

- Division (/)



A screenshot of the Python IDLE Shell window. The title bar says "IDLE Shell 3.9.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, Help. The Python version is listed as "Python 3.9.5 (tags/v3.9.5:0a0dcbdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32". The shell prompt shows the following code and output:  
```python  
>>> x = 8
>>> y = 4
>>> x/y
2.0
>>>```

- Modulus (%)

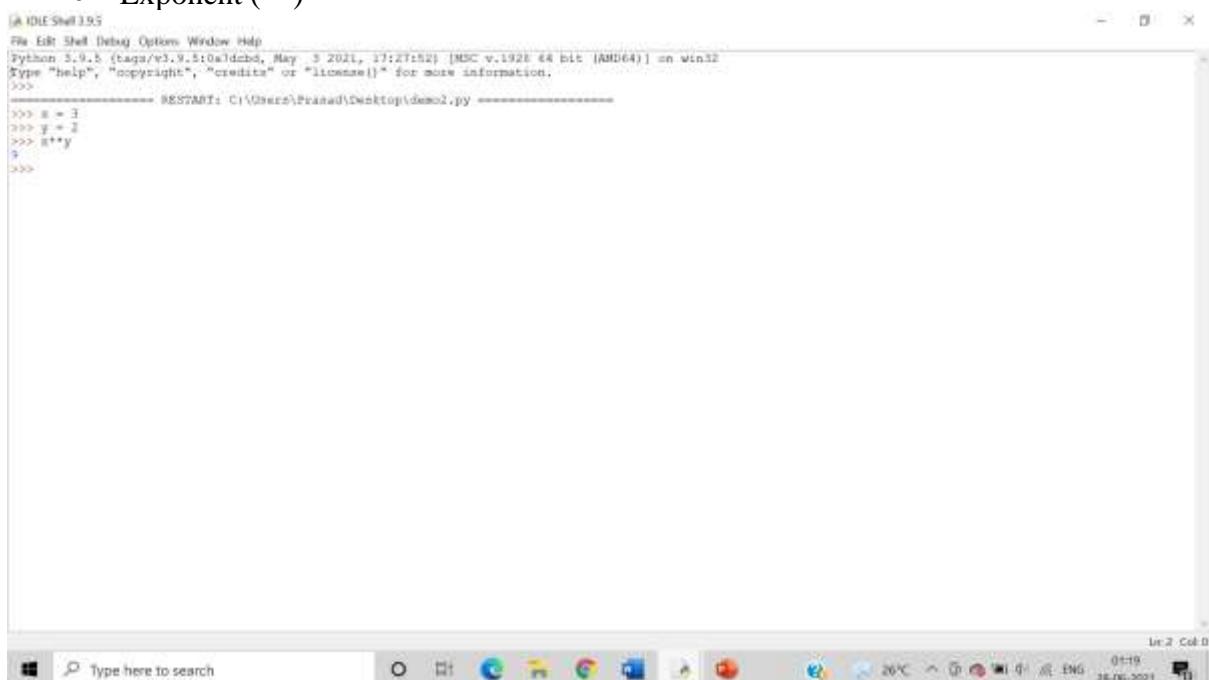


A screenshot of the Python IDLE Shell window. The title bar says "IDLE Shell 3.9.5". The shell displays the following code and output:

```
>>> x = 8
>>> y = 2
>>> x%y
0
>>>
```

The status bar at the bottom shows "Line 8 Col 6". The taskbar below the window includes icons for File Explorer, Task View, Edge, File History, Google Chrome, File Explorer, File History, and Task View.

- Exponent (**)



A screenshot of the Python IDLE Shell window. The title bar says "IDLE Shell 3.9.5". The shell displays the following code and output:

```
>>> x = 3
>>> y = 2
>>> x**y
9
>>>
```

The status bar at the bottom shows "Line 2 Col 6". The taskbar below the window includes icons for File Explorer, Task View, Edge, File History, Google Chrome, File Explorer, File History, and Task View.

- Floor Division (//)

The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.5". The code input area contains the following Python code:

```
>>> 1//3
1.666666666666666
>>> 9//8
1.0
>>> 8//3
2
>>>
```

The output window shows the results of the floor division operations.

The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.5". The code input area contains the following Python code:

```
>>> -8//3
-2.666666666666666
>>> 9//8
1.0
>>> 8//3
2
>>> -55//1
-18.33333333333332
>>> -55//3
-18
>>>
```

The output window shows the results of the floor division operations, including negative numbers.

ii. Assignment operators

Algorithm: - We use the assignment operators ($=$, $+=$, $-=$, $*=$, $/=$, $**=$, $//=$) to perform the operations of addition, subtraction, multiplication, division, floor division as well as to find the exponent of the variable a with a integer.

- $=$

When we type $x = 5$, that time we **assign** value of x as 5.



The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
a = 5
print('The value of variable a is ' + str(a))
```

The output window shows:

```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:27:52) [MSC v.1938 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo3.py -----
The value of variable a is 5
>>>
```

The taskbar at the bottom shows the date as 28-06-2021 and the time as 11:06.

- $+=$

When we type $x+=5$, it means $x = x + 5$.



The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
a = 5
a += 5
print('The value of variable a is ' + str(a))
```

The output window shows:

```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:27:52) [MSC v.1938 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----
The value of variable a is 25
>>>
```

The taskbar at the bottom shows the date as 28-06-2021 and the time as 10:58.

- -=

When we type $x=5$, it means $x = x - 5$.

```

@ idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demo2.py -----
The value of variable a is -16.3
>>>

```

- *=

When we type $x*=5$, it means $x = x * 5$.

```

@ idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demo2.py -----
The value of variable a is 106.5
>>>

```

- /=

When we type $x/=5$, it means $x = x / 5$.

```
a = 5
a /=21.3
print("The value of variable a is "+str(a))

>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
The value of variable a is 0.2347417840375587
>>>
```

- %=

When we type $x\%5$, it means $x = x \% 5$.

```
a = 5
a %=21.3
print("The value of variable a is "+str(a))

>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
The value of variable a is 5.0
>>>
```

- $//=$

When we type $x//=$, it means $x = x // 5$.

```
a = 25
a //= 5
print("The value of variable a is " + str(a))
```

```
The value of variable a is 5.0
```

- $**=$

When we type $x**=$, it means $x = x**5$

```
a = 2
a **= 5
print("The value of variable a is " + str(a))
```

```
The value of variable a is 32.0
```

iii. Logical operators

Algorithm: - Logical operators are used on conditional statements (either True or False). They perform **Logical AND**, **Logical OR** and **Logical NOT** operations. In a Logical AND operator Logical operator returns True if both the operands are True else it returns False. In a Logical OR operator, Logical or operator returns True if either of the operands is True. Logical not operator work with the single Boolean value. If the Boolean value is True it returns False and vice-versa.

- AND

```
a = 5
b = 10
c = 92
if (a>b and a>c):
    print ("TRUE")
else:
    print ("FALSE")
```

```
>>>
RESTART: C:\Users\Prasad\Desktop\demo2.py
FALSE
```

```
a = 92
b = 92
c = 92
if (a>b and a>c):
    print ("TRUE")
else:
    print ("FALSE")
```

```
>>>
RESTART: C:\Users\Prasad\Desktop\demo2.py
TRUE
```

- OR

```
a = 3  
b = 5  
c = 3  
if (a==b or b==c):  
    print ("TRUE")  
else:  
    print("FALSE")
```

```
IDLE Shell 1.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:5f9df22, May 3 2021, 17:27:52) [MSC v.1926 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
TRUE  
>>>
```

```
Type here to search
```

```
IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:5f9df22, May 3 2021, 17:27:52) [MSC v.1926 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:\Users\prasad\Desktop\demo2.py -----  
FALSE  
>>>
```

```
Type here to search
```

- NOT

The screenshot shows the Python IDLE Shell 3.9.5 window. The code in the editor is:

```
a = 3  
b = 5  
c = 8  
if not (a==b or a==c):  
    print ("TRUE")  
else:  
    print ("FALSE")
```

The output in the shell is:

```
|# IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1935 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
TRUE  
>>> |
```

The screenshot shows the Python IDLE Shell 3.9.5 window. The code in the editor is identical to the one above:

```
a = 3  
b = 5  
c = 8  
if not (a==b or a==c):  
    print ("TRUE")  
else:  
    print ("FALSE")
```

The output in the shell is:

```
|# IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1935 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
FALSE  
>>> |
```

iv. Membership operators

Algorithm: - The ‘in’ operator is used to check if a value exists in a sequence or not. It evaluates to true if it finds a variable in the specified sequence and false otherwise. In a NOT IN membership operator, the output evaluates to true if it does not find a variable in the specified sequence and false otherwise.

- in

The screenshot shows a Windows desktop with two windows open. The top window is the Python IDLE Shell 1.9.5, which displays the following code and output:

```
[1] ian@ian-OptiPlex-5070: ~ [0]:0% 
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Bhub" in a)
True
>>>
```

The bottom window is the taskbar, showing other application icons like File Explorer, Edge, and Google Chrome.

The screenshot shows a Windows desktop with two windows open. The top window is the Python IDLE Shell 2.9.5, which displays the following code and output:

```
[1] ian@ian-OptiPlex-5070: ~ [0]:0% 
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Bhub" not in a)
False
>>>
```

The bottom window is the taskbar, showing other application icons like File Explorer, Edge, and Google Chrome.

- not in

```

cmd -> C:\Users\Prasad\Desktop\demo1.py (0.0s)
File Edit Shell Options Window Help
x = ["L1", "L2", "Honda City", "Maruti"]
print("F10" not in x)

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 17:37:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
True
True
True

```

Using strings:

- in

```

cmd -> C:\Users\Prasad\Desktop\demo2.py (0.0s)
File Edit Shell Options Window Help
x = ["How are you today?"]
print("How" in x)

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 17:37:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
True
True
True

```

- not in

```
[&] idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.9.5)
File Edit Shell Debug Options Window Help
x = ["How are you today?"]
print("How" not in x)

[&] IDLE Shell 1.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
False
>>>
```

Type here to search 26°C 00:44 28-06-2021

```
[&] idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.9.5)
File Edit Shell Debug Options Window Help
x = ["How are you today?"]
print("When" not in x)

[&] IDLE Shell 1.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
True
>>>
```

Type here to search 26°C 00:44 28-06-2021

v. Identity operators

Algorithm: - In Identity operators, the IS Identity operator Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. While, the IS NOT Identity operator evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

- is

```
a = 6
b = 6
print(a==b)
print(a is b)
```

```
RESTART: C:\Users\Prasad\Desktop\demo2.py
True
True
>>>
```

```
RESTART: C:\Users\Prasad\Desktop\demo2.py
True
False
>>>
```

- list

```
x = ['a','b','c']
y = ['a','b','c']
print(x==y)
print(x is y)
```

```
RESTART: C:\Users\Prasad\Desktop\demo2.py
True
True
>>>
```

```
RESTART: C:\Users\Prasad\Desktop\demo2.py
True
False
>>>
```

```
a = 5  
b = 6  
print(a+b)  
print(a, b)  
  
x = ("a","b","c")  
y = ("a","b","c")  
z=x  
print(x is z)
```

```
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
=>>> RESTART: C:\Users\Prasad\Desktop\demo1.py <----  
True  
True  
True  
=>>>
```

```
a = 5  
b = 6  
print(a+b)  
print(a, b)  
  
x = ("a","b","c")  
y = ("a","b","c")  
z=x  
print(x is z)
```

```
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
=>>> RESTART: C:\Users\Prasad\Desktop\demo2.py <----  
True  
True  
True  
=>>>
```

character

```
a = "abc"
b = "abc"
print(a==b)
print(a != b)

x = ("a","b","c")
y = ("a","b","c")
z=x
print(x is z)
```

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2bc, May 3 2021, 11:21:32) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
False
False
True
>>>
```

- is not

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2bc, May 3 2021, 11:21:32) [MSC v.1928 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
>>> x = 5
>>> type(x)
<class 'int'>
>>> x = 5.1
>>> type(x)
<class 'float'>
>>> x = 5
>>> type(x) is int
True
>>> type(5.1) is int
False
>>> x = 5
>>> type(x) is int
True
>>> type(x) is float
False
```

Python for Bioinformatics

Exercise 2

II-Title: Programs demonstrating use of Control structures

1. Write set of programs for demonstrating the usage of following loops and loop controls
 - i. If

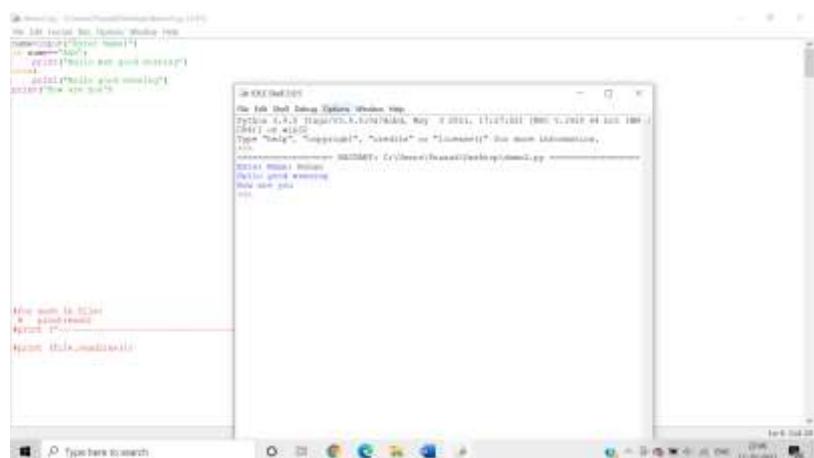
Algorithm: -

- 1) An if statement consists of a boolean expression followed by one or more statements. If the condition following the if statement evaluates to True, then the block of code indented inside the if statement gets executed.
 - 2) An if statement can be followed by an optional else statement, which executes when the Boolean expression is FALSE. If an If condition does not evaluate to True, then the block of code indented inside the else condition gets executed.



```
python if.py
Hello world
```

- ii. If-else

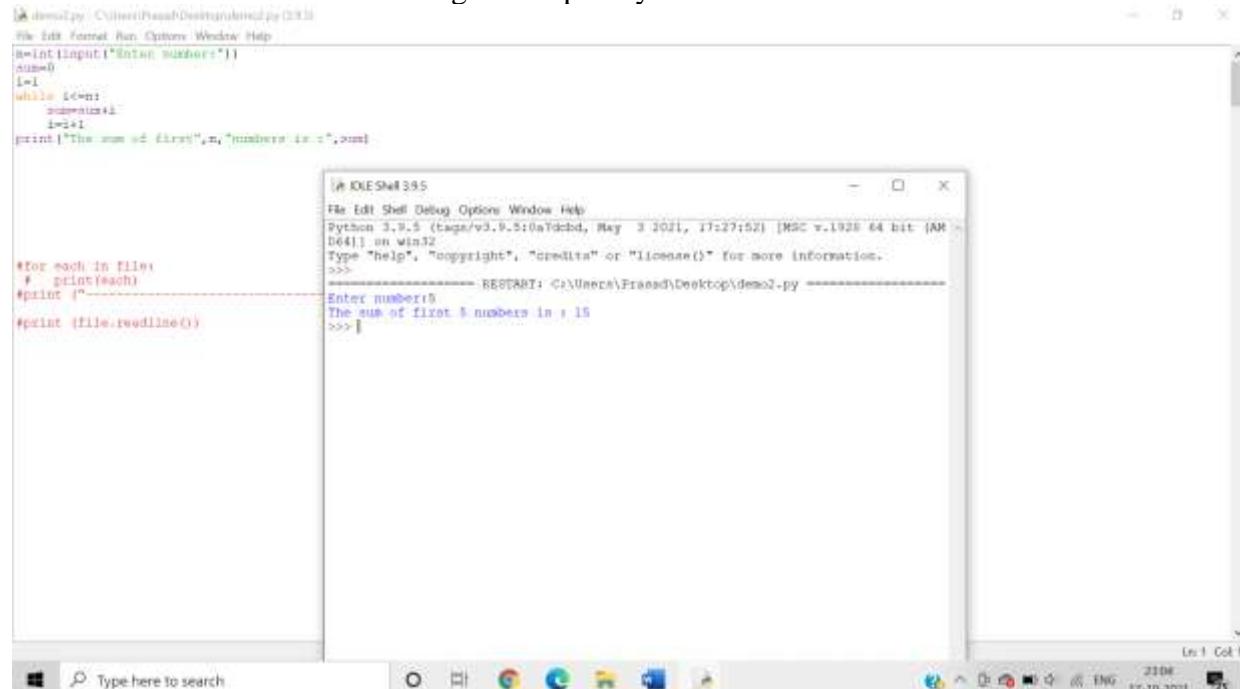


```
python ifelse.py
Goodbye world
```

iii. While

Algorithm:-

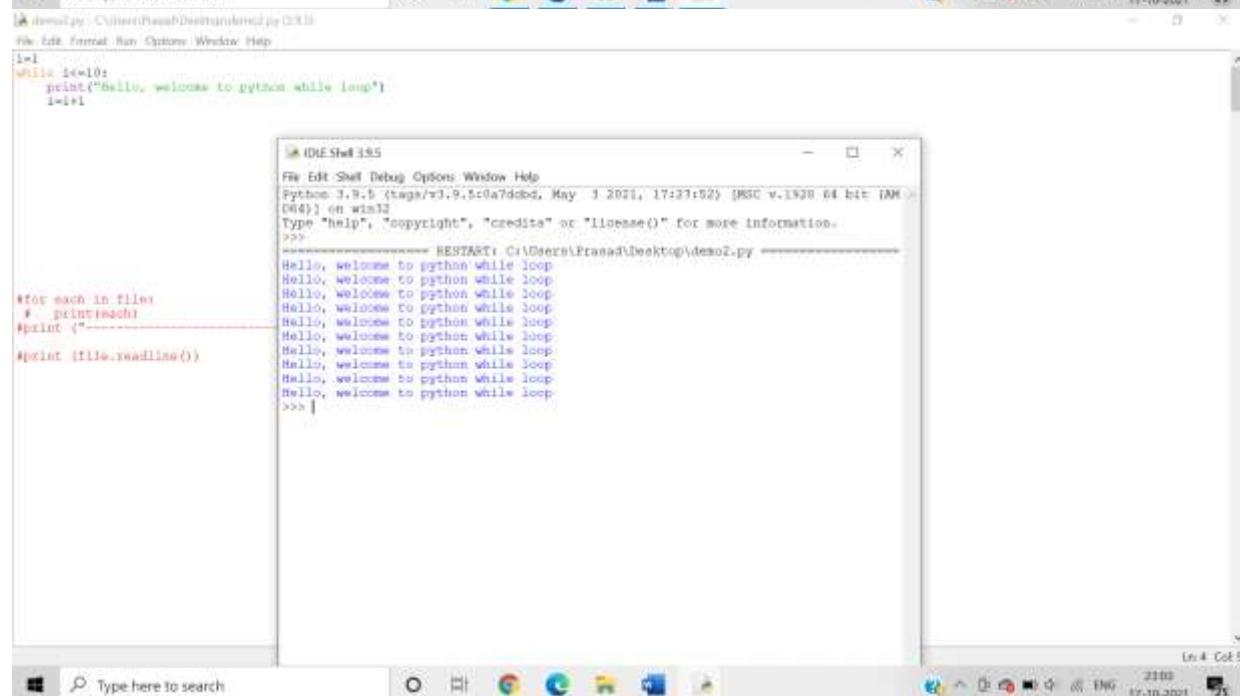
A While loop repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.



```
# demo1.py : C:\Users\Praesad\Desktop\demo1.py (23L)
file = open('C:\Users\Praesad\Desktop\sum.txt', 'r')
sum=0
for i in file:
    n = int(i)
    sum+=n
print("The sum of first", n, "numbers is : ", sum)

# for each in file:
#     print(each)
# print ("-----")
# print (file.readline())

# IDLE Shell 3.9.5
# File Edit Shell Debug Options Window Help
# Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on Win32
# Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praesad\Desktop\demo1.py
Enter number5
The sum of first 5 numbers is : 15
>>>
```



```
1=1
while 1<=10:
    print("Hello, welcome to python while loop")
    1+=1

# for each in file:
#     print(each)
# print ("-----")
# print (file.readline())

# IDLE Shell 3.9.5
# File Edit Shell Debug Options Window Help
# Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on Win32
# Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praesad\Desktop\demo2.py
Hello, welcome to python while loop
>>>
```

vi. For

A **for loop** executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. A **for with range loop** executes a sequence of statements multiple times and abbreviates the code that manages the loop variable for the given range

The screenshot shows the Python IDLE interface. On the left is the code editor window with the file 'demo1.py' containing:

```
# demo1.py - C:\Users\Prasad\Desktop\demo1.py (23.0)
file = open("Batch.txt")
for i in range(6):
    print("The character present at {} index is {}".format(i,i))
    i+=1
```

On the right is the IDLE Shell window with the output:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo1.py
The character present at 0 index is 0
The character present at 1 index is 1
The character present at 2 index is 2
The character present at 3 index is 3
The character present at 4 index is 4
The character present at 5 index is 5
>>>
```

The taskbar at the bottom shows the file 'demo1.py' is currently open.

vii. For with range

The screenshot shows the Python IDLE interface. On the left is the code editor window with the file 'demo2.py' containing:

```
# demo2.py - C:\Users\Prasad\Desktop\demo2.py (23.0)
for x in range(10):
    print("Hello welcome to python for loop")

# for each in files:
#     print(each)
# print(file.readlines())
```

On the right is the IDLE Shell window with the output:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Hello welcome to python for loop
>>>
```

The taskbar at the bottom shows the file 'demo2.py' is currently open.

A screenshot of a Windows desktop environment. In the center, there is a window titled "IDLE Shell 3.5". The window contains a Python script and its output. The script reads a file named "file.txt" and prints each line. The output shows the numbers 1 through 10. The Python version is 3.9.5. The desktop taskbar at the bottom shows icons for File Explorer, Google Chrome, and Microsoft Edge. A search bar is also visible on the taskbar.

```
#!/usr/bin/python3  
# Author: Prasad Deshpande (prasad.deshpande1996@gmail.com)  
#  
# This program prints all the elements present in file1.txt  
# one by one.  
  
try:  
    f = open('file1.txt')  
    print(f.read())  
  
except:  
    print("An error occurred")
```

```
IDLE Shell 3.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:ccb2d82, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
print(file1.readlines())  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
>>>
```

The screenshot shows a Windows desktop environment. In the center, there is a Notepad window titled "demo2.py" containing the following Python code:

```
#for x in range(21):
#    print(x)
#print ("-----")
#print (file.readline())
```

To the right of the Notepad window, an "IDLE Shell 3.8.5" window is open, showing the output of the code execution. The shell window has the following text:

```
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580d1c6, May  3 2021, 17:27:52) |MSC v.1929 64 bit (AMD64) | on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
>>> |
```

The taskbar at the bottom of the screen shows several pinned icons, including File Explorer, Microsoft Edge, and File History. The system tray indicates the date as 17-10-2021 and the time as 23:57.

vii. Break

Algorithm:-

- Break Terminates the loop statement and transfers execution to the statement immediately following the loop.

```
# demo1.py - CounterPrasadDesirajuLekha1.py (2021)
file edit Format Run Options Window Help
for i in range(10):
    if i==7:
        print("processing is enough..pls break")
        break
    print(i)

# for each in file:
#     print(each)
#print ("-----")

#processing is enough..pls break
>>>
```



```
# demo2.py - CounterPrasadDesirajuLekha1.py (2021)
file edit Format Run Options Window Help
cart=[10,20,30,400,60,70]
for item in cart:
    if item==500:
        print("To place this order insurance must be required")
        break
    print(item)

# for each in file:
#     print(each)
#print ("-----")

To place this order insurance must be required
>>>
```

viii. Continue

- 1) Algorithm:- Continue Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

The screenshot shows the Python IDLE interface. On the left is the code editor window titled 'demo1.py' containing the following code:for i in range(10):
 if i%2==0:
 continue
 print(i)

 #for each in files:
 # print(each)
 #print(file.readline())The right side shows the 'IDLE Shell 3.9.5' window with the following output:>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
1
3
5
7
9
>>>The status bar at the bottom indicates 'In 4 Col 13'.

The screenshot shows the Python IDLE interface. On the left is the code editor window titled 'demo2.py' containing the following code:cart=[10,20,500,700,50,60]
for item in cart:
 if item>500:
 print("We cannot process this item : ",item)
 continue
 print(item)

 #for each in files:
 # print(each)
 #print(file.readline())The right side shows the 'IDLE Shell 3.9.5' window with the following output:>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
We cannot process this item : 500
We cannot process this item : 700
>>>The status bar at the bottom indicates 'In 6 Col 21'.

ix. Pass

Algorithm: The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled "IDLE Shell 1.9.5" running Python 3.9.5. The code in the editor window is:

```
# demo1.py
for letter in "charan's book":
    if letter == ' ':
        pass
    print("This is pass block")
    print ('Current Letter :', letter)

print ("Good bye!")

# for each in files:
#     print(each)
#print ("-----")

#print (files.readme())
```

The terminal output shows the execution of the code:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
Current Letter : c
Current Letter : h
Current Letter : a
Current Letter : r
Current Letter : a
Current Letter : n
Current Letter : ' '
Current Letter : b
Current Letter : u
Current Letter : k
Current Letter : o
Current Letter : o
Current Letter : n
Good bye!
>>>
```

At the bottom of the terminal window, it says "Ln: 1 Col: 1".

2. Accept number from user and calculate the sum of all number between 1 and given number
Algorithm:

Step 1: -Allows a user to enter the number (n) he wishes to calculate the sum and average. The program accepts user input using the [input function](#).

Step 2: -Next, run loop till the entered number using the for loop and [range\(\) function](#).

Step 3: -Next, calculate the sum using a sum = sum + current number formula.

Step 4:-At last, after the loop ends, calculate the average using average = sum / n. n is a number entered by the user.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled 'Untitled - C:\Users\Pramad\Desktop\demo2.py (Python 3.9.5)'. It contains Python code for calculating the sum of numbers from 1 to a user-specified number. Below it is a terminal window titled 'DLE-Shell 3.9.5' showing the execution of the script. The terminal output shows the user entering '4' and the script printing 'Sum of numbers is: 10'. The taskbar at the bottom has icons for File Explorer, Edge, and other applications.

```
#> Untitled - C:\Users\Pramad\Desktop\demo2.py (Python 3.9.5)
File Edit Shell Debug Options Window Help
n = int(input("Enter number"))
sum = 0
for num in range(1, n + 1, 1):
    sum = sum + num
print("Sum of numbers is ", sum)

#file = open ("C:/Users/Pramad/Documents/myfiles.txt", "w")
#This will print the content
#for each in file:
#    print(each)
#print ("")
#print (file.readlines())
#>>>
```

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:5f9f7da, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Pramad\Desktop\demo2.py =====
Enter number 4
Sum of numbers is: 10
>>>
```

3. Write a Python program which iterates the integers from 1 to 50. For multiples of three print "Hello" instead of the number and for the multiples of five print "Welcome". For numbers which are multiples of both three and five print "Good Morning".

Algorithm: - We make use of a for loop to iterate all the numbers in the range of 1 to 50. We then use the if and if else loops with conditions where if the number a multiple of 3 then the output prints "Hello" is. When the number is a multiple of 5 the output prints the ord "Welcome" and when the number is the multiple of both 3 and 5 we print "Good Morning".



The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled 'KREShell 3.9.7' running Python 3.9.7. The code in the terminal window is as follows:

```
for x in range(51):
    if (x==0):
        continue
    if (x%15)==0:
        print("Good Morning")
    elif (x%3)==0:
        print("Hello")
    elif (x%5)==0:
        print("Welcome")
    else:
        print(x)
```

The output of the script is displayed below the code:

```
Good Morning
Hello
welcome
Good Morning
Hello
welcome
Good Morning
Hello
welcome
>>>
```

At the bottom of the terminal window, it says 'RESTART: C:\Users\Prasad\Desktop\demohai.py'. The status bar at the bottom right of the desktop shows 'Loc 3 Col 16' and the date '23-10-2021'.

4. Write a program to check whether a number is prime or not

Algorithm:-We use a for loop to divide the input number with number ranging from 1 to input num -1 using the remainder operator % in a for loop . If the remainder is zero in any instance then the input number is divisible by other numbers and hence is not a prime number.

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "IDLE Shell 3.9.5" running Python 3.9.5. The code in the terminal window is as follows:

```
if __name__ == "__main__":
    num = int(input("Enter a number: "))
    if num > 1:
        for i in range(2, num):
            if num % i == 0:
                print(f"{num} is not a prime number")
                print(f"\t{i} times {num//i} is {num}")
                break
        else:
            print(f"{num} is a prime number")
    else:
        print(f"{num} is not a prime number")
# This will print the content of myfile.txt
with open("myfile.txt") as file:
    for each in file:
        print(each)
print(f"\n{file.read()}\n")
```

The output of the script is:

```
>>> 31 is a prime number
>>>
```

In the background, there is a code editor window titled "idle39.py" showing the same Python script. The taskbar at the bottom of the screen displays icons for File Explorer, Edge browser, and File Explorer again.

5. Write a program to determine whether the number entered is Armstrong or not

Algorithm:-

Step 1:- The user has to enter any number.

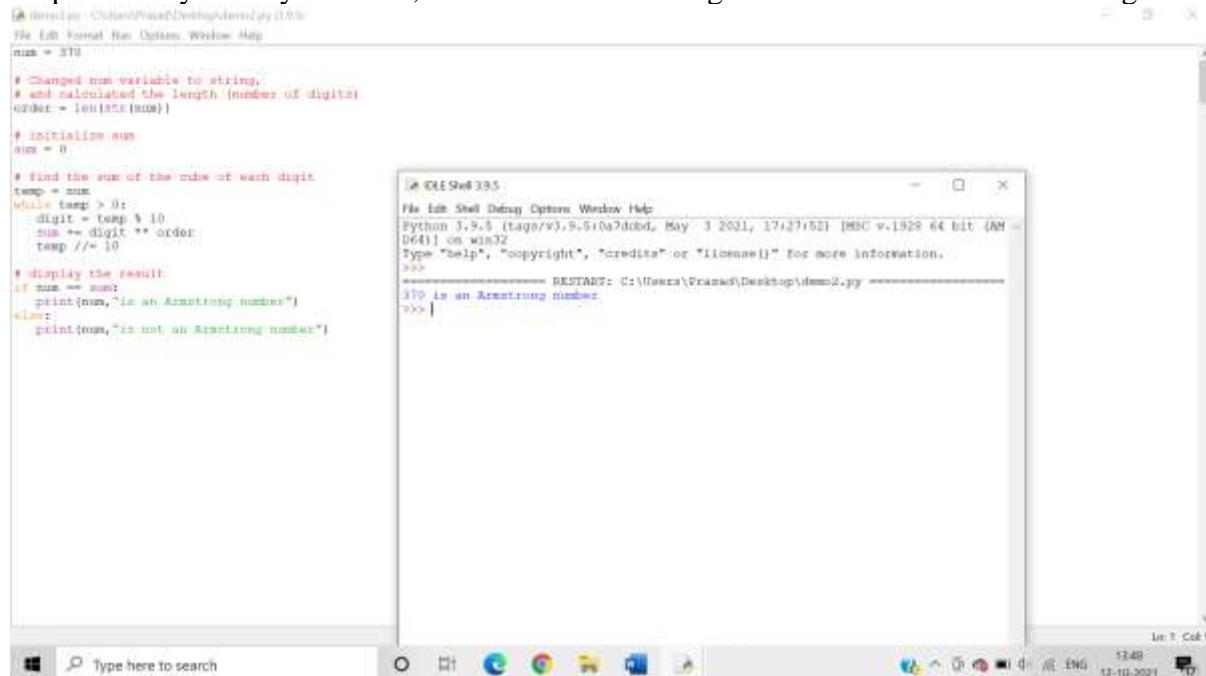
Step 2:- Count the Number of individual digits (For Example, 370 means 3).

Step 3:- Divide the given number into individual digits (For Example, Divide 370 into 3, 7, and 0).

Step 4:- Calculate the power of n for each individual and add those numbers.

Step 5:- Compare the original value with Sum value.

Step 6:- If they exactly matched, then it is an Armstrong number else it is not Armstrong



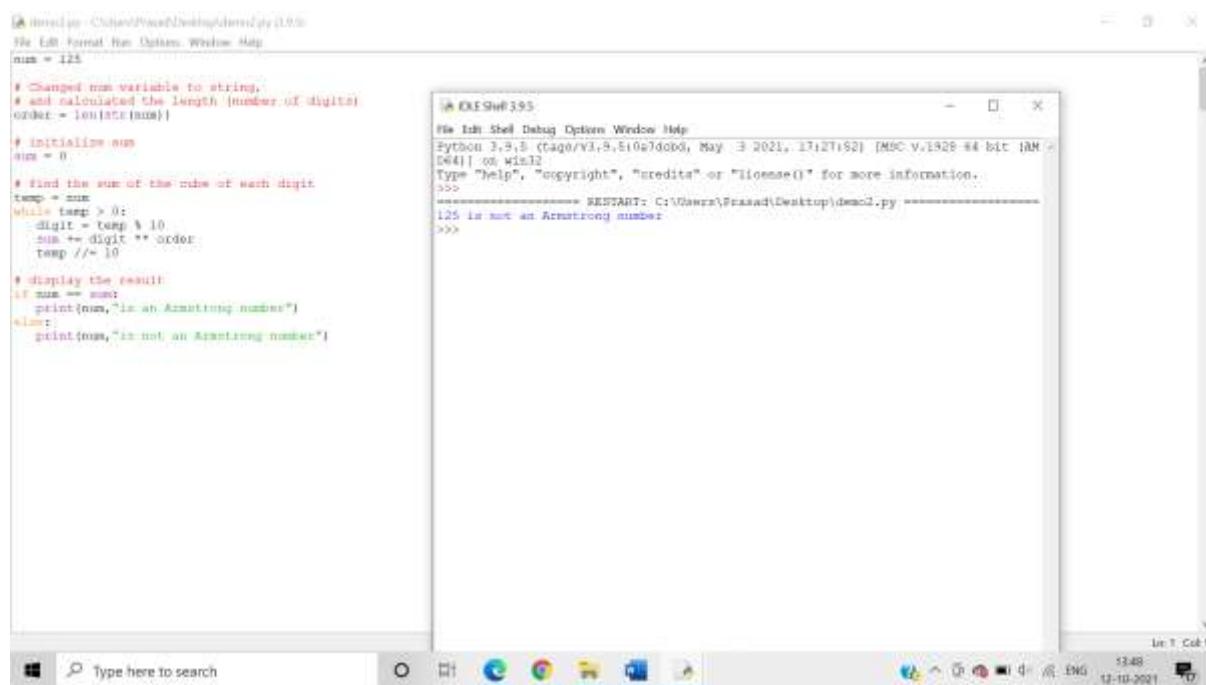
```
# changed num variable to string,
# and calculated the length (number of digits)
order = len(str(num))

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** order
    temp //= 10

# display the result
if sum == num:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

```
RESTART: C:\Users\Prasad\Desktop\dmemo2.py
370 is an Armstrong number
>>>
```



```
# changed num variable to string,
# and calculated the length (number of digits)
order = len(str(num))

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** order
    temp //= 10

# display the result
if sum == num:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

```
RESTART: C:\Users\Prasad\Desktop\dmemo2.py
125 is not an Armstrong number
>>>
```

6. Write a program to calculate the factorial of the number entered by the user

Algorithm:- We use recursive method to multiple all the number ranging from the input to 1 .

The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled "IDE Shell 3.8.5" running Python 3.9.5. The code in the terminal window is as follows:

```
# factorial.py - C:\Users\Praveen\Desktop\factorial.py (1.0.0)
File Edit Shell Debug Options Window Help
# Python program to find the factorial of a number provided by the user.
# change the value for a different result
num = 9

# To take input from the user
#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print("The factorial of",num,"is",factorial)

RESTART: C:\Users\Praveen\Desktop\factorial.py
>>> The factorial of 9 is 362880
```

Below the terminal window, there is a code editor window titled "factorial.py" containing the same Python code. The system tray at the bottom right shows the date as 12-10-2021 and the time as 13:51.

7. Print all odd and even numbers separately entered in the user defined range

Algorithm:- A number is even if it is perfectly divisible by 2. When the number is divided by 2, we use the remainder operator % to compute the remainder. If the remainder is not zero, the number is odd.

The screenshot shows a Windows desktop environment. In the foreground, there is an open Python script named 'demo2.py' in a code editor window. The code is as follows:

```
# Iterating each number in list
for num in range(start, end + 1):
    # checking condition
    if num % 2 == 0:
        print(num, end = " ")
```

Below the code editor is the Python IDLE Shell window, titled 'IDLEShell 3.9.5'. It displays the following interaction:

```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f0fbd9d, May 3 2021, 17:27:53) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Enter the start of range: 3
Enter the end of range: 7
4 6
```

The taskbar at the bottom of the screen shows several pinned icons, including File Explorer, Edge, and File Manager. The system tray indicates the date as 12-10-2021 and the time as 15:08.

8. Write a program to reverse words in a given String in Python

Algorithm:-

Step 1 Initialize the string.

Step 2 Split the string on space and store the resultant list in a variable called words.

Step 3 Reverse the list words using reversed function.

Step 4 Convert the result to list.

Step 5 Join the words using the join function and print it.

The screenshot shows a Windows desktop environment. In the center, there is a terminal window titled "IDLE Shell 3.9.5" running Python 3.9.5. The terminal displays the following text:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3f, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\dmn3.py =====
student & am I
>>>
```

To the left of the terminal is a code editor window titled "dmn3.py". It contains the following Python code:

```
# initializing the string
string = "I am a student"
## splitting the string on space
words = string.split()
## reversing the words using reversed() function
words = list(reversed(words))
## joining the words and printing
print(" ".join(words))

# for each in file:
#     print(each)
# print(f'{file.readlines()}

# print(file.readlines())
```

9. Construct the following pattern, using a nested for loop.

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

Algorithm:-

Step 1:- Accept the number of rows from a user using [input\(\)](#) function or decide the size of a pattern.

