

INDEX

9/9/2023

Python Practical File

Full Time Diploma in Computer Engineering
5th Semester

Abhishek Roka
10621019

INDEX

Practical no.	Practical Name	Remarks
1	Calculate the multiplication and sum of two numbers	
2	Print the sum of the current number and the previous number	
3	Print characters from a string that are present at an even index number	
4	Remove first n characters from a string	
5	Check if the first and last number of a list is the same	
6	Display numbers divisible by 5 from a list	
7	Return the count of a given substring from a string	
8	Print the following pattern 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5	
9	Check Palindrome Number	

10	Create a new list from a two list using the following condition – new list should contain odd numbers from the first list and even numbers from the second list									
11	Write a Program to extract each digit from an integer in the reverse order.									
12	Calculate income tax for the given income by adhering to the below rules <table><tr><th>Taxable Income</th><th>Rate (in %)</th></tr><tr><td>First \$10,000</td><td>0</td></tr><tr><td>Next \$10,000</td><td>10</td></tr><tr><td>The remaining</td><td>20</td></tr></table>	Taxable Income	Rate (in %)	First \$10,000	0	Next \$10,000	10	The remaining	20	
Taxable Income	Rate (in %)									
First \$10,000	0									
Next \$10,000	10									
The remaining	20									
13	Print multiplication table from 1 to 10									
14	Print downward Half-Pyramid Pattern with Star (asterisk) * * * * * * * * * * * * * * *									
15	Write a function called exponent(base, exp) that returns an int value of base raises to the power of exp.									
16	Write a function called exponent(base, exp) that returns an int value of base raises to the power of exp.									

17	<p>Write a program to create function func1() to accept a variable length of arguments and print their value.</p> <p>Note: Create a function in such a way that we can pass any number of arguments to this function, and the function should process them and display each argument's value.</p>	
18	Write a program to create function calculation() such that it can accept two variables and calculate addition and subtraction. Also, it must return both addition and subtraction in a single return call	
19	<p>Write a program to create a function show_employee() using the following conditions.</p> <ul style="list-style-type: none"> • It should accept the employee's name and salary and display both. • If the salary is missing in the function call then assign default value 9000 to salary 	
20	<p>Create an inner function to calculate the addition in the following way</p> <ul style="list-style-type: none"> • Create an outer function that will accept two parameters, a and b • Create an inner function inside an outer function that will calculate the addition of a and b • At last, an outer function will add 5 into addition and return it 	
21	Write a program to create a recursive function to calculate the sum of numbers from 0 to 10.	
22	<p>Assign a different name to function and call it through the new name</p> <p>Below is the function display_student(name, age). Assign a new name show_tudent(name, age) to it and call it using the new name.</p>	

23	Generate a Python list of all the even numbers between 4 to 30	
24	Find the largest item from a given list	
25	Write a program that defines a function <code>count_lower_upper()</code> that accepts a string and calculates the number of uppercase and lowercase alphabets in it. It should return these values as a dictionary. Call this function for some sample strings.	
26	Write a program that defines a function <code>compute()</code> that calculates the value of $n + nn + nnn + nnnn$, where n is digit received by the function. Test the function for digits 4 and 7.	
27	Write a program that defines a function <code>create_array()</code> to create and return a 3D array whose dimensions are passed to the function. Also initialize each element of this array to a value passed to the function.	
28	Write a program that defines a function <code>create_list()</code> to create and return a list which is an intersection of two lists passed to it.	
29	Write a program that defines a function <code>sanitize_list()</code> to remove all duplicate entries from the list that it receives	
30	Write a program to receive three integers from keyboard and get their sum and product calculated through a user-defined function <code>cal_sum_prod()</code> .	
31	Pangram is a sentence that uses every letter of the alphabet. Write a program that checks whether a given string is pangram or not, through a user-defined function <code>ispangram()</code> .	
32	<p>Write a Python program that accepts a hyphen-separated sequence of words as input and calls a function <code>convert()</code> which converts it into a hyphen-separated sequence after sorting them alphabetically.</p> <p>For example, if the input string is here-come-the-dots-followed-by-dashes Then, the output must be: by-come-dashes-dots-followed-here-the</p>	

33	Write a python function to create and return a list containing tuples of the form (x, x2, x3) for all x between 1 and 20 (both included).	
34	Write a program that defines a function ispalindrome() which checks whether a given string is a palindrome or not. Ignore spaces and case mismatch while checking for palindrome.	
35	Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically.	
36	Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically.	
37	Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically.	
38	Write a program that defines two functions called create_sent1() and create_sent2(). Both receive following 3 lists: subjects = ['He', 'She'] verb = ['loves', 'hates'] objects = ['TV Serials', 'Netflix'] Both functions should form sentences by picking elements from the lists and returns them. Use for loops in create_sent1() and list comprehension in create_sent2()	
39	Perform the following operations on a list of names. – Create a list of 5 names – 'Anil', 'Amol', 'Aditya', 'Avi', 'Alka' – Insert a name 'Anuj' before 'Aditya' – Append a name 'Zulu' – Delete 'Avi' from the list – Replace 'Anil' with 'AnilKumar' – Sort all the names in the list – Print reversed sorted list	

40	<p>Perform the following operations on a list of numbers.</p> <ul style="list-style-type: none"> – Create a list of 5 odd numbers – Create a list of 5 even numbers – Combine the two lists – Add prime numbers 11,17,29 at the beginning of the combined list – Report how many elements are present in the list – Replace last 3 numbers in the list with 100 , 200 , 300 – Delete all the numbers in the list – Delete the list 	
41	Write a program to implement a Stack data structure. Stack is a Last In First Out (LIFO) list, in which addition and deletion takes place at the same end.	
42	Write a program to implement a Queue data structure. Queue is a First In First Out (FIFO) list, in which addition takes place at the rear end of the queue and deletion takes place at the front end of the queue.	
43	Write a program to generate and store in a list 20 random numbers in the range 10 to 100. From this list delete all those entries which have value between 20 and 50. Print the remaining list.	
44	Write a program to add two 3x4 matrices.	
45	Pass a tuple to divmod() function and obtain the quotient and the remainder.	
46	<p>Write a Python program to perform the following operations:</p> <ul style="list-style-type: none"> – Pack first 10 multiples of 10 into a tuple – Unpack the tuple into 10 variables, each holding 1 value – Unpack the tuple such that first values gets stored in variable x, last value in y and all values in between into disposable variables _ – Unpack the tuple such that first values gets stored in variable i, last value in j and all values in between into disposable variables _ 	
47	A list contains names of boys and girls as its elements. Boys' names are stored as tuples. Write a Python program to find out number of boys and girls in the list.	
48	A list contains tuples containing roll number, names and age of student. Write a Python Program to gather all the names from this list into another list.	