Step 2:- Iterate the number of rows using outer for loop and [range\(\) function](#)

Step 3:- Next, the inner loop or nested for loop to handle the number of columns. Inner loop iteration depends on the values of the outer loop.

Step 4:- asterisk in a Pyramidal pattern using the print() function.

Step 5:- Add a new line after each row, i.e. after each iteration of outer for loop so you can display pattern appropriately.

The screenshot shows a Windows desktop environment. In the foreground, there is a Python script named 'dem2.py' open in a code editor. The code contains nested loops to print a pyramidal pattern of asterisks. Below the code editor is the Python IDLE Shell window, which displays the output of the script. The output shows a pyramid of asterisks with 5 rows. The desktop taskbar at the bottom includes icons for File Explorer, Edge browser, File Manager, Task View, and others. The system tray shows the date and time as '12-10-2021 19:22'. The status bar at the bottom right indicates 'In 10 Col 13'.

```
for i in range(1):  
    for j in range(i):  
        print('* ', end="")  
    print()  
  
for i in range(0,-1):  
    for j in range(i+1):  
        print('* ', end="")  
    print()  
  
#for each in file:  
#    print(each)  
#print('*'  
#print(file.readline())
```

```
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f020d98, May 3 2021, 17u27t52) [MSC v.1919 64 bit (AM  
D64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
-->>> RESTART: C:\Users\Prasad\Desktop\dem2.py  
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * *  
* * * *  
* * *  
* *  
*  
*
```

10. Print the following pattern using for loop

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

Algorithm:-

Step 1:- Accept the number of rows from a user using [input\(\)](#) function or decide the size of a pattern.

Step 2:- Iterate the number of rows using outer for loop and [range\(\) function](#)

Step 3:- Next, the inner loop or nested for loop to handle the number of columns. Inner loop iteration depends on the values of the outer loop.

Step 4:- Print number using the print() function.

Step 5:- Add a new line after each row, i.e. after each iteration of outer for loop so you can display pattern appropriately.

The screenshot shows a Windows desktop environment. In the foreground, there is an 'OLE Shell 3.95' window which is a Python shell interface. It displays the following text:

```
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:0a0d0dd0d, May 1 2021, 17437158) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
5  
5 4 3 2 1  
4 3 2 1  
3 2 1  
2 1  
1  
>>> |
```

Below the shell window, the taskbar is visible with several icons. To the left of the shell window, there is another window titled 'idle.pyw - C:\Users\Praaad\Desktop\democ2.py (1.0%)'. This window contains the following Python code:

```
rows = 5  
for i in range(0, rows + 1):  
    for j in range(rows - i, 0, -1):  
        print(j, end=' ')  
    print()  
  
for each in file:  
    print(each)  
print('*'*50)  
print(file.readline())
```

Python for Bioinformatics

Exercise 3

III-Title: Programs demonstrating use of Lists and Tuples

1. Write a program demonstrating declaration, accessing- Indexing and splicing of lists and its elements.

Algorithm:-

Step 1: In python a list is declared by placing all the items (elements) inside square brackets [], separated by commas and assigning it to a variable.

Step 2: The list is accessed by printing the variable to which the list has been assigned to.

Step 3: To access the list items refer to the index number inside the index operator[] (INDEXING) [].

Step 4: To print a subset of values present in lists, input the start and end number of the subset in the index operator []

Answer: Program demonstrating splicing:

The image shows three separate windows of the Python IDLE shell. Each window displays a code editor at the top and a command-line interface below. The code in each window is identical:`# demo1.py - C:\Users\Praasad\Desktop\demo1.py (1.0)
File Edit Shell Options Window Help
mycolors = ["Red", "Green", "White", "Yellow", "Blue", "Orange"]
print(mycolors[2:6])`
The first window's command line shows the output: `['White', 'Yellow', 'Blue', 'Orange']`.
The second window's command line shows the output: `['White', 'Yellow', 'Blue', 'Orange']`.
The third window's command line shows the output: `['White', 'Yellow', 'Blue', 'Orange']`.

```
[#] C:\Users\Pranav\Desktop>idle2.py (1.9.5)
File Edit Shell Options Window Help
mycolors = ["Red","Pink","White","Green","Blue","Orange"]
print(mycolors [5])

# IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May  3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Pranav\Desktop\demo2.py -----
['Red', 'Pink', 'White', 'Green', 'Blue']
>>> 
```

Using negative index:

```
[#] C:\Users\Pranav\Desktop>idle2.py (1.9.5)
File Edit Shell Options Window Help
mycolors = ["Red","Pink","White","Green","Blue","Orange"]
print(mycolors [-1])

# IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May  3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Pranav\Desktop\demo2.py -----
Orange
>>> 
```

Splicing using negative index:

The screenshot shows a Windows desktop environment. In the center, there are two windows of the Python IDLE Shell 3.9.5. The left window contains the following code:

```
[> IDLE.py - C:\Users\Prasad\Desktop\demo2.py (1/1)]  
File Edit Shell Options Window Help  
mycolor = ["Red", "Pink", "White", "Green", "Blue", "Cyan"]  
print(mycolor[-5:-1])
```

The right window shows the output of the code execution:

```
[> IDLE Shell 3.9.5  
File Edit Shell Options Window Help  
Python 3.9.5 (tags/v3.9.5:f9f7f1e, May  3 2021, 17:27:43) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> -----  
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
['Pink', 'White', 'Green', 'Blue']  
>>> ]
```

At the bottom of the screen is a taskbar with several icons, including the Start button, a search bar, and pinned application icons for File Explorer, Edge, and others. The system tray shows the date as 06-07-2021.

2. Write a program demonstrating input of list elements from the user.
- Step 1: Use the `input()` function to take user input and assign it to a variable
- Step 2: Use the `split` function to split each value in the variable with a “,” and assign it a new list variable.
- Step 3: Print the list variable

```

#demul1.py - C:\Users\Prasad\Desktop\demul1.py (1.8k)
File Edit Shell Options Window Help
BUCK = []
n = int(input("Enter the number of elements"))
print("go on entering the values")
for i in range(0,n):
    s = int(input("Enter the value"))
    BUCK.append(s)
print(BUCK)

mylist = ['Red', 'Pink', 'Yellow', 'Blue']
s = " ".join(mylist)
#print(s)

mylist = mystr.split(",") #splitting the string
#print(mylist)
mylist = mystr.splitlines() #splitting the string into lines
#print(mylist)

def myfunc(x):
    # return len(x)
    mycolour = str(x) + myfunc
    #print(mycolour)
    return mycolour

for i in range(0, len(BUCK)):
    print(myfunc(BUCK[i]))

```

```

#demul2.py - C:\Users\Prasad\Desktop\demul2.py (1.8k)
File Edit Shell Options Window Help
mycolors = ['Red', 'Pink', 'White', 'Green', 'Blue', 'Orange']
#print(mycolors)


```

Using list constructor:

```
# demo2.py : C:\Users\Prasad\Desktop\demo2.py (1/3)
File Edit Shell Options Window Help
File Edit Shell Options Window Help
mylist = [x**2 for x in range(1,6)]
print(mylist)

[A] IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d 2021-06-29 19:12:32) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
>>> [1, 4, 9, 16, 25]
>>>
```

Type here to search 1340 10:29 AM 06-07-2021

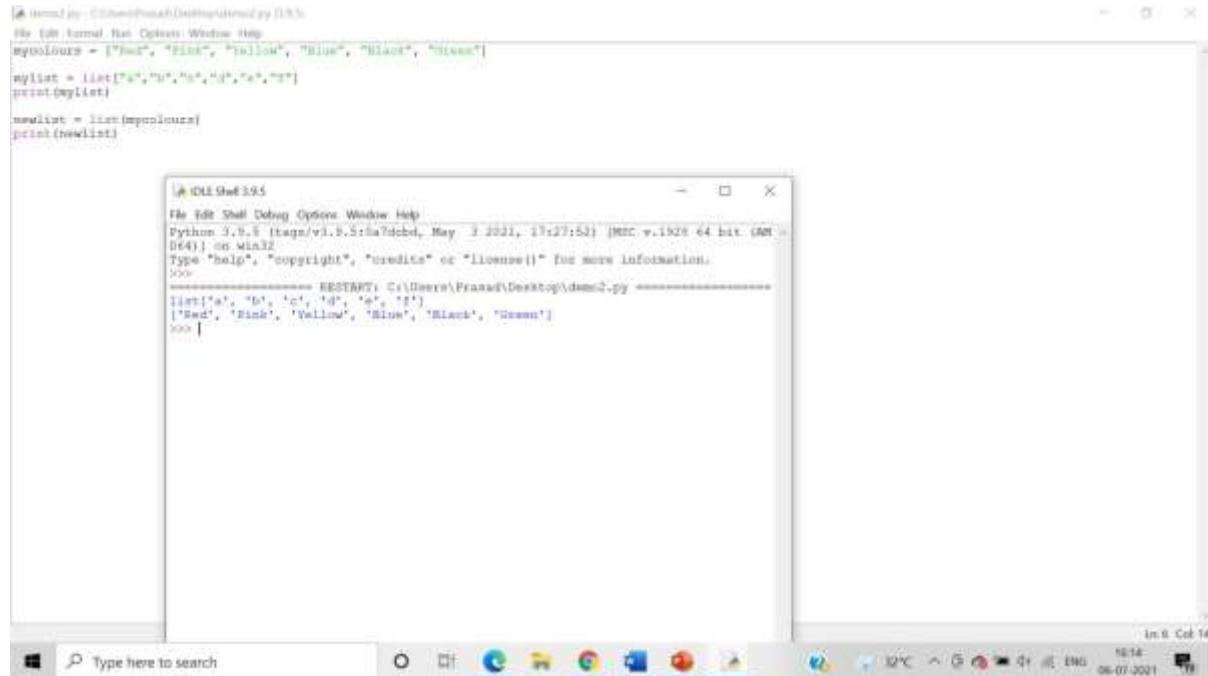
Creating copy of list:

```
# demo2.py : C:\Users\Prasad\Desktop\demo2.py (1/3)
File Edit Shell Options Window Help
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]
mylist = [x**2 for x in range(1,6)]
print(mylist)
print(mycolours)

[A] IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d 2021-06-29 19:12:32) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
>>> [1, 4, 9, 16, 25]
>>> ['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green']
>>>
```

Type here to search 1559 10:29 AM 06-07-2021

Creating copy of list using list constructor:



```
[dmc2.py] C:\Users\Prasad\Desktop\dmc2.py [18%]
File Edit Shell Debug Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]
mylist = list("a", "b", "c", "d", "e", "f")
print(mylist)
newlist = list(mycolours)
print(newlist)

[Python 3.9.5 (tags/v3.9.5:541377f, May 3 2021, 17:33:52) [MSC v.1928 64 bit (AMD64)] on Windows]
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\dmc2.py
['a', 'b', 'c', 'd', 'e', 'f']
['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green']
```

3. Write a program demonstrating printing of list elements using for in and range() function

Algorithm:-

Step 1: Declare a list by assigning it to a variable

Step 2: The range() gives us a sequence of numbers in between the two integers given to it.

Step 3: We can use the in and range() function to print the list elements using a for loop.

```
File Edit Shell Debug Options Window Help
>>> Enter the number of elements to be inserted: 3
[1]
for i in range(0,n):
    elem=int(input("Enter elements: "))
    x.append(elem)
avg=sum(x)/n
print("Average of elements in the list",round(avg,2))

>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Enter the number of elements to be inserted: 3
Enter element: 45
Enter element: 33
Enter element: 33
Average of elements in the list 31.8
>>>

from __future__ import division
num = []
n = int(input("Enter the number of elements to be inserted: "))
for i in range(0,n):
    elem = int(input("Enter elements: "))
    num.append(elem)
print("Average of elements in the list",round(sum(num)/n,2))

num = [1, 2, 3, 4, 5]
print(num)
```

4. Write a program demonstrating len() and del() function.

Algorithm:-

Step 1: The len() function is used to obtain the length of the list

Step 2: The del() function deletes all the elements in range starting from index 'a' till 'b' mentioned in arguments.

Len()

```

# demo2.py : C:\Users\Prasad\Desktop\demo2.py (1/3)
# File Edit Shell Options Window Help
# Python 3.9.5 (tags/v3.9.5:5f9f085, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
# Type "help", "copyright", "credits" or "license()" for more information.
#>>> print("Average")
# Average
#>>> print(len(mycolours))
# 6
#>>> print(mylist)
# ['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green']

# mylist = []
# for x in mylist:
#     mylist = x
# print(mylist)

# num = 1
# n = int(input())
# print("go on")
# for i in range(n):
#     a = int(i)
#     mylist.append(a)
# print(num)

```

del()

```

# demo2.py : C:\Users\Prasad\Desktop\demo2.py (1/3)
# File Edit Shell Options Window Help
# Python 3.9.5 (tags/v3.9.5:5f9f085, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
# Type "help", "copyright", "credits" or "license()" for more information.
#>>> print(mycolours)
# mycolours
#>>> print(mycolours)

# ("Lemon", "yellow", "Light", "Green")
# mylist = mylist + mycolours
# mylist = list("a", "b", "c", "d")
#>>> print(mylist)
# mylist = list(mycolours)
#>>> print(newlist)
# NameError: name 'mycolours' is not defined
#>>>

```

5. Write a program demonstrating usage of ‘+’ and ‘*’ operators on lists Algorithm:-

Step 1: The ‘+’ operator is used to concatenate the 2 lists into a single list.

Step 2: The ‘*’ operator is used to multiply the list “n” times.

Program demonstrating usage of ‘*’ operator:

Code: Program demonstrating usage of ‘+’ operator:

```
[& demo1.py -C:\Users\Praveen\Desktop\demo1.py] [0:0]
File Edit Shell Options Window Help
mylist = ['Red', 'Pink', 'Yellow', 'Green', 'Black', 'Orange']
mylist = ["Creme", "Yellow", "Light Green"]

mylist = mylist + mylist
print(mylist)
mylist = list("a", "b", "c")
print(mylist)

File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf67d3, May  3 2021, 17:53:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> **** RESTART: C:\Users\Praveen\Desktop\demo1.py ****
['Red', 'Pink', 'Yellow', 'Green', 'Black', 'Orange', 'Red', 'Pink', 'Yellow', 'Green', 'Black', 'Orange', 'Creme', 'Yellow', 'Light Green']
>>>
```

```
[& demo1.py -C:\Users\Praveen\Desktop\demo1.py] [0:0]
File Edit Shell Options Window Help
list1 = ["Red", "Pink", "Yellow"]
list2 = [1, 2, 3]
list3 = list1 * 3
print(list3)

File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf67d3, May  3 2021, 17:53:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> **** RESTART: C:\Users\Praveen\Desktop\demo1.py ****
['Red', 'Pink', 'Yellow', 'Red', 'Pink', 'Yellow', 'Red', 'Pink', 'Yellow']
```

```
spaceInList = []
#>= 1st approach
#Print (x)
#>= 2nd approach
#>= x.append(x)
#>= print(x)
#Print("Message - a")
```

The screenshot shows a Windows desktop environment with a Python IDLE Shell window open. The window title is "IDLE Shell 3.9.5". The code in the editor pane is:list1 = [1,2,3,4]
list2 = [1,2,3]
list3 = list1 * 3
print(list3)

Python 3.9.5 (tags/v3.9.5:0c0f23c, May 3 2021, 17:27:42) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
[1, 2, 3, 1, 2, 3, 4]The command prompt shows a blank line after the restart message. The taskbar at the bottom includes icons for File Explorer, Task View, Edge, Google Chrome, File Explorer, and Task View, along with system status indicators like battery level and date/time.

6. Write a program demonstrating “in” and “not in” keywords on lists

Algorithm:-

Step 1: - The ‘in’ operator is used to check if an element is present in the list or not. Returns true if element is present in list else returns false.

Step 2: The ‘not in’ operator is used to check if an element is not present in the list or not. Returns true if element is not present in list else returns false.

Answer: Program demonstrating “in” keyword:

```
[file demo1.py]
#>!/usr/bin/python3
#># This program demonstrates the 'in' keyword.
#>mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
#>"Green" in mycolours
#>print("Colour present")
#>else:
#>    print("Colour absent")

[> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0a0bd, May 3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
----- RESTART: C:\Users\Prasad\Desktop\demo1.py -----
colour present
>>>

#>x = len(mycolours)
#>print(x)

#>n=int(input())
#>i=1
#>for i in range(1,n+1):
#>    s.append(mycolours[i])
#>print(s)
#>print("Average")
```

Program demonstrating “not in” keyword:

```
[file demo2.py]
#>#!/usr/bin/python3
#># This program demonstrates the 'not in' keyword.
#>mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
#>"Purple" not in mycolours
#>print("Colour absent")
#>else:
#>    print("Colour present")

[> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0a0bd, May 3 2021, 17:12:52) [MSC v.1928 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
colour absent
>>>

#>x = len(mycolours)
#>print(x)

#>n=int(input())
#>i=1
#>for i in range(1,n+1):
#>    s.append(mycolours[i])
#>print(s)
#>print("Average")
```

7. Write set of programs for demonstrating the usage of all the different Lists methods along with their variations

Algorithm:-

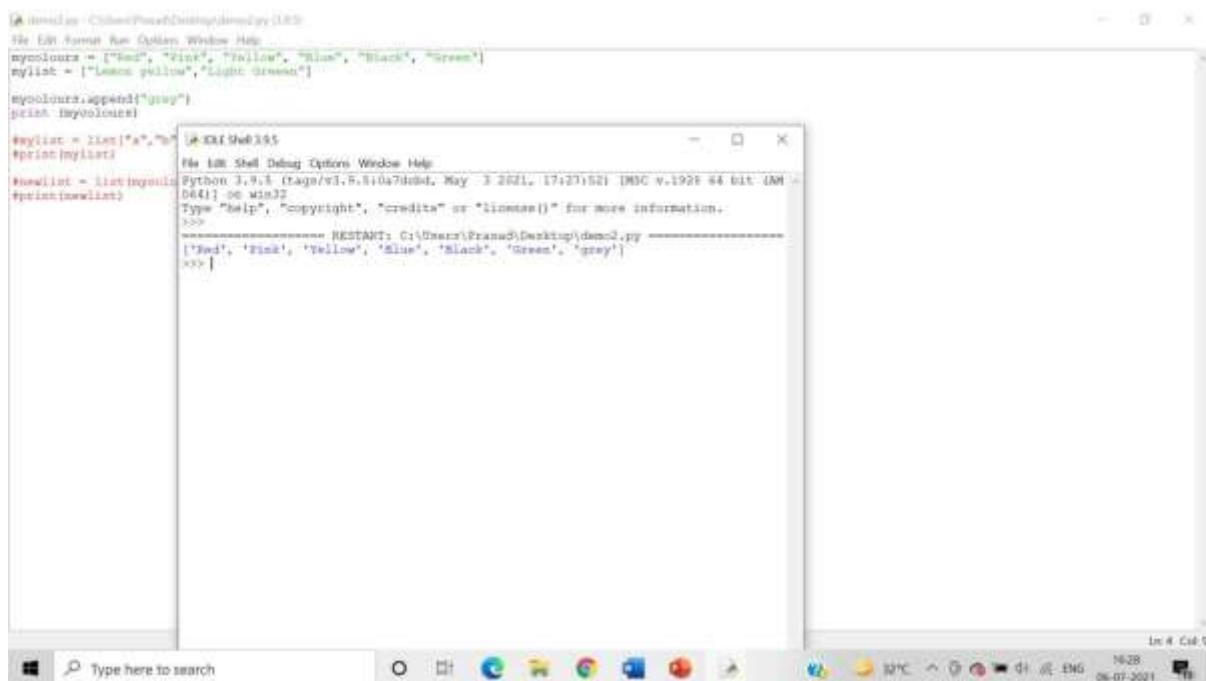
Step 1: The append() function is used to add values to a given list.

Step 2: The clear() function is used to erase all the elements of list. After this operation, list becomes empty.

Step 3: The copy() function is used to copy elements in a list and to assign it to a new variable.

Step 4: The count() function counts the number of occurrences of elements in list.

- Append()

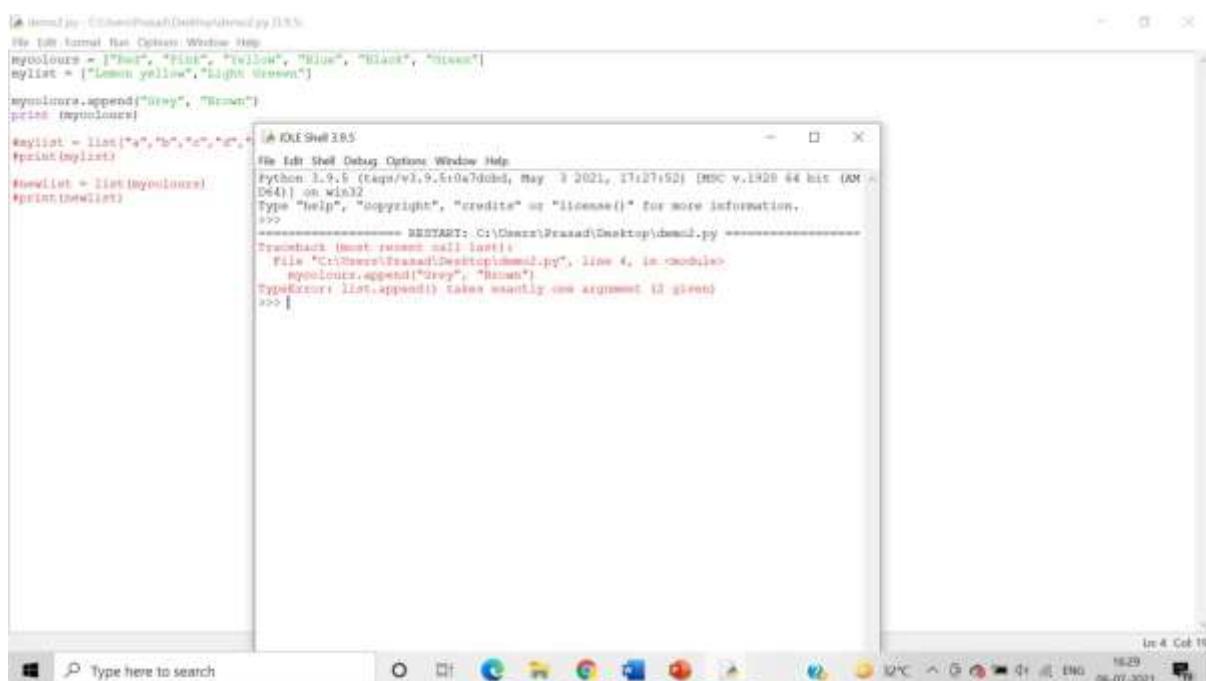


```
# demo1.py - C:\Users\Praaad\Desktop\demol1.py (1.0)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]
mylist = ["Lemon yellow", "Light Green"]
mycolours.append("grey")
print(mycolours)

mylist = list(*a*, *b*)
print(mylist)

newlist = list(mycolours)
#print(newlist)
>>> RESTART: C:\Users\Praaad\Desktop\demol1.py
['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green', 'grey']
>>>
```

Adding 2 elements:



```
# demo1.py - C:\Users\Praaad\Desktop\demol1.py (1.0)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]
mylist = ["Lemon yellow", "Light Green"]
mycolours.append("Grey", "Brown")
print(mycolours)

mylist = list(*a*, *b*, *c*, *d*)
print(mylist)

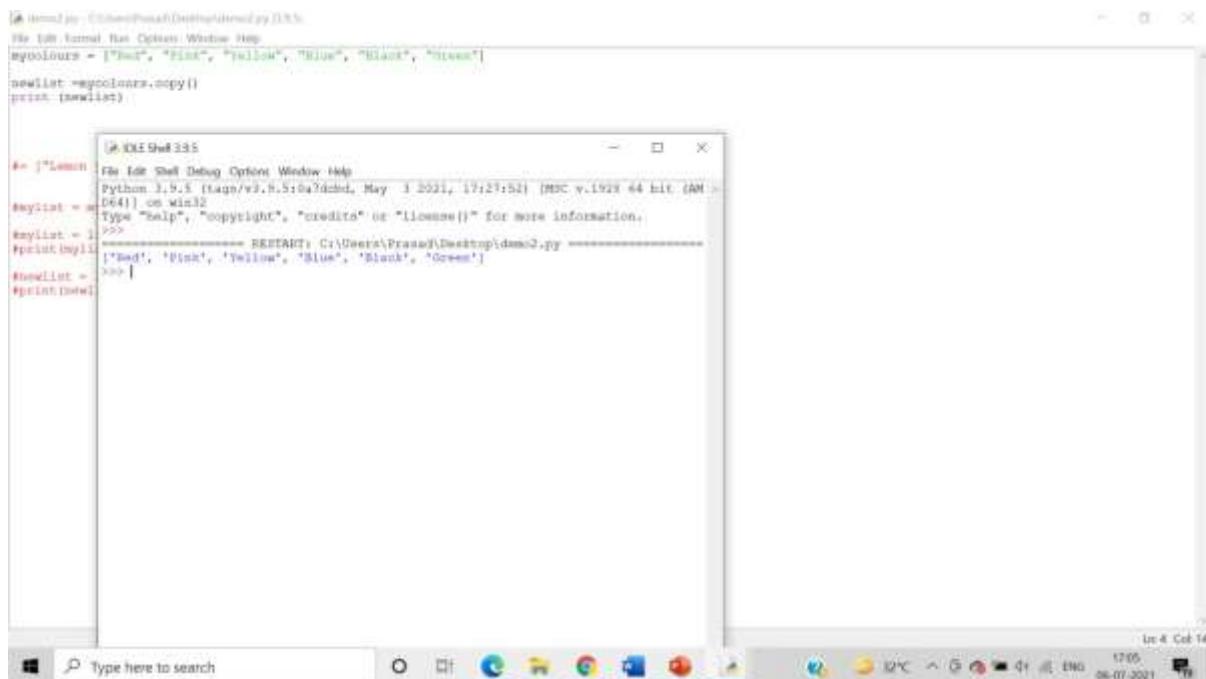
newlist = list(mycolours)
#print(newlist)
>>> RESTART: C:\Users\Praaad\Desktop\demol1.py
Traceback (most recent call last):
  file "C:\Users\Praaad\Desktop\demol1.py", line 4, in <module>
    mycolours.append("Grey", "Brown")
TypeError: list.append() takes exactly one argument (2 given)
>>>
```

```
[# demo2.py - C:\Users\Prasad\Desktop\Demo2\demo2.py (1/1)]  
File Edit Shell Run Options Window Help  
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]  
mylist = ["Lemon yellow", "Light Green"]  
mycolours.append(mylist)  
print(mycolours)  
  
mylist = list("a","b","c")  
print(mylist)  
  
newlist = list(mycolours)  
print(newlist)  
  
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green', ['Lemon yellow', 'Light Green']]  
#>>>
```

- **Clear()**

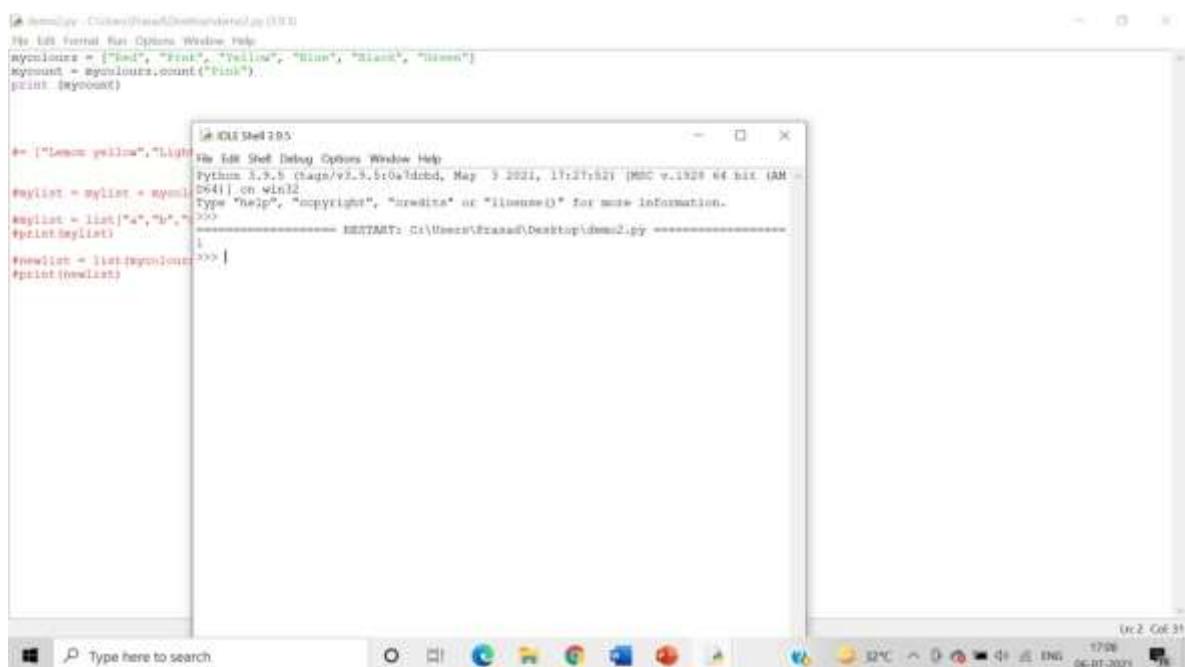
```
[# demo2.py - C:\Users\Prasad\Desktop\Demo2\demo2.py (1/1)]  
File Edit Shell Run Options Window Help  
mycolours = ["Red", "Pink", "Yellow", "Blue", "Black", "Green"]  
mycolours.clear()  
print(mycolours)  
  
## ("Lemon yellow", "Light Green")  
mylist = mylist + mycolours  
mylist = list("a","b","c")  
print(mylist)  
  
newlist = list(mycolours)  
print(newlist)  
  
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
[]  
#>>>
```

- Copy()



```
# democopy.py - ClintonPrasad\Democopy\democopy.py (1.0.0)
#> IDLE Shell 3.9.5
#> File Edit Shell Options Window Help
#> Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 1 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
#> type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\democopy\democopy.py
['Red', 'Pink', 'Yellow', 'Blue', 'Black', 'Green']
>>>
```

- Count()



```
# democount.py - ClintonPrasad\Democount\democount.py (1.0.0)
#> IDLE Shell 2.0.5
#> File Edit Shell Options Window Help
#> Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 1 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
#> type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\democount\democount.py
1
>>>
```

```
[file demo1.py] C:\Users\Prasad\Desktop\democ1.py (1/3)
File Edit Shell Options Window Help
mylist = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mycount = mycolours.count("Pink")
print(mycount)

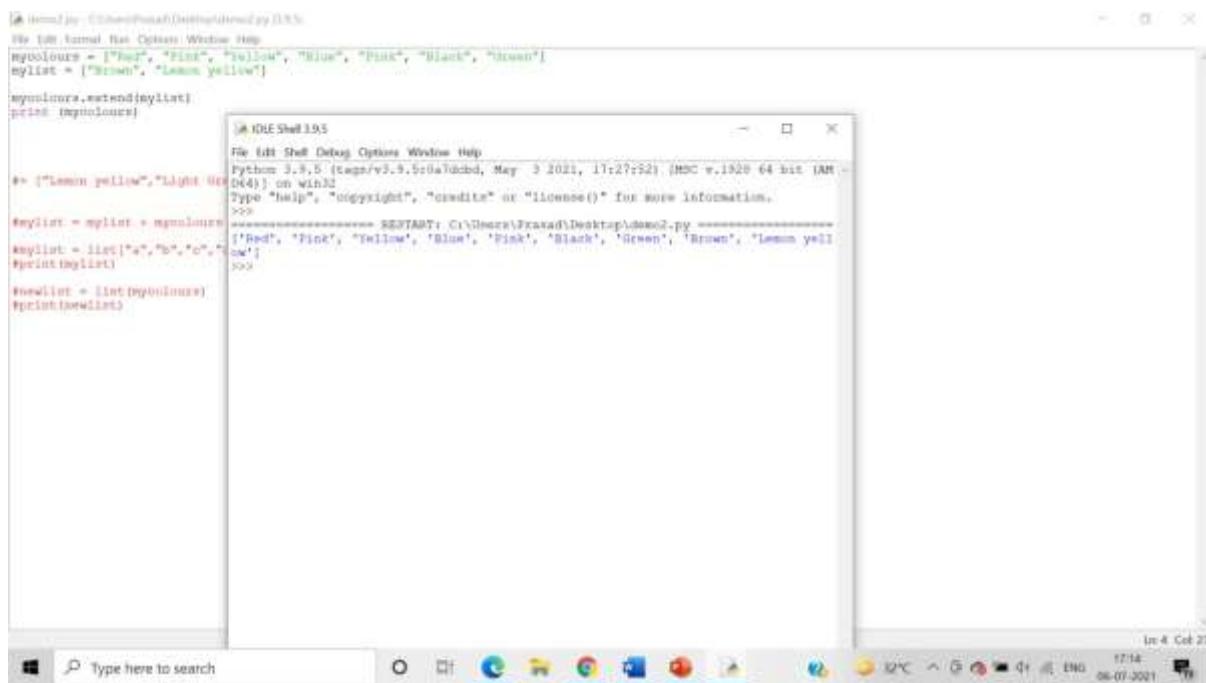
>>> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a073cbf4, May 3 2021, 17:27:02) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\democ1.py
>>> 2
>>> 
```

- Extend()

```
[file demo2.py] C:\Users\Prasad\Desktop\democ2.py (1/3)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mylist = ["Green", "Lemon Yellow"]

mycolours.extend(["purple"])
print(mycolours)

>>> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a073cbf4, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\democ2.py
['Red', 'Pink', 'Yellow', 'Blue', 'Pink', 'Black', 'Green', 'purple']
>>> 
```



```
[f] dmc2.py - C:\Users\Prasad\Desktop\dmc2.py (1/3)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Brown"]
mylist = ["Brown", "Lemon yellow"]

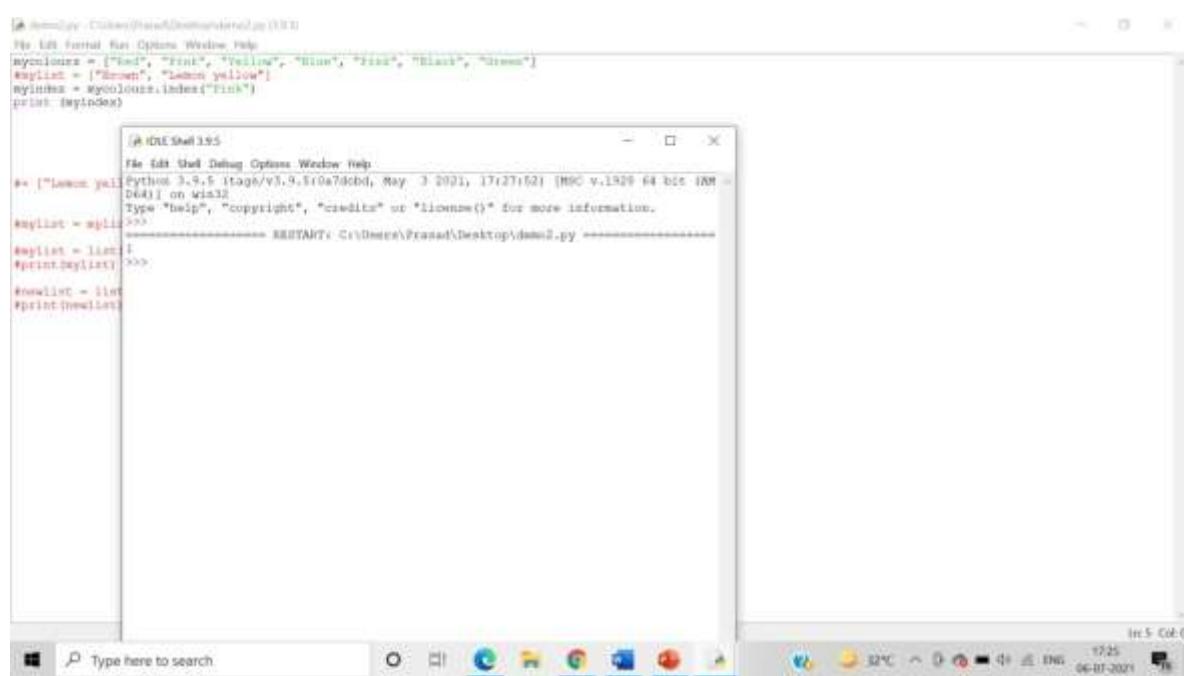
mylist.extend(mylist)
print (mycolours)

# ("Lemon yellow","Light orange")
#mylist = mylist + mycolours
#mylist = list("a","b","c","d")
#print(mylist)

#newlist = list(mycolours)
#print(newlist)
```

The screenshot shows the Python script dmc2.py running in the IDLE Shell. The script defines a list 'mycolours' and a list 'mylist'. It then extends 'mylist' by adding elements from 'mycolours' and prints the new list. The output in the shell shows the original 'mycolours' list followed by the extended 'mylist'.

- Index()



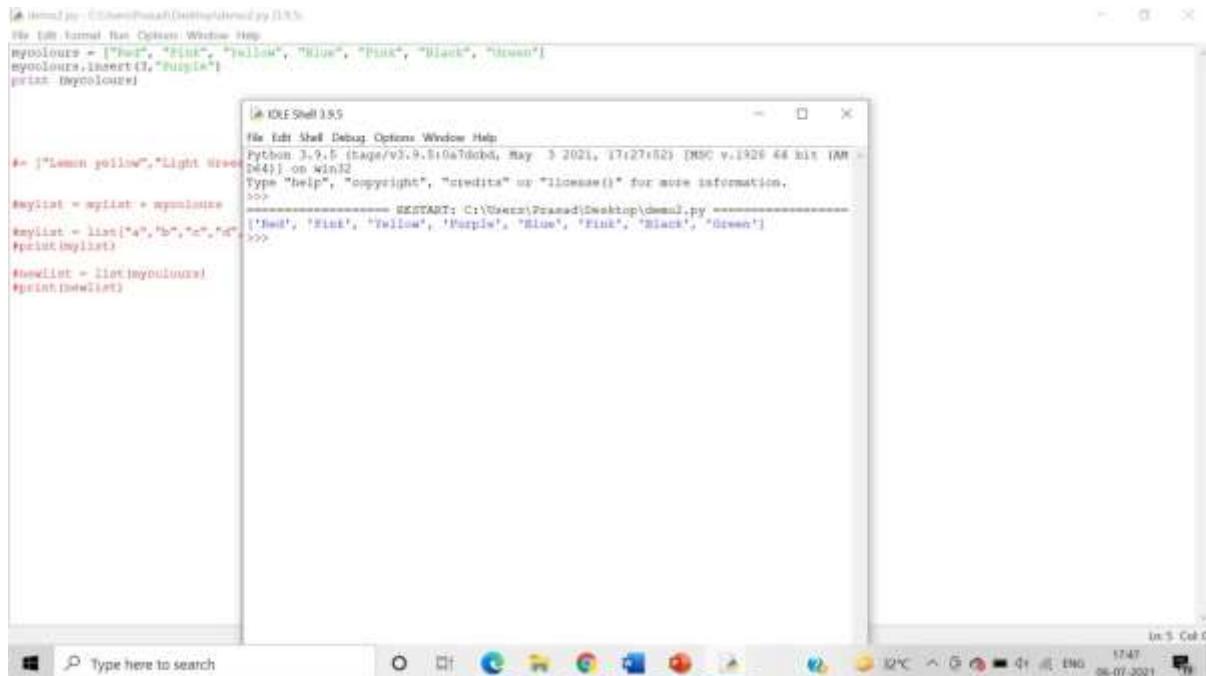
```
[f] dmcl2.py - C:\Users\Prasad\Desktop\dmcl2.py (1/3)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Brown"]
mylist = ["Brown", "Lemon yellow"]
myindex = mycolours.index("Pink")
print (myindex)

# ("Lemon yellow","Light orange")
#mylist = mylist + mycolours
#mylist = list("a","b","c","d")
#print(mylist)

#newlist = list(mycolours)
#print(newlist)
```

The screenshot shows the Python script dmcl2.py running in the IDLE Shell. The script defines a list 'mycolours' and prints its index for the element 'Pink'. The output in the shell shows the index of 'Pink' in the list.

- Insert()



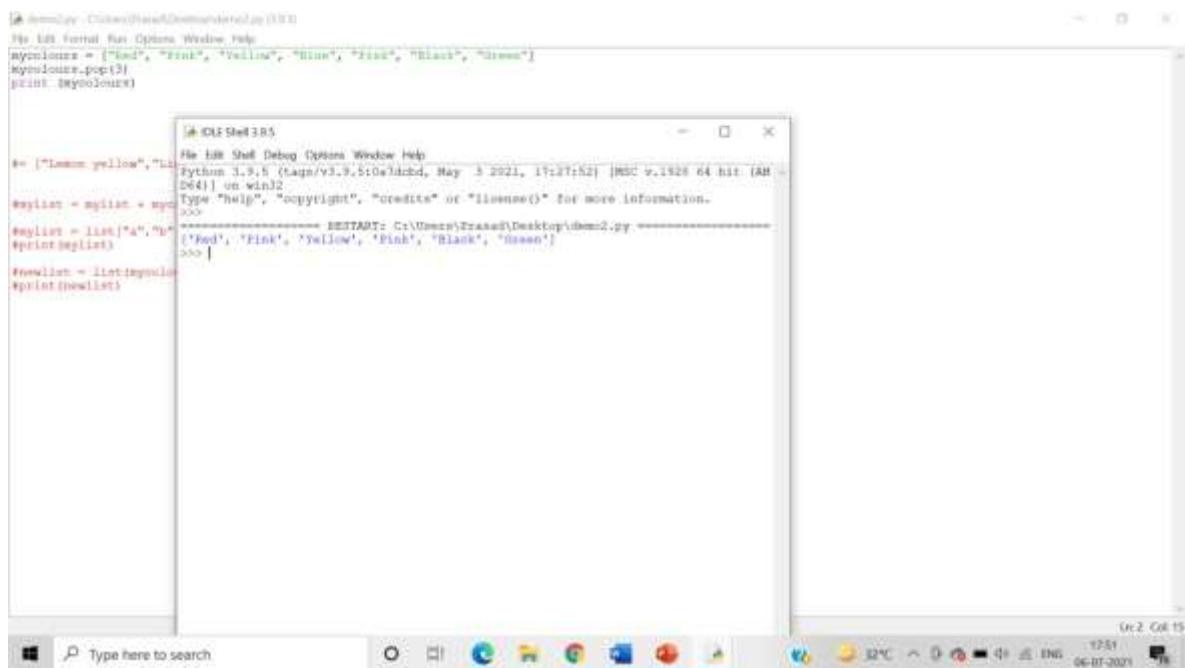
```
# idle3.py - C:\Users\Prasad\Desktop\idle3.py (1/1)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mycolours.insert(3, "Purple")
print mycolours

#<-- Lemon yellow,"Light Green"
mylist = mylist + mycolours
mylist = list("a", "b", "c", "d", >>>
#newlist = list(mycolours)
#print(newlist)
```

```
RESTART: C:\Users\Prasad\Desktop\idle3.py
['Red', 'Pink', 'Yellow', 'Purple', 'Blue', 'Pink', 'Black', 'Green']

in S Col 0
```

- Pop()



```
# idle3.py - C:\Users\Prasad\Desktop\idle3.py (1/1)
File Edit Shell Options Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mycolours.pop(3)
print mycolours

#<-- Lemon yellow,"Light Green"
mylist = mylist + mycolours
mylist = list("a", "b", >>>
#newlist = list(mycolours)
#print(newlist)
```

```
RESTART: C:\Users\Prasad\Desktop\idle3.py
['Red', 'Pink', 'Yellow', 'Pink', 'Black', 'Green']

in S Col 15
```

- Remove()

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
# demo2.py - C:\Users\Praasad\Desktop\demo2.py (1/1)
#> #!/usr/bin/python3
#> # This program has colors. Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mycolours.remove("Pink")
print(mycolours)
```

The IDLE shell window displays the output of the code:

```
[& IDLE Shell 1.95
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:\Users\Praasad\Desktop\demo2.py
['Red', 'Yellow', 'Blue', 'Pink', 'Black', 'Green']
>>>]
```

The taskbar at the bottom shows various application icons.

- Reverse()

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
# demo2.py - C:\Users\Praasad\Desktop\demo2.py (1/1)
#> #!/usr/bin/python3
#> # This program has colors. Window Help
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
mycolours.reverse()
print(mycolours)
```

The IDLE shell window displays the output of the code:

```
[& IDLE Shell 1.95
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> RESTART: C:\Users\Praasad\Desktop\demo2.py
['Green', 'Black', 'Pink', 'Blue', 'Yellow', 'Pink', 'Red']
>>>]
```

The taskbar at the bottom shows various application icons.

- Sort()

```

# demo1.py : C:\Users\Praasad\Desktop\demo1.py (1.8k)
# File Edit Shell Debug Options Window Help
# Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AM
# D64)] on win32
# Type "help", "copyright", "credits" or "license()" for more information.
>>> mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
>>> mycolours.sort()
>>> print(mycolours)
['Lemon yellow', 'Light blue', 'Pink', 'Pink', 'Red', 'Yellow', 'Black', 'Green']

A IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mylist = mylist + mycolours
>>> mylist = list(*a,*b,*c)
>>> print(mylist)
>>> newList = list(mycolours)
>>> print(newList)

```

The screenshot shows the Python IDLE interface. The code in the editor window is `mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]` followed by `mycolours.sort()`. The output window shows the sorted list: `['Lemon yellow', 'Light blue', 'Pink', 'Pink', 'Red', 'Yellow', 'Black', 'Green']`. The status bar at the bottom right indicates it's in line 2, column 14.

Using reverse parameters:

```

# demo1.py : C:\Users\Praasad\Desktop\demo1.py (1.8k)
# File Edit Shell Debug Options Window Help
# Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AM
# D64)] on win32
# Type "help", "copyright", "credits" or "license()" for more information.
>>> mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
>>> mycolours.sort(reverse=True)
>>> print(mycolours)
['Yellow', 'Red', 'Pink', 'Pink', 'Black', 'Green', 'Blue', 'Lemon yellow']

A IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> mylist = mylist + mycolours
>>> mylist = list(*a,*b,*c)
>>> print(mylist)
>>> newList = list(mycolours)
>>> print(newList)

```

The screenshot shows the Python IDLE interface. The code in the editor window is `mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]` followed by `mycolours.sort(reverse=True)`. The output window shows the sorted list in descending order: `['Yellow', 'Red', 'Pink', 'Pink', 'Black', 'Green', 'Blue', 'Lemon yellow']`. The status bar at the bottom right indicates it's in line 2, column 28.

Declaring a function:

The screenshot shows a Windows desktop environment. In the top-left corner, there is a file named "demo2.py" with a size of 0.00 KB. To its right is an "IDLE Shell 3.9.5" window. The shell window has the following content:

```
# demo2.py, C:\Users\Pradeep\Desktop\demo2.py [18%]
# file format: Unix: LF, Windows: CRLF
# This file contains a function definition and its usage.
mycolours = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
def myfunc(x):
    return len(x)
mylist = myfunc(mycolours)
print(mylist)

# New list creation
newlist = list(mylist)
newlist.append("Lemon yellow")
newlist = newlist + mylist
newlist = list("a", "b")
print(newlist)
# New list creation
newlist = list(mylist)
print(newlist)
```

The output from the shell shows the execution of the code and the resulting lists:

```
>>> RESTART: C:\Users\Pradeep\Desktop\demo2.py
['Red', 'Pink', 'Yellow', 'Blue', 'Pink', 'Black', 'Green']
>>>
['Red', 'Pink', 'Yellow', 'Blue', 'Pink', 'Black', 'Green', 'Lemon yellow']
['Red', 'Pink', 'Yellow', 'Blue', 'Pink', 'Black', 'Green', 'Lemon yellow']
['a', 'b']
```

At the bottom of the screen, there is a taskbar with several icons and a system tray showing the date and time as 08-07-2021.

8. Write a program demonstrating conversion of string to list and list to a string

Algorithm:-

Step 1: We use the list() function to convert a string into a list.

Step 2: We use the join() function to convert the values present in the list to a single string.

Answer: Program demonstrating conversion of string to list and list to a string: Splitting a string into a list (String Methods) –

1. split()

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "Terminal" with the path "C:\Users\Franas\Desktop\demol2.py [1/1]". It contains the following Python code:

```
mystr = "this is my string"
mylist = mystr.split()
print(mylist)

mycolors = ["red", "green", "blue"]
mylist = mycolors.split(",")
print(mylist)

newlist = list("a")
print(newlist)
```

Below the terminal is the Python IDLE Shell window titled "IDLE Shell 3.9.5". It shows the Python interpreter prompt (">>>>") and the output of the executed code. The output includes the list split from the string and the list created from the character "a".

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "Terminal" with the path "C:\Users\Franas\Desktop\demol2.py [1/1]". It contains the same Python code as the previous screenshot:

```
mystr = "thisismystring"
mylist = mystr.split()
print(mylist)

mycolors = ["red", "green", "blue"]
mylist = mycolors.split(",")
print(mylist)

newlist = list("a")
print(newlist)
```

Below the terminal is the Python IDLE Shell window titled "IDLE Shell 3.9.5". It shows the Python interpreter prompt (">>>>") and the output of the executed code. The output includes the list split from the string and the list created from the character "a".

Splitting at the occurrence of “i”

```
# Demo2.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
# File Edit View Insert Options Window Help
mystr = "Thisisimystring"
mylist = mystr.split('i')
print(mylist)

#def myfunc(x):
#    return len(x)
#mycolours.sort(key = myfunc)
#print(mycolours

# ("Lemon yellow

#mylist = mylist
#mylist = list(*4)
#print(mylist)

#newlist = list(mylist)
#print(newlist)
```

The screenshot shows a Windows desktop with two Python IDLE windows open. Both windows have the title bar 'Demo2.py - C:\Users\Prasad\Desktop\demo2.py (1.0)'. The left window contains the code provided above, which splits the string 'Thisisimystring' into ['Th', 's', 's', 'm', 'y', 'string']. The right window shows the same code with a few lines removed, specifically the ones starting with '#def myfunc(x):', '#mylist = mylist', and '#mylist = list(*4)'. This results in an error message: 'SyntaxError: invalid syntax'.

```
# Demo2.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
# File Edit View Insert Options Window Help
mystr = "Thisisimystring"
mylist = mystr.split('i',1)
print(mylist)

#def myfunc(x):
#    return len(x)
#mycolours.sort(key = myfunc)
#print(mycolours

# ("Lemon yellow", "Light Green

#mylist = mylist + mycolours
#mylist = list(*4, *4, *4)
#print(mylist)

#newlist = list(mycolours)
#print(newlist)
```

This screenshot shows the same code as the previous one, but with a different parameter for the split function: 'mystr.split('i',1)'. This results in a single-element list ['Thisisimystring'] because the 'i' character is the first character of the string and there is no second 'i' to split it further.

2. rsplit()

```
[4] In [1]: jay: C:\Users\Prasad\Desktop\dsmc2.py [D:\DSMC]
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d 05-08-2021) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\dsmc2.py =====
('Thisismystr', 'hg')
>>>
#def myfunction():
#    return len(x)
#mycolours.sort(key = myfunction)
#print( mycolours)

#*= {"Lemon yellow","Light

#mylist = mylist + mycolours

#mylist = list(*a,*b,*c)
#print(mylist)

#newlist = list(mycolours)
#print(newlist)
```

```
# demo1.py - C:\Users\Prasad\Desktop\Democode\demo1.py [1.5]
File Edit Shell Options Window Help
mystr = "Thisisamazing"
mylist = mystr.split("i",2)
print(mylist)
mylist = mystr.split("i",2)
print(mylist)

#mycolours = ['red','blue','green','yellow']
#mylist = mylist + mycolours
#print(mylist)

#def myfunc(x):
#    return len(x)
#mycolours.sort(key = myfunc)
#print(mycolours)

#="Lemon yellow"

#mylist = mylist + list("a")
#print(mylist)
#newlist = list(mylist)
#print(newlist)
```

```
[1] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo1.py -----
['This', 'is', 'amazing']

[2] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo1.py -----
This
is
my
string
>>>
```

3. splitlines()

```
# demo2.py - C:\Users\Prasad\Desktop\Democode\demo2.py [1.5]
File Edit Shell Options Window Help
mystr = "This\nis\namazing"
print(mystr)

#mylist = mystr.splitlines()
#print(mylist)
#newlist = list(mylist)
#print(newlist)

#mycolours = ['red','blue','green','yellow']
#mylist = mylist + mycolours
#print(mylist)

#def myfunc(x):
#    return len(x)
#mycolours.sort(key = myfunc)
#print(mycolours)

#="Lemon yellow"

#mylist = mylist + list("a")
#print(mylist)
#newlist = list(mylist)
#print(newlist)
```

```
[1] IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
This
is
my
string
>>>
```

```
# hexdump.py - C:\Users\Prasad\Desktop\democ2.py [135]
# file type: Python Script Window Help
mystr = "Thisisinsmystring"
mylist = mystr.splitlines()
print(mystr)
print(mylist)

#mylist = []
#print(mylist)

#def myfunction():
#    return []
#mylist.append("This")
#mylist.append("is")
#mylist.append("my")
#mylist.append("string")

#>>> print(mylist)

#=> ["This", "is", "my", "string"]

#def myfunction():
#    return []
#mylist.append("This")
#mylist.append("is")
#mylist.append("my")
#mylist.append("string")

#>>> print(mylist)

#=> ["This", "is", "my", "string"]
```

```
# hexdump.py - C:\Users\Prasad\Desktop\democ2.py [135]
# file type: Python Script Window Help
mystr = "Thisisinsmystring"
mylist = mystr.splitlines()
print(mystr)
print(mylist)

#mylist = []
#print(mylist)

#def myfunction():
#    return []
#mylist.append("This")
#mylist.append("is")
#mylist.append("my")
#mylist.append("string")

#>>> print(mylist)

#=> ["This", "is", "my", "string"]

#def myfunction():
#    return []
#mylist.append("This")
#mylist.append("is")
#mylist.append("my")
#mylist.append("string")

#>>> print(mylist)

#=> ["This", "is", "my", "string"]
```

Joining a list into a string –

1. join() – list method

```
# demol.py - C:\Users\Prasad\Desktop\demol.py (1/3)
File Edit Format Run Options Window Help
mylist = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
x = ","
join(mylist)
print(x)

#mylist = mystr.split()
#print (mylist)
#mylist = mystr.rsplit()
#print (mylist)

#mylist = "This\nis\nme"
#mylist = mystr.rstrip()
#print (mystr)
#print (mylist)

#def myfunc(s):
#    return len(s)
#mycolours.sort(key = myfunc)
#print (mycolours)

#e ("Lemon yellow")

mylist = mylist +
mylist = list("a","b")
#print(mylist)
```

The IDLE Shell window shows the output of the code:

```
>>> RESTART: C:\Users\Prasad\Desktop\demol.py
bad-Pink-Yellow-Blue-Pink-Black-Green
```

```
# demol.py - C:\Users\Prasad\Desktop\demol.py (1/3)
File Edit Format Run Options Window Help
mylist = ["Red", "Pink", "Yellow", "Blue", "Pink", "Black", "Green"]
x = ","
join(mylist)
print(x)

#mylist = mystr.split()
#print (mylist)
#mylist = mystr.rsplit()
#print (mystr)
#print (mylist)

#mylist = "This\nis\nme"
#mylist = mystr.rstrip()
#print (mystr)
#print (mylist)

#def myfunc(s):
#    return len(s)
#mycolours.sort(key = myfunc)
#print (mycolours)

#e ("Lemon yellow","b")

mylist = mylist +
mylist = list("a","b")
#print(mylist)
```

The IDLE Shell window shows the output of the code:

```
>>> RESTART: C:\Users\Prasad\Desktop\demol.py
bad-Pink-Yellow-Blue-Pink-Black-Green
```

```
[1] demo1.py - C:\Users\Prasad\Desktop\demol1.py (1.85)
File Edit Shell Options Window Help
mylist = ["Red", "Pink", "Yellow", "Blue", "Fink", "Black", "Green"]
x = ", ".join(mylist)
print(x)

[2] IDLE Shell 1.95
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demol1.py
Red Pink Yellow Blue Fink Black Green
>>>

mylist = mystr.split(",")
print(mylist)
mylist = mystr.replace(",","")
print(mylist)

mystr = "This\nis\nmy"
mystr = mystr.split("\n")
print(mystr)
print(mylist)

def myfunc(x):
    # return len(x)
    mycolours = sort(key = x)
    print(mycolours)

#> ("Lemon yellow", "Lg")

mylist = mylist + mystr
mylist = list("a", "b", c)
print(mylist)

[3] Type here to search  O  C:\  E:\  F:\  G:\  H:\  I:\  J:\  K:\  L:\  M:\  N:\  O:\  P:\  Q:\  R:\  S:\  T:\  U:\  V:\  W:\  X:\  Y:\  Z:\  15:48  06-07-2021
```

3. Use 'for in'

```
[1] demo1.py - C:\Users\Prasad\Desktop\demol2.py (1.85)
File Edit Shell Options Window Help
mylist = ["Red", "Pink", "Yellow", "Blue", "Fink", "Black", "Green"]
mystr = ""
for x in mylist:
    mystr = mystr + x
print(mystr)

[2] IDLE Shell 1.95
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demol2.py
RedPinkYellowBlueFinkBlackGreen
>>>

num = []
n = int(input("go on"))
for i in range(0, n):
    x = int(input(" "))
    num.append(x)

x = ", ".join(str(num))
print(x)

mylist = mystr
print(mylist)
mylist = mystr
print(mylist)

mystr = "This
myList = mystr
print(mylist)

[3] Type here to search  O  C:\  E:\  F:\  G:\  H:\  I:\  J:\  K:\  L:\  M:\  N:\  O:\  P:\  Q:\  R:\  S:\  T:\  U:\  V:\  W:\  X:\  Y:\  Z:\  21:03  06-07-2021
```

9. Write a program demonstrating declaration, accessing- Indexing and splicing of tuples and its elements.

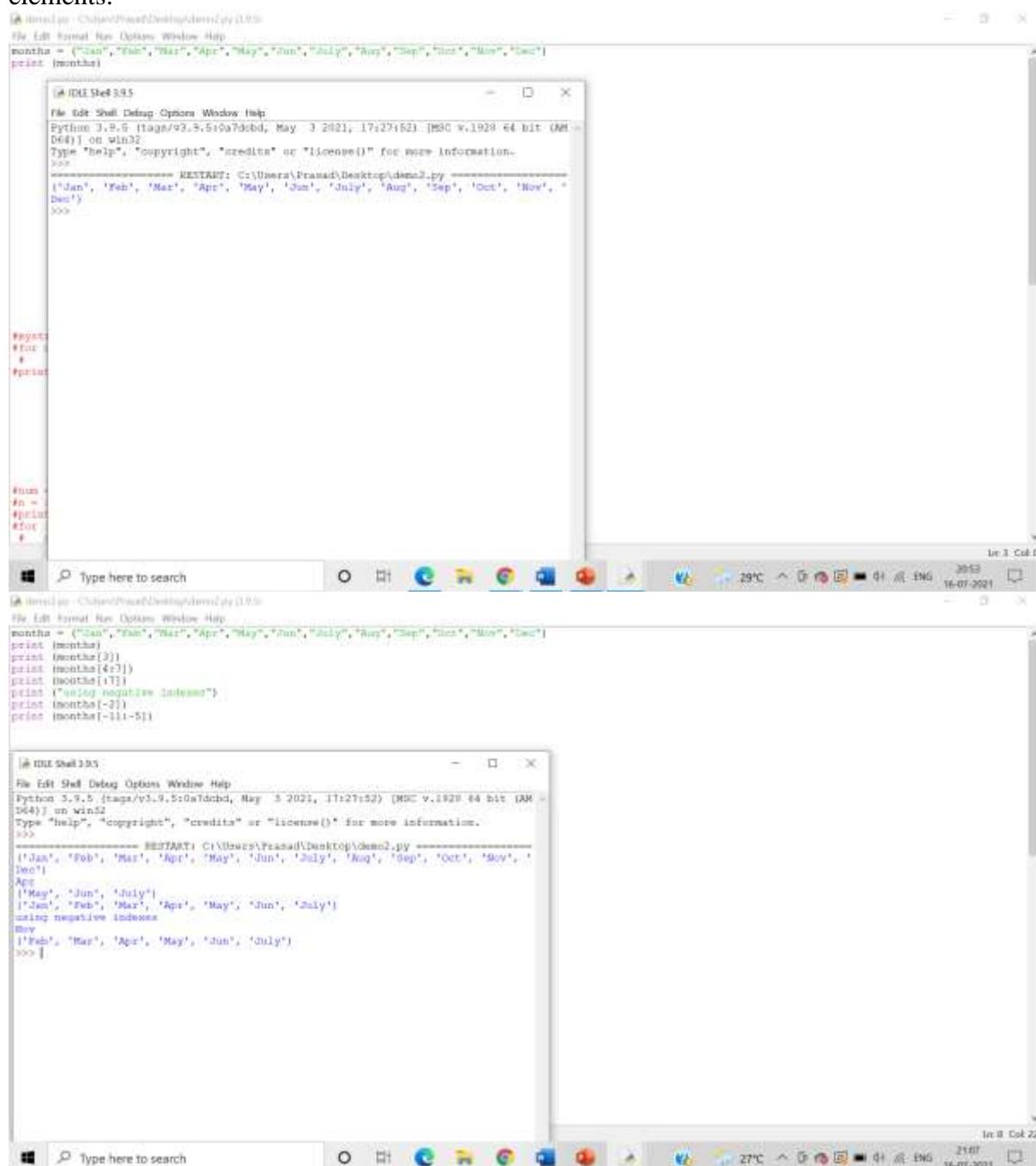
Algorithm:-

Step 1: A tuple is declared by placing all the items (elements) inside parentheses (), separated by commas and assigning it to a variable.

Step 2: To access the tuple items refer to the index number insed the index operator[] (Indexing).

Step 3: By inputting a range of indexes by specifying where to start and where to end the range we can print a subset of the tuple (splicing)

Answer: Program demonstrating declaration, accessing- Indexing and splicing of tuples and its elements:



```

#> C:\Users\Prasad\Desktop>python demo1.py
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d2, May 3 2021, 17:37:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
>>> print(months)
('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')

#> C:\Users\Prasad\Desktop>python demo2.py
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d2, May 3 2021, 17:37:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
>>> print(months[3])
print(months[4])
print(months[1:3])
print("using negative indexes")
print(months[-2])
print(months[-11:-5])

```

The image shows three identical Python IDLE windows stacked vertically. Each window has a title bar "IDLE (Python 3.9.5)" and a menu bar "File Edit Shell Debug Options Window Help". The status bar at the bottom of each window shows "In 8 Col 0" and the date "16-07-2021".

The code in all three windows is:months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")months[12] = "New"print(months)

The output in the first window shows a syntax error:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
SyntaxError (most recent call last):
  File "C:\Users\Prasad\Desktop\demo1.py", line 2, in <module>
    months[12] = "New"
TypeError: 'tuple' object does not support item assignment
>>>
```

The output in the second window shows a syntax error:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
SyntaxError (most recent call last):
  File "C:\Users\Prasad\Desktop\demo2.py", line 2, in <module>
    x = ('abc', -4.24e93, 10+6.0j, 'xyz')
TypeError: 'tuple' object does not support item assignment
>>>
```

The output in the third window shows a syntax error:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo3.py
SyntaxError (most recent call last):
  File "C:\Users\Prasad\Desktop\demo3.py", line 2, in <module>
    x = ('abc', -4.24e93, 10+6.0j, 'xyz')
TypeError: 'tuple' object does not support item assignment
>>>
```

10. Write a program demonstrating input of tuple elements from the user

Algorithm:-

Step 1: Use the input() function to take user input and assign it to a variable

Step 2: Use the split function to split each value in the variable with a “,” and assign it a new tuple variable.

Step 3: Print the tuple variable

Answer: Program demonstrating input of tuple elements from the user:

The screenshot shows a Windows desktop environment. In the top-left corner, there is a Notepad window titled "Untitled - C:\Users\Praasad\Desktop\demo2.py (1.00)". The code in the Notepad is:

```
# First create an empty list
num = []
# Enter number of elements as input
n = int(input("Enter number of elements : "))
print ("For entering values")
# Iterating till the range
for i in range (0,n):
    x = int(input("Enter here<=3"))
    num.append(x) # adding the element
print(num)
tupnum = tuple(num)
print (tupnum)
```

In the bottom-right corner, there is an IDLE Shell window titled "IDLE Shell 3.9.5". The shell output is:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb4, May  3 2021, 17:27:52) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praasad\Desktop\demo2.py =====
Enter number of elements : 4
For entering values
enter here<=3
enter here<=3
enter here<=3
enter here<=3
(22, 36, 2, 39)
(22, 36, 2, 39)
>>>
```

At the bottom of the desktop screen, there is a taskbar with icons for File Explorer, Edge browser, File Manager, Task View, and others. The system tray shows the date as 16-07-2021 and the time as 22:59.