49	<p>Given the following tuple ('F','l','a','b','b','e','r','g','a','s','t','e','d')</p> <p>Write a Python program to carry out the following operations:</p> <ul style="list-style-type: none"> – Add an ! at the end of the tuple – Convert a tuple to a string – Extract ('b','b') from the tuple – Find out number of occurrences of 'e' in the tuple – Check whether 'r' exists in the tuple – Convert the tuple to a list – Delete characters 'b','b','e','r' from the tuple 	
50	<p>What will be the output of the following program?</p> <pre> a = {10,20,30,40,50,60,70} b = {33,44,51,10,20,50,30,33} print(a b) print(a & b) print(a - b) print(b - a) print(a ^ b) print(a >=b) print(a <= b) </pre>	
51	<p>What will be the output of the following program?</p> <pre> a = {1,2,3,4,5,6,7} b = {1,2,3,4,5,6,7} c = {1,2,3,4,5,6,7} d = {1,2,3,4,5,6,7} e= {3,4,1,0,2,5,8,9} a = e print(a) b &= e print(b) c -=e print(c) d ^= e print(d) </pre>	
52	<p>Write a program to carry out the following operations on the given set s = {10, 2,-3,4,5,88}</p> <ul style="list-style-type: none"> – number of items in set s – maximum element in set s – minimum element in set s – sum of all elements in set s 	

	<ul style="list-style-type: none"> – obtain a new sorted set from s, set s remaining unchanged – report whether 100 is an element of set s – report whether -3 is an element of set s 	
53	<p>What will be the output of the following program?</p> <pre>l = [10,20,30,40,50] t = ('Sundeeep', 25, 79.58) s = 'set theory' s1 = set(l) s2 = set(t) s3 = set(s) print(s1) print(s2) print(s3)</pre>	
54	<p>What will be the output of the following program?</p> <pre>s= {1, 2, 3, 7, 6, 4) s.discard(10) s.remove(10) print(s)</pre>	
55	<p>What will be the output of the following program?</p> <pre>s1= {10, 20, 30, 40, 50}) s2 =(10, 20, 30, 40, 50} print(id(s1), id(s2))</pre>	
56	<p>What will be the output of the following program?</p> <pre>s1={10, 20, 30, 40, 50} s2 = {10, 20, 30, 40, 50} s3 = (*s1, *s2}</pre>	
57	<p>What will be the output of the following program?</p> <pre>s= set('KanLabs') t= s[:-1] print(t)</pre>	
58	<p>What will be the output of the following program?</p> <pre>num = {10, 20, {30, 40}, 50) print(num)</pre>	
59	<p>What will be the output of the following program?</p> <pre>s=('Tiger', 'Lion', 'Jackal') del(s) print(s)</pre>	
60	<p>What will be the output of the following program?</p> <pre>fruits = ('Kiwi, 'Jack Fruit', 'Lichi')</pre>	

	<pre>fruits.clear() print(fruits)</pre>	
61	<p>What will be the output of the following program?</p> <pre>s = {10, 25, 4, 12, 3, 8} s = sorted(s) print(s)</pre>	
62	<p>What will be the output of the following program?</p> <pre>s = {} t = {1,4,5,2,3} print(type(s), type(t))</pre>	
63	<p>A set contains names which begin either with A or with B. write a program to separate out the names into two sets, one containing names beginning with A and another containing names beginning with B.</p>	
64	<p>Create an empty set. Write a program that adds five new names to this set, modifies one existing name and deletes two names existing in it.</p>	
65	<p>Write a program to create a set containing 10 randomly generated numbers in the range 15 to 45. Count how many of these numbers are less than 30. Delete all numbers which are greater than 35.</p>	
66	<p>What will be the output of the following program?</p> <pre>s = {'Mango', 'Banana', 'Guava', 'Kiwi'} s.clear() print(s) del(s) print(s)</pre>	

Practical 1

Calculate the multiplication and sum of two numbers

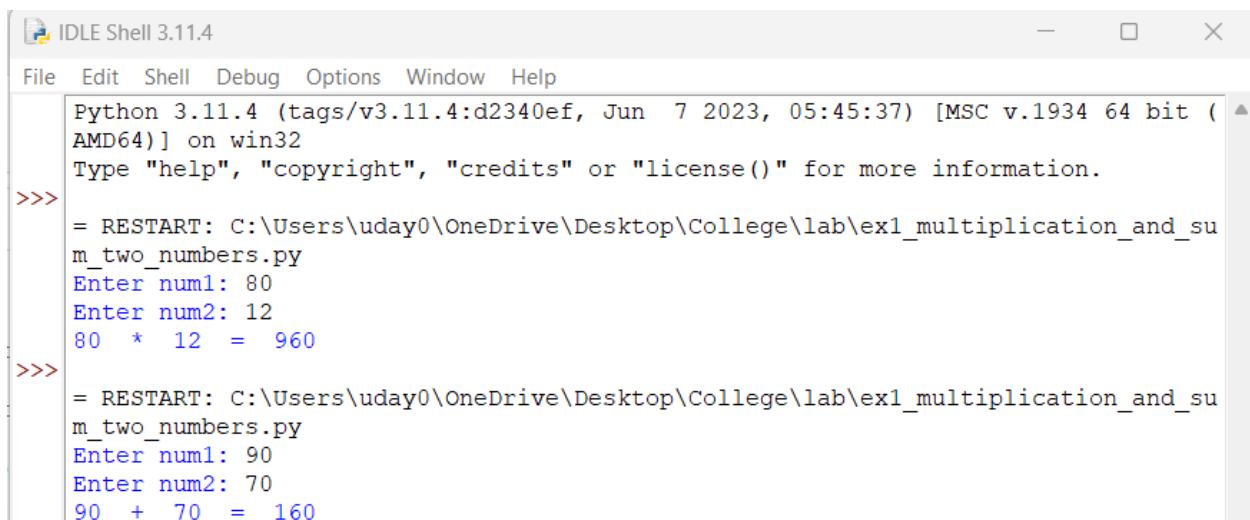
Code

```
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))

mul = num1*num2

if mul <= 1000:
    print(num1, " * ", num2, " = ", mul)
else:
    print(num1, " + ", num2, " = ", num1+num2)
```

Output:



```
IDLE Shell 3.11.4
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex1_multiplication_and_sum_two_numbers.py
Enter num1: 80
Enter num2: 12
80 * 12 = 960
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex1_multiplication_and_sum_two_numbers.py
Enter num1: 90
Enter num2: 70
90 + 70 = 160
```

Practical 2

Print the sum of the current number and the previous number

Code:

```
prev_num = 0
rangelen = range(10)
currentnum=0
for i in rangelen:
    currentnum = prev_num + i
    print("Previous number is ", prev_num, " currentnum is ", currentnum)
    prev_num = currentnum
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex2_print_sum_of_current_
num_and_prev_num.py
Previous number is 0  currentnum is 0
Previous number is 0  currentnum is 1
Previous number is 1  currentnum is 3
Previous number is 3  currentnum is 6
Previous number is 6  currentnum is 10
Previous number is 10  currentnum is 15
Previous number is 15  currentnum is 21
Previous number is 21  currentnum is 28
Previous number is 28  currentnum is 36
Previous number is 36  currentnum is 45
>>>
```