11. Write a program demonstrating printing of tuple elements using for in and range() function

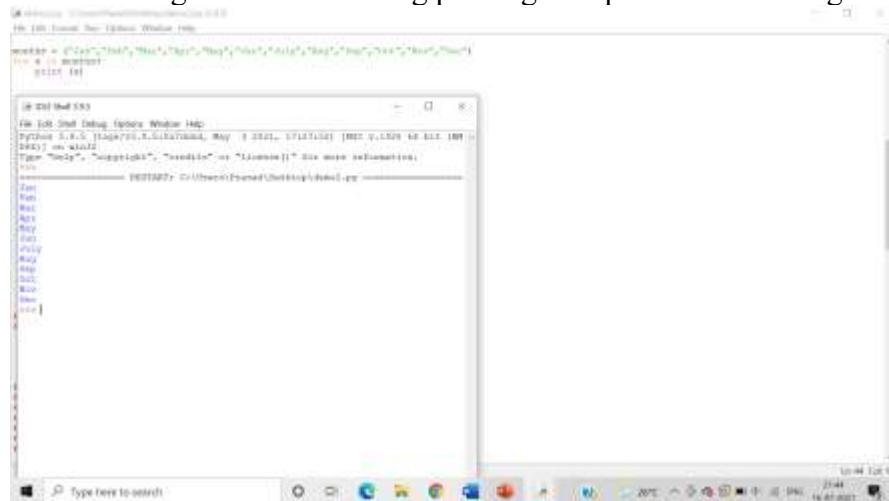
Algorithm:-

Step 1: Declare a tuple by assigning it to a variable

Step 2: The range() gives us a sequence of numbers in between the two integers given to it.

Step 3: We can use the in and range() function to print the tuple elements using a for loop.

Answer: Program demonstrating printing of tuple elements using “for in”:



```
tuple_in.py
tuple = ("Java", "Python", "C", "C++", "C#", "Java Script", "VBScript", "HTML", "CSS", "JavaScript", "SQL", "MySQL", "Oracle", "DBMS")
for i in range(0, len(tuple)):
    print(tuple[i])
```

Program demonstrating printing of tuple elements using range() function:



```
tuple_range.py
tuple = ("Java", "Python", "C", "C++", "C#", "Java Script", "VBScript", "HTML", "CSS", "JavaScript", "SQL", "MySQL", "Oracle", "DBMS")
for i in range(0, len(tuple)):
    print(tuple[i])
```

12. Write a program demonstrating len() and del() function on tuple.

Algorithm:-

Step 1: The `len()` function is used to obtain the length of the tuple

Step 2: The **del()** function **deletes all the elements in range** starting from index ‘a’ till ‘b’ mentioned in arguments.

Answer: Program demonstrating len() function on tuple:

```
# Python Program demonstrating len() function on tuple.
# len() function has Update Window Help
months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
print (len)

[!] Python 3.9.5 (tags/v3.9.5:f9ad8d2, May 3 2021, 13:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\dem02.py -----
12
>>>

#mystr = ""
#for x in mylist:
#    mystr = mystr + x
#print (mystr)
```

Program demonstrating del() function on tuple:

```
Program: C:\Users\Prasad\Desktop\demo1.py
File Edit Format View Options Window Help
months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
del(months)
print(months)

iPython Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbf, May 3 2021, 17:27:52) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
Traceback (most recent call last):
  File "C:\Users\Prasad\Desktop\demo2.py", line 1, in <module>
    print(months)
NameError: name 'months' is not defined
>>>
```

13. Write a program demonstrating usage of ‘+’ and ‘*’ operators on tuple

Algorithm:-

Step 1: The ‘+’ operator is used to concatenate the 2 tuples into a single tuple.

Step 2: The ‘*’ operator is used to multiply the tuple “n” times

Answer: Program demonstrating usage of ‘+’ operators on tuple:

The screenshot shows two windows. The top window is a code editor with the following Python script:

```
tuple1 = ("a","b","c")
tuple2 = (1,2,3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

The bottom window is the Python IDLE Shell 3.9.5, showing the output of the script:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo1.py
('a', 'b', 'c', 1, 2, 3)
```

The taskbar at the bottom shows the date as 16-07-2021 and the time as 21:23.

Program demonstrating usage of ‘*’ operators on tuple:

The screenshot shows two windows. The top window is a code editor with the following Python script:

```
tuple1 = ("a","b","c")
tuple2 = (1,2,3)
tuple3 = tuple1 * 4
print(tuple3)
```

The bottom window is the Python IDLE Shell 3.9.5, showing the output of the script:

```
[idle] >>> RESTART: C:\Users\Prasad\Desktop\demo2.py
('a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c')
```

The taskbar at the bottom shows the date as 16-07-2021 and the time as 21:27.

14. Write a program demonstrating “in” and “not in” keywords on tuple

Algorithm:-

Step 1: - The ‘in’ operator is used to **check if an element is present** in the tuple or not. Returns true if element is present in tuple else returns false.

Step 2: The ‘not in’ operator is used to **check if an element is not present** in the tuple or not. Returns true if element is not present in tuple else returns false.

Answer: Demonstrating “in” keyword on tuple:

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
if "Jan" in months:
    print("exists")
else:
    print("does not exist")
```

The IDLE shell window shows the output of the code:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
exists
```

The taskbar at the bottom of the screen shows various application icons and the system clock indicating 22:01 on 16-07-2021.

Demonstrating “not in” keyword on tuple:

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
if "Alma" not in months:
    print("true")
else:
    print("false")
```

The IDLE shell window shows the output of the code:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
true
```

The taskbar at the bottom of the screen shows various application icons and the system clock indicating 22:02 on 16-07-2021.

15. Write set of programs for demonstrating the usage of all the different Tuple methods along with their variations

Algorithm:-

Step 1: The count() method returns the number of times a specified value occurs in a tuple
Step 2: The index () searches the tuple for a specified value and returns the position of where it was found. It raises an exception if item is not found

Answer: Programs for demonstrating the usage of count() :

The screenshot shows the Python IDLE shell with the following code and output:

```
1 <input> C:\Users\Praaad\Desktop\demo1.py (1.0)#
2 File Edit Kernel Help Options Window Help
3 months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
4 print(months)
5 print(months[3])
6 print(months[4+7])
7 print(months[-1])
8 print("Using negative indexes")
9 print(months[-2])
10 print(months[-31:-5])
11
12 x = months.count("Feb")
13 print(x)
```

```
1 <input> IDE Shell 3.9.5
2 File Edit Shell Debug Options Window Help
3 Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AM
4 D64)] on win32
5 Type "help", "copyright", "credits" or "license()" for more information.
6 >>>
7 RESTART: C:\Users\Praaad\Desktop\demo1.py
8 ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')
9
10 ('May', 'Jun', 'July')
11 ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July')
12 using negative indexes
13 Nov
14 ('Feb', 'Mar', 'Apr', 'May', 'Jun', 'July')
15
16 >>>
```

Programs for demonstrating the usage of index() :

The screenshot shows the Python IDLE shell with the following code and output:

```
1 <input> C:\Users\Praaad\Desktop\demo2.py (1.0)#
2 File Edit Shell Debug Options Window Help
3 months = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec")
4 print(months)
5 print(months[3])
6 print(months[4+7])
7 print(months[-1])
8 print("Using negative indexes")
9 print(months[-2])
10 print(months[-31])
11
12 x = months.index("Feb")
13 print(x)
```

```
1 <input> IDE Shell 3.9.5
2 File Edit Shell Debug Options Window Help
3 Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AM
4 D64)] on win32
5 Type "help", "copyright", "credits" or "license()" for more information.
6 >>>
7 RESTART: C:\Users\Praaad\Desktop\demo2.py
8 ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')
9
10 ('May', 'Jun', 'July')
11 ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July')
12 using negative indexes
13 Nov
14 ('Feb', 'Mar', 'Apr', 'May', 'Jun', 'July')
15
16 >>>
```

rindex() is not applicable in tuples:

The screenshot shows a Windows desktop environment. In the center is a terminal window titled 'cmd' with the path 'C:\Users\Prasad\Desktop'. The window contains Python code demonstrating the use of the rindex method on a tuple. The code defines a tuple of months and prints its elements. It then attempts to use the rindex method on the tuple, which results in an AttributeError. The terminal window has a dark theme. Below it is a standard Windows taskbar with icons for Start, Task View, File Explorer, Edge, Google Chrome, and others. The system tray shows the date as 16-07-2021, the time as 23:27, and the temperature as 26°C. The status bar at the bottom right indicates 'In 13. Cok 12'.

```
prasad@prasad-OptiPlex-5090: ~
$ python -c "months = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'); print(months[3]); print(months[4:7]); print(months[-1]); print(months[-7]); print(months[-11:-5]); x = months.rindex('Feb'); print(x)

[1] 11188 python3.9 -c "months = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'); print(months[3]); print(months[4:7]); print(months[-1]); print(months[-7]); print(months[-11:-5]); x = months.rindex('Feb'); print(x)
prasad@prasad-OptiPlex-5090: ~
$
```

```
>>> months = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')
>>> print(months[3])
>>> print(months[4:7])
>>> print(months[-1])
>>> print(months[-7])
>>> print(months[-11:-5])
>>> x = months.rindex('Feb')
>>> print(x)

[1] 11188 python3.9 -c "months = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'); print(months[3]); print(months[4:7]); print(months[-1]); print(months[-7]); print(months[-11:-5]); x = months.rindex('Feb'); print(x)
prasad@prasad-OptiPlex-5090: ~
$
```

```
File "C:\Users\Prasad\Desktop\demo2.py", line 11, in <module>
  x = months.rindex("Feb")
AttributeError: 'tuple' object has no attribute 'rindex'
>>>
```

16. Write a Python Program to Calculate the Average of Numbers entered by the user in the user-defined list

Algorithm:-

Step 1: Declare a User defined list and assign it to a variable.

Step 2: For the user defined range use the $\text{avg}=\text{sum}(a)/n$ formula to calculate the the average of the numbers present in the user defined list.

The screenshot shows a Windows desktop environment. In the center is a terminal window titled 'IDLE Shell 3.8'. The code inside the window is a Python script named 'demo2.py' which calculates the average of a list of numbers. The script uses a for loop to iterate through a range of numbers, appending them to a list. It then calculates the sum of the list and divides it by the number of elements to find the average. The terminal window also shows the command prompt and some system information at the bottom. Below the terminal is a taskbar with various icons, and at the very bottom is a system tray with icons for battery, volume, and date/time.

```
# demo2.py : C:\Users\Praashad\Desktop\demo2.py [1.8]
# An IDE is not needed. Windows has its own built-in Python shell.
# Enter the number of elements to be inserted: 4
# Enter element: 33
# Enter element: 340
# Enter element: 44
# Enter element: 33
# Average of elements in the list: 113.5
>>>

#num = []
#n = int(input("Enter the number of elements to be inserted: "))
#for i in range(0,n):
#    a = int(input())
#    num.append(a)
#print(num)

#x = " ".join(str(i) for i in num)
#print(x)
```

17. Write a Program to Generate Random Numbers from 1 to 10 and add them in a defined list

Algorithm:-

Step 1: Import random function

Step 2: Input the range of the list

Step 3: Initialize an empty list

Step 4: Set i=0 and use a while loop where when $i < 0$ the loop executes its task.

Step 5: Using The randint() function to generate random integer and use the append() function to add the integer to the list.

Step6: Print the list.

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "DOS Shell 3.3.5" running Python 3.9.5. The terminal displays a script named "dmc01.py" which generates a list of random integers. The user enters "4" when prompted for the number of elements. The output shows the list [97, 82, 44, 69]. Below the terminal is a code editor window titled "Untitled - ChainedProject\2021\damc01.py (1.0.0)". The code in the editor is as follows:

```
import random
num=int(input("Enter the number of elements in a list"))
list=[]
i = 0
while i < num:
    n = random.randint(0,100)
    list.append(n)
    i+=1
print(list)

# a file named "myfile", will be opened with r
file = open ("C:/Users/Praaad/Documents/myfile"
#this will print the content

#for each in file:
#    print(each)
#print ("-----")
#print (file.readline())

```

18. Write a program demonstrating a creation of 2D list and print it.

Algorithm:-

Step 1: Initialise the number of rows and columns

Step 2: Use Nested for loops to print out the 2D matrix in given dimensions.

The screenshot shows a Windows desktop environment. In the center is a terminal window titled "IDLE Shell 3.9.5". The code inside the window is:

```
# IDLE: C:\Users\Prasad\Desktop\demo2.py
# File Edit Shell Debug Options Window Help
# Python 3.9.5 (tags/v3.9.5:0.51.0, May 3 2021, 17:21:52) [MSC v.1928 64 bit (AMD64)] on win32
# Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
>>>
```

The terminal window has a status bar at the bottom indicating "Line: 4 Col: 0". Below the terminal window is a taskbar with several icons, including the Start button, File Explorer, Task View, Edge browser, and others. A search bar on the taskbar contains the placeholder "Type here to search".

19. Write a program to create a user defined 2D list and print it. Take dimensions from the user.

Algorithm:-

Step 1: Input the number of rows and columns required.

Step 2: Initialize an empty array list

Step 3: Use Nested for loops so as to initialize the matrix to the user defined rows and columns.

Step 4: Input the values of the matrix.

Step 5: Use the append function to append the values to the matrix of user defined rows and columns.

The screenshot shows a Windows desktop environment. On the left is a code editor window titled "idle.py - C:\Users\Prasad\Desktop\idle.py (1.0.0)" containing Python code for creating a 2D list. On the right is a terminal window titled "IDLE Shell 3.9.5" showing the execution of the script and its output.

```
#> idle.py - C:\Users\Prasad\Desktop\idle.py (1.0.0)
File Edit Shell Options Window Help
rows = int(input("Enter your rows:"))
cols = int(input("Enter your cols:"))
arr = []
for i in range(rows):
    for j in range(cols):
        print ("Enter your entry")
        arr.append(int(input()))
    print(arr)

# a file named "myfile", will be opened with reading mode
file = open('C:/Users/Prasad/Documents/myfile.txt','r')
#This will print the content:
for each in file:
    print(each)
print ("-----")
print(file.readline())

#> Enter your rows:3
#> Enter your cols:2
#> Enter your entry
#> 34
#> [34]
#> Enter your entry
#> 532
#> [34, 532]
#> Enter your entry
#> 1
#> [34, 532, 1]
#> Enter your entry
#> 44
#> [34, 532, 1, 44]
#> Enter your entry
#> 33
#> [34, 532, 1, 44, 33]
#> -----
```

20. Write a program for addition of two matrices

Algorithm:-

Step 1: Input the number of rows and columns.

Step 2: Initialize 2 empty arrays and a result matrix where all values are 0.

Step 3: Use Nested for loops so as to initialize the matrix's to the user defined rows and columns.

Step 4: Input the values of the 2 matrix's so as to obtain a 3x3 matrix.

Step 5: Use nested for loops and use the index position of each values of both the matrix as to add each value in a given position of a matrix to the corresponding value of the same value in the second matrix.

The screenshot shows a Windows desktop with a Python script named 'dem03.py' open in a code editor and its execution in the IDLE Shell 3.9.5. The code defines two 3x3 matrices, adds them, and prints the result. The IDLE shell shows the input of matrix elements and the resulting sum.

```
#!/usr/bin/python3.9.5
# Enter the Number of Rows
rows = int(input("Enter the Number of rows : "))
# Enter the Number of Columns
columns = int(input("Enter the Number of Columns : "))

print("Enter the elements of First Matrix:")
matrix_a = [[int(input()) for i in range(columns)] for i in range(rows)]
print("First Matrix is : ")
for n in matrix_a:
    print(n)

print("Enter the elements of Second Matrix:")
matrix_b = [[int(input()) for i in range(columns)] for i in range(rows)]
for n in matrix_b:
    print(n)

result=[[0 for i in range(columns)] for i in range(rows)]

for i in range(rows):
    for j in range(columns):
        result[i][j] = matrix_a[i][j]+matrix_b[i][j]

print("The Sum of Above Two Matrices is : ")
for k in result:
    print(k)
# a file named "myfile", will be opened with reading mode.
#file = open ("C:/Users/Pramad/Documents/myfile.txt",'r')
#this will print the content.

#for each in file:
#    print(each)
#print ("-----")
#print( file.readline())
#
```

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d0dd0, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:/Users/Pramad/Desktop/dem03.py
Enter the Number of rows : 3
Enter the Number of Column : 3
Enter the elements of First Matrix:
32
32
32
37
37
37
48
48
48
First Matrix is:
[32, 32]
[37, 37]
[48, 48]
Enter the elements of Second Matrix:
31
47
5
1
[31, 47]
[5, 1]
The Sum of Above Two Matrices is :
[63, 89]
[42, 56]
>>>
```

21. Write a program for subtraction of two matrices

Algorithm:-

Step 1: Input the number of rows and columns.

Step 2: Initialize 2 empty arrays and a result matrix where all values are 0.

Step 3: Use Nested for loops so as to initialize the matrix's to the user defined rows and columns.

Step 4: Input the values of the 2 matrix's so as to obtain a 3x3 matrix.

Step 5: Use nested for loops and use the index position of each values of both the matrix as to subtract each value in a given position of a matrix to the corresponding value of the same value in the second matrix.

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tag: v3.9.5, 0a7dcd, May 3 2021, 17:27:53) [N
-- 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
7
Enter the Number of rows for the first matrix: 3
Enter the Number of Columns for the First matrix: 3
Enter the elements of First Matrix:
4
5
3
23
45
3
First Matrix is:
[[4, 5, 3],
 [23, 45, 3]]
Enter the Number of Columns for the second matrix: 2
Enter the elements of Second Matrix:
3
3
2
45
42
5
[3, 3]
[2, 45]
[42, 5]

Matrix_A X Matrix_B is:
[148, 252]
[281, 3109]
>>>
```

22. Write a program for multiplication of two matrices

Algorithm:-

Step 1: Input the number of rows and columns.

Step 2: Initialize 2 empty arrays and a result matrix where all values are 0.

Step 3: Use Nested for loops so as to initialize the matrix's to the user defined rows and columns.

Step 4: Input the values of the 2 matrix's so as to obtain a 3x3 matrix.

Step 5: Use nested for loops and use the index position of each values of both the matrix as to multiply each value in a given position of a matrix to the corresponding value of the same value in the second matrix.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled 'idle2.py - Untitled Document - Microsoft Word'. It contains Python code for matrix multiplication. In the background, there is a Python IDLE Shell window titled 'IDLE Shell 3.9.5'. The shell window shows the output of the executed code, which prints the multiplication result of matrix A and B. The result is a 3x3 matrix:

```
[[10, 36, 42], [66, 81, 96], [102, 126, 150]]
```

Python for Bioinformatics

Exercise 4

IV-Title: Programs demonstrating use of Sets and Dictionaries

1. Write a program demonstrating declaration and accessing of Sets and its elements.

Algorithm:

Step 1:- A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function.

Step 2:- You can access the set by printing the variable to which the set is assigned.

Answer: Program demonstrating accessing of Sets:

The screenshot shows a Windows desktop environment. In the foreground, there is a Notepad window titled "Untitled - C:\Users\Prasad\Desktop\demo2.py" containing the following Python code:

```
fruits = {"apple", "watermelon", "banana", "mango", "papaya"}  
print(fruits)
```

Below the Notepad window is the Python IDLE Shell 3.9.5 window, which displays the output of the code execution:

```
IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:5f9df22, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
{'papaya', 'banana', 'Watermelon', 'apple', 'mango'}  
>>>
```

The desktop taskbar at the bottom shows icons for File Explorer, Task View, Edge browser, Google Chrome, and File Explorer again. The system tray indicates the date as 17-07-2021, the time as 01:24, and the temperature as 27°C.

2. Write a program demonstrating len() and del() function on Sets.

Algorithm:

Step 1: The len() function is used to obtain the length of the set.

Step 2: The del() function **deletes all the elements in range** starting from index 'a' till 'b' mentioned in arguments of the set.

Answer: Program demonstrating len() function on Sets:

```
#!/usr/bin/python -C:\Users\Prasad\Desktop\demol.py(1.0)
# File Edit Shell Options Window Help
fruits = ["apple", "watermelon", "banana", "orange", "papaya"]
print (len(fruits))

#first create a
#list = []
#number of elem
n = int(input())
#print ("Go on")
#iterating till
#for i in range:
#    n = int(i)
#    n.append()
#print(n)
```

Program demonstrating del() function on Sets:

```
#!/usr/bin/python -C:\Users\Prasad\Desktop\demol.py(1.0)
# File Edit Shell Options Window Help
fruits = ["apple", "watermelon", "banana", "orange", "papaya"]
del fruits
print (fruits)

#first create a
#list = []
#number of elem
n = int(input())
#print ("Go on")
#iterating till
#for i in range:
#    n = int(i)
#    n.append()
#print(n)
```

3. Write a program demonstrating input of Set elements from the user

Algorithm:-

Step 1:-Take user input

Step 2:-Use the split() function to split the values stored and assign it to the lists variable/

Step3:- Use the set() function to store the values stored in the lists variable to convert it onto a set, Assign it to a sets variable

Step 4:- Print the sets variable.

Answer: Program for demonstrating input of Set elements from the user

```
[4] 11:45 AM - ChatredPrasad\Desktop\demol2.py (1.0.0)
File Edit Shell Debug Options Window Help
mss = set()
n = int(input("Enter number of elements : "))
print ("Ok on entering the values")
#iterating till the range
for i in range (0,n):
    x = int(input("Enter here-->"))
    mss.add(x) #adding the elements
print(mss)
```

The screenshot shows a Windows operating system interface. In the center is a Python IDLE Shell window titled 'IDLE Shell 3.9.5'. The window displays a Python script named 'demol2.py' with code for creating a set from user input. The script prompts the user for the number of elements and then iterates through a range of values, adding each to a set. The output shows the set containing the elements 34, 2, and 44. Below the IDLE window is a taskbar with various icons for applications like File Explorer, Edge, and Google Chrome. To the right of the taskbar is a system tray showing the date (17-07-2021), time (01:16), battery status (27°C), and language (ENG).

```
[4] 11:45 AM - ChatredPrasad\Desktop\demol2.py (1.0.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d0, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> ===== RESTART: C:\Users\ChatredPrasad\Desktop\demol2.py =====
Enter number of elements : 4
Ok on entering the values.
enter here-->34
enter here-->56
enter here-->44
enter here-->2
{56, 34, 2, 44}
>>>
```

4. Write set of programs for demonstrating the usage of all the different Set methods along with their variations

Algorithm:-

Step 1:- To add one item to a set use the add() method.

Step 2:- The clear() method empties the set

Step3:- The copy() method returns a copy of the set.

Answer: Program for demonstrating add()

```
#> idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry", "apple", "orange", "banana", "mango", "watermelon", "cherry", "apple", "mango", "banana", "papaya", "watermelon"}
```

The screenshot shows the Python IDLE Shell window. The code defines three sets: fruits1, fruits2, and fruits3. The fruits1 set contains {"apple", "watermelon", "banana", "mango", "papaya"}. The fruits2 set contains {"orange", "apple", "cherry", "mango"}. The fruits3 set contains {"papaya", "strawberry", "apple", "orange", "banana", "mango", "watermelon", "cherry", "apple", "mango", "banana", "papaya", "watermelon"}. A command is entered to add "cherry" to fruits1: fruits1.add("cherry"). The output shows the updated set fruits3, which now includes "cherry". The status bar at the bottom right indicates "In 9 Col 19" and the date/time "16-07-2021 23:28".

```
#> idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry", "apple", "orange", "banana", "mango", "watermelon", "cherry", "apple", "mango", "banana", "papaya", "watermelon"}
```

The screenshot shows the Python IDLE Shell window. The code defines three sets: fruits1, fruits2, and fruits3. The fruits1 set contains {"apple", "watermelon", "banana", "mango", "papaya"}. The fruits2 set contains {"orange", "apple", "cherry", "mango"}. The fruits3 set contains {"papaya", "strawberry", "apple", "orange", "banana", "mango", "watermelon", "cherry", "apple", "mango", "banana", "papaya", "watermelon"}. A command is entered to add "apple" to fruits1: fruits1.add("apple"). The output shows the updated set fruits3, which now includes "apple". The status bar at the bottom right indicates "In 15 Col 10" and the date/time "16-07-2021 23:32".

Program for demonstrating clear()

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
#fruits = ["apple", "watermelon", "banana", "mango", "papaya"]
#fruits2 = ["orange", "apple", "cherry", "mango"]
#fruits3 = ["apple", "banana", "strawberry"]

fruits1.clear()
print(fruits1)
```

The output window shows:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tag:b793.9.5.0a7d0b3, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
>>>
```

At the bottom of the IDLE window, there is a status bar displaying "In 3 Col 14".

Program for demonstrating copy()

The screenshot shows a Windows desktop with a Python IDLE window open. The code in the editor is:

```
#fruits = ["apple", "watermelon", "banana", "mango", "papaya"]
#fruits2 = ["orange", "apple", "cherry", "mango"]
#fruits3 = ["apple", "banana", "strawberry"]

fruits1.add("apple")
print(fruits1)

newFruit = fruits1.copy()
print(newFruit)
```

The output window shows:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tag:b793.9.5.0a7d0b3, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
['banana', 'mango', 'apple', 'papaya', 'watermelon']
['banana', 'mango', 'apple', 'papaya', 'watermelon']
>>>
```

At the bottom of the IDLE window, there is a status bar displaying "In 20 Col 0".

Program for demonstrating difference()

Algorithm:-

Step 1:-Difference() method returns a set containing the difference between two or more sets.

Step 2:-Difference_update() removes the items in this set that are also included in another, specified set.

Step3:- Discard() method removes the specified item.

```
# idle3.py - Created by Praasad on 16-07-2021.
# File Edit Format Run Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry"}

diff = fruits1.difference(fruits2)
print (diff)

#first create an
#num = []
#number of elem
#n = int(input())
#print ("Go on")
#iterating till
#for i in range
#    s = int(input())
#    num.append(s)
#    print (num)
```

```
# idle3.py - Created by Praasad on 16-07-2021.
# File Edit Format Run Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry"}

diff1 = fruits1.difference(fruits2)
print (diff1)
diff2 = fruits2.difference(fruits1)
print (diff2)

#first create an
#num = []
#number of elem
#n = int(input())
#print ("Go on")
#iterating till
#for i in range
#    s = int(input())
#    num.append(s)
#    print (num)
```

Program for demonstrating update()

```
#fruits1 = ['apple','watermelon','banana','mango','papaya']
fruits2 = ['orange','apple','cherry','mango']
fruits3 = ['papaya','strawberry']

fruits1.difference_update(fruits3)
print (fruits1)
```

```
[iDE Shell 3.9.5]
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:fdbd, May  3 2021, 17:27:53) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\desol.py =====
['watermelon', 'papaya', 'banana']
>>>
```

Program for demonstrating difference_update()

```
#fruits1 = ['apple','watermelon','banana','mango','papaya']
fruits2 = ['orange','apple','cherry','mango']
fruits3 = ['papaya','strawberry']

fruits2.difference_update(fruits3)
print (fruits2)
```

```
[iDE Shell 3.9.5]
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:fdbd, May  3 2021, 17:27:53) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demn2.py =====
['cherry', 'orange']
>>>
```

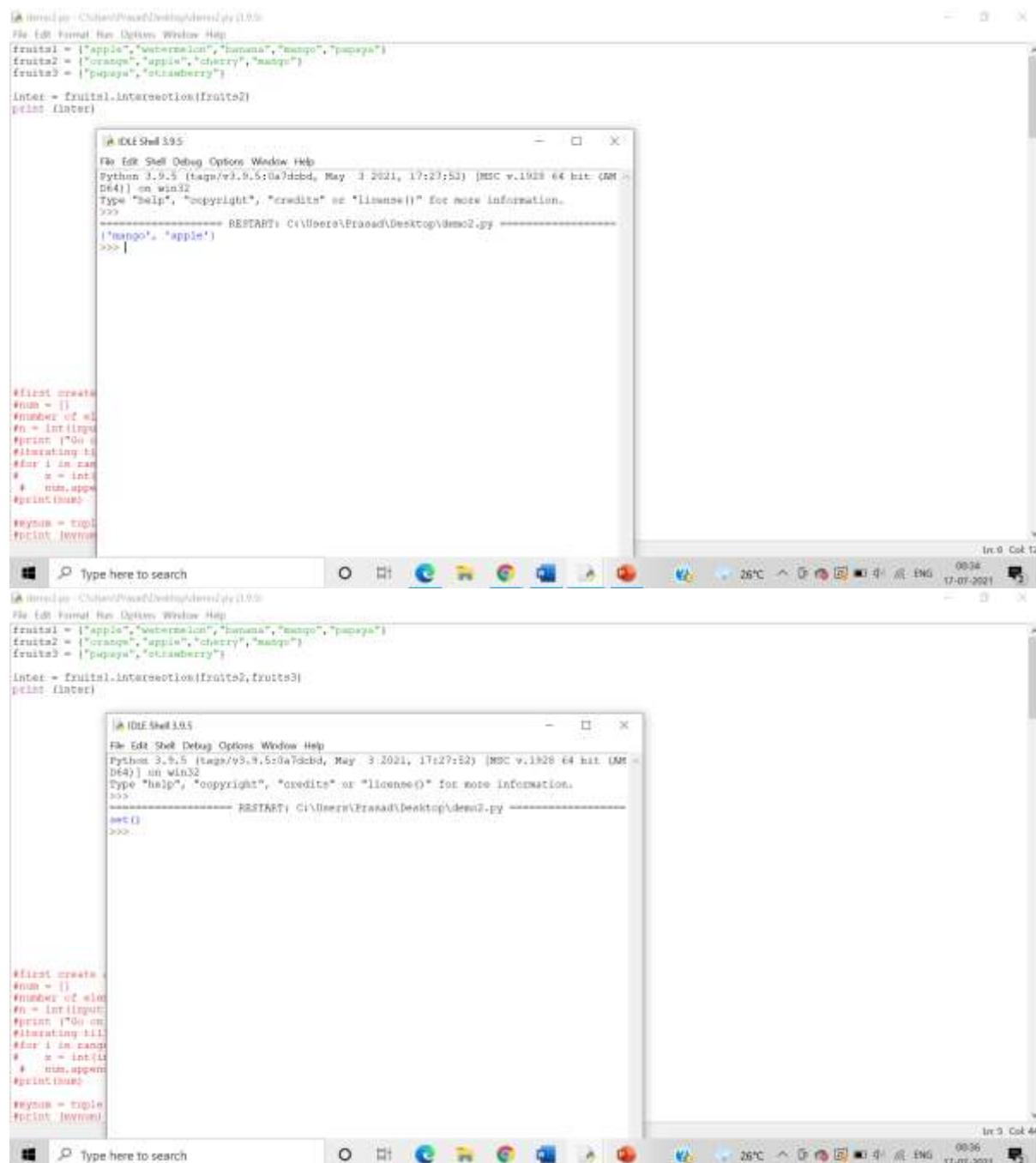
Program for demonstrating intersection()

Algorithm:-

Step 1:- intersection() method returns a set, that is the intersection of two other sets

Step 2:- intersection_update() method removes the items in this set that are not present in other, specified set.

Step3:- isdisjoint() method returns whether two sets have a intersection or not.



```
#include <iostream>
#include<vector>
using namespace std;
int main()
{
    vector<string> v1, v2, v3;
    int n;
    cout << "Enter number of elements in first set: ";
    cin >> n;
    cout << "Enter elements of first set: ";
    for(int i=0; i<n; i++)
    {
        string s;
        cin >> s;
        v1.push_back(s);
    }
    cout << "Enter number of elements in second set: ";
    cin >> n;
    cout << "Enter elements of second set: ";
    for(int i=0; i<n; i++)
    {
        string s;
        cin >> s;
        v2.push_back(s);
    }
    cout << "Enter number of elements in third set: ";
    cin >> n;
    cout << "Enter elements of third set: ";
    for(int i=0; i<n; i++)
    {
        string s;
        cin >> s;
        v3.push_back(s);
    }
    set<string> s1 = set<string>(v1.begin(), v1.end());
    set<string> s2 = set<string>(v2.begin(), v2.end());
    set<string> s3 = set<string>(v3.begin(), v3.end());
    cout << "Intersection of first and second set: ";
    cout << s1 << endl;
    cout << "Intersection of first and third set: ";
    cout << s1 << endl;
    cout << "Intersection of second and third set: ";
    cout << s2 << endl;
    cout << "Intersection of all three sets: ";
    cout << s1 << endl;
}
```

```
fruits1 = {"apple", "watermelon", "banana", "mango", "pawaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"pawaya", "strawberry"}
```

```
inter = fruits1.intersection(fruits2)
print (inter)
```

```
fruits1 = {"apple", "watermelon", "banana", "mango", "pawaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"pawaya", "strawberry"}
```

```
inter = fruits1.intersection(fruits3)
print (inter)
```

```
fruits2 = {"orange", "apple", "cherry", "mango"}  
fruits3 = {"pawaya", "strawberry"}
```

```
inter = fruits2.intersection(fruits3)
print (inter)
```

```
#> interact.py - C:\Users\Prasad\Desktop\interact.py(1.0):<
File Edit Shell Debug Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry", "apple"}

inter = fruits1.intersection(fruits2,fruits3)
print(inter)

#> IODE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb3, May  3 2021, 17:27:53) [MSC v.1932 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'apple'}
```

```
#> IODE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb3, May  3 2021, 17:27:53) [MSC v.1932 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'apple'}
```

Program for demonstrating intersection_update()

```
#> interact.py - C:\Users\Prasad\Desktop\interact.py(1.0):<
File Edit Shell Debug Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"papaya", "strawberry", "apple"}

fruits1.intersection_update(fruits2,fruits3)
print(fruits1)

#> IODE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb3, May  3 2021, 17:27:53) [MSC v.1932 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'apple'}
```

```
#> IODE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcb3, May  3 2021, 17:27:53) [MSC v.1932 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'apple'}
```

Program for demonstrating isdisjoint()

```
#> idle> C:\Users\Prasad\Desktop\demo2.py (1.0.0)
File Edit Shell Options Window Help
fruits1 = {"apple", "watermelon", "banana", "mango", "pawaya"}
fruits2 = {"orange", "apple", "cherry", "mango"}
fruits3 = {"pawaya", "strawberry", "apple"}

print (fruits1.isdisjoint(fruits2))

#>>> False
#>>>

#First create an empty
num = []
#number of elements
n = int(input("Enter"))
#input 100 on screen
#iterating till the n
for i in range (0,n):
    # i = int(input())
    # num.append(i)
    #print (num)

#num = tuple(num)
#print (len(num))

#> idle> C:\Users\Prasad\Desktop\demo2.py (1.0.0)
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
False
#>>>

#First create an empty
num = []
#number of elements
n = int(input("Enter"))
#input 100 on screen
#iterating till the n
for i in range (0,n):
    # i = int(input())
    # num.append(i)
    #print (num)

#num = tuple(num)
#print (len(num))

#> idle> C:\Users\Prasad\Desktop\demo2.py (1.0.0)
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d8, May 3 2021, 17:37:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
True
#>>>
```

Program for demonstrating issubset()

Algorithm:-

Step 1:- issubset() methods returns whether another set contains this set or not.

Step 2:- issuperset() method returns whether this set contains another set or not.

Step3:- pop() method removes an element from the set.

Step 4:- remove() method removes the specified element.

The image shows two side-by-side screenshots of the Python IDLE shell interface. Both shells have identical code and output displayed.

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<string> fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"};
    vector<string> fruits2 = {"orange", "apple", "cherry", "mango"};
    vector<string> fruits3 = {"apple", "banana"};

    cout << (fruits3.isSubset(fruits1));
}
```

The first shell's output is:

```
>>> True
```

The second shell's output is:

```
>>> False
```

Both shells are running Python 3.9.5 on Windows 10. The taskbar at the bottom shows other open applications like File Explorer, Edge, and Google Chrome.

Program for demonstrating issuperset()

```
#> idle.py - C:\Users\Praasad\Desktop\demo1.py (1.0)
File Edit Format Run Options Window Help
fruits1 = ['apple', 'watermelon', 'banana', 'mango', 'papaya']
fruits2 = ['orange', 'apple', 'cherry', 'mango']
fruits3 = ['apple', 'banana']

print (fruits3.issuperset(fruits1))

#> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praasad\Desktop\demo1.py =====
True
>>>
```

The screenshot shows the Python IDLE interface. The code in the editor window is:

```
#> idle.py - C:\Users\Praasad\Desktop\demo1.py (1.0)
File Edit Format Run Options Window Help
fruits1 = ['apple', 'watermelon', 'banana', 'mango', 'papaya']
fruits2 = ['orange', 'apple', 'cherry', 'mango']
fruits3 = ['apple', 'banana']

print (fruits3.issuperset(fruits1))
```

The output window shows the result of running the script:

```
#> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praasad\Desktop\demo1.py =====
True
>>>
```

Program for demonstrating symmetric_difference()

Algorithm:-

Step 1:- symmetric_difference() method returns a set with the symmetric differences of two sets.

Step 2:- symmetric_difference_update() method inserts the symmetric differences from this set and another.

Step3:- union() method return a set containing the union of sets .

Step 4:-update() method update the set with the union of this set and others.

```
#> idle.py - C:\Users\Praasad\Desktop\demo2.py (1.0)
File Edit Format Run Options Window Help
fruits1 = ['apple', 'watermelon', 'banana', 'mango', 'papaya']
fruits2 = ['orange', 'apple', 'cherry', 'mango']
fruits3 = ['apple', 'banana']

diff1 = fruits1.symmetric_difference(fruits2)
print (diff1)

#> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praasad\Desktop\demo2.py =====
['watermelon', 'orange', 'cherry', 'papaya', 'banana']
>>>
```

The screenshot shows the Python IDLE interface. The code in the editor window is:

```
#> idle.py - C:\Users\Praasad\Desktop\demo2.py (1.0)
File Edit Format Run Options Window Help
fruits1 = ['apple', 'watermelon', 'banana', 'mango', 'papaya']
fruits2 = ['orange', 'apple', 'cherry', 'mango']
fruits3 = ['apple', 'banana']

diff1 = fruits1.symmetric_difference(fruits2)
print (diff1)
```

The output window shows the result of running the script:

```
#> IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praasad\Desktop\demo2.py =====
['watermelon', 'orange', 'cherry', 'papaya', 'banana']
>>>
```

Program for demonstrating symmetric_difference_update()

```
#fruits = ["apple", "watermelon", "banana", "mango", "papaya"]
#fruits2 = ["orange", "apple", "cherry", "mango"]
#fruits3 = ["apple", "banana"]

fruits1.symmetric_difference_update(fruits2)
print(fruits1)

#First creates an empty list
num = []
#Number of elements
n = int(input("No. of elements"))
#Print ("Go on set")
#Iterating till the number of elements
for i in range(0, n):
    x = int(input("Enter element"))
    num.append(x)
#Print (num)
#num = tuple(num)
#print (num)
```

Type here to search

Program for demonstrating union()

```
# Program for demonstrating union()
# Runned by : C:\Users\Prasad\Desktop\demo1.py

# In Python, we can't directly create a union type like in C/C++.
# Instead, we can use the built-in tuple type to achieve similar functionality.

# First, create a
#num = []
#Number of elements
n = int(input())
#print ("Go on")
#Iterating till
#For i in range
#    x = int(input())
#    num.append(x)
#print (num)

#myset = tuple(num)
#print (myset)

#fruits1 = ["apple", "watermelon", "banana", "mango", "papaya"]
#fruits2 = ["orange", "apple", "cherry", "mango"]
#fruits3 = ["apple", "banana"]

#myset = fruits1.union(fruits2,fruits3)
#print (myset)

#>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
#('apple', 'orange', 'papaya', 'banana', 'watermelon', 'mango', 'cherry')
#>>>
```

Program for demonstrating update()

The image shows three separate windows on a Windows desktop, each displaying a Python script and its output. The top window is titled 'idle.py - C:\Users\Prasad\Desktop\demo1.py (1.0.0)'. The middle window is titled 'KRE Shell 3.05' and the bottom window is also titled 'KRE Shell 3.05'. All three windows show the same Python code and its execution results.

```
#first create a
#num = []
#number of ele
#n = int(input())
#print ("n")
#print ("n")
#iterating till
#for i in range
#    x = intin
#    n.append
#print (num)

#num = tuple()
#print (num)
```

```
fruits1 = ('apple','watermelon','banana','mango','papaya')
fruits2 = ('orange','apple','cherry','mango')
fruits3 = ('apple','banana','strawberry')

fruits1.update(fruits3)
print (fruits1)
```

```
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
('watermelon', 'strawberry', 'banana', 'papaya', 'mango', 'apple')
>>>
```

```
#first create a
#num = []
#number of ele
#n = int(input())
#print ("n")
#print ("n")
#iterating till
#for i in range
#    x = intin
#    n.append
#print (num)

#num = tuple()
#print (num)
```

```
fruits1 = ('apple','watermelon','banana','mango','papaya')
fruits2 = ('orange','apple','cherry','mango')
fruits3 = ('apple','banana')

fruits1.update(fruits3)
print (fruits1)
```

```
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
('watermelon', 'mango', 'apple', 'papaya', 'banana')
>>>
```

```
#first create a
#num = []
#number of ele
#n = int(input())
#print ("n")
#print ("n")
#iterating till
#for i in range
#    x = intin
#    n.append
#print (num)

#num = tuple()
#print (num)
```

5. Demonstrate declaration and usage of frozen sets

Algorithm:-

Step 1:- A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function.

Step 2:- A frozen set is created by using the frozenset() method.

Step3:- If no parameters are passed, it returns an empty frozenset.

Answer: Program for declaration and usage of frozen sets

```
fruits1 = {"apple", "watermelon", "banana", "mango", "papaya"}  
fruits2 = {"orange", "apple", "cherry", "mango"}  
fruits3 = {"apple", "banana", "strawberry"}  
  
fset = frozenset(fruits3)  
print(fset)
```



```
# first creates a  
#num = []  
#number of elem  
n = int(input())  
print("Go on")  
#iterating till  
for i in range(n):  
    x = int(input())  
    num.append(x)  
  
resnum = tuple(num)  
print(resnum)
```



```
#tuple of vowels  
vowels = ('a','e','i','o','u')  
fset = frozenset(vowels)  
print("The frozen set is:", fset)  
print("The empty frozen set is:", frozenset())  
#frozensets are immutable  
fset.add('e')
```



```
# IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:0a0d0dd0, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
=>  
----- RESTART: C:\Users\Prasad\Desktop\demo1.py -----  
{'apple', 'banana', 'mango', 'papaya'}  
  
# IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:0a0d0dd0, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
=>  
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----  
The frozen set is: frozenset({'a', 'e', 'i', 'o', 'u'})  
The empty frozen set is: frozenset()  
Traceback (most recent call last):  
  File "C:\Users\Prasad\Desktop\demo2.py", line 36, in <module>  
    fset.add('e')  
AttributeError: 'frozenset' object has no attribute 'add'  
=>
```

6. Write a program demonstrating declaration and accessing dictionaries and its elements.

Algorithm:-

Step 1:- A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function. You can declare a set by assigning it to a variable.

Step 2:- You can access the set by using the key in a square bracket '[]'.

Answer: Program demonstrating accessing dictionaries and its elements:

```
l:\> idle3.py -C:\Users\Pranav\Desktop\demo1.py 1.0.0
File Edit Shell Options Window Help
Python 3.6.5 (tags/v3.6.5:fdbf626, May  3 2018, 17:27:52) [MSC v.1925 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Pranav\Desktop\demo1.py
{'BID19001': 'Karan', 'BID19002': 'Aaru', 'BID19003': 'Geeta', 'BID19004': 'Neha'}
Aaru
>>>
```

7. Write a program demonstrating looping through the dictionaries, printing keys and values separately and together

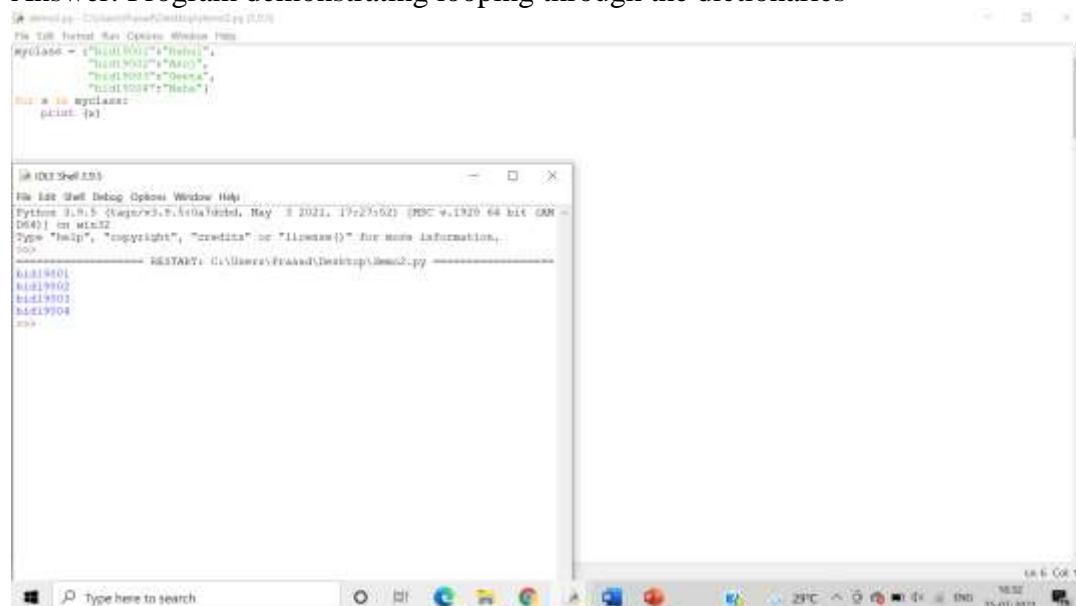
Algorithm:-

Step 1:- Using a for loop where x is a key in a set, we can print keys separately by printing 'x'.

Step 2:- Using a for loop where y is a key in a set, we can print the values separately by printing 'days[y]' which will print the value assigned to the key 'y'.

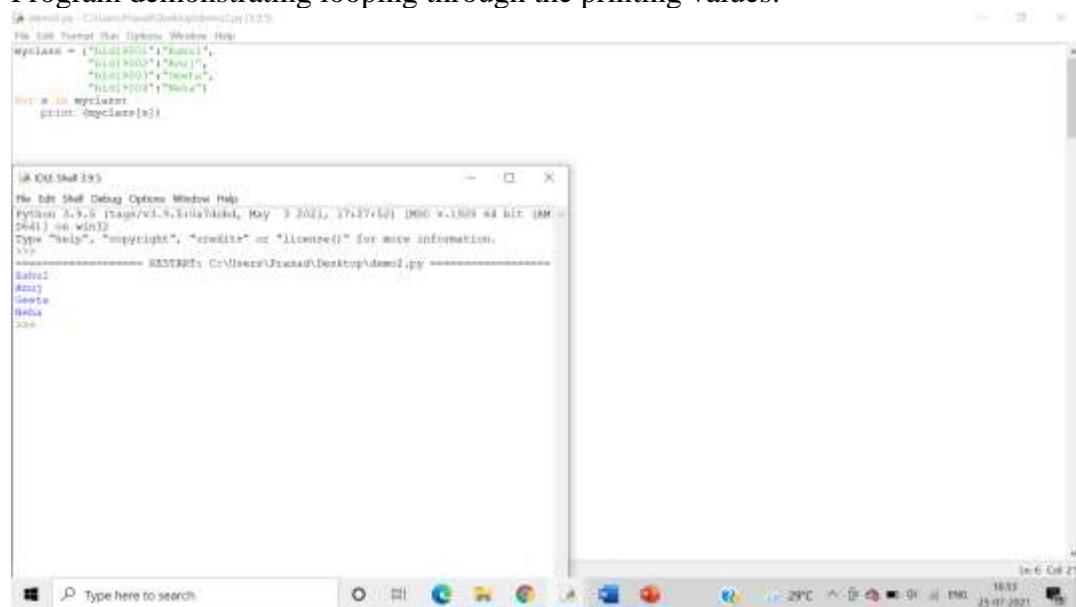
Step3:- Using a for loop where z is a key in a set, we can print keys and values together by printing 'z' which will print the keys and days['z'],which will print the value assigned to the key 'z'.

Answer: Program demonstrating looping through the dictionaries



```
#> demo.py -> C:\Users\Prasad\Desktop\demo1.py (3.9)
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:5af3f2c, May  2 2021, 17:27:46) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
Math
Science
English
Hindi
>>>
```

Program demonstrating looping through the printing values:



```
#> demo.py -> C:\Users\Prasad\Desktop\demo1.py (3.9)
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:5af3f2c, May  2 2021, 17:27:46) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
Math
Science
English
Hindi
>>>
```

Program demonstrating looping through the printing values and keys:

The screenshot shows a Windows desktop environment. In the center is a Python IDLE Shell window titled 'IDLE Shell 3.9.5'. The code inside the shell is as follows:

```
myclass = {"B1D19001": "Kamal",
           "B1D19002": "Anuj",
           "B1D19003": "Deeta",
           "B1D19004": "Neha"}  
for x in myclass:  
    print(x + " " + myclass[x])
```

When run, the output is:

```
B1D19001>Kamal  
B1D19002>Anuj  
B1D19003>Deeta  
B1D19004>Neha
```

Below the IDLE window is the Windows taskbar, which includes icons for File Explorer, Task View, Start, Edge, and others. The system tray shows the date (25-07-2021), time (16:35), battery level (83%), and language (ENG).

8. Write a program demonstrating “in” and “not in” keywords on Dictionaries

Algorithm:-

Step 1: - The ‘in’ operator is used to **check if an element is present** in the dictionary or not. Returns true if element is present in dictionary else returns false.

Step 2: The ‘not in’ operator is used to **check if an element is not present** in the dictionary or not. Returns true if element is not present in dictionary else returns false.

Answer: A program demonstrating “in” keyword on Dictionaries:

```
lak@lak-OptiPlex-5070: ~
```

```
File Edit Format Run Options Window Help
```

```
myclass = {"bid19001": "Kamal",
```

```
           "bid19002": "Anu",
```

```
           "bid19003": "Amit",
```

```
           "bid19004": "Rahul"}
```

```
if "bid19001" in myclass:
```

```
    print(myclass["bid19001"])
```



```
lak@lak-OptiPlex-5070: ~
```

```
File Edit Shell Debug Options Window Help
```

```
Python 3.9.5 (tags/v3.9.5:0a0a7dcb3, May 3 2021, 17:27:52) [MSC v.1938 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>> ----- RESTART: C:\Users\lak\Desktop\demo1.py -----
```

```
Anu
```

```
>>>
```

A program demonstrating “not in” keyword on Dictionaries

```
lak@lak-OptiPlex-5070: ~
```

```
File Edit Format Run Options Window Help
```

```
myclass = {"bid19001": "Kamal",
```

```
           "bid19002": "Anu",
```

```
           "bid19003": "Amit",
```

```
           "bid19004": "Rahul"}
```

```
if "bid19001" not in myclass:
```

```
    print("Absent")
```



```
lak@lak-OptiPlex-5070: ~
```

```
File Edit Shell Debug Options Window Help
```

```
Python 3.9.5 (tags/v3.9.5:0a0a7dcb3, May 3 2021, 17:27:52) [MSC v.1938 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>> ----- RESTART: C:\Users\lak\Desktop\demo2.py -----
```

```
Absent
```

```
>>>
```

9. Write a program demonstrating len() and del() function on Dictionaries.

Algorithm:-

Step 1: The len() function is used to obtain the length of the dictionary

Step 2: The del() function **deletes all the elements in range** starting from index 'a' till 'b' mentioned in arguments.

Answer: Program demonstrating len() function on Dictionaries:

```
# idle3.py - C:\Users\Prasad\Desktop\demo1.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d0dd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo1.py =====
4
>>>
```

Program demonstrating del() function on Dictionaries:

```
# idle3.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d0dd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
{'bid19001': 'Rahul', 'bid19002': 'Ama', 'bid19003': 'Geeta'}
001 (myclass["001"])
print (myclass)
```

The screenshot shows a Windows desktop environment. In the center, there is an 'IDE Shell 3.9.5' window with a light gray background. The title bar says 'IDE Shell 3.9.5'. Inside the window, there is Python code and its output:

```
[IDEShell] C:\Users\Prasad\Desktop\diamond.py(1,0)
File Edit Shell Debug Options Window Help
myclass = {"bid19001": "Kamal",
           "bid19002": "Anur",
           "bid19003": "Sneha",
           "bid19004": "Rehana"}
m1 = myclass
print(m1)

[IDEShell] C:\Users\Prasad\Desktop\diamond.py(1,0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2ca, May  3 2021, 11:21:32) [MSC v.1920 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ****SETHALT: C:\Users\Prasad\Desktop\diamond.py
Traceback (most recent call last):
  File "C:\Users\Prasad\Desktop\diamond.py", line 6, in <module>
    print(myclass)
NameError: name 'myclass' is not defined
>>>
```

Below the window, the Windows taskbar is visible, featuring the Start button, a search bar with the placeholder 'Type here to search', and several pinned icons for applications like File Explorer, Edge, and File History.

10. Write a program demonstrating Nested Dictionaries

Algorithm:-

Step 1:- In Python, a nested dictionary is a dictionary inside a dictionary. It's a collection of dictionaries into one single dictionary.

Step 2:- Here each key in a dictionary has its own dictionary

Answer: Program demonstrating Nested Dictionaries:

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled "IDLE Shell 3.9.5" running Python 3.9.5. The code in the terminal window is as follows:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:5f9f7c5, May 2 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
{'bid19001': {'name': 'John', 'address': 'Bhargha'}, 'bid19002': {'name': 'Raj', 'address': 'Yashvi'}, 'bid19003': {'name': 'Kanu', 'address': 'Pune'}, 'bid19004': {'name': 'Pachi', 'address': 'Khar'}, 'bid19005': {'name': 'Raj', 'address': 'Yashi'}}
```

Below the terminal window is a code editor window titled "Untitled - C:\Users\Prasad\Desktop\demo2.py (1.0.0)". The code in the editor is:

```
myclass = {'id':19001}
    "name": "John",
    "address": "Bhargha"
}
{
    "id":19002
}
    "name": "Raj"
    "address": "Yashvi"
}
{
    "id":19003
}
    "name": "Kanu"
    "address": "Pune"
}
{
    "id":19004
}
    "name": "Pachi"
    "address": "Khar"
}
{
    "id":19005
}
    "name": "Raj"
    "address": "Yashi"
}

print(myclass)
print (myclass["id19002"])
```

At the bottom of the screen, the taskbar shows various icons for applications like File Explorer, Edge, and others, along with system status indicators for battery level, temperature (29°C), and date/time (25-07-2021).

11. Write a program demonstrating input of Dictionary elements from the user

Algorithm:

Step 1:- Take user input for number of elements in the dictionary as well that of both the keys and values in a for loop.

Step 2:- We initialize a empty dictionary to which we can then add the values of both the keys and values using the update() function

Code-Program demonstrating input of Dictionary elements from the user:-

The screenshot shows a Windows desktop environment. On the left is a code editor window titled 'idle3.py' with Python code for creating a dictionary. On the right is a terminal window titled 'IDLE Shell 3.9.5' running Python 3.9.5, showing the execution of the code and its output.

```
idle3.py  C:\Users\Prasad\Desktop\idle3.py (1.0%)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8c5, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\idle3.py =====
Enter number of key-value pairs: 5
go on entering the values
enter here<23
enter here<55
enter here<432
enter here<21
enter here<380
('380', '21', '23', '55', '432')
go on entering the values
enter here<44
enter here<556
enter here<30
enter here<99
enter here<745
('380', '21', '23', '556', '30', '55', '99', '432', '745')
>>>
```

12. Write set of programs for demonstrating the usage of all the different dictionary methods along with their variations

Answer: Programs for demonstrating

Algorithm:-

Step 1:- The clear() method empties the dictionary.

Step 2:- The copy() method returns a copy of the dictionary.

clear()

```
#> idle1.py - C:\Users\Prasad\Desktop\demo1.py (1.0)
File Edit Shell Options Window Help
myclass = {'bid19001':{
    "name": "John",
    "address": "Kharghar"
  },
  "bid19002": {
    "name": "Raj",
    "address": "Vashi"
  },
  "bid19003": {
    "name": "Rani",
    "address": "Pune"
  },
  "bid19004": {
    "name": "Prachi",
    "address": "Pune"
  },
  "bid19005": {
    "name": "Prasad",
    "address": "Chhat"
  }
}
print myclass
print myclass.clear()

#fruits2 = ["orange","apple"]
#fruits3 = ["apple","banana"]
```

copy()

```
#> idle2.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Shell Options Window Help
myclass = {'bid19001':{
    "name": "John",
    "address": "Kharghar"
  },
  "bid19002": {
    "name": "Raj",
    "address": "Vashi"
  },
  "bid19003": {
    "name": "Rani",
    "address": "Pune"
  },
  "bid19004": {
    "name": "Prachi",
    "address": "Pune"
  },
  "bid19005": {
    "name": "Prasad",
    "address": "Chhat"
  }
}
print myclass
print myclass.copy()

#fruits2 = ["orange","apple"]
#fruits3 = ["apple","banana"]
```

```
[C:\Users\Praaad\Desktop\demo1.py] 1.0.0
File Edit Shell Options Window Help
myclass = {"bid19001": "Raja",
           "bid19002": "Kavya",
           "bid19003": "John",
           "bid19004": "Prachi",
           "bid19005": ""}

dict1 = myclass
newdict = myclass.copy()

if dict1 is myclass:
    print("yes")
else:
    print("no")

File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> newdict
{'bid19001': 'Raja', 'bid19002': 'Kavya', 'bid19003': 'John', 'bid19004': 'Prachi', 'bid19005': ''}
>>> RESTART: C:\Users\Praaad\Desktop\demo2.py
no
no
no
no
>>>
```

fromkeys()

Algorithm:-

Step 1:- fromkeys() returns a dictionary with the specified keys and value

Step 2:- get() returns the value of the specified key

Step 3:- items() returns a list containing a tuple for each key value pair

Step 4:- keys() returns a list containing the dictionary's keys

```
[C:\Users\Praaad\Desktop\demo1.py] 1.0.0
File Edit Shell Options Window Help
x = ["a", "b", "c", "d"]
y = ["pink", "blue", "red", "yellow"]
colours = dict.fromkeys(x,y)
print(colours)

File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9bd, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> {'a': 'pink', 'b': 'blue', 'c': 'red', 'd': 'yellow'}
{'a': 'pink', 'b': 'blue', 'c': 'red', 'd': 'yellow'}
>>> RESTART: C:\Users\Praaad\Desktop\demo1.py
{'a': 'pink', 'b': 'blue', 'c': 'red', 'd': 'yellow'}
{'a': 'pink', 'b': 'blue', 'c': 'red', 'd': 'yellow'}
{'a': 'pink', 'b': 'blue', 'c': 'red', 'd': 'yellow'}
```

```

[1] mridul@Celeron:~/Desktop$ idle2.py(1.0)
File Edit Shell Debug Options Window Help
x = ["a","b","c","d"]
y = ["pink"]
colours = dict.fromkeys(x,y)
print(colours)

[1] mridul@Celeron:~/Desktop$ idle2.py(1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a09ecf, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': 'pink', 'b': 'pink', 'c': 'pink', 'd': 'pink'}
```



```

[1] mridul@Celeron:~/Desktop$ idle2.py(1.0)
File Edit Shell Debug Options Window Help
x = ["a","b","c","d"]
y = ["pink","red"]
colours = dict.fromkeys(x,y)
print(colours)

[1] mridul@Celeron:~/Desktop$ idle2.py(1.0)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a09ecf, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': 'pink', 'b': 'pink', 'c': 'pink', 'd': 'pink'}
>>> -----
RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': {'pink', 'red'}, 'b': {'pink', 'red'}, 'c': {'pink', 'red'}, 'd': {'pink', 'red'}}
```

```

[1] iuridipy - C:\Users\Prasad\Desktop\demo1.py(1.0): 
File Edit Shell Debug Options Window Help
x = ["a", "b", "c", "d"]
y = ["34"]
colours = dict.fromkeys(x,y)
print(colours)

[2] iuridipy - C:\Users\Prasad\Desktop\demo2.py(1.0): 
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:186bb26e, May 1 2021, 17:27:42) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': 'pink', 'b': 'pink', 'c': 'pink', 'd': 'pink'}
>>> 
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': {'pink', 'red'}, 'b': {'pink', 'red'}, 'c': {'pink', 'red'}, 'd': {'pink', 'red'}}
>>> 
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
{'a': '34', 'b': '34', 'c': '34', 'd': '34'}
>>> 

[3] iuridipy - C:\Users\Prasad\Desktop\demo3.py(1.0): 
File Edit Shell Debug Options Window Help
x = ["a", "b", "c", "d"]
colours = dict.fromkeys(x)
print(colours)

[4] iuridipy - C:\Users\Prasad\Desktop\demo3.py(1.0): 
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:186bb26e, May 1 2021, 17:27:42) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
----- RESTART: C:\Users\Prasad\Desktop\demo3.py -----
{'a': None, 'b': None, 'c': None, 'd': None}
>>> 

[5] iuridipy - C:\Users\Prasad\Desktop\demo3.py(1.0): 
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:186bb26e, May 1 2021, 17:27:42) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
----- RESTART: C:\Users\Prasad\Desktop\demo3.py -----
{'a': None, 'b': None, 'c': None, 'd': None}
>>> 

```

```
get()
```

```
File Edit Shell Debug Options Window Help
```

```
myclass = {"BID19001": "Raj",  
          "BID19002": "Raj",  
          "BID19003": "John",  
          "BID19004": "Prachi",  
          "BID19005": "I"}
```

```
x = myclass.get("BID19001")
```

```
print(x)
```

```
[?] IDLE Shell 3.9.5
```

```
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f0fbd9, May 3 2021, 17:27:53) [MSC v.1932 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>
```

```
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
```

```
John
```

```
>>>
```

```
#dicti = myclass  
#newdict = myclass.copy()  
  
#if dict in myclass:  
#    print ("yes")  
#else:  
#    print ("no")  
  
#if newdict in myclass:  
#    print ("yes")  
#else:  
#    print ("no")
```

```
File Edit Shell Debug Options Window Help
```

```
Python 3.9.5 (tags/v3.9.5:f0fbd9, May 3 2021, 17:27:53) [MSC v.1932 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>>
```

```
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
```

```
Node
```

```
>>>
```

```
#dicti = myclass  
#newdict = myclass  
  
#if dict in myclass:  
#    print ("yes")  
#else:  
#    print ("no")  
  
#if newdict in myclass:  
#    print ("yes")  
#else:  
#    print ("no")
```

```
File Edit Shell Debug Options Window Help
```

```
Python 3.9.5 (tags/v3.9.5:f0fbd9, May 3 2021, 17:27:53) [MSC v.1932 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>>
```

```
----- RESTART: C:\Users\Prasad\Desktop\demo2.py -----
```

```
Node
```

```
>>>
```

The image displays two separate Windows desktop sessions. Both sessions show an open Python IDLE Shell window and a taskbar at the bottom.

Session 1 (Left):

- Python Code:**

```

myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prasad", "bid19005": "T"}
x = myclass.get("bid19004", "abc")
print(x)

```
- IDLE Shell Output:**

```

# IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:5f9ca7dcd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
abc
>>>

```
- Taskbar:** Shows icons for File Explorer, Task View, Start, Edge, Google Chrome, File Manager, and File Explorer again.
- System Tray:** Shows battery level (28%), signal strength, and date/time (22:11, 25-07-2021).

Session 2 (Right):

- Python Code:**

```

myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prasad", "bid19005": "T"}
x = myclass.get("bid19004", "abc")
print(x)

```
- IDLE Shell Output:**

```

# IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:5f9ca7dcd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
Prash
>>>

```
- Taskbar:** Shows icons for File Explorer, Task View, Start, Edge, Google Chrome, File Manager, and File Explorer again.
- System Tray:** Shows battery level (28%), signal strength, and date/time (22:12, 25-07-2021).

items()

```
#include <ConsolePseudoTerminal.h>(1.0)
File Edit Format Run Options Window Help
myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prachi", "bid19005": "J"}
x = myclass.items()
print (x)

IDE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
dict_items([('bid19001', 'Raj'), ('bid19002', 'Raj'), ('bid19003', 'John'), ('bid19004', 'Prachi'), ('bid19005', '')])
>>>
```

```
#include <ConsolePseudoTerminal.h>(1.0)
File Edit Format Run Options Window Help
myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prachi", "bid19005": "J"}
x = myclass.items()
print (x)
print(x[1])

IDE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f9ad8d2, May 3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
dict_items([('bid19001', 'Raj'), ('bid19002', 'Raj'), ('bid19003', 'John'), ('bid19004', 'Prachi'), ('bid19005', ''))]
Traceback (most recent call last):
  File "C:\Users\Prasad\Desktop\demo2.py", line 31, in <module>
    print(x[1])
TypeError: 'dict_items' object is not subscriptable
>>>
```

keys()

```
myclass = {"bid19001": "Raj", "bid19002": "Raju", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Ishu"}  
x = myclass.keys()  
print (x)  
  
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py  
dict_keys(['bid19001', 'bid19002', 'bid19003', 'bid19004', 'bid19005'])  
>>>  
  
#dict1 = {}  
#mydict = {}  
  
#if dict is  
# print  
# else  
# print  
  
#dict1 = myclass  
#mydict = myclass  
  
#if dict is  
# print  
# else  
# print
```

The screenshot shows a Windows desktop with two windows open. The top window is a Python IDLE shell showing the output of `myclass.keys()` which prints the keys of the dictionary as a list of strings. The bottom window is a standard Windows file explorer showing a folder named 'Python'.

values()

```
myclass = {"bid19001": "Raj", "bid19002": "Raju", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Ishu"}  
x = myclass.keys()  
print (x)  
  
y = myclass.values()  
print (y)  
  
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py  
dict_keys(['bid19001', 'bid19002', 'bid19003', 'bid19004', 'bid19005'])  
dict_values(['Raj', 'Raju', 'John', 'Prachi', 'Ishu'])  
>>>  
  
#dict1 = myclass  
#mydict = myclass  
  
#if dict is  
# print  
# else  
# print  
  
#dict1 = myclass  
#mydict = myclass  
  
#if dict is  
# print  
# else  
# print
```

The screenshot shows a Windows desktop with two windows open. The top window is a Python IDLE shell showing the output of `myclass.values()` which prints the values of the dictionary as a list of strings. The bottom window is a standard Windows file explorer showing a folder named 'Python'.

pop()

Algorithm:-

Step 1:- pop() removes the element with the specified key.

Step 2:- popitem() removes the last inserted key-value pair

Step 3:- setdefault() returns the value of the specified key. If the key does not exist: insert the key, with the specified value

```
myclass = {"bid19001": "Raj", "bid19002": "Raju", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Roja"}  
x = myclass.pop("bid19004")  
print(x)  
  
#dict1 = myclass  
#newdict = myclass  
  
#if dict in myclass:  
#    print("yes")  
#else:  
#    print("no")  
  
#if newdict in myclass:  
#    print("yes")  
  
#dict1 = myclass  
#newdict = myclass.pop()  
  
#if dict in myclass:  
#    print("yes")  
#else:  
#    print("no")  
  
#if newdict in myclass:
```

The screenshots show two separate Windows desktop sessions. Each session has a taskbar at the bottom with icons for File Explorer, Task View, Edge, File Explorer, File Explorer, and Task View. The desktop background is white. In each session, there is a window titled 'IDLE Shell 3.9.5' running Python 3.9.5. The code in both windows is identical, demonstrating the use of the pop() and popitem() methods on a dictionary named 'myclass'. The output in the top window shows the removal of the item with key 'bid19004', while the bottom window shows the removal of the last item from the dictionary.

The image shows two separate Windows desktop sessions. Both sessions have identical taskbars at the bottom with icons for File Explorer, Edge, File Explorer, Task View, Start, Taskbar settings, and system status (29°C, 32:24, 25-07-2021).

Session 1 (Top):

- File menu: File, Edit, Shell, Debug, Options, Window, Help.
- Python version: Python 3.9.5 (tags/v3.9.5:0a7d9d3, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
- Code window:

```

myclass = {"bid19001": "Raj",
           "bid19002": "Raju",
           "bid19003": "John",
           "bid19004": "Prachi",
           "bid19005": "Pooja"}  

x = myclass.pop("bid19001", "abc")
print (x)
print (myclass)

```

- Output window (IDLE Shell 3.9.5):

```

>>> RESTART: C:\Users\Prachi\Desktop\demo1.py
Poja
{'bid19001': 'Raj', 'bid19002': 'Raju', 'bid19003': 'John', 'bid19004': 'Prachi'}
>>>

```

Session 2 (Bottom):

- File menu: File, Edit, Shell, Debug, Options, Window, Help.
- Python version: Python 3.9.5 (tags/v3.9.5:0a7d9d3, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
- Code window:

```

myclass = {"bid19001": "Raj",
           "bid19002": "Raju",
           "bid19003": "John",
           "bid19004": "Prachi",
           "bid19005": "Pooja"}  

x = myclass.pop("bid19001", "abc")
print (x)
print (myclass)

```

- Output window (IDLE Shell 3.9.5):

```

>>> RESTART: C:\Users\Prachi\Desktop\demo1.py
abc
{'bid19001': 'Raj', 'bid19002': 'Raju', 'bid19003': 'John', 'bid19004': 'Prachi',
 'bid19005': 'Pooja'}
>>>

```

popitems()

```
myclass = {'a':10, 'b':20, 'c':30, 'd':40}
x = myclass.popitem()
print(x)
print(myclass)

# Output:
# ('d', 40)
# dict_keys(['a', 'b', 'c'])

# In the terminal window:
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
('d', 40)
dict_keys(['a', 'b', 'c'])

In [2]: Col 14
```

```
# In the code editor:
x = myclass.popitem()
print(x)
print(myclass)

# Output:
# ('d', 40)
# dict_keys(['a', 'b', 'c'])

# In the terminal window:
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
('d', 40)
dict_keys(['a', 'b', 'c'])

In [2]: Col 14
```

setdefault()

```
myclass = {'a':10, 'b':20, 'c':30, 'd':40}
x = myclass.setdefault("e", 50)
print(x)
print(myclass)

# Output:
# 50
# dict_keys(['a', 'b', 'c', 'd', 'e'])

# In the terminal window:
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
50
dict_keys(['a', 'b', 'c', 'd', 'e'])

In [2]: Col 25

# In the code editor:
x = myclass.setdefault("e", 50)
print(x)
print(myclass)

# Output:
# 50
# dict_keys(['a', 'b', 'c', 'd', 'e'])

# In the terminal window:
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
50
dict_keys(['a', 'b', 'c', 'd', 'e'])

In [2]: Col 25
```

The image shows two side-by-side screenshots of a Windows desktop environment. Both screenshots feature a Python IDE window (IDLE Shell 3.9.5) and a taskbar at the bottom.

Left Screenshot:

```

myclass = {"a": "000", "b": "111", "c": "222", "d": "333"}
x = myclass.setdefault("q")
print(x)
print(myclass)

# Output in IDLE Shell:
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
#>>> None
#>>> {'a': '000', 'b': '111', 'c': '222', 'd': '333', 'q': None}
#>>>

```

Right Screenshot:

```

myclass = {"a": "000", "b": "111", "c": "222", "d": "333"}
x = myclass.setdefault("q", "333")
print(x)
print(myclass)

# Output in IDLE Shell:
#>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
#>>> 333
#>>> {'a': '000', 'b': '111', 'c': '222', 'd': '333', 'q': '333'}
#>>>

```

The taskbar at the bottom of both screenshots shows several icons, including the Start button, File Explorer, Task View, Edge browser, Google Chrome, File Manager, and File Explorer again. The system tray indicates the date as 25-07-2021, the time as 23:26, and the battery level as 28%.