Practical 3

Print characters from a string that are present at an even index number

Code:

```
user_string = input("Enter string: ")
length = len(user_string)
range_len = range(0,length,2)
for i in range_len:
    print(user_string[i])
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex3_print_chars_from_string_at_even_index_num.py
Enter string: ABHISHEK
A
H
S
E
>>>
```

Practical 4

Remove first n characters from a string

Code:

```
user_string = input("Enter string: ")
num_of_chars = int(input("Enter number of characters to remove from beginning: "))
new_string = user_string[num_of_chars:-1]
print("Your string after removing first ", num_of_chars, " is: ", new_string)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex4_remove_first_n_chars_
from_string.py
Enter string: Hello, Python! I am a Developer.
Enter number of characters to remove from beginning: 10
Your string after removing first 10 is: hon! I am a Developer
>>>
```

Practical 5

Check if the first and last number of a list is the same

Code:

```
num = int(input("Enter number of elements to add in list: "))
rangeLen = range(num)
user_list = []
for i in rangeLen:
    print("Enter element no.", i+1, " : ", end="")
    user_num = int(input())
    user_list.append(user_num)
if user_list[0] == user_list[-1]:
    print("First and last numbers are same.")
else:
    print("First and last numbers are not same.")
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex5_check_first_and_last_
num_of_list_same.py
Enter number of elements to add in list: 5
Enter element no. 1 : 32
Enter element no. 2 : 45
Enter element no. 3 : 67
Enter element no. 4 : 12
Enter element no. 5 : 14
First and last numbers are not same.
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex5_check_first_and_last_
num_of_list_same.py
Enter number of elements to add in list: 5
Enter element no. 1 : 32
Enter element no. 2 : 45
Enter element no. 3 : 67
Enter element no. 4 : 12
Enter element no. 5 : 32
First and last numbers are same.
>>>
```

Practiaal 6

Display numbers divisible by 5 from a list

Code:

```
user_list = []
length = int(input("Enter number of elements to be entered in list: "))
rangeLen = range(length)
for i in rangeLen:
    print("Enter element no.", i, " : ", end="")
    num = int(input())
    user_list.append(num)

for i in user_list:
    if i%5==0:
        print(i," is divisible by 5.")
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex6_display_nums_divisibl
e_by_5_from_list.py
Enter number of elements to be entered in list: 5
Enter element no. 0 : 123
Enter element no. 1 : 145
Enter element no. 2 : 23670
Enter element no. 3 : 234
Enter element no. 4 : 5675
145 is divisible by 5.
23670 is divisible by 5.
5675 is divisible by 5.
>>>
```


Practical 7

Return the count of a given substring from a string

Code:

```
string_to_examin = "Hello, World! I am a Python Developer. I am a freelancer.  
Also work on web development. Hello, Development"
```

```
to_find_string = "Hello"
```

```
print(to_find_string, " occurred ", string_to_examin.count(to_find_string), " times  
in ", string_to_examin)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex7_return_count_of_give_  
substring_from_string.py  
Hello occurred 2 times in Hello, World! I am a Python Developer. I am a freel  
ancer. Also work on web development. Hello, Development  
>>>
```

Practical 8

Print the following pattern

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

Code:

```
num= int(input("Enter number for triangle: "))
heightRangeLen = range(num)
for i in heightRangeLen:
    lengthRangelen = range(i+1)
    for j in lengthRangelen:
        print(i+1, end=" ")
    print()
```

Output:

```
=== RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex8_print_pattern.py ==
Enter number for triangle: 5
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Practical 9

Check Palindrome Number

Code:

```
user_num = input("Enter number: ")

if user_num == user_num[::-1]:
    print(user_num, " is a palindrome number.")
else:
    print(user_num, " is not a palindrome number.")
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex9_check_palindrome_num.
PY
Enter number: 12521
12521  is a palindrome number.
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex9_check_palindrome_num.
PY
Enter number: 12345
12345  is not a palindrome number.
```

Practical 10

Create a new list from a two list using the following condition

- new list should contain odd numbers from the first list and even numbers from the second list.

Code:

```
list1= [23,44,567,67,88,987]
```

```
list2 = [24, 45, 89, 90, 33]
```

```
new_list = []
```

```
for i in list1:
```

```
    if i%2==1:
```

```
        new_list.append(i)
```

```
for i in list2:
```

```
    if i%2==0:
```

```
        new_list.append(i)
```

```
print("New list is: ", new_list)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex10_merge_2_list_odd_fro
m_first_even_from_second.py
New list is:  [23, 567, 67, 987, 24, 90]
```

Practical 11

Write a Program to extract each digit from an integer in the reverse order.

Code:

```
num = int(input("Enter integer: "))
new_num=0
store_orignal_num = num
while num>0:
    new_num = new_num*10 + num%10
    num = num // 10
print("Digits of ", store_orignal_num, " extracted in reverse order is: ", new_num)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex11_extract_digit_from_i
nt_in_reverse_order.py
Enter integer: 12345
Digits of 12345 extracted in reverse order is: 54321
```

Practical 12

Calculate income tax for the given income by adhering to the below rules

Taxable Income	Rate (in %)
First \$10,000	0
Next \$10,000	10
The remaining	20

Code:

```
"""
```

```
Taxable income      Rate
```

```
10,000              0
```

```
next 10,000         10
```

```
remaining           20
```

```
"""
```

```
income_tax = 0
```

```
your_income = float(input("Your income is $ "))
```

```
income_division = []
```

```
if your_income < 10000:
```

```
    income_tax = 0
```

```
elif your_income < 20000:
```

```
    income_division.append(10000)
```

```
    income_division.append(your_income-10000)
```

else:

```
income_division.append(10000)
```

```
income_division.append(10000)
```

```
income_division.append(your_income-20000)
```

```
length = len(income_division)
```

```
rangeLen = range(length)
```

```
for i in rangeLen:
```

```
    if i == 1:
```

```
        income_tax += income_division[i]*0.1
```

```
    elif i > 1:
```

```
        income_tax += income_division[i]*0.2
```

```
print("Your income tax is: $", income_tax, "/-")
```

Output:

```
>>> = RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex12_calculate_income_tax
      .py
      Your income is $ 50000000
      Your income tax is: $ 9997000.0 /-
```

Practical 13

Print multiplication table form 1 to 10

Code:

```
table_of = range(1,11)
table_numbers = range(1,11)
for i in table_of:
    print("Table of ", i)
    for j in table_numbers:
        print(i, " x ", j, " = ", i*j)
    print()
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex13_multiplication_table
_1_to_10.py
Table of 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Table of 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```