```

#> myclass.py - C:\Users\Praasad\Desktop\myclass.py(1,0)
File Edit Shell Options Window Help
myclass = {"a": "10", "b": "20", "c": "30", "d": "40"}
x = myclass.setdefault("a", "333")
print(x)
print(myclass)

#> IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praasad\Desktop\demo2.py
60
{'a': '60', 'b': '20', 'c': '30', 'd': '40'}
>>>

#> myclass = {"a": "10", "b": "20", "c": "30", "d": "40"}
#> print(myclass)

```

update()

Algorithm:-

- Step 1:- update() updates the dictionary with the specified key-value pairs
- Step 2:-values() returns a list of all the values in the dictionary

```

#> myclass.py - C:\Users\Praasad\Desktop\myclass.py(1,0)
File Edit Shell Options Window Help
myclass = {"bid19001": "Raj", "bid19002": "Raju", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Pooja"}
myclass.update({"bid19001": "Siddh"})
print(myclass)

#> IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7d9d3, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praasad\Desktop\demo2.py
{'bid19001': 'Raj', 'bid19002': 'Raju', 'bid19003': 'John', 'bid19004': 'Prachi', 'bid19005': 'Pooja'}
>>>

#> myclass = {"a": "10", "b": "20", "c": "30", "d": "40"}
#> print(myclass)

```

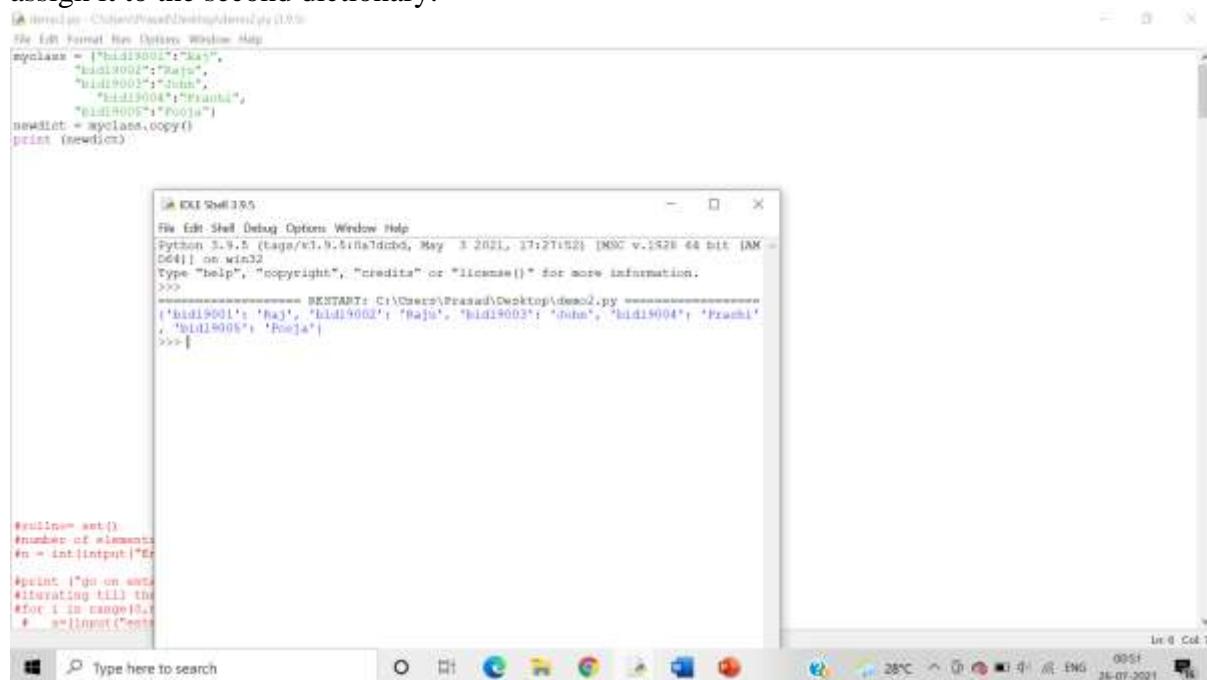
```
[F:\class2.py - Python Shell] F:\class2.py(1,1)  
File Edit Shell Debug Options Window Help  
myclass = {"bid19001": "Raj",  
           "bid19002": "Raju",  
           "bid19003": "John",  
           "bid19004": "Prachi",  
           "bid19005": "Pooja"}  
myclass.update({"bid19003": "John"})  
print(myclass)  
  
#  
#myclass = {"a": "s78",  
#           "b": "23",  
#           "c": "21",  
#           "d": "10"}  
  
#>>> myclass  
{'bid19001': 'Raj', 'bid19002': 'Raju', 'bid19003': 'John', 'bid19004': 'Prachi',  
 'bid19005': 'Pooja', 'bid19006': '23'}  
#>>>  
  
#myclass = {"a": "s78",  
#           "b": "23",  
#           "c": "21",  
#           "d": "10"}  
  
#>>> myclass  
{'bid19001': 'Raj', 'bid19002': 'Raju', 'bid19003': 'John', 'bid19004': 'Prachi',  
 'bid19005': 'Pooja', 'bid19006': '23'}  
#>>>
```

13. Demonstrate the usage of copy method

Algorithm:-

Step 1:-We create a dictionary and declare it.

Step 2:-We can then use copy() function to copy the keys and values In the first dictionary and assign it to the second dictionary.

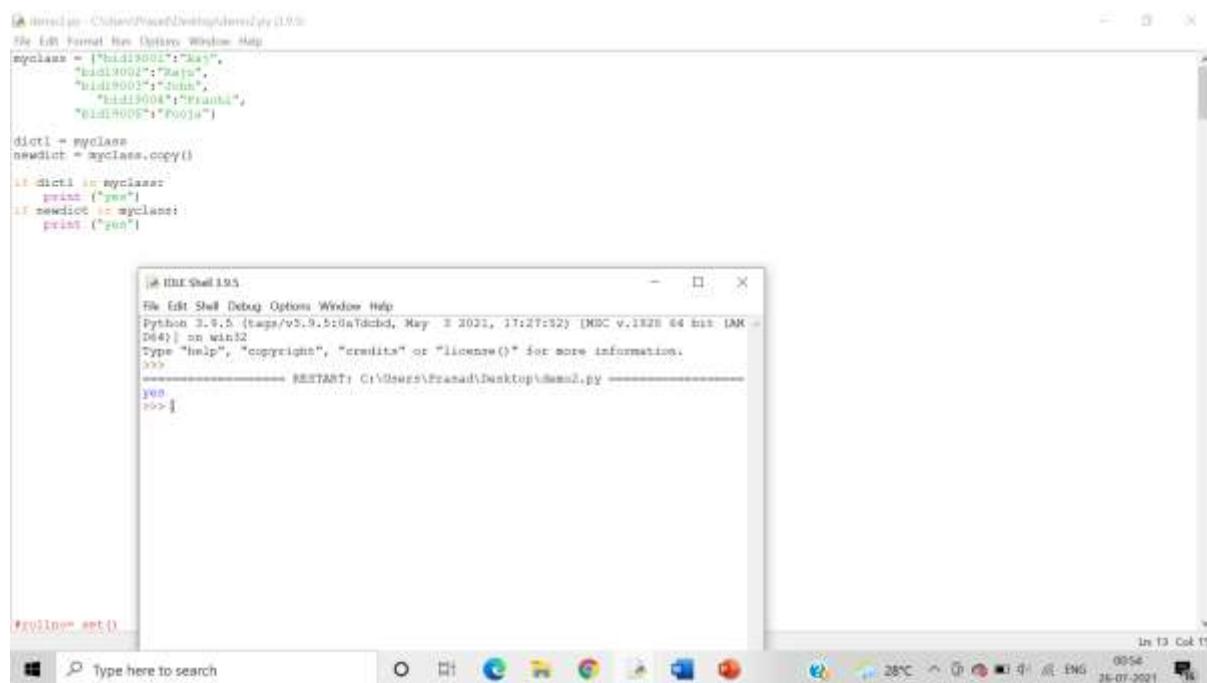


```
#> idle2.py - C:\Users\Praasad\Desktop\idle2.py (1.0)
File Edit Shell Options Window Help
myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Rooja"}
newdict = myclass.copy()
print (newdict)

#>>> RESTART: C:\Users\Praasad\Desktop\idle2.py
{'bid19001': 'Raj', 'bid19002': 'Raj', 'bid19003': 'John', 'bid19004': 'Prachi', 'bid19005': 'Rooja'}
>>>

#> idle2.py - C:\Users\Praasad\Desktop\idle2.py (1.0)
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 17:27:02) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
#>>> RESTART: C:\Users\Praasad\Desktop\idle2.py
>>> print ("yes")
yes
>>> newdict = myclass
>>> print ("yes")
yes

#>>>
```



```
#> idle2.py - C:\Users\Praasad\Desktop\idle2.py (1.0)
File Edit Shell Options Window Help
myclass = {"bid19001": "Raj", "bid19002": "Raj", "bid19003": "John", "bid19004": "Prachi", "bid19005": "Rooja"}
dict1 = myclass
newdict = myclass.copy()

if dict1 is myclass:
    print ("yes")
if newdict is myclass:
    print ("yes")

#>>> RESTART: C:\Users\Praasad\Desktop\idle2.py
yes
>>> print ("yes")
yes
>>> newdict = myclass
>>> print ("yes")
yes

#>>>
```

A screenshot of a Windows desktop environment. In the center, there is a terminal window titled "IDLE Shell 3.9.5" which displays Python code and its execution. Below the terminal is a taskbar with various icons and system status indicators.

```
[A] mreday - C:\Users\Prasad\Desktop\demod2.py (1.0)
File Edit Shell Debug Options Window Help
myclass = {"id119001": "kav",
           "id119002": "Raju",
           "id119003": "John",
           "id119004": "Frankie",
           "id119005": "Pooja"}
dict1 = myclass
newdict = myclass.copy()
for dict1 in newdict:
    print ("yes")
else:
    print ("no")
if newdict in myclass:
    print ("yes")
else:
    print ("no")
[1] RESTART: C:\Users\Prasad\Desktop\demod2.py
>>> yes
no
>>>
```

14. Write a program to merge two dictionaries.

Algorithm:-

Step 1:- We create a merge function where The first dictionary gets update with the elements of the second dictionary using the update() method which updates the dictionary with the specified key-value pairs

Step 2:- We then use this function to merge 2 dictionaries.

The screenshot shows a Windows desktop environment. In the center is a terminal window titled 'IDLE Shell 3.9.5' running Python 3.9.5. The code inside the terminal is:

```
dict_1 = {1: 'a', 2: 'b'}
dict_2 = {3: 'c', 4: 'd'}
dict_3 = dict_2.copy()
dict_3.update(dict_1)
print(dict_3)
```

The output of the code is:

```
{2: 'b', 3: 'c', 1: 'a'}
```

To the left of the terminal window is a Notepad document titled 'demo2.py'. The code in the Notepad is identical to what was run in the terminal:

```
#for each in file:
#    print(each)
#print(file.readline())
#file.close()
```

15. Write a program to sort the elements of a dictionary in two ways - i) according to keys and ii) according to values

16. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

Algorithm:-

Step 1:- Create an empty dictionary.

Step 2:- Using a for loop assign the keys and values to the empty dictionary where each value is the square of the key using the range() function.

The screenshot shows a Windows desktop environment. In the foreground, there is an 'IDLE Shell 3.9.5' window titled 'idle1'. It contains Python code that creates a dictionary where keys are integers from 1 to 15 and values are their squares. The output of the code execution is shown in the shell window. Below the shell window, there is a code editor window with more Python code, including a file reading operation. The taskbar at the bottom shows several pinned icons, and the system tray indicates the date as 18-10-2021 and the time as 07:15.

```
idle1.py - ChainedProject\Working\idle1.py (1.0.0)
File Edit Format Run Options Window Help
idle1.py
for k in range(1,16):
    dic[k] = k**2
print(dic)

#For each in file:
#    print(each)
#print("-----")
#Print file.readlines()

RESTART: C:\Users\Bpaxasd\Desktop\dic1.py
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}
>>>
```

Python for Bioinformatics

Exercise 5

V-Title: Programs demonstrating implementation of Functions

1. Write a program to determine whether the number entered is Armstrong or not (should not be restricted to no. of digits)

Algorithm:-

Step 1:- The user has to enter any number.

Step 2:- Count the Number of individual digits (For Example, 370 means 3).

Step 3:- Divide the given number into individual digits (For Example, Divide 370 into 3, 7, and 0).

Step 4:- Calculate the power of n for each individual and add those numbers.

Step 5:- Compare the original value with Sum value.

Step 6:- If they exactly matched, then it is an Armstrong number else it is not Armstrong

The screenshot shows the Spyder Python 3.7 IDE interface. The left pane displays a Python script named 'temp.py' with the following code:

```
1 num = 153
2
3 # changed num variable to string.
4 # and calculated the length (number of digits)
5 order = len(str(num))
6
7 sum = 0
8
9 # find the sum of the cube of each digit
10 temp = num
11 while temp > 0:
12     digit = temp % 10
13     sum += digit ** order
14     temp //= 10
15
16 # printing the result
17 if num == sum:
18     print(num,"is an Armstrong number")
19 else:
20     print(num,"is not an Armstrong number")
```

The right pane shows the IPython console output:

```
In [7]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
153 is not an Armstrong number

In [8]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
1534 is an Armstrong number

In [9]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
1384 is not an Armstrong number

In [10]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
153 is an Armstrong number

In [11]:
```

2. Write a program to calculate the factorial of the number entered by the user (function recursion)

Algorithm:- We use recursive method to multiple all the number ranging from the input to 1 .

The screenshot shows the Spyder Python 3.7 IDE interface. The code editor window contains a script named 'temp.py' with the following content:

```
1 num = 8
2 # To take input from the user
3 num = int(input("Enter a number: "))
4
5 factorial = 1
6
7 # check if the number is negative, positive or zero
8 if num < 0:
9     print("Sorry, factorial does not exist for negative numbers")
10 elif num == 0:
11     print("The factorial of 0 is 1")
12 else:
13     for i in range(1,num + 1):
14         factorial = factorial*i
15     print("The factorial of",num,"is",factorial)
```

The IPython console window shows the execution of the script:

```
In [12]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
The factorial of 7 is 5040

In [13]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
The factorial of 3 is 6

In [14]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
The factorial of 8 is 720

In [15]:
```

The status bar at the bottom indicates the file is open with permissions RW.

3. Print all odd and even numbers separately entered in the user defined range (Use Lambda function)

Algorithm:-

Step 1:-Take user input.

Step 2:- If user input i.e. num%2 is equal to 0 then the num is even

Step 3:-if num%2 is not equal to 0 then it is odd.

The screenshot shows the Spyder Python 3.7 IDE interface. The code editor window contains a script named 'temp.py' with the following content:

```
1 num = int(input("Enter a number: "))
2 if (num % 2) == 0:
3     print("{} is Even".format(num))
4 else:
5     print("{} is Odd".format(num))
```

The IPython console window shows the execution of the script. It prompts the user to enter a number, and then prints whether the number is even or odd. The output for two different inputs is shown:

```
In [15]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
Enter a number: 32
32 is Even

In [16]: 33
Out[16]: 33

In [17]: runfile('C:/Users/Prasad/.spyder-py3/temp.py', wdir='C:/Users/Prasad/.spyder-py3')
Enter a number: 65
65 is Odd

In [18]:
```

The taskbar at the bottom of the screen also displays various application icons.

4. Write a program to create grade calculator using dictionaries -
 (Creating a dictionary which consists of the student roll no, assignment marks (4 subjects), midterm exam marks (4 subjects), final exam marks (4 subjects) and their practical results (2 subjects). Calculate the aggregate, their percentage and grades accordingly and display. Calculate the average percentage and grade of the entire class

```
Pooja = { "rollno": "BID10015",
    "assignment" : [8,10,7,9],
    "midterm" : [24,26,28, 25],
    "theory" : [45,50,43,47],
    "lab" : [44, 45]
}
```

Algorithm:-

Step 1:- Creating a dictionary which consists of the student roll no, assignment marks, midterm exam marks, final exam ma and their practical results.

Step 2:- Create a program that calculates the aggregate, their percentage and grades accordingly.

Step 3:- Create a program that also calculate the average percentage and grade of the entire class

Step 4:- Print the average as well as grade of all the students

```

# Name of the student
Jahnvi = { "name": "Jahnvi",
    "assignment" : [90, 10, 40, 20],
    "test" : [35, 75],
    "lab" : [12.20, 73.20]
}

Neha = { "name": "Neha",
    "assignment" : [132, 24, 44, 30],
    "test" : [80, 80],
    "lab" : [67.80, 68.72]
}

Santosh = { "name": "Santosh",
    "assignment" : [67, 92, 23, 29],
    "test" : [78, 27],
    "lab" : [80, 40]
}

Prachi = { "name": "Prachi",
    "assignment" : [47, 45, 77, 21],
    "test" : [40, 50],
    "lab" : [89, 48.54]
}

Shreya = { "name": "Shreya",
    "assignment" : [143, 88, 60, 18],
    "test" : [65, 36],
    "lab" : [80, 40.8]
}

# Function calculates average
def get_average(marks):
    total_sum = sum(marks)
    total_num = float(total_sum)
    return total_sum / len(marks)

# Function calculates total average
def calculate_total_average(students):
    assignment = get_average(students["assignment"])
    test = get_average(students["test"])
    lab = get_average(students["lab"])

    # Return the result based on
    # formula
    return ((0.2 * assignment) + (0.3 * test) + (0.5 * lab))

# Main function
if __name__ == "__main__":
    students = [
        {"name": "Jahnvi", "marks": {"assignment": [90, 10, 40, 20], "test": [35, 75], "lab": [12.2, 73.2]}},
        {"name": "Neha", "marks": {"assignment": [132, 24, 44, 30], "test": [80, 80], "lab": [67.8, 68.72]}},
        {"name": "Santosh", "marks": {"assignment": [67, 92, 23, 29], "test": [78, 27], "lab": [80, 40]}},
        {"name": "Prachi", "marks": {"assignment": [47, 45, 77, 21], "test": [40, 50], "lab": [89, 48.54]}},
        {"name": "Shreya", "marks": {"assignment": [143, 88, 60, 18], "test": [65, 36], "lab": [80, 40.8]}}
    ]
    print("Average Marks of Jahnvi is : ", get_average(students["Jahnvi"]["marks"]))
    print("Letter Grade of Jahnvi is : E")

    print("Average Marks of Neha is : ", get_average(students["Neha"]["marks"]))
    print("Letter Grade of Neha is : C")

    print("Average Marks of Santosh is : ", get_average(students["Santosh"]["marks"]))
    print("Letter Grade of Santosh is : E")

    print("Average Marks of Prachi is : ", get_average(students["Prachi"]["marks"]))
    print("Letter Grade of Prachi is : E")

    print("Average Marks of Shreya is : ", get_average(students["Shreya"]["marks"]))
    print("Letter Grade of Shreya is : E")

    print("Class Average is : ", calculate_total_average(students))
    print("Letter Grade of the class is : E")

```

```
[A demohelpy - C:\Users\Pasad\Desktop\demohelpy (3.9.7)
File Edit Format Run Options Window Help
# Function calculate total average
def calculate_total_average(students):
    assignment = get_average(students["assignment"])
    test = get_average(students["test"])
    lab = get_average(students["lab"])

    # Returns the result based
    # on weightage supplied
    # 10 % from assignments
    # 10 % from test,
    # 20 % from lab-work
    return (.1 * assignment +
            .7 * test + .2 * lab)

# Calculate letter grade of each student
def assign_letter_grade(score):
    if score >= 90: return "A"
    if score >= 80: return "B"
    if score >= 70: return "C"
    if score >= 60: return "D"
    else: return "F"

# Function to calculate the total
# average marks of the whole class
def class_average_ls(student_list):
    result_list = []

    for student in student_list:
        stud_avg = calculate_total_average(student)
        result_list.append(stud_avg)
    return get_average(result_list)

# Student list consisting the
# dictionary of all students
students = [Jahnvi, Neha, Samnash, Prachi, Shreya]

# Iterate through the students list
# and calculate their respective
# average marks and letter grade
for i in students:
    print(i["name"])
```

5. Write a program to calculate Molecular weight of the peptide sequence submitted by the user.
Subtract the weight of water molecules.

Algorithm:-

Step 1:-Create a dictionary consisting of the molecular weight of all the amino acids.

Step 2:- Create a if loop where it will print the values of the key that matches with the input amino acid and adds the values (M.W of the amino acid) to an empty list.

Step 3:-Find the sum of all values in the list and apply the formula to get the melcular weight of the peptide sequence.

Step 4:- formula =Sum of the M.W of all the amino acids - the length of peptide sequence minus 1 into 18 g/mol (mol of waters)

6. Write a program to calculate the frequency of bases in the DNA sequence submitted by the user

Algorithm:-

Step 1:-Initialize the counts of A, C, G, T to 0 and take user input.

Step 2:-Create if loop's where if the base of the DNA sequence matches with either A, C, G, T, then their count gets incremented.

Step 3:- Print the final count of all the bases in the DNA sequence.

The image shows a Windows desktop environment. In the foreground, there is a taskbar with several icons. On the desktop, there are two windows: a Notepad window titled 'demo1.py' and an IDLE Shell window titled 'IDLE Shell 3.3.5'. The Notepad window contains the following Python code:

```
def count_dna(seq, allowed_bases=['A','T','C','G']):
    seq = seq.upper()
    total_dna_bases = 0
    for base in allowed_bases:
        total_dna_bases += seq.count(base)
    dna_fraction = total_dna_bases / len(seq)
    return(dna_fraction * 100)
print(count_dna("ACCGGGTTTCCCCCTTAAATTT"))
print(count_dna("ACTTGATTCATCCATCAATG", ["A","T","C","G"]))
print(count_dna("actggatccatccatcaatg"))
```

The IDLE Shell window shows the output of running this code. It displays three lines of text:

```
>>> RESTART: C:\Users\Prasad\Desktop\demo1.py
0.363636363636
90.9090909090909
0.363636363636
```

6. Write a program to perform 1st frame [translation](#) of a dna sequence into a protein sequence.

Algorithm:-

Step 1:- Create a dictionary consisting of all the codon sequences and their corresponding amino acids.

Step 2:-Create a for loop with range(length of the sequence -2) so as to get the proper reading frame.

Step 3:- If the codon sequence matches with the appropriate key in the dictionary then print the corresponding value of that key.

Step 4:- Print the protein sequence of DNA sequence.

```
File Edit Shell Options Windows Help
seq = input("Enter a gene sequence: ")
new_seq = seq.replace("T", "U")

def translate(codon):
    cod_dict = {
        "UUU": "F", "UUC": "F", "UUA": "L", "UUG": "L",
        "CUU": "S", "CUC": "S", "CUA": "L", "CUG": "L",
        "AUU": "I", "AUC": "I", "AUA": "N", "AUG": "M",
        "GUU": "V", "GUC": "V", "GUA": "A", "GUG": "D"
    }
    return cod_dict[codon]

print("Enter a gene sequence: ATGCUAUCUACGU")
print(translate("ATG"))
print(translate("CUA"))
print(translate("UUC"))
print(translate("GAA"))
print(translate("UAC"))
print(translate("GCU"))
print(translate("AUU"))
print(translate("GUA"))
print(translate("UUA"))
print(translate("GUC"))
print(translate("AUU"))
print(translate("GUA"))
print(translate("UAC"))
print(translate("GCA"))
print(translate("UCA"))
print(translate("UCC"))
print(translate("UAU"))
print(translate("GUA"))
print(translate("UAA"))
print(translate("UAG"))

RESTART: C:\Users\Pranad\Desktop\demhai.py
>>> Enter a gene sequence: ATGCUAUCUACGU
ATG
CUA
UUC
GAA
UAC
GCU
AUU
GUA
UUA
GUC
AUU
GUA
UAC
GCA
UCA
UCC
UAU
GUA
UAA
UAG
>>> 
```

Python for Bioinformatics

Exercise 6

VI: Programs demonstrating use of modules

1. Create a package incorporating a module for calculating GC percentage of the nucleotide sequence, a module for translating the nucleotide sequence and a module for calculation of Molecular Weight of the generated peptide sequence

Algorithm:-

Main File:-

Step 1:-Create 2 modules for calculating GC percentage module along with calculation of Molecular Weight of peptide sequence.

Step 2:-Import both the modules in the main file.

Step 3:- Take user input for their choice of function to execute.

Step 4:- If user input is 1 use the gc_percentage() function from the imported modules.

Step 5:- If user input is 2 use the mol_wt() function from the imported modules.

GC percentage module:-

Step 1:-In the GC percentage module, you initialize the count of G and C to 0.

Step 2:-Create an if loop where when the sequence has either G or C the count increases by 1.

Step 3:- Calculate the sum of the counts of both the G and C and then divide it by the length of the sequence so as to calculate the percentage of the gc in the sequence.

Mol wt module:-

Step 1:-Create a dictionary consisting of the molecular weight of all the amino acids.

Step 2:- Create a if loop where it will print the values of the key that matches with the input amino acid and adds the values (M.W of the amino acid) to an empty list.

Step 3:-Find the sum of all values in the list and apply the formula to get the molecular weight of the peptide sequence.

Step 4:- formula =Sum of the M.W of all the amino acids - the length of peptide sequence minus 1 into 18 g/mol (mol of waters)

```

[A:\] mosharafa - C:\Users\Praxed\Desktop\bemal.py [19.7]
File Edit Insert Run Options Window Help
Sequence = input("Enter Seq:")
def count_dna(seq, allowed_bases=['A', 'C']):
    seq=seq.upper()
    total_dna_bases=0
    for base in allowed_bases:
        total_dna_bases+=seq.count(base)
    dna_fraction=total_dna_bases/len(seq)
    dna_fraction=(dna_fraction*100)
    print("Frequency of GC is:")
    print(count_dna(Sequence))
>>>

```

File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7f0f963, Aug 30 2021, 20:19:10) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>

RESTART: C:\Users\Praxed\Desktop\bemal.py

Enter SequenceCATGCATGCACTGATGATGACTGATGACG
Frequency of GC is:
45.454545454545454

```

print('press 1 if you want to calculate gc percentage')
print('\n')
print('press 2 if you want to calculate mol.wt')
print('\n')
x=input("Select your function: ")
if x=='1':
    print('you Have selected function 1 --> calculate gc percentage')
    from python_packages import gcper
if x=='2':
    g=print('you Have selected function 2 --> calculate mol.wt')
    from python_packages import molwt

```

```

def gc_percentage(x):
    A=0
    C=0
    G=0
    T=0
    for i in x:
        if i=='A':
            A=A+1
        if i=='C':
            C=C+1
        if i=='G':
            G=G+1
        if i=='T':
            T=T+1
    for f in x:
        if f!='A':
            if f!='C':
                if f!='G':
                    if f!='T':
                        print(str(f) + ' is not a valid base hence the sequence is wrong')

    break
    print(((C+G)/len(x))*100)

gc_percentage(input('Enter your sequence: '))

```

```

def molwt(y):
    mol_wt={ #Dictionary consisting of M.W of the amino acids
        'A':89.09,
        'C':121.16,
        'E':147.13,
        'F':165.19,
        'D':133.10,
        'G':75.07,
        'H':155.16,
        'I':131.18,
        'K':146.19,
        'L':131.18,
        'M':149.21,
        'N':132.12,
        'P':115.13,
        'Q':146.15,
        'R':174.20,
        'S':105.09,
        'T':119.12,
        'V':117.15,
        'W':204.23,
        'Y':181.19,
    }
    {
    }

    values=[]
    seq=y

    def checkpep(seq):
        for x in mol_wt:
            if seq == x:
                wt=mol_wt[x]      #will print the values of the key that matches with the input amino acid
                values.append(wt) #adds the values (M.W of the amino acid) to an empty list

    for x in seq:
        checkpep(x)
    print(values)
    total=0
    checkpep(x)
    for x in seq:
        checkpep(x)
    print(values)
    total=0
    checkpep(x)
    for ele in range(0, (len(values)-1)):
        total = total + values[ele]
        #to find the sum of all the values in the list
    #formula -sum of the M.W of all the amino acids - the length of peptide sequence minus 1 into 18 g/mol (mol of waters)

    final_value= total-(len(seq)-1)*18
    print(final_value)

molwt(input('Enter your sequence: '))

```

press 1 if you want to calculate gc percentage

press 2 if you want to calculate mol.wt

```

Select your function: 1
you Have selected function 1 --> calculate gc percentage
Enter your sequence: ACGATA
33.33333333333333
>>> |

```

```
press 1 if you want to calculate gc percentage  
  
press 2 if you want to calculate mol.wt  
  
Select your function: 2  
you Have selected function 2 --> calculate mol.wt  
Enter your sequence: ADF  
[89.09, 133.1, 165.19]  
351.38  
>>> |
```

Python and Bio-Python

Practical no: 7

Practical Solving:

Perform the following questions using functions:

1. Open a file and print the contents of the file on the screen

Algorithm:-

Step 1:- Open a file using the open() function where you enter the file location in the brackets.

Step 2:-use the .read() function to read the contents of the file and to print them in the console.

The screenshot shows two windows side-by-side. The left window is titled 'idle.py - C:\Users\Praaad\Desktop\demo.py (1.0)'. It contains the following Python code:

```
# a file named "myfile", will be opened with reading mode.
file = open ("C:/Users/Praaad/Desktop/Documents/myfiles.txt",'r')
#this will print the content
for each in file:
    print(each)
print ("")
```

The right window is titled 'IDLE Shell 3.9.5'. It shows the output of the code execution:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbf, May  3 2021, 17:27:52) |MSC v.1928 64 bit (AMD64)| on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:/Users/Praaad/Desktop/demo.py =====
Computer can understand only the language of Digital Electronics. Digital Electronics deals with presence and absence of voltages.

Within the computer there are two logics can play their role. These logics are & 0
Positive Logic &#39; Here presence of voltage will be denoted by 1 and absence of voltage will be denoted by 0
Negative Logic &#39; Here presence of voltage will be denoted by 0 and absence of voltage will be denoted by 1

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability
with its use of significant indentation, its language constructs as well as its object-oriented approach aim to help programmers
write clear, logical code for small and large-scale projects.
Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured
(particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included"
language due to its comprehensive standard library.
>>>
```

Below the windows, the Windows taskbar is visible with icons for File Explorer, Task View, Start, Taskbar settings, and system status.