Table of 3

3	x	1	=	3
3	x	2	=	6
3	x	3	=	9
3	x	4	=	12
3	x	5	=	15
3	x	6	=	18
3	x	7	=	21
3	x	8	=	24
3	x	9	=	27
3	x	10	=	30

Table of 4

4	x	1	=	4
4	x	2	=	8
4	x	3	=	12
4	x	4	=	16
4	x	5	=	20
4	x	6	=	24
4	x	7	=	28
4	x	8	=	32
4	x	9	=	36
4	x	10	=	40

Table of 5

5	x	1	=	5
5	x	2	=	10
5	x	3	=	15
5	x	4	=	20
5	x	5	=	25
5	x	6	=	30
5	x	7	=	35
5	x	8	=	40
5	x	9	=	45
5	x	10	=	50

Table of 6

6	x	1	=	6
6	x	2	=	12
6	x	3	=	18
6	x	4	=	24
6	x	5	=	30
6	x	6	=	36
6	x	7	=	42
6	x	8	=	48
6	x	9	=	54
6	x	10	=	60

Table of 7

7	x	1	=	7
7	x	2	=	14
7	x	3	=	21
7	x	4	=	28
7	x	5	=	35
7	x	6	=	42
7	x	7	=	49
7	x	8	=	56
7	x	9	=	63
7	x	10	=	70

Table of 8

8	x	1	=	8
8	x	2	=	16
8	x	3	=	24
8	x	4	=	32
8	x	5	=	40
8	x	6	=	48
8	x	7	=	56
8	x	8	=	64
8	x	9	=	72
8	x	10	=	80

Table of 9

9	x	1	=	9
9	x	2	=	18
9	x	3	=	27
9	x	4	=	36
9	x	5	=	45
9	x	6	=	54
9	x	7	=	63
9	x	8	=	72
9	x	9	=	81
9	x	10	=	90

Table of 10

10	x	1	=	10
10	x	2	=	20
10	x	3	=	30
10	x	4	=	40
10	x	5	=	50
10	x	6	=	60
10	x	7	=	70
10	x	8	=	80
10	x	9	=	90
10	x	10	=	100

Practical 14

Print downward Half-Pyramid Pattern with Star (asterisk)

* * * * *

* * * *

* * *

* *

*

Code:

```
length_of_triangle_height = int(input("Enter height of triangle: "))
order_of_height = range(length_of_triangle_height,0,-1)
for i in order_of_height:
    elements_range = range(i)
    for j in elements_range:
        print("*",end=" ")
    print()
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex14_half_pyramid_invert
d.py
Enter height of triangle: 5
* * * * *
* * * *
* * *
* *
*
*
```

Practical 15

Write a function called `exponent(base, exp)` that returns an int value of base raises to the power of exp.

Code:

```
def exponent(base, exp):  
    return base ** exp  
  
base = float(input("base= "))  
exp = float(input("exponent= "))  
print(base, " ^ ", exp, " = ", exponent(base,exp))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\lab\ex15_exponent_function.py  
base= 4  
exponent= 5  
4.0 ^ 5.0 = 1024.0
```

Practical – 16

Write a program to create a function that takes two arguments, name and age, and print their value.

Code:

```
def function(name, age):  
    print("name is: ", name)  
    print("age is: ", age)  
  
function("John", "20")
```

Output:

```
= RESTART: C:\Use  
ate_function.py  
name is:  John  
age is:  20  
>>>
```

Practical 17

Write a program to create function func1() to accept a variable length of arguments and print their value.

Note: Create a function in such a way that we can pass any number of arguments to this function, and the function should process them and display each argument's value.

Code:

```
def func1(*args):  
    print("sum",args, " is: ", sum(args))
```

```
func1(20,30,40)
```

```
func1(10,30,50,60,100)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\De  
iabl_number_args.py  
sum (20, 30, 40) is: 90  
sum (10, 30, 50, 60, 100) is: 250
```

Practical-18

Write a program to create function calculation() such that it can accept two variables and calculate addition and subtraction. Also, it must return both addition and subtraction in a single return call

Code:

```
def calculation(num1,num2):  
    return num1+num2, num1-num2  
  
x = float(input("x = "))  
y = float(input("y = "))  
result = calculation(x,y)  
print("x + y = ", result[0])  
print("x - y = ", result[1])
```

Output:

```
= RESTART: C:\Users\uday0\Or  
urn_multiple_values.py  
x = 17866437423  
y = 28520759751  
x + y = 46387197174.0  
x - y = -10654322328.0  
...
```

Practical 19

Write a program to create a function show_employee() using the following conditions.

- It should accept the employee's name and salary and display both.
- If the salary is missing in the function call then assign default value 9000 to salary

Code:

```
def show_employee(name, salary=9000):  
  
    print("Employee name: ", name)  
  
    print("Employee salary: ", salary)  
  
name = input("Enter name: ")  
  
salary = int(input("Enter salary: "))  
  
print("\nPassing name only:-")  
  
show_employee(name)  
  
print("\nPassing name and salary")  
  
show_employee(name, salary)
```

Output:

```
= RESTART: C:\Users\uday0\O  
ault_arg.py  
Enter name: Rohan  
Enter salary: 4000  
  
Passing name only:-  
Employee name: Rohan  
Employee salary: 9000  
  
Passing name and salary  
Employee name: Rohan  
Employee salary: 4000
```

Practical – 20

Create an inner function to calculate the addition in the following way

- Create an outer function that will accept two parameters, **a** and **b**
- Create an inner function inside an outer function that will calculate the addition of **a** and **b**
- At last, an outer function will add 5 into addition and return it

Code:

```
def calculation(a,b):  
  
    result = 0  
  
    def addition():  
  
        return a+b  
  
    result += addition()  
  
    result += 5  
  
    return result  
  
print(calculation(3,5))
```

Output:

```
= RESTART: C:\u  
er_function.py  
13
```


Practical – 21

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

Code:

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

Output:

```
= RESTART: C:  
ursive.py  
55
```

Practical – 22

Assign a different name to function and call it through the new name

Below is the function `display_student(name, age)`. Assign a new name `show_tudent(name, age)` to it and call it using the new name.