The screenshot shows two windows side-by-side, identical to the one above. The left window is titled 'idle.py - C:\Users\Praaad\Desktop\demo.py (1.0)'. It contains the same Python code as before:

```
# a file named "myfile", will be opened with reading mode.
file = open ("C:/Users/Praaad/Desktop/Documents/myfiles.txt",'r')
#this will print the content
for each in file:
    print(each)
print ("")
```

The right window is titled 'IDLE Shell 3.9.5'. It shows the output of the code execution:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbf, May  3 2021, 17:27:52) |MSC v.1928 64 bit (AMD64)| on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:/Users/Praaad/Desktop/demo.py =====
Computer can understand only the language of Digital Electronics. Digital Electronics deals with presence and absence of voltages.

Within the computer there are two logics can play their role. These logics are & 0
Positive Logic &#39; Here presence of voltage will be denoted by 1 and absence of voltage will be denoted by 0
Negative Logic &#39; Here presence of voltage will be denoted by 0 and absence of voltage will be denoted by 1

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability
with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers
write clear, logical code for small and large-scale projects.
Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured
(particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included"
language due to its comprehensive standard library.
>>>
```

Below the windows, the Windows taskbar is visible with icons for File Explorer, Task View, Start, Taskbar settings, and system status.

The image displays two side-by-side screenshots of a Windows desktop environment. Both screens show a Python script named `demo1.py` and its execution in the IDLE Shell 3.9.5 interface.

Top Screenshot:

- File Path:** C:\Users\Prasad\Desktop\demo1.py (1.00)
- Code:**

```
# A file named "myfile", will be opened with reading mode.  
file = open ("C:/Users/Prasad/Documents/myfiles.txt",'r')  
#This will print the content.  
  
#For each in file:  
#    print(each)  
#    print ("---")  
  
print (file.readline())
```
- IDLE Shell Output:**

```
IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f0fbd9d, May 3 2021, 17:27:53) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Prasad\Desktop\demo1.py ======  
Computer can understand only the language of  
>>> |
```

Bottom Screenshot:

- File Path:** C:\Users\Prasad\Desktop\demo2.py (1.00)
- Code:**

```
# A file named "myfile", will be opened with reading mode.  
file = open ("C:/Users/Prasad/Documents/myfiles.txt",'r')  
#This will print the content.  
  
#For each in file:  
#    print(each)  
#    print ("---")  
  
print (file.readline())
```
- IDLE Shell Output:**

```
IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:f0fbd9d, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Prasad\Desktop\demo2.py ======  
Computer can understand only the language of Digital Electronics. Digital Electronics deals with presence and absence of voltages.  
>>> |
```

2. Write a Python program to append text to a file and display the text

Algorithm:-

Step 1:- Open a file using the open() function where you enter the file location in the brackets.

Use 'r' in the brackets to read the file

Step 2:-use the .read() function to read the contents of the file and to print them in the console.

Step 3:- Open the second file using the open() function where you enter the file location in the brackets. Use 'a' in the brackets to append text to the file

Step 4:- Use .close() to close the file.

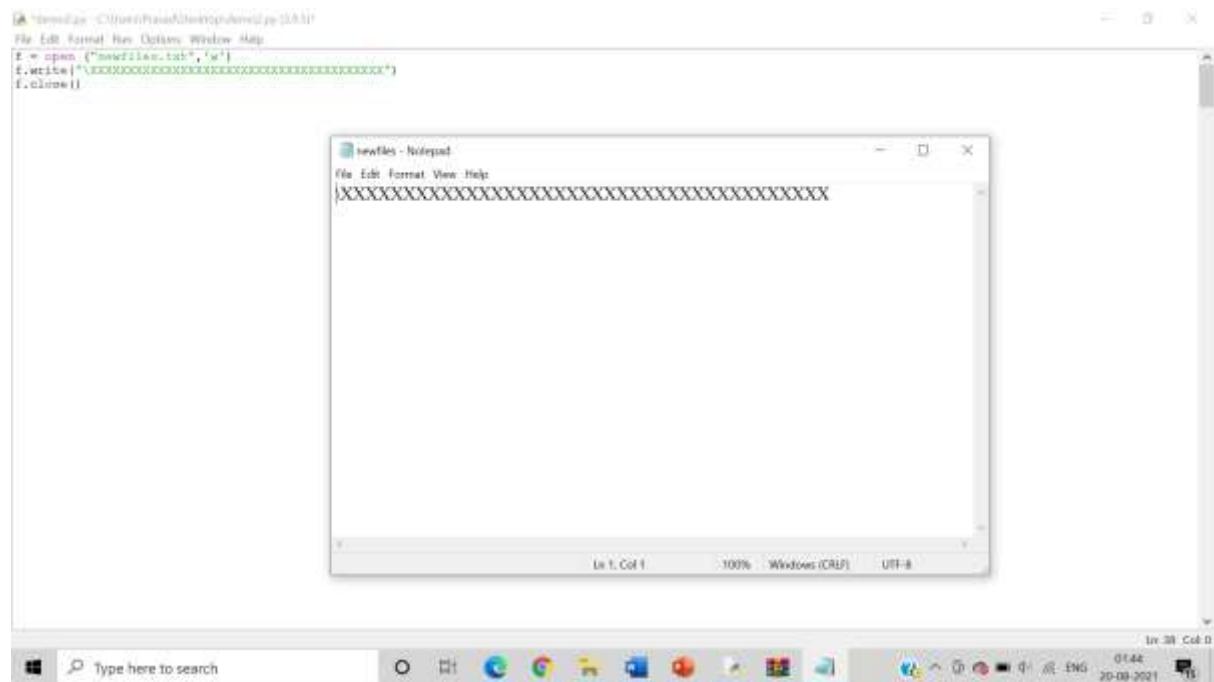
The screenshot shows a Windows desktop environment with two windows open. The top window is a terminal window titled 'cmd' with the path 'C:\Users\Prasad\Desktop\demol.py (1.0.0)'. It contains the following Python code:

```
f = open("newfile1.txt","a")
f.write("Now the file has more content!")
f.close()

#open and read the file after appending
f = open("newfile1.txt",'r')
print(f.read())

f = open("newfile1.txt","w")
f.write("Now the files has more content!")
f.close()
```

The bottom window is a Notepad window titled 'newfile1 - Notepad'. It displays the text "Now the files has more content!".



3. Write a program asking for user details and print those details onto a file

Algorithm:-

Step 1:- Open a file using the open() function where you enter the file location in the brackets. Use 'x' in the brackets where x is open for exclusive creation, failing if the file already exists

Step 2:- Use the input() function to take user input.

Step 3:- Then use the with open method to open the text file as output use the .write() function to add the user input to the text file.

Step 4:- Use the .read() function to read the file.

The screenshot shows a Windows desktop environment. In the foreground, there is a code editor window titled 'idle3.py - C:\Users\Prasad\Desktop\idle3.py (1.0.0)'. It contains the following Python code:

```
#Print User Input to A File
x = input("Name : ")
y = open("thisfile.txt", "w")
y.write(x)

x = input("Roll No : ")
y = open("thisfile.txt", "r")
y.write(x)
```

Below the code editor is a Python terminal window titled 'OLE Shell 3.9.5'. The terminal shows the following output:

```
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f0f5057, May 3 2021, 11:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\idle3.py
Name : Omjan
Roll No : 4533
```

The taskbar at the bottom of the screen shows the Start button, a search bar with 'Type here to search', and several pinned icons for applications like File Explorer, Edge, and File History. The system tray on the right side of the taskbar displays the date (20-08-2021), time (21:45), battery level (70%), and language (ENG).

4. Write a program to copy contents of one file to another

Algorithm:-

Step 1:- Open a file using the open() function where you enter the file location in the brackets.

Use 'r+' to open a disk file for updating

Step 2:-

Step 3:- Then use the with open method to open the text file as output use the .write() function to add the user input to the text file.

Step 4:- Use the .read() function to read the file.

```
#!/usr/bin/python
f = open("file.txt", "r+")
str=""

for x in f:
    str+=x
f.close()

print(str)
print("-----")

mylist=[]
mylist = str.split("\n")
print(mylist)

f = open("newfile.txt", "w")
f.write(str)
f.close()
```

The screenshot shows a Windows desktop environment with two open windows. The left window is a code editor titled 'Untitled - C:\Users\Praaad\Desktop\demo2.py (1.00)'. It contains Python code that reads 'file.txt' and writes its contents to 'newfile.txt'. The right window is a terminal titled 'IDLE Shell 3.9.5' running Python 3.9.5. It shows the output of the script, which prints 'this is a file' to the console.

```
#> idle.py - C:\Users\Praaad\Desktop\demo2.py (1.00)
File Edit Format Run Options Window Help
f = open("file.txt",'r')
strw=""
for s in f:
    strw+=s
f.close()

print(str)
print("")

mylist=[]
mylist = str.split("\n")
print(mylist)

f = open("newfile.txt",'w')
f.write(str)
f.close()

#> idle.py - C:\Users\Praaad\Desktop\demo2.py (1.00)
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:f020d98, May  3 2021, 17:27:02) [MSC v.1938 64 bit (AMD64)] on Win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -----
this is a file
['this is a file']
>>>
```

5. Write a program to reverse the content of a file and store it in another file (apply full reversing : “hello world” => “dlrow olleh” and words reversing : “hello world” – “world hello”)

Algorithm:-

Step 1: Start.

Step 2: Open the files.

Step 3: Access the elements of the file in reverse and write them in thefile.

Step 4: Open the file in read mode.

Step 5: Close the file.

Step 6: So far we have reversed the entire string.

Step 7: To reverse only the alphabets in the strings, we will have to split the string.

Step 8: reverse() returns an iterator that accesses the given sequence in the reverse order.

Step 9: Write this in the file.

Step 10: Open the file in read mode.

Step 11: Close the file.

Step 12: Stop.

A screenshot of a Windows operating system desktop. At the top, there is a taskbar with various icons. Below the taskbar, there are three windows open: 1) A terminal window titled 'cmd' with the path 'C:\Users\Vaibhav\Dev\Python\Chapter 09'. It contains Python code to reverse a file's content. 2) A Notepad window titled 'Complish - Notepad' with the text 'dlrow olleh'. 3) Another Notepad window titled '9fe - Notepad' with the text 'hello world'. The desktop background is white.

```
f1 = open("newfile.txt", "w")  
with open("file.txt", "r") as myfile:  
    data = myfile.read()  
  
data_i = data[::1]  
  
f1.write(data_i)  
f1.close()
```

Python and Bio-Python

Assignment 9

Programs demonstrating use of Randomization in Python

1. Demonstrate mixing of lists elements and generate a funny story

Algorithm:-

Step 1:-Import the random function.

Step 2:-Create 2 lists 'a' and 'b'.

Step 3:-We then use list indexing where the index value is randomly generated using the randint() function.

Step 4:- We then use the for loop to print multiple funny stories.

The screenshot shows a Windows desktop environment. In the center, there is a Python IDLE Shell window titled 'IDLE Shell 3.8.5'. The code editor window is titled 'idle2.py'. Both windows are running on a desktop with a light blue background. The taskbar at the bottom shows icons for File Explorer, Edge browser, Google Chrome, FileZilla, and Task View. The system tray on the right shows battery status, signal strength, and the date '05-09-2021'.

```
idle2.py - C:\Users\Praaad\Desktop\idle2.py (1.0)
File Edit Format Run Options Window Help
import random
a = ["teacher", "dancer", "singer", "actor", "driver"]
b = ["teaching", "dancing", "singing", "acting", "driving"]
for i in range(10, 5): print(a[random.randint(0, 4)], b[random.randint(0, 4)])
```

```
[idle2.py] IDLE Shell 3.8.5
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:5a8df873cbdb, May  3 2021, 17:07:52) [MSC v.1920 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Praaad\Desktop\idle2.py -----
dancer is singing
dancer is acting
driver is acting
driver is dancing
actor is dancing
>>>
```

```
#f = open ("newfile.txt","w")
#f.write("Good Day")
#f.close()

ff = open ("newfile.txt","r")
print(ff.read())
ff.close()
```

2. Write a program to create a “Guess the Number” game

Algorithm:-

Step 1:-Import the random function and use the randint() function to generate random numbers which are then assigned to the number variable.

Step 2:-Create multiple if else conditions where if the user input matches with the random number assigned to the number variable then the loop exits and the game is won.

Step 3:-In case the user input doesn't match then if loop condition the code runs the next loop condition until the chances are exhausted.

The code in the editor window:

```
# Importing the required module
import random

# First take the Inputs
lower = int(input("Please enter your Lower bound:- "))

upper = int(input("Please enter your Upper bound:- "))

x = random.randint(lower, upper)
print("Well! You have only",
      round(math.log(upper - lower + 1, 2)),
      " chances to guess the integer!\n")

# Initializing the number of guesses.
count = 0

while count < math.log(upper - lower + 1, 2):
    count += 1

    # taking guessing number as input.
    guess = int(input("Please guess a number:- "))

    # Condition testing
    if x == guess:
        print("Well! You did it in",
              count, " try")
        # Once guessed, loop will break
        break
    elif x > guess:
        print("No, Too small!")
    elif x < guess:
        print("No, too large number!")

if count >= math.log(upper - lower + 1, 2):
    print("\nThe number is", x)
    print("\nGood try!")



```

The terminal window output:

```
Please enter your Lower bound:- 1
Please enter your Upper bound:- 33
You have only 6 chances to guess the integer!
Please guess a number:- 45
No, too large number!
Please guess a number:- 88
No, too large number!
Please guess a number:- 12
No, Too small!
Please guess a number:- 33
No, Too small!
Please guess a number:- 46
No, too large number!
Please guess a number:- 75
No, too large number!
The number is 21
        Good try!
>>>
```

3. Write a program to perform insertion, substitution and deletion mutation in a user entered DNA string

Algorithm:-

Step 1:-User input is given for the type of function the user wants to perform.

Step 2:-Import the random function.

Step 3:-Create multiple if-else loops for the multiple functions.

Step 4:-we use the choice() and randint() function to perform insertion, substitution and deletion mutations randomly in a random position in the user given input sequence.

The screenshot shows a Windows desktop environment. In the foreground, there is a terminal window titled 'cmdshell325' running Python 3.9.5. The command prompt shows the path 'C:\Users\Prasad\Desktop\dasm1.py'. The user has typed 'Enter the dna sequence ATCC' and pressed Enter. The terminal then displays the output 'AT ATCC'. Below the terminal window, the taskbar shows the Start button, a search bar with 'Type here to search', and several pinned icons for Google Chrome, File Explorer, Task View, and others. The system tray at the bottom right indicates the date as '05-09-2021' and the time as '10:50 AM'.

```
dasm1.py - Created: Friday, September 3, 2021 10:50:00 AM
File Edit Format Run Options Window Help
import random

dna_string = input ("Enter the dna sequence")

def insertion(dna, index):
    return dna[:index] + "T" + dna[index:]

random_index = random.randint(0, len(dna_string))
print (insertion ("ATCC", random_index))

#>>> Enter the dna sequence ATCC
AT ATCC
>>>
```

```
#f = open ('newfileex.txt','w')
#f.write("Good Day")
#f.close()

f1 = open ('newfile1.txt','r')
print(f1.read())
```

Python and Bio-Python

Assignment 10

Practical Solving:

Perform the following questions using regular expressions

1. Check whether a string starts and ends with the same character or not

Algorithm:-

Step 1: Import the re module

Step 2: Define the regex pattern

Step 3: Check the pre-defined inputs for the pattern

Step 4: IF the pattern exists : print ("Valid")

Step 5: Else: print ("Invalid")

The screenshot shows a Windows desktop environment. On the left, there is a code editor window titled 'demo2.py' containing Python code. On the right, there is an 'IDLE Shell 3.9.5' window showing the execution of the script.

```
# than comment the regular expression
regex = r"^(a|z){2}^$"
# now function for checking the string
def check(string):
    if re.search(regex, string):
        print("It is valid")
    else:
        print("It is invalid")
if __name__ == '__main__':
    sample1 = "this is a cat"
    sample2 = "this is a dog"
    sample3 = "this is a string"
check(sample1)
check(sample2)
check(sample3)
```

The IDLE Shell window displays the following output:

```
[A] IDLE Shell 3.9.5
File Edit Shell Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0d2d5, May 3 2021, 1710762) [GCC v10.2.0 64-bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\Praaad\Desktop\demo2.py =====
It is valid
It is invalid
It is valid
>>>
```

2. Write a program to Extract IP address from file using Python
Algorithm:-

Step 1: import re

Step 2: open the .txt file

Step 3: Define the regex pattern for an ip address

Step 4: Search for pattern within the file

Step 5: if found : print(empty)

```
#> In [1]: C:\Users\Prasad\Desktop\devel\devel.py (1.0)
#> File Edit Shell Debug Options Window Help
#> def isValidIP(s):
#>     # First check the number of periods
#>     if s.count('.') != 3:
#>         return "It is an invalid Ip address"
#>
#>     l = list(map(str, s.split('.')))
#>
#>     # now check the range of each number between periods
#>     for ele in l:
#>         if int(ele) < 0 or int(ele) > 255:
#>             return "It is an invalid Ip address"
#>
#>     return "It is a Valid Ip address"
#>
#> # Driver Code
#> print(isValidIP("192.168.1.1"))

#> RESTART: C:\Users\Prasad\Desktop\devel.py
#> In [2]: It is a Valid Ip address
#> In [3]:
```

```
#> In [1]: C:\Users\Prasad\Desktop\devel\devel.py (1.0)
#> File Edit Shell Debug Options Window Help
#> def isValidIP(s):
#>     # First check the number of periods
#>     if s.count('.') != 3:
#>         return "It is an invalid Ip address"
#>
#>     l = list(map(str, s.split('.')))
#>
#>     # now check the range of each number between periods
#>     for ele in l:
#>         if int(ele) < 0 or int(ele) > 255:
#>             return "It is an invalid Ip address"
#>
#>     return "It is a Valid Ip address"
#>
#> # Driver Code
#> print(isValidIP("192.1.1"))

#> RESTART: C:\Users\Prasad\Desktop\devel.py
#> In [2]: It is an invalid Ip address
#> In [3]:
```

3. Write a program to Categorize Password as Strong or Weak using Regex in Python
Algorithm:-

Step 1: Import re

Step 2: Check the length of password

Step 3: if length >=8:

Step 4: Check for special characters, capital letters, small letters, numbers within the input

Step 5: if all of the above are present: print("your password is strong")

Step 6: else : print("your password is weak")

Step 7: else : print("Invalid password")

The screenshot shows a Windows desktop environment. On the left is a code editor window titled 'Untitled - Untitled Document - Python (3.9.5)'. It contains Python code for password validation. On the right is an 'IDLE Shell 3.9.5' window, which is a Python interactive shell. The shell window shows the execution of the code and its output.

```
import re

def password(v):
    if v == "\n" or v == " ":
        return "Password cannot be a blankline or space!"

    # the password length should be in between 8 and 20
    if 8 <= len(v) <= 20:
        if re.search("[A-Z][a-z]", v):
            if re.search("[0-9]", v):
                if re.search("[!@#$%^&*()_+=-]", v):
                    return "Strong Password"
                else:
                    return "Weak Password: The same character repeats three or more times in a row"
            else:
                return "Weak Password: It is a same string pattern repetition"
        else:
            return "Strong Password"

    else:
        return "Password length must be 8-20 characters!"

# Main method
if __name__ == "__main__":
    # Driver code
    print(password("Elephant@123"))
    print(password("Carina@1"))
    print(password("One password1"))
    print(password("junkpassword"))
    print(password("00 123"))
    print(password("this is a password 123"))

# Driven Code
if __name__ == "__main__":
    main()
```

```
[IDLE Shell 3.9.5]
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbf, May  3 2021, 17:23:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\demo2.py =====
Strong Password!
Password length must be 8-20 characters!
Strong Password!
Strong Password!
Strong Password!
Password length must be 8-20 characters!
Password length must be 8-20 characters!
>>>
```

4. Write a program to validate an Email address entered by the user
Algorithm:-

Step 1: Import re

Step 2: Take user defined input

Step 3: Define the regex pattern

Step 4: if (input==regex pattern): print("valid email id")

Step 5: else: print ("Invalid")

The screenshot shows a Windows desktop environment. On the left is a code editor window titled 'demo2.py - Untitled Document - Python (1.0.0)'. It contains Python code for validating email addresses. On the right is an 'IDLE Shell 3.9.5' window showing the output of running the script.

```
File Edit Insert Run Options Window Help
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a0a0bd, May 3 2021, 17:27:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Pramad\Desktop\demo2.py
It is a valid Email
It is a valid Email
It is an invalid Email
>>>
```

The code in the editor is:

```
import re

regex = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,10}$'

def check(email):
    if re.fullmatch(regex, email):
        print("It is a valid Email")
    else:
        print("It is an Invalid Email")

# Driver Code
if __name__ == "__main__":
    # Now you have to enter the email
    email = "sanjanadeepshankar@gmail.com"

    # calling run function
    check(email)

    email = "donotanswertothisemail"
    check(email)

    email = "email2@max.com"
    check(email)
```

5. Write a program to identify no of motifs and their positions in a protein sequence entered by the user. Print the motifs present along with count and positions.(5 motifs)

Algorithm:

Step 1: import re

Step 2: define the regex pattern

Step 3: initialize $x = \text{True}$

Step 4: result1 = re.findall(pattern,sequence)

Step 5: `result2 = refinditer(pattern,sequence)`

Step 6: initialize $b = 0$

Step 7: for x in result2: print("Motif position = " + str(x[0]) + " and motif sequence = " + str(x[1]))

Step 8: $b = b + 1$

Step 9: calculate the total no of results using len()

The screenshot shows a Windows desktop environment. In the foreground, the Python IDLE shell window is open, displaying a script and its execution results. The script uses regular expressions to find matches in a string. The output shows the number of matches found and the details of each match. The desktop background is visible, along with taskbar icons for various applications like File Explorer, Edge, and FileZilla.

```
positions = []
mydict = {"YAHOO/COM": "C:\\A\\H(A-Z){1,2}\\.L(YAHOO)\\C(A-Z){1,2}C\\G(YAHOO)", "MSN": "C:\\A\\H(A-Z){1,2}\\.L(MSN)\\C(A-Z){1,2}C\\G(MSN)", "GMAIL": "C:\\A\\H(A-Z){1,2}\\.L(GMAIL)\\C(A-Z){1,2}C\\G(GMAIL)"}

for position in positions:
    print(position)
    print(f"no of matches found are {len(positions)}")

for position in positions:
    matches = re.findall(position, s)
    if len(matches) > 0:
        for match in matches:
            print(f"match found -> {match}")

for position in positions:
    print(f"position->{position}, no of matches found->{len(positions)}")
```

```
IDLE Shell 19.7
File Edit Shell Debug Options Window Help
Python 3.9.7 |tags/v3.9.7:f0fbd0f3, Aug 30 2021, 20:19:30| (MSC v.1929 64 bit (AMD64)) on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Prasad\Desktop\dschach.py =====
no of matches found are 2
position-> 10 match found-> CYHRMEOCTY
position-> 29 match found-> CTHULUBDCA
>>>
```

```
[file:///C:/Users/Praasad/Desktop/demochai.py] (1.0.7)
File Edit Format Run Options Window Help
import re
positions={}
mystr = "RENTABELE HOMES FOR RENT IN CHENNAI"
match = re.findall("(?=(A-Z)(A-Z)(L)|W)(?=(A-Z)(A-Z)(D))",mystr)
#print (match)
pos = re.finditer("C(?=(A-Z)(A-Z)(L)|W)(?=(A-Z)(A-Z)(D))",mystr)
for m in pos:
    positions.append(m.start())
#print (positions)

mydict={}
for value in match:
    for key in positions:
        mydict[key]=value
    positions.remove(key)
    break

#print (mydict)
for x in mydict:
    print ("position->",x, "match found->",mydict[x])


```

```
[file:///C:/Users/Praasad/Desktop/demochai.py] (1.0.7)
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Praasad\Desktop\demochai.py =====
no of matches found are 2
position-> 10 match found-> CVERHESFCY
position-> 29 match found-> CTNGLCRCA
>>>
```

```
[file:///C:/Users/Praasad/Desktop/demochai.py] (1.0.7)
File Edit Format Run Options Window Help
import re
positions={}
mystr = "Julia's Shubh 223491467hp37588lines#7389156"
match = re.findall("(?=(A-Z)(A-Z)(L)|W)",mystr)
#print (match)
pos = re.finditer("C(?=(A-Z)(A-Z)(L)|W)",mystr)
for m in pos:
    positions.append(m.start())
#print (positions)

mydict={}
for value in match:
    for key in positions:
        mydict[key]=value
    positions.remove(key)
    break

#print (mydict)
for x in mydict:
    print ("position->",x, "match found->",mydict[x])
```

```
[file:///C:/Users/Praasad/Desktop/demochai.py] (1.0.7)
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Praasad\Desktop\demochai.py =====
no of matches found are 6
position-> 5 match found-> T89
position-> 12 match found-> 2224
position-> 20 match found-> 47
position-> 25 match found-> 796899
position-> 33 match found-> 47799
position-> 39 match found-> 796
>>>
```


6. Write a program that parses a fasta sequence file (use file handling) and print the raw sequence onto the console. Also parse the accession Id and print it.

Algorithm:

- Step 1: import re
Step 2: open the fasta file
Step 3: read the fasta file
Step 4: create an empty string seq
Step 5: for l in data :
Step 6: if the characters == any special symbol : skip
Step 7: else : seq += l
Step 8: print("Sequence=")

The screenshot shows a Windows command prompt window titled "dennihay - C:\Users\Priyadarshini\Desktop\dennihay (3.9.7)". The code in the window is as follows:

```
# dennihay - C:\Users\Priyadarshini\Desktop\dennihay (3.9.7)
File Edit Format Run Options Window Help
import re
f = open("inclusion.fasta", "r+")
data = f.readlines()
#PRINT (data)
seq=""
for line in data:
    if re.match("^>*", line):
        next
    elif re.match("//<*>", line):
        next
    elif re.match("//>*", line):
        next
    else:
        seq+=line
print (seq)
rawseq = re.sub("//>","",seq)
print ("the sequence is-->" + rawseq)
```

The output of the program is displayed below the code in the terminal window:

```
The sequence is --> AQKPLSDEEKFLFVDRHFTVNNPLAQADWSAKKLVWVPSKENNG
PEAASISKEERGDEVTEVLEQEHGERVTLSEKDDIQMHNPPFKFSEVEDMAELTCLENEASVLHLRE
RYPFGLIYTYSGLPCVVINHPYKQLPIYSERNIIIMYNGHPRRMENPPHIYIAIAUTAYRSHLQDRE
DQSILCTGESGAQKTETKRVVIQYLAVVASSHHOKHDTTSITQGPFSTYGELEKQLLQAMFILE
AFQNAKTVRDHSRRFGRKFIRINFDVTGYIVGANIETYILLEKSRAIRQARDERTFHIFYLLIA
GASEQMRNDLLEGGFNNHTFLSNGHVPPIPAQQDDEMFQETLEANTIMGFTKEEQTSLRVVSS
VLCQGNIIVFKKERINTQASHMPDTAAQIKVCHLMGNTIVTVDFTRSILTFRIRKVGRDVVQHAGTKE
QADFAIKALAWAKTERLFRWILTRVWVALDKTRPQGASFLOILDIAIGFEIFINSFEQLCINH
THEKLQQLFHNTHTFILEQEKEYQEEGIEWNFIDFGQLDLQFCIELIERFTFPF0VLLDEECNF
PKATDTISFVERKLIGEQOHNAKTFQKSEQLKMDNTEFCILHYAGKVTTHASAWLTHNMDPLHNRVT
SLNQSSDKFVAEMLWDVDRIVGLDQMARMTESSLPSASKTKEGHFRTVGQLYHEQLYTKLHMTT
LHNINHPPVRCIIIPHMEEKRAGKLDQAEVLVEQLRCNGVLEGIRICRQGFPNRIVFQKFRQHRYEI
LAANAIPKGPFMDQFQACILMIKALELDPNLYRIGQSKIFTRTGVLANLEEEERDLKITDVIIAF
QACRGYLLARKAFARROQQLGSCDFSEEQTAEFREAFQLFDTGDKILYSQCGDVMBALGQN
PTNAEVHGCVLGHFPRSDDEHNLRTLKFQFLPNSQQTIAKHDQGCEDYVEGLRVFDMEGHGTVM
GAEIRHVLVTLGEKVNTEEEVEQLVAGHEDSNGCINHYKELVBMVLSG
>>>
```

7. Write a program that parses a genbank sequence file (use file handling) and print the annotation in one file and raw sequence in another file.

Algorithm:

Step 1: import re

Step 2: open your .gb file

Step 3: readfile

Step 4: create empty seq set

Step 5: create empty string result

Step 6: search for the annotation and sequence pattern within the file

Step 7: if found :

Step 8: create a new file : write annotated seq

Step 9: Create another file : write raw sequence



```
demohelpy - C:\Users\Prasad\Desktop\demohelpy (3.9.7)
File Edit Format Run Options Window Help
import re
f = open("insulin.gb", "r+")
data = f.readlines()
#print (data)
seq=[]
annot=[]
result = ""
for i in data:
    if re.match("//", i):
        break
    elif (re.match("^\n\nID [A-Z][a-z]*;[1-9]", i)):
        result = re.findall("//\n\nID [A-Z][a-z]*;[1-9]", i)
        seq.append(result)
    else:
        next
#print (seq)
flatseq = [item for sublist in seq for item in sublist]
#print (flatseq)
rawseq = "\n".join(flatseq)
#print (rawseq)
rawseq = re.sub("//", "", rawseq)
print (rawseq)

alldata = "\n".join(data)
#print (alldata)
myannot = alldata.split("//\n\nID")[1]
#print (myannot)
#print (type(myannot))
print ("My annotation is >>>", myannot[0])
```

8. Write a program that parses helix or sheet regions from the PDB file and print onto the console

Algorithm:

Step 1: import re

Step 2: open your .pdb file

Step 3: readfile

Step 4: create empty seq set

Step 5: create empty string result

Step 6: search for the annotation and sequence pattern within the file

Step 7: if found :

Step 8: print("x")

CODE:-

```
# demohelix.py -C:\Users\Prasad\Desktop\demohelix (3.9.7)
File Edit Format Run Options Window Help
import re
f = open("insulin.pdb", "r")
data = f.readlines()
# print (data)
seq = []
for i in data:
    if re.match("TER", i):
        break
    elif (re.match("HELIX[\n\r\f\v\b\v\f]+", i)):
        result = re.findall("HELIX[\n\r\f\v\b\v\f]+", i)
        seq.append(result)
    elif (re.match("SHEET[\n\r\f\v\b\v\f]+", i)):
        result = re.findall("SHEET[\n\r\f\v\b\v\f]+", i)
        seq.append(result)
    else:
        break
#print (seq)
flatseq = [item for sublist in seq for item in sublist]
#print (flatseq)
for s in flatseq:
    print(s)
```

Python for Bioinformatics

Exercise 10

X: Title – Programs demonstrating Object Oriented Programming in Python

1. Create a class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle. Show its implementation on IDLE

Algorithm:

Step 1: Create a class and using a constructor initialize values of that class.

Step 2: Create a method called as area and return the area of the class.

Step 3: Create an object for the class.

Step 4: Using the object, call the method area() with the parameters as the length and breadth

Step 5: Print the area

The screenshot shows the Python IDLE interface. The top window is the code editor with the following content:

```
idle.py - C:\Users\Prasad\Desktop\demo2.py (1.0)
File Edit Format Help Options Windows Help
class Rectangle():
    def __init__(self, l, w):
        self.length = l
        self.width = w
    def rectangle_area(self):
        return self.length * self.width
newRectangle = Rectangle(12, 10)
print(newRectangle.rectangle_area())
```

The bottom window is the interactive shell with the following content:

```
IDLE Shell 1.9.5
File Edit Shell Debug Options Windows Help
Python 3.9.5 (tags/v3.9.5:ad82fcd, May  3 2021, 17:27:52) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> RESTART: C:\Users\Prasad\Desktop\demo2.py
>>> 120
>>> 
```

The taskbar at the bottom shows several open application icons, including a browser, file explorer, and command prompt.

2. Create a class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle. Implement it in another python script that takes input from the user and also asks the choice of calculation to the user.

Algorithm:

Step 1: Create a class and using a constructor initialize values of that class.

Step 2: Create a method called as area and return the area of the class.

Step 3: Create a method called perimeter and return the perimeter of the class

Step 4: Create an object for the class.

Step 5: Using the object, call the method area() with the parameters as the length and breadth

Step 6: Print the area and perimeter

Code:-

```
from circle_class import circle
circ=circle(input("enter your radius:- "))
print("The choices of calculations are:-")
print("1.Area of a circle")
print("2.Perimeter of a circle")
choice=int(input("Select your option :- "))
if choice==1:
    print(circ.area())
if choice==2:
    print((circ.perimeter()))
```

Output:-

```
enter your radius:- 2
The choices of calculations are:-
1.Area of a circle
2.Perimeter of a circle
Select your option :- 1
The area of circle is 12
>>>
enter your radius:- 2
The choices of calculations are:-
1.Area of a circle
2.Perimeter of a circle
Select your option :- 2
The perimeter of circle is 12
>>> |
```

3. Create a class named studentinfo that has information of students roll no and name and has a method with welcome message for each student. Create a sub class named studentsubject that implements class studentinfo and has information about rollno and marks of four subjects. It has a method that calculates percentage of each student and displays the grade. (Single Inheritance)

Algorithm:

- Step 1: Create a class studentinfo and using a constructor initialize values of that class.
Step 2: Create a method called as `_init_` and return the `_init_`
Step 3: Create an object for the class.
Step 4: Create a method called as msg and return the msg
Step 5: Create an object for the class.
Step 6: Create a class student subject and using a constructor initialize values of that class.
Step 7: Create a method called as init and return the `_init_`
Step 8: Create an object for the class.
Step 9: Create a method called as percentage and return the percentage
Step 10: Create an object for the class.
Step 11: Using the object, call the method `_init_()` with the parameters as name, roll_no,marks,percentage
Step 12: Print (result)

```
from studentQ3 import studentinfo
from studentQ3 import studentsubject
stud1=studentsubject("John Lennon",1,92,82,72,62)
stud2=studentsubject("Ringo star",2,78,67,56,76)
stud3=studentsubject("Paul McCartney",3,87,67,98,67)
stud4=studentsubject("George harrison",4,65,87,98,89)

print(stud1.welcome())
print(stud1.percent())
print(stud1.grade())
print('\n')
print(stud2.welcome())
print(stud2.percent())
print(stud2.grade())
print('\n')
print(stud3.welcome())
print(stud3.percent())
print(stud3.grade())
print('\n')
print(stud4.welcome())
print(stud4.percent())
print(stud4.grade())
```

```

class studentinfo:
    def __init__(self,name,rno):
        self.name=name
        self.rno=rno

    def welcome(self):
        return "Welcome {} ,your Roll.no is {}".format(self.name, self.rno)

class studentsubject(studentinfo):
    def __init__(self,name,rno,sub1,sub2,sub3,sub4):
        studentinfo.__init__(self.name,rno)
        self.sub1=sub1
        self.sub2=sub2
        self.sub3=sub3
        self.sub4=sub4

    def percent(self):
        return ("the percentage is {}".format((self.sub1 + self.sub2 + self.sub3 + self.sub4) / 4))

    def grade(self):
        marks=(self.sub1 + self.sub2 + self.sub3 + self.sub4)
        if marks>90:
            return("grade:- A")
        elif marks>80:
            return("grade:- B")
        elif marks>70:
            return("grade:- C")
        elif marks>60:
            return("grade:- D")
        else:
            return("grade:- F")

```

Output:-

```
Welcome John Lennon ,your Roll.no is 1
the percentage is 77.0
grade:- A
```

```
Welcome Ringo star ,your Roll.no is 2
the percentage is 69.25
grade:- B
```

```
Welcome Paul McCartney ,your Roll.no is 3
the percentage is 79.75
grade:- A
```

```
Welcome George harrison ,your Roll.no is 4
the percentage is 84.75
grade:- A
````
```

4. Create a class named studentinfo1 that has information of students roll no and name and has a method with welcome message for each student. Create another class studentinfo2 that has info on rollno and marks of four subjects with a method calculating the aggregate and displaying it. Create a sub class that implements both of these classes and has information on practical marks of 2 subjects along with a method that calculates the percentage and displays the grade for each student. (Multiple Inheritance)

Algorithm:

Step 1: Create a class studentinfo1 and using a constructor initialize values of that class.