Code:

```
def display_student(name, age):  
    print("Student name is: ", name)  
    print("Student age is: ", age)
```

```
show_student = display_student
```

```
show_student("Rohan",18)
```

Output:

```
= RESTART: C:\Users\uday0\  
_name_to_func.py  
Student name is: Rohan  
Student age is: 18
```

Practical – 23

Generate a Python list of all the even numbers between 4 to 30

Code:

```
def even_numbers_4_to_30():  
    return list(range(4,30,2))  
  
print("List is: ", even_numbers_4_to_30())
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab  
n_nos_bw_4_and_30.py  
List is: [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

Practical – 24

Find the largest item from a given list

Code:

```
myList = [4,6,5,8,80,90,10,3,2]
print("Largest item in the list: ", max(myList))
```

Output:

```
= RESTART: C:\Users\uday0\OneDri
gest_of_list.py
Largest item in the list:  90
```

Practical-25

Write a program that defines a function `count_lower_upper()` that accepts a string and calculates the number of uppercase and lowercase alphabets in it. It should return these values as a dictionary. Call this function for some sample strings.

Code:

```
def count_lower_upper(userstring):  
    record = {"lower":0, "upper":0}  
    characterRange = list(range(65,91)) + list(range(97,123))  
    for i in userstring:  
        if ord(i) in characterRange:  
            if i.islower():  
                record["lower"]+=1  
            elif i.isupper():  
                record["upper"]+=1  
    return record  
  
while True:  
    user = input("Enter String: ")  
    print(count_lower_upper(user))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\I  
ingLowerandUpper.py  
Enter String: 123#thesunofthewest  
{'lower': 15, 'upper': 0}  
Enter String: He110  
{'lower': 1, 'upper': 1}  
Enter String: •
```

Practical – 26

Write a program that defines a function compute() that calculates the value of $n + nn + nnn + nnnn$, where n is digit received by the function. Test the function for digits 4 and 7.

Code:

```
def compute(n):  
    return n * (1+11+111+1111)  
  
print(compute(4))  
  
print(compute(7))
```

Output:

```
= RESTART: C:\Usr  
te_function.py  
4936  
8638
```

Practical – 27

Write a program that defines a function create_array() to create and return a 3D array whose dimensions are passed to the function. Also initialize each element of this array to a value passed to the function.

Code:

```
def create_array(dims,values):  
    l=0  
    array = []  
    if len(values)==dims[0]*dims[1]*dims[2]:  
        for i in range(dims[0]):  
            subarray=[]  
            for j in range(dims[1]):  
                subsubarray=[]  
                for k in range(dims[2]):  
                    subsubarray.append(values[l])  
                    l+=1  
                subarray.append(subsubarray)  
            array.append(subarray)  
    return array
```



```
aList = []

for i in range(3*8*3):

    aList.append(i)

print(create_array((3,8,3), aList))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\1_09_2023\q3_create_3d_array_and_initialize_and_return.py
[[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11], [12, 13, 14], [15, 16, 17], [18, 19, 20], [21, 22, 23]], [
[24, 25, 26], [27, 28, 29], [30, 31, 32], [33, 34, 35], [36, 37, 38], [39, 40, 41], [42, 43, 44], [45, 46, 47]], [[48, 49, 50], [51, 52, 53], [54, 55, 56], [57, 58, 59], [60, 61, 62], [63, 64, 65], [66, 67, 68], [69, 70, 71]]]
```

Practical – 28

Write a program that defines a function `create_list()` to create and return a list which is an intersection of two lists passed to it.

Code:

```
def create_array(list1,list2):  
  
    list3=[]  
  
    for i in list1:  
  
        if i in list2:  
  
            list3.append(i)  
  
    return list3  
  
print(create_array([3, 2, 5, 2, 8, 8, 8, 7, 4, 2],[8, 1, 7, 7, 1, 8, 5, 4, 2, 6]))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive  
thon\lab\1_09_2023\q4_create_ret_l  
_two_lists.py  
[2, 5, 2, 8, 8, 8, 7, 4, 2]
```

Practical – 29

Write a program that defines a function `sanitize_list()` to remove all duplicate entries from the list that it receives.

Code:

```
def sanitize_list(givenList):  
    newList = []  
    for i in set(givenList):  
        newList.append(i)  
    return newList  
  
print(sanitize_list([8, 8, 6, 4, 1, 6, 1, 5, 8, 5]))
```

Output:

```
= RESTART: C:\Users\ud  
ython\lab\1_09_2023\q5  
[1, 4, 5, 6, 8]
```

Practical – 30

Write a program to receive three integers from keyboard and get their sum and product calculated through a user-defined function cal_sum_prod().

Code:

```
def cal_sum_prod(num1,num2,num3):  
    return num1+num2+num3, num1*num2*num3  
  
n1 = int(input("Enter first integer: "))  
n2 = int(input("Enter second integer: "))  
n3 = int(input("Enter third integer: "))  
result = cal_sum_prod(n1,n2,n3)  
  
print("Sum is: ", result[0])  
print("Product is: ", result[1])
```

Output:

```
= RESTART: C:\Users\uday0\OneDri  
_int_sum_prod_from_function.py  
Enter first integer: 5  
Enter second integer: 2  
Enter third integer: 5  
Sum is: 12  
Product is: 50
```

Practical – 31

Pangram is a sentence that uses every letter of the alphabet. Write a program that checks whether a given string is pangram or not, through a user-defined function ispangram().

Code:

```
def ispangram(sentence):  
  
    alphabets = [chr(i) for i in range(65,91)] + [chr(i) for i in range(97,123)] #list of  
    all alphabets lowercase and uppercase  
  
    for i in set(sentence):  
  
        if i not in alphabets:  
  
            return False  
  
    return True  
  
statement = input("Enter statment to check for pangram: ")  
  
if ispangram(statement):  
  
    print("It is a pangram.")  
  
else:  
  
    print("It is not a pangram.")
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\P  
_sentence_for_pangram_of_alphabets.py  
Enter statment to check for pangram: Hello  
It is a pangram.  
>>>  
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\P  
_sentence_for_pangram_of_alphabets.py  
Enter statment to check for pangram: Hello, World!  
It is not a pangram.
```

Practical – 32

Write a Python program that accepts a hyphen-separated sequence of words as input and calls a function convert() which converts it into a hyphen-separated sequence after sorting them alphabetically. For example, if the input string is here-come-the-dots-followed-by-dashes

Then, the output must be:

by-come-dashes-dots-followed-here-the

Code:

```
def convert(sentence):  
    return "-".join(sorted(sentence.split("-")))  
  
user_input = input("Enter string: ")  
  
print(convert(user_input))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Pr  
hyphen_separated.py  
Enter string: here-come-the-dots-followed-by-dashes  
by-come-dashes-dots-followed-here-the  
>>>
```

Practical – 33

Write a python function to create and return a list containing tuples of the form (x, x², x³) for all x between 1 and 20 (both included).

Code:

```
def x_square_cube():  
  
    returnList = []  
  
    for x in range(1,21):  
  
        returnList.append((x,x**2,x**3))  
  
  
    return returnList  
  
print(x_square_cube())
```

Output:

```
>>> = RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\8_09_2023\p4_return_list_of_tuple_of_x_square_cube.py  
[(1, 1, 1), (2, 4, 8), (3, 9, 27), (4, 16, 64), (5, 25, 125), (6, 36, 216), (7, 49, 343), (8, 64, 512), (9, 81, 729), (10, 100, 1000), (11, 121, 1331), (12, 144, 1728), (13, 169, 2197), (14, 196, 2744), (15, 225, 3375), (16, 256, 4096), (17, 289, 4913), (18, 324, 5832), (19, 361, 6859), (20, 400, 8000)]  
>>>
```


Practical – 34

Write a program that defines a function ispalindrome() which checks whether a given string is a palindrome or not. Ignore spaces and case mismatch while checking for palindrome.

Code:

```
def ispalindrome(arg_string):  
    arg_string = arg_string.replace(" ", "").lower()  
    if arg_string == arg_string[::-1]:  
        return True  
    return False  
  
user_string = input("Enter string: ")  
print("Palindrome condition: ", ispalindrome(user_string))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\Col:
indrome.py
Enter string: Ni tin
Palindrome condition:  True
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\Col:
indrome.py
Enter string: Malayalam
Palindrome condition:  True
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\Col:
indrome.py
Enter string: Murder for a jar of red rum
Palindrome condition:  True
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\Col:
indrome.py
Enter string: Rats live on no evil star
Palindrome condition:  True
>>>
= RESTART: C:\Users\uday0\OneDrive\Desktop\Col:
indrome.py
Enter string: Python
Palindrome condition:  False
>>>
```

Practical – 35

Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically.

Code:

```
def convert(arg_string):  
  
    x = sorted(set(arg_string.split(" ")))  
  
    return " ".join(x)
```

```
sentence = input("Enter string: ")  
  
print(convert(sentence))
```

Output:

```
... = RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\8_09_2023\p6_string_remove_space_duplicate_sort.py  
Enter string: Sakhi was a singer because her mother was a singer, and Sakhi's mother was a singer because her father was a singer.  
Sakhi Sakhi's a and because father her mother singer singer, singer. was
```

Practical – 36

Write a program that defines a function `count_alphabets_digits()` that accepts a string and calculates the number of alphabets and digits in it. It should return these values as a dictionary. Call this function for some sample strings.

Code:

```
def count_alphabets_digits(userstring):  
    record = {"alphabets":0, "digits":0}  
    characterRange = list(range(65,91)) + list(range(97,123)) + list(range(48,58))  
    for i in userstring:  
        if ord(i) in characterRange:  
            if i.isalpha():  
                record["alphabets"]+=1  
            elif i.isdigit():  
                record["digits"]+=1  
    return record  
  
while True:  
    user = input("Enter String: ")  
    print(count_alphabets_digits(user))
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Dr  
_alphas_and_digits.py  
Enter String: Hel10, W0rld!  
{ 'alphabets': 5, 'digits': 5}  
Enter String:
```

Practical – 37

Write a program that defines a function called frequency() which computes the frequency of words present in a string passed on it. The frequencies should be returned in sorted order by words in the string.

Code:

```
def frequency(arg_string):  
    returnDict = {}  
  
    arg_string = arg_string.split(" ")  
  
    for i in set(arg_string):  
        returnDict[i] = 0  
  
    for i in arg_string:  
        returnDict[i] +=1  
  
    return returnDict  
  
user_string = input("Enter string: ")  
  
print(frequency(user_string))
```

Output:

```
Enter string.  
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\8_09_2023\p8_freque  
ncy_of_word.py  
Enter string: It is true for all that that that that that refers to is not  
the same that that that that refers to  
{'It': 1, 'for': 1, 'the': 1, 'to': 2, 'refers': 2, 'is': 2, 'that': 10, 'not':  
1, 'true': 1, 'all': 1, 'same': 1}
```

Practical – 38

Write a program that defines two functions called `create_sent1()` and `create_sent2()`.

Both receive following 3 lists:

```
subjects = ['He', 'She']
```

```
verb = ['loves', 'hates']
```

```
objects = ['TV Serials', 'Netflix']
```

Both functions should form sentences by picking elements from the lists and returns them. Use for loops in `create_sent1()` and list comprehension in `create_sent2()`.

Code:

```
def create_sent1(sub,verb,obj):  
    # using for loop  
    sentences = []  
    for i in sub:  
        for j in verb:  
            for k in obj:  
                sentences.append(" ".join((i,j,k)))  
    return sentences  
  
def create_sent2(sub,verb,obj):
```

```
# Using list comprehension

return [" ".join((i,j,k)) for i in sub for j in verb for k in obj]


subject = ['He','She']
verb = ['loves', 'hates']
obj = ['TV Serial','Netflix']


print("Using create_sent1() function which uses for loop:")
for i in create_sent1(subject,verb,obj):
    print(i)


print("\nUsing create_sent2() function which uses list comprehension:")
for i in create_sent2(subject,verb,obj):
    print(i)
```


Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\8_09_2023\p9_sente  
nce_creation.py  
Using create_sent1() function which uses for loop:  
He loves TV Serial  
He loves Netflix  
He hates TV Serial  
He hates Netflix  
She loves TV Serial  
She loves Netflix  
She hates TV Serial  
She hates Netflix  
  
Using create_sent2() function which uses list comprehension:  
He loves TV Serial  
He loves Netflix  
He hates TV Serial  
He hates Netflix  
She loves TV Serial  
She loves Netflix  
She hates TV Serial  
She hates Netflix  
>>>
```

Practical – 39

Perform the following operations on a list of names.

- **Create a list of 5 names –
‘Anil’, ‘Amol’, ‘Aditya’, ‘Avi’, ‘Alka’**
- **Insert a name ‘Anuj’ before ‘Aditya’**
- **Append a name ‘Zulu’**
- **Delete ‘Avi’ from the list**
- **Replace ‘Anil’ with ‘AnilKumar’**
- **Sort all the names in the list**
- **Print reversed sorted list**

Code:

```
#initializing the list
```

```
namelist=['Anil','Amol','Aditya','Avi','Alka']
```

```
#inserting Anuj before Aditya in list.
```

```
namelist.insert(namelist.index('Aditya'),'Anuj')
```

```
print("Name list after inserting 'Anuj' before 'Aditya' : ", namelist)
```

```
# Appending name Zulu to the list
```

```
namelist.append('Zulu')
```

```
print("Name list after appending name 'Zulu' to list: ", namelist)
```

```
# removing Avi from the list

namelist.remove('Avi')

print("Name list after removing 'Avi' from the list: ", namelist)


# replacing Anil with AnilKumar

namelist[namelist.index('Anil')]='AnilKumar'

print("Name list after replacing 'Anil' with 'AnilKumar' in the list: ", namelist)


# Sorting the list

namelist.sort()

print("Name list after names in the list: ", namelist)


# reversing the list

namelist.reverse()

print("Name list after reversing the order of names in the list: ", namelist)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2023\q1_operation_on_names_list.py •
Name list after inserting 'Anuj' before 'Aditya' : ['Anil', 'Amol', 'Anuj', 'Aditya', 'Avi', 'Alka']
Name list after appending name 'Zulu' to list: ['Anil', 'Amol', 'Anuj', 'Aditya', 'Avi', 'Alka', 'Zulu']
Name list after removing 'Avi' from the list: ['Anil', 'Amol', 'Anuj', 'Aditya', 'Alka', 'Zulu']
Name list after replacing 'Anil' with 'AnilKumar' in the list: ['AnilKumar', 'Amol', 'Anuj', 'Aditya', 'Alka', 'Zulu']
Name list after names in the list: ['Aditya', 'Alka', 'Amol', 'AnilKumar', 'Anuj', 'Zulu']
Name list after reversing the order of names in the list: ['Zulu', 'Anuj', 'AnilKumar', 'Amol', 'Alka', 'Aditya']
```

Practical – 40

Perform the following operations on a list of numbers.

- **Create a list of 5 odd numbers**
- **Create a list of 5 even numbers**
- **Combine the two lists**
- **Add prime numbers 11,17,29 at the beginning of the combined list**
- **Report how many elements are present in the list**
- **Replace last 3 numbers in the list with 100 , 200 , 300**
- **Delete all the numbers in the list**
- **Delete the list**

Code:

```
odd_num_list = list(range(1,10,2)) # created list of 5 odd numbers
even_num_list = list(range(2,11,2)) # created list of 5 even numbers
combined_list = odd_num_list + even_num_list # combining both list
print("Combinded list: ", combined_list)
combined_list = [11,17,29] + combined_list
print("After adding prime number to the beginning of the list: ", combined_list)
print("Number of elements present in th e list = ", len(combined_list))
combined_list[-3:len(combined_list)] = [100,200,300]
print("list after replacing last 3 numbers with 100, 200, 300: ", combined_list)
del combined_list[0:]
```

```
print("list after deleting all numbers: ", combined_list)
```

```
del combined_list
```

```
print("List deleted!")
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2023\q2_operations_on_numbers_list.py
Combinded list:  [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
After adding prime number to the beginning of the list:  [11, 17, 29, 1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
Number of elements present in th e list =  13
list after replacing last 3 numbers with 100, 200, 300:  [11, 17, 29, 1, 3, 5, 7, 9, 2, 4, 100, 200, 300]
list after deleting all numbers:  []
List deleted!
```

Practical – 41

Write a program to implement a Stack data structure. Stack is a Last In First Out (LIFO) list, in which addition and deletion takes place at the same end.

Code:

```
stack = []

def push(element):
    stack.append(element)

def popping():
    print("Popped: ",stack.pop())

while True:
    print("""\n\nOperations to be done on stack:
1. Push
2. Pop
3. Display stack
e. Exit
Enter: """, end=" ")
    option = input()
    if option=='1':
```

```
    element = input("Enter element to push: ")

    push(element)

elif option=='2':

    popping()

elif option=='3':

    print("Stack is: ", stack)

elif option.lower() == 'e':

    print("Program Terminated...")

    break

else:

    print("Invalid option!")
```

Output:

Push

```
Operations to be done on stack:
1. Push
2. Pop
3. Display stack
e. Exit
Enter: 1
Enter element to push: 1

Operations to be done on stack:
1. Push
2. Pop
3. Display stack
e. Exit
Enter: 1
Enter element to push: 2

Operations to be done on stack:
1. Push
2. Pop
3. Display stack
e. Exit
Enter: 1
Enter element to push: 3
```


Pop

```
Operations to be done on stack:  
1. Push  
2. Pop  
3. Display stack  
e. Exit  
Enter: 2  
Popped: 3
```

Display (Traverse)

```
Operations to be done on stack:  
1. Push  
2. Pop  
3. Display stack  
e. Exit  
Enter: 3  
Stack is: ['1', '2']
```

Exit

```
Operations to be done on stack:  
1. Push  
2. Pop  
3. Display stack  
e. Exit  
Enter: e  
Program Terminated...
```

Practical – 42

Write a program to implement a Queue data structure. Queue is a First In First Out (FIFO) list, in which addition takes place at the rear end of the queue and deletion takes place at the front end of the queue.

Code:

```
queue = []

def enqueue(element):

    queue.append(element)

def dequeue():

    global queue

    print("Dequeued element: ", queue[0])

    queue=queue[1:]

while True:

    print("""\n\nOperations to be done on queue:

1. Enqueue

2. Dequeue

3. Display queue

e. Exit
```

```
Enter: "", end=" ")

option = input()

if option=='1':

    element = input("Enter element to enqueue: ")

    enqueue(element)

elif option=='2':

    dequeue()

elif option=='3':

    print("queue is: ", queue)

elif option.lower() == 'e':

    print("Program Terminated...")

    break

else:

    print("Invalid option!")
```

Output:

Enqueue

```
Operations to be done on queue:
1. Enqueue
2. Dequeue
3. Display queue
e. Exit •
Enter: 1
Enter element to enqueue: 1
```

```
Operations to be done on queue:
1. Enqueue
2. Dequeue
3. Display queue
e. Exit
Enter: 1
Enter element to enqueue: 2
```

```
Operations to be done on queue:
1. Enqueue
2. Dequeue
3. Display queue
e. Exit
Enter: 1
Enter element to enqueue: 3
```

Dequeue

```
Operations to be done on queue:
1. Enqueue
2. Dequeue
3. Display queue
e. Exit
Enter: 2
Dequeued element: 1
```

Display (Traverse)

```
Operations to be done on queue:  
1. Enqueue  
2. Dequeue  
3. Display queue  
e. Exit  
Enter: 3  
queue is: ['2', '3']
```

Exit

```
Operations to be done on queue:  
1. Enqueue  
2. Dequeue  
3. Display queue  
e. Exit  
Enter: e  
Program Terminated...
```

Practical – 43

Write a program to generate and store in a list 20 random numbers in the range 10 to 100. From this list delete all those entries which have value between 20 and 50. Print the remaining list.

Code:

```
import random

random_num_list = [random.randrange(10,101) for i in range(20)]

print("Random number list: ", random_num_list)

i=0

while i<len(random_num_list):

    if random_num_list[i]>=20 and random_num_list[i]<=50:

        del random_num_list[i]

    else:

        i+=1

print("List after removing elements between 20 and 50: ", random_num_list)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2023\q5_random_list_remove_between_20_50.py
Random number list:  [86, 23, 58, 57, 74, 73, 11, 34, 14, 93, 20, 46, 88, 56, 35, 92, 65, 44, 45, 42]
List after removing elements between 20 and 50:  [86, 58, 57, 74, 73, 11, 14, 93, 88, 56, 92, 65]
```

Practical – 44

Write a program to add two 3x4 matrices.

Code:

```
mat1=[[1,2,3,4],
      [5,6,7,8],
      [9,10,11,12]]
mat2=[[1,2,3,4],
      [5,6,7,8],
      [9,10,11,12]]
mat3=[[0,0,0,0],
      [0,0,0,0],
      [0,0,0,0]]
i,j=0,0
while i<3:
    while j<4:
        mat3[i][j] = mat1[i][j] + mat2[i][j]
        j+=1
    i+=1
    j=0
print("Sum is: ", mat3)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2023\q6_matr  
ix_addition.py  
Sum is:  [[2, 4, 6, 8], [10, 12, 14, 16], [18, 20, 22, 24]]
```


Practical – 45

Pass a tuple to divmod() function and obtain the quotient and the remainder.