Step 2: Create a method called as `_init_` and return the `_init_`

Step 3: Create an object for the class.

Step 4: Create a method called as `msg` and return the `msg`

Step 5: Create an object for the class.

Step 6: Create a class studentinfo2 and using a constructor initialize values of that class.

Step 7: Create a method called as `init` and return the `_init_`

Step 8: Create an object for the class.

Step 9: Create a method called as `percentage` and return the percentage

Step10: Create an object for the class.

Step11: Using the object, call the method `_init_()` with the parameters as name, roll\_no,marks,percentage

Step12: Print (result)

```
from studentQ4 import subclss
stud1=subclss("John Lennon",1,92,82,72,62,43,65)
stud2=subclss("Ringo star",2,78,67,56,76,56,65)
stud3=subclss("Paul McCartney",3,87,67,98,67,76,67)
stud4=subclss("George harrison",4,65,87,98,89,87,77)

print(stud1.welcome())
print(stud1.aggregate())
print(stud1.percent())
print(stud1.grade())
print('\n')
print(stud2.welcome())
print(stud2.aggregate())
print(stud2.percent())
print(stud2.grade())
print('\n')
print(stud3.welcome())
print(stud3.aggregate())
print(stud3.percent())
print(stud3.grade())
print('\n')
print(stud4.welcome())
print(stud4.aggregate())
print(stud4.percent())
print(stud4.grade())
```

```

class studentinfo:
 def __init__(self, name, roll):
 self.name=name
 self.roll=roll

 def welcome(self):
 print("Welcome {} , your Roll.no is {} ".format(self.name, self.roll))

class studentinfo2:
 def __init__(self, name, roll, sub1, sub2, sub3, sub4):
 self.name=name
 self.roll=roll
 self.sub1=sub1
 self.sub2=sub2
 self.sub3=sub3
 self.sub4=sub4

 def aggregate(self):
 return ("the aggregate is {} ".format((self.sub1 + self.sub2 + self.sub3 + self.sub4)/4))

 def studentinfo1(self, name, roll, sub1, sub2, sub3, sub4):
 self.name=name
 self.roll=roll
 self.sub1=sub1
 self.sub2=sub2
 self.sub3=sub3
 self.sub4=sub4

 def percent(self):
 print ("The percentage is {} ".format((self.sub1 + self.sub2 + self.sub3 + self.sub4)/4))
 grade = self.roll
 marks=(self.sub1 + self.sub2 + self.sub3 + self.sub4)/4
 if marks >=90:
 return ("grade:-A")
 elif marks>80:
 return ("grade:-B")
 elif marks>70:
 return ("grade:-C")
 else:
 return ("grade:-F")

 def marks(self):
 print("Grade:-C")
 print("Grade:-B")
 print("Grade:-C")
 print("Grade:-D")
 print("Grade:-E")
 print("Grade:-F")

 def welcome(self):
 print("Welcome John Lennon ,your Roll.no is 1")
 print("the aggregate is 308")
 print("the percentage is 69.33333333333333")
 print("grade:-C")

 def welcome(self):
 print("Welcome Ringo star ,your Roll.no is 2")
 print("the aggregate is 277")
 print("the percentage is 66.33333333333333")
 print("grade:-C")

 def welcome(self):
 print("Welcome Paul McCartney ,your Roll.no is 3")
 print("the aggregate is 319")
 print("the percentage is 77.0")
 print("grade:-B")

 def welcome(self):
 print("Welcome George harrison ,your Roll.no is 4")
 print("the aggregate is 339")
 print("the percentage is 83.83333333333333")
 print("grade:-B")

```

5. Create a class named “Card” storing information on Deck of cards – suite and their values.suite = ['Hearts','Diamonds','Clubs','Spades'] value = ['A','2','3','4','5','6','7','8','9','10','J','Q','K']. Create another class Deck that is implementing the class Card and having a method that displays the Suite and Deck in systematic order for all four suites. Also stores the values in a new list – mycardset.(Data should appear as – “A of Hearts, 2 of Hearts.....A of Diamond.....”). Create another class shufflecards that implements the Deck class. It will have method that checks whether mycardset is full (no of cards is full in the set), if yes then it shuffles and return the shuffled set if no then returns an error. Once the set is shuffled write another method that will pop up a random card every time the method executes (Multilevel Inheritance)

Algorithm:

- Step 1: Create a class card and using a constructor initialize values of that class.
- Step 2: Create a class shufflecard and using a constructor initialize values of that class.
- Step 3: Create a method called as `_init_` and return the `_init_`
- Step 4: Create an object for the class.
- Step 5: Create a method called as `shuffle` and return `shuffle`
- Step 6: Create an object for the class.
- Step 7: Create a method called as `pop` and return `pop`
- Step 8: Create an object for the class.
- Step 9: Using the object, call the method `_init_()` with the parameters as suites and values
- Step 10: Print (result)

```

demodehp - C:\Users\Priyadarshini\Desktop\demodehp (3.9.7)
File Edit Format Run Options Window Help
File Edit Format Run Options Window Help
from random import shuffle

class Card:
 def __init__(self):
 self.suite = ['Hearts','Diamonds','Clubs','Spades']
 self.value = ['A','2','3','4','5','6','7','8','9','10','J','Q','K']
 def __str__(self):
 return ("{} {} in class Card".format(self.value[-1], self.suite[-1]))

class Deck(Card):
 def __init__(self):
 Card.__init__(self)
 self.mycardset = []
 for s in self.suite:
 for v in self.value:
 str = v + " of " + s
 self.mycardset.append(str)

 def info(self):
 return len(self.mycardset)

class ShuffleCard(Deck):
 def __init__(self):
 Deck.__init__(self)

 def shuffle(self):
 if len(self.mycardset)<52:
 raise ValueError("Only full decks can be shuffled")
 else:
 shuffle(self.mycardset)
 return self.mycardset
 return ("cards shuffled")

 def popcard(self):
 if len(self.mycardset) == 0:
 raise ValueError("All cards have been popped")
 else:
 return self.mycardset.pop()

```

```
['A of Hearts', '2 of Hearts', '3 of Hearts', '4 of Hearts', '5 of Hearts', '6 of Hearts', '7 of Hearts', '8 of Hearts', '9 of Hearts', '10 of Hearts', 'J of Hearts', 'Q of Hearts', 'K of Hearts', 'A of Diamonds', '2 of Diamonds', '3 of Diamonds', '4 of Diamonds', '5 of Diamonds', '6 of Diamonds', '7 of Diamonds', '8 of Diamonds', '9 of Diamonds', '10 of Diamonds', 'J of Diamonds', 'Q of Diamonds', 'K of Diamonds', 'A of Clubs', '2 of Clubs', '3 of Clubs', '4 of Clubs', '5 of Clubs', '6 of Clubs', '7 of Clubs', '8 of Clubs', '9 of Clubs', '10 of Clubs', 'J of Clubs', 'Q of Clubs', 'K of Clubs', 'A of Spades', '2 of Spades', '3 of Spades', '4 of Spades', '5 of Spades', '6 of Spades', '7 of Spades', '8 of Spades', '9 of Spades', '10 of Spades', 'J of Spades', 'Q of Spades', 'K of Spades']
```

```
Card popped is K of Spades
```

```
Card popped is Q of Spades
```

```
Card popped is 2 of Clubs
```

```
>>> |
```

6. Create a class named shapes constructed by radius of circle and length and width of rectangle and separate methods for calculating the area of circle and rectangle. Create subclass1 that implements shapes and is constructed by length and breadth information of ellipse. This class has method to calculate perimeter of circle and perimeter of ellipse.  
 . Create subclass2 that implements shapes and constructed by bases, height and sides of trapezoid. It has methods to calculate perimeter of rectangle and trapezoid.  
 (Hierarchical Inheritance)

Algorithm:

- Step 1: Create a class shapes and using a constructor initialize values of that class.
- Step 2: Create a method called as `_init_` and return the `_init_`
- Step 3: Create an object for the class.
- Step 4: Create a method called as `area_rect` and return `area_rect`
- Step 5: Create an object for the class.
- Step 6: Create a method called as `area_circ` and return `area_circ`
- Step 7: Create an object for the class.
- Step 8: Create a class subclass1 and using a constructor initialize the values of that class
- Step 9: Create a method called as `_init_` and return the `_init_`
- Step 10: Create an object for the class.
- Step 11: Create a method called as `perimeter_circle` and return `perimeter_circle`
- Step 12: Create an object for the class.
- Step 13: Create a method called as `perimeter_ellipse` and return `perimeter_ellipse`
- Step 14: Create an object for the class.
- Step 15: Create a class subclass2 and using a constructor initialize the values of that class
- Step 16: Create a method called as `_init_` and return the `_init_`
- Step 17: Create an object for the class.
- Step 18: Create a method called as `perimeter_trap` and return `perimeter_trap`
- Step 19: Create an object for the class.
- Step 20: Create a method called as `perimeter_rect` and return `perimeter_rect`
- Step 21: Create an object for the class.
- Step 22: from shape import shape
- Step 23: from shape import subclass1
- Step 24: from shape import subclass2
- Step 25: Using the object, call the method `_init_()` with the parameters as suites and values
- Step 26: Print (result)

```

class shapes:
 def __init__(self,radius,length,width):
 self.radius=radius
 self.length=length
 self.width=width
 def area_rect(self):
 return "The area of rectangle is "+str(self.length*self.width)
 def area_circ(self):
 return "The area of circle is "+str((float(3.14)*float(self.radius)**2))
 class subclass1(shapes):
 def __init__(self,radius,length,width,breadth):
 self.radius=radius
 self.length=length
 self.breadth=breadth
 shapes.__init__(self,radius,length,width)
 def perimeter_circle(self):
 return "The perimeter of circle is "+str((float(2)*float(3.14)*float(self.radius)))
 def perimeter_ellipse(self):
 return "The perimeter of ellipse is "+str((float(2)*float(3.14)*float((float(self.radius)+float(self.breadth))/float(2))))+str((float(self.radius)*float(self.breadth))/float(2))**2)
 class subclass2(shapes):
 def __init__(self,radius,length,width,sides1,sides2,bases1):
 self.radius=radius
 self.length=length
 self.breadth=breadth
 self.sides1=sides1
 self.sides2=sides2
 self.bases1=bases1
 shapes.__init__(self,radius,length,width)
 def perimeter_rect(self):
 return "The perimeter of rectangle is "+str((float(2)*(self.length+self.breadth)))
 def perimeter_trap(self):
 return "The perimeter of trapezoid is "+str((self.sides1+ self.sides2 +self.bases1 + self.bases2))

```

```
from shapesclass import shapes
from shapesclass import subclass1
from shapesclass import subclass2

shape1=shapes(2,3,4)
print(shape1.area_rect())
print(shape1.area_circ())
print('\n')
shape2=subclass1(2,2,3,6,7)
print(shape2.perimeter_circle())
print(shape2.perimeter_ellipse())
print('\n')
shape3=subclass2(2,3,4,5,6,7,8)
print(shape3.perimeter_rect())
print(shape3.perimeter_trap())
```

```
The area of rectangle is 12
The area of circle is 12.56
```

```
The perimeter of circle is 12.56
The perimeter of ellipse is 133
```

```
The perimeter of circle is 24.0
The perimeter of ellipse is 26
>>> |
```

## Python for Bioinformatics

### Exercise 11

#### Title: Programs demonstrating usage of CGI Python

1. Create a web page for Students portal seeking information about name, roll no, sex, favorite book, participation in extracurricular activities and any other information (include all form elements). Ask students to enter marks of 4 theory and 2 practical subjects, incorporate the grade calculator program and display the grade of respective students along with other information.

Algorithm:

Step 1: Start and initialize Xammp and start Apache.

Step 2: Open the html file for the student's portal which asks for the input in the browser.

Step 3: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 4: In the python file set the directory of the python at the top of the code.

Step 5: Save both the html and python file in the htdocs folder in the directory for xammp.

Step 6: Give the input for the students information in the html file and submit.

The image shows two screenshots. The top screenshot is of the XAMPP Control Panel v3.3.0. It lists services: Apache (running on port 80), MySQL (running on port 3306), and others like Ftpd, Memcached, and Tomcat. The Apache service has a 'Stop' button highlighted. The bottom screenshot is a Windows desktop with a command prompt window open. The code in the command prompt is as follows:

```
pract1 = int (form.getvalue('pract1'))
pract2 = 0
grade = percentage_grade.letterGrade(pract1, pract2)
print ("Content-type:text/html\n\n")
print ("<html>")
print ("<head>")
print ("<title>Grade</title>")
```

```
stu_info.html - Notepad
File Edit Format View Help
<html>
<head>
<title>Student Information</title>
<link rel="stylesheet" href="stu_info.css">
</head>
<body>
<form action = "http://localhost/cgi/stu_info.py" method = "post" target = "_blank">
<h2>STUDENT DETAILS:</h2>
Name: <input type = "text" name = "name">

Roll Number: <input type = "text" name = "roll_no">

Sex: <input type = "radio" name = "sex" value = "Male">Male
<input type = "radio" name = "sex" value = "Female">Female
<input type = "radio" name = "sex" value = "Other">Other

Favourite Book:<select name = "dropdown">
<option>100 years of solitude</option>
<option>Catch 22</option>
<option>Notes from the underground</option>
<option>Kafka on the shore</option>
<option>Norwegian Wood</option>
</select>

What are your hobbies?

<input type = "checkbox" name = "hobbies" value = "Sports"/>
<label>Sports</label>

<input type = "checkbox" name = "hobbies" value = "Reading"/>
<label>Reading</label>

<input type = "checkbox" name = "hobbies" value = "Writing"/>
<label>Writing</label>


```

```
stu_info.html - Notepad
File Edit Format View Help
<input type = "checkbox" name = "hobbies" value = "Writing"/>
<label>Writing</label>

<input type = "checkbox" name = "hobbies" value = "Cooking"/>
<label>Cooking</label>

Tell us about your extra-curricular activities:

<textarea rows = "6" cols = "75" name = "description"></textarea>
<h2>ACADEMIC DETAILS:</h2>
<h3>Please enter your marks one by one in the given boxes:</h3>
Subject 101: <input type = "number" name = "sub1"/>

Subject 102: <input type = "number" name = "sub2"/>

Subject 103: <input type = "number" name = "sub3"/>

Subject 104: <input type = "number" name = "sub4"/>

<h2>ACADEMIC DETAILS:</h2>
<h3>Please enter your marks one by one in the given boxes:</h3>
Subject 101: <input type = "number" name = "sub1"/>

Subject 102: <input type = "number" name = "sub2"/>

Subject 103: <input type = "number" name = "sub3"/>

Subject 104: <input type = "number" name = "sub4"/>

Practical 11: <input type = "number" name = "pract1"/>

Practical 12: <input type = "number" name = "pract2"/>

<input type = "submit" value = "Submit"/>
<input type = "reset" value = "Reset"/>
</form>
</body>
</html>
```

```
[A]stu_info.py - Chompp\Hdocs\stu_info.py (3.3.7)
File Edit Format Run Options Window Help
#!C:/Users/Frasedz/AppData/Local/Programs/Python/Python38/python.exe
import cgi, cgitb
form = cgi.FieldStorage()

name = form.getvalue('name')
roll_no = form.getvalue('roll_no')
dropdown = form.getvalue('dropdown')
description = form.getvalue('description')
hobbies = form.getvalue('hobbies')

try:
 sub1 = int(form.getvalue('sub1'))
except TypeError:
 sub1 = 0

try:
 sub2 = int(form.getvalue('sub2'))
except TypeError:
 sub2 = 0

try:
 sub3 = int(form.getvalue('sub3'))
except TypeError:
 sub3 = 0

try:
 sub4 = int(form.getvalue('sub4'))
except TypeError:
 sub4 = 0

try:
 pract1 = int(form.getvalue('pract1'))
except TypeError:
 pract1 = 0

try:
 pract2 = int(form.getvalue('pract2'))
except TypeError:
 pract2 = 0

 import percentage_grade
grade = percentage_grade.letsgo(sub1, sub2, sub3, sub4, pract1, pract2)
print ("Content-type:text/html\n\n")
print ("<html>")
print ("<head>")
print ("<title>Details</title>")

In: 4 Col: 20
2256 21-10-2021
```

```
[A]stu_info.py - Chompp\Hdocs\stu_info.py (3.3.7)
File Edit Format Run Options Window Help
 sub1 = int (form. getvalue("sub1"))
except TypeError:
 sub1 = 0

try:
 sub2 = int (form. getvalue("sub2"))
except TypeError:
 sub2 = 0

try:
 sub3 = int (form. getvalue("sub3"))
except TypeError:
 sub3 = 0

try:
 sub4 = int (form. getvalue("sub4"))
except TypeError:
 sub4 = 0

try:
 pract1 = int (form.getvalue("pract1"))
except TypeError:
 pract1 = 0

try:
 pract2 = int (form.getvalue("pract2"))
except TypeError:
 pract2 = 0

 import percentage_grade
grade = percentage_grade.letsgo(sub1, sub2, sub3, sub4, pract1, pract2)
print ("Content-type:text/html\n\n")
print ("<html>")
print ("<head>")
print ("<title>Details</title>")
print ("</head>")
print ("<body>")
print ("<h1>Hello sir/n>" + name)
print ("<h2>Your roll number is " + roll_no + "" % roll_no)
print ("<h2>You are a <i>" + dropdown + "</i></h2> % dropdown")
print ("<h2>Your favourite book is " + book + "" % dropdown)
print ("<h2>You mentioned your hobbies are " + hobbies + "" % hobbies)
print ("<h2>Your co-curricular activities are " + activities + "" % description)
print ("<h2>The marks are awarded grade " + grade + "" % grade)
print ("</body>")
print ("</html>")

In: 4 Col: 20
2256 21-10-2021
```

```

percentage_grade.py - C:\xampp\htdocs\percentage_grade.py (3.9.7)
File Edit Format Run Options Window Help
def percentage_grade(sub1, sub2, sub3, sub4, practical1, practical2):
 sum = sub1+sub2+sub3+sub4+practical1+practical2
 perc = sum/6
 if perc > 90:
 grade = "A"
 elif perc > 70:
 grade = "B"
 elif perc > 60:
 grade = "C"
 elif perc > 50:
 grade = "D"
 else:
 grade = "Failed"
 return grade

```

## Output:

STUDENT DETAILS: Name: Gurjan Deshpande

Roll Number: BID 1900

Sex: Female  Male  Other

Favourite Book: Kafka on the shore

What are your hobbies?

|          |         |
|----------|---------|
| Cricket  | Sports  |
| Blogging | Writing |
| Painting | Cooking |

Tell us about your extra-curricular activities:  
I love swimming and I enjoy travelling.

### ACADEMIC DETAILS:

Please enter your marks one by one in the given boxes:

|                 |
|-----------------|
| Subject 101: 67 |
| Subject 102: 98 |
| Subject 104: 88 |

### ACADEMIC DETAILS:



WhatsApp

Download file - your conversion

Info (332) - parulvedt1900@yhs

Student Information

Favourite Book:

What are your hobbies?

|          |         |
|----------|---------|
| Cricket  | Sports  |
| Blogs    | Writing |
| Par Bhop | Cooking |

Tell us about your extra-curricular activities:  
I love swimming and I enjoy travelling.

**ACADEMIC DETAILS:**

Please enter your marks one by one in the given boxes:

|                 |
|-----------------|
| Subject 101: 67 |
| Subject 102: 98 |
| Subject 104: 80 |

**ACADEMIC DETAILS:**

Please enter your marks one by one in the given boxes:

|                  |
|------------------|
| Subject 101: 12  |
| Subject 102: 50  |
| Subject 104: 87  |
| Practical 11: 97 |

Type here to search

22:50 21-10-2021

WhatsApp

Download file

Info (332)

Shapes

Details

404 Not Found

Student Information

STUDENT DETAILS: Name:

Roll Number:

Sex:  Male  Female  Other

Favourite Book:

What are your hobbies?

Sports  
 Reading  
 Writing  
 Cooking

Tell us about your extra-curricular activities:

Type here to search

18:00 22-10-2021

2. Create a web page that calculates areas of 2D and 3D shapes. Give choices to the users and perform calculation as per user's selection. Make use of form elements and images.

Algorithm:

Step 1: Start and initialize Xammp and start Apache.

Step 2: Open the html file for the shapes which asks for the input in the browser.

Step 3: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 4: In the python file set the directory of the python at the top of the code.

Step 5: Save both the html and python file in the htdocs folder in the directory for xammp along with the module files.

Step 6: Give the input for the in the html file and submit.

3. Create a fancy & creative web page for BMI Calculator giving various choices and options to users.

Algorithm:

Step 1: Start and initialize Xammp and start Apache.

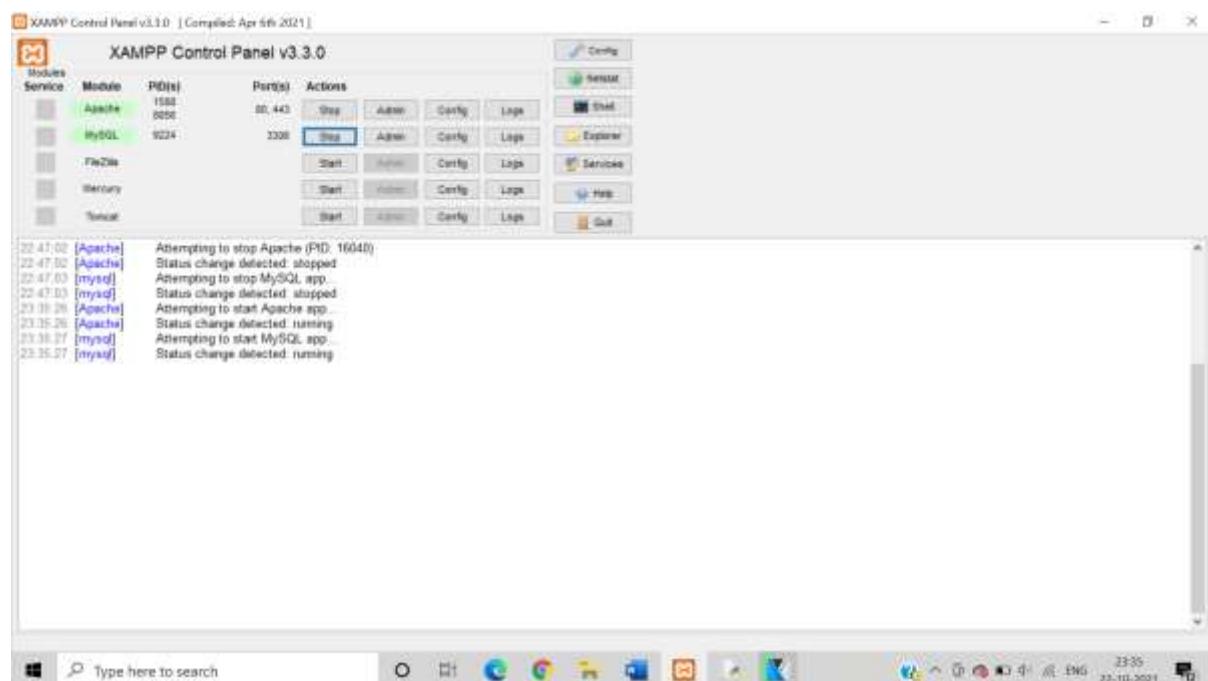
Step 2: Open the html file for the BMI which asks for the input in the browser.

Step 3: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 4: In the python file set the directory of the python at the top of the code.

Step 5: Save both the html and python file in the htdocs folder in the directory for xammp.

Step 6: Give the input for the BMI in the html file and submit.



```
#> BMI.py - ChallengingHttpd/BMI.py (3.9.7)
File Edit Format Run Options Window Help
#C:\Users\Praveen\AppData\Local\Programs\Python\Python39\python.exe
import cgi, cgitb
form = cgi.FieldStorage()

try:
 age = form.getvalue('age')
except TypeError:
 age = "Not selected. Try again."
try:
 gender = form.getvalue('gender')
except TypeError:
 gender = "Not selected. Try again."
try:
 weight = float(form.getvalue('weight'))
except TypeError:
 weight = "please enter a value"
try:
 height = float(form.getvalue('height'))
except TypeError:
 height = 0

import real_formula as rf
bmi = rf.bmi_calculator(weight, height)

print ("Content-type:text/html\n")
print ("")
print ("")
print ("Bmi calculator")
print ("")
print ("")
print ('<link rel="stylesheet" href="style.css">')
print ('<head>')
print ('<meta>')
print ('<hr><hr><hr><hr><hr><hr><hr>')
print ('<table class = "B3">')
print ('<tr style="color:blue;font-family:arial black;font-size:20px;">')
if bmi < 16:
 print ('<td>Your BMI is <u>{0}</u>. You are suffering from severe thinness.</td>'.format(bmi))
elif bmi < 17:
 print ('<td>Your BMI is <u>{0}</u>. You are suffering from moderate thinness.</td>'.format(bmi))
elif bmi < 18.5:
 print ('<td>Your BMI is <u>{0}</u>. You are suffering from mild thinness.</td>'.format(bmi))

In 13 Col 17
```

```
C:\Users\Hdico\BMISite - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

<html>
<head>
<title>BMI Calculator</title>
</head>
<body>
<header>
<link rel="stylesheet" href="bmisite.css">

<hr><hr>
<h1>Calculator</h1>
<div class="d2">
<form action = "http://localhost/cgi/BMI.py" method = "get" target = "_blank">
<p style="color:black; font-family:Arial;font-size:10px; align: left">
BMI Calculator

<input type = "number" name = "age">

<input type = "radio" name = "gender" value = "Male">Male

<input type = "radio" name = "gender" value = "Female">Female

<input type = "radio" name = "gender" value = "Other">Other

Weight(in kgs):

<input type = "number" name = "weight">

Height(in cms):

<input type = "number" name = "height">

<input type = "submit" value = "Submit">
</p>
</form>
</div>

</header>
</body>
</html>
```

The screenshot shows a Microsoft Windows desktop environment. At the top, there is a taskbar with several icons and windows. One window titled "BMI Calculator" is open in the center. The calculator interface includes fields for Age (20), Gender (Female selected), weight (52), and Height (160). A "Submit" button is at the bottom. The taskbar also shows icons for WhatsApp, File Explorer, Task View, Start, and other system utilities.

**BMI Calculator**

**Age:**

**Gender:**  
Male   
Female   
Other

**weight(in kgs):**

**Height(in cms):**

**Submit**

## **Python for Bioinformatics**

### **Exercise 12**

#### **Title – Programs demonstrating usage of DBI Python**

1. [Create](#) a Database named MyDB, create a table having student's information with fields roll no, name, email id and marks of 4 subjects. Insert about 10 rows in the table. Calculate percentage of each student and create another table having roll no and percentage of all students

Algorithm:

Step 1: Start and initialize Xammp and start Apache and mysql.

Step 2: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 3: Import the Mysql connector and enter the host, username and password

Step 4: Using CREATE DATABASE and SHOW DATABASE create and show the database.

Step 5: Run a python file which asks for input and inserts the information into the database.

Step 6: Save the python files in the htdocs folder in the directory for xampp.

Step 7: Go to the phpmyadmin page and see the info and output.

2. Create a Web page for a Departmental Store that takes input of list of items from the client. Link it to a python script that calculates the cost of items and stores the information of client name, order id and price in a table.

Algorithm:

Step 1: Start and initialize Xammp and start Apache and mysql.

Step 2: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 3: Import the Mysql connector and enter the host, username and password

Step 4: Using CREATE DATABASE and SHOW DATABASE create and show the database.

Step 5: Run a python file which asks for input and inserts the information into the database.

Step 6: Save the python files in the htdocs folder in the directory for xampp.

Step 7: Go to the phpmyadmin page and see the info and output.

3. Create a Web page that takes an input of a protein sequence from the user. Link it to a python script that parses the sequence and the accession id separately, also calculates the molecular weight of the sequence. Store the information of accession id, sequence, sequence length and molecular weight in a table

Algorithm:

Step 1: Start and initialize Xammp and start Apache and mysql.

Step 2: Give the directory of the python file for the form action in the html file. Set the method to post.

Step 3: Import the Mysql connector and enter the host, username and password

Step 4: Using CREATE DATABASE and SHOW DATABASE create and show the database.

Step 5: Run a python file which asks for input and inserts the information into the database.

Step 6: Save the python files in the htdocs folder in the directory for xampp.

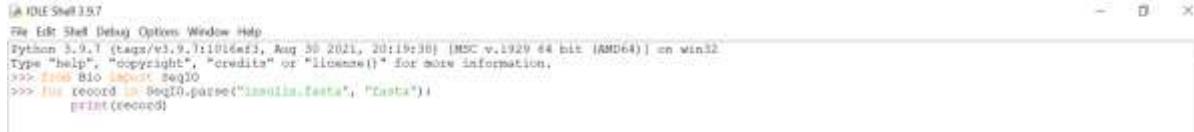
Step 7: Go to the phpmyadmin page and see the info and output

## Python for Bioinformatics

### Exercise 13

### XIII: Title – Programs demonstrating usage of BIOPYTHON Modules

1. Write any three programs demonstrating the usage of Biopython Modules



```
idle Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:f9087da, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from Bio import SeqIO
>>> ins record = SeqIO.parse("insulin.fasta", "fasta")
>>> print(record)
```

- Concatenating or adding sequences.

**Algorithm:-** You may often have many sequences to add together, Like Python strings, Biopython Seq also has a .join method.



```
idle Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:f9087da, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from Bio.Seq import Seq
>>> protein_seq = Seq("EVIRHET")
>>> dna_seq = Seq("AGCT")
>>> protein_seq + dna_seq
Seq('EVIRHAKACGT')
```

- Comparing Seq objects.

**Algorithm:-** Sequence comparison is actually a very complicated topic, and there is no easy way to decide if two sequences are equal. The basic problem is the meaning of the letters in a sequence are context dependent - the letter “A” could be part of a DNA, RNA or protein sequence. Biopython can track the molecule type, so comparing two Seq objects could mean considering this too.

```
[A] IDLE Shell 3.9.1+
File Edit Shell Debug Option Window Help
Python 3.9.1 (tags/v3.9.1:1e5d61f, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from Bio.Seq import Seq
>>> protein_seq = Seq("EVPRRAKQTT")
>>> dna_seq = Seq("ACGT")
>>> protein_seq + dna_seq
Seq('EVPRRAKQTTACGT')
>>> Bio.Seq import Seq
>>> seq1 = Seq("ACGT")
>>> "ACGT" == seq1
True
>>> "ACGT" == seq1
False
>>>
```

### C. Slicing a sequence.

Algorithm:- 'Seq' objects follow the usual indexing conventions for Python strings, with the first element of the sequence numbered 0. Also like a Python string, you can do slices with a start, stop and *stride* (the step size, which defaults to one).

```
[A] IDLE Shell 3.9.1+
File Edit Shell Debug Option Window Help
Python 3.9.1 (tags/v3.9.1:1e5d61f, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from Bio.Seq import Seq
>>> my_seq = Seq("GTCATATGGCCATATGGCCATATGGCC")
>>> my_seq[4:12]
Seq('GGCCATATGG')
>>>
```