Code:

```
def divmod(numtuple):  
    return numtuple[0]//numtuple[1], numtuple[0]%numtuple[1]  
  
result = divmod((10,3))  
print("Quotient is: ", result[0])  
print("Remainder is: ", result[1])
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\De  
od_quotient_remainder.py  
Quotient is:  3  
Remainder is: 1
```

Practical – 46

Write a Python program to perform the following operations:

- Pack first 10 multiples of 10 into a tuple
- Unpack the tuple into 10 variables, each holding 1 value
- Unpack the tuple such that first values gets stored in variable x, last value in y and all values in between into disposable variables _
- Unpack the tuple such that first values gets stored in variable i, last value in j and all values in between into disposable variables _

Code:

```
multiple_of_ten = tuple([i for i in range(10,110,10) if i%10==0])
```

```
v1,v2,v3,v4,b5,v6,v7,v8,v9,v10 = multiple_of_ten
```

```
x,*_,y = multiple_of_ten
```

```
i,*_,j = multiple_of_ten
```

```
print(x,_,y)
```

```
print(i,_,j)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\Coll  
ation_on_tuple.py  
10 [20, 30, 40, 50, 60, 70, 80, 90] 100  
10 [20, 30, 40, 50, 60, 70, 80, 90] 100
```

Practical – 47

A list contains names of boys and girls as its elements. Boys' names are stored as tuples. Write a Python program to find out number of boys and girls in the list.

Code:

```
names_list =
["Nidhi", "Niharika", "Meena", ("Abhishek", "Bunty", "Manan", "Rohit"), "Rashmi", "Drishti"]

girls = 0

boys=0

i=0

while i<len(names_list):

    if type(names_list[i]) is tuple:

        boys=len(names_list[i])

    else:

        girls+=1

    i+=1

print("Number of boys=", boys, "\nNumber of girls=", girls)
```

Output:

```
= RESTART: C:\Users\uday0\O
f_boys_and_girls.py
Number of boys= 4
Number of girls= 5
```

Practical – 48

A list contains tuples containing roll number, names and age of student. Write a Python Program to gather all the names from this list into another list.

Code:

```
data = [(1,"Abhishek",19),(2,"Bunty",18),(3,"Manan",19),(4,"Rohit",19)]

names = []

for i in data:

    names.append(i[1])

print("Names of student in data: ", names)
```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2
her_names.py
Names of student in data:  ['Abhishek', 'Buntv', 'Manan', 'Rohit']
```

Practical – 49

Given the following tuple

(‘F’,‘l’,‘a’,‘b’,‘b’,‘e’,‘r’,‘g’,‘a’,‘s’,‘t’,‘e’,‘d’)

Write a Python program to carry out the following operations:

- Add an ! at the end of the tuple
- Convert a tuple to a string
- Extract (‘b’,‘b’) from the tuple
- Find out number of occurrences of ‘e’ in the tuple
- Check whether ‘r’ exists in the tuple
- Convert the tuple to a list
- Delete characters ‘b’,‘b’,‘e’,‘r’ from the tuple

Code:

```
data = ('F','l','a','b','b','e','r','g','a','s','t','e','d') #create tuple
```

```
# adding '!' in end of tuple.
```

```
data = list(data) #unpacking tuple
```

```
data.append('!')
```

```
data=tuple(data)#packing tuple
```

```
print("tuple after appending '!' in it: ", data)
```

```

# converting tuple to string

tupletostring = "".join(data)

print("After converting tuple to string: ", tupletostring)


# extracting ('b','b') from given tuple

if data[data.index('b')+1] == 'b':

    print(data[data.index('b'):data.index('b')+2])


# Counting occurrence of 'e' in given tuple

print("Number of occurrence of 'e' is ", data.count('e'))


# checking 'r' exists in data or not

print("'r' exists in given tuple: ", 'r' in data)


# converting tuple to list

data = list(data)

print("After converting tuple to list: ", data)


# deleting characters 'b','b','e','r' from given tuple

data = tuple("".join(data).replace('bber',''))

print("tuple after deleting characters 'b','b','e','r': ", data)

```

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Desktop\College\Python\lab\15_09_2023\q11_tuple_operations.py
tuple after appending '!' in it: ('F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd', '!')
After converting tuple to string: Flabbergasted!
('b', 'b')
Number of occurrence of 'e' is 2
'r' exists in given tuple: True
After converting tuple to list: ['F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd', '!']
tuple after deleting characters 'b','b','e','r': ('F', 'l', 'a', 'g', 'a', 's', 't', 'e', 'd', '!')
```

Practical-50

What will be the output of the following program?

`a = {10,20,30,40,50,60,70}`

`b = {33,44,51,10,20,50,30,33}`

`print(a | b)`

`print(a & b)`

`print(a - b)`

`print(b - a)`

`print(a ^ b)`

`print(a >=b)`

`print(a <= b)`

Output:

```
= RESTART: C:\Users\uday0\OneDrive\Deskt
{33, 70, 40, 10, 44, 50, 51, 20, 60, 30}
{50, 10, 20, 30}
{40, 60, 70}
{33, 51, 44}
{33, 70, 40, 44, 51, 60}
False
False
```


Practical – 51

What will be the output of the following program?

a = {1,2,3,4,5,6,7}

b = {1,2,3,4,5,6,7}

c = {1,2,3,4,5,6,7}

d = {1,2,3,4,5,6,7}

e= {3,4,1,0,2,5,8,9}

a |= e

print(a)

b &= e

print(b)

c -=e

print(c)

d ^= e

print(d)

Output:

```
= RESTART: C:\Users\uday0\OneD
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5}
{6, 7}
{0, 6, 7, 8, 9}
```

Practical – 52

Write a program to carry out the following operations on the given set

$s = \{10, 2, -3, 4, 5, 88\}$

- number of items in set s
- maximum element in set s
- minimum element in set s
- sum of all elements in set s
- obtain a new sorted set from s, set s remaining unchanged
- report whether 100 is an element of set s
- report whether -3 is an element of set s

Code:

```
s = {10,2,-3,4,5,88}
```

```
print("Number of items in s= ",len(s))
```

```
print("Maximum element in s= ", max(s))
```

```
print("Minimum element in s= ", min(s))
```

```
print("Sum of all elements in s= ", sum(s))
```

```
k=set(sorted(list(s)))
```

```
print("New set is: ", k)
```

```
print("100 is element of s: ", 100 in s)
```

```
print("-3 is element of s: ", -3 in s)
```

Output:

```
Number of items in s= 6
Maximum element in s= 88
Minimum element in s= -3
Sum of all elements in s= 106
New set is: {2, 4, 5, 10, 88, -3}
100 is element of s: False
-3 is element of s: True
```

Practical – 53

What will be the output of the following program?

```
l = [10,20,30,40,50]
```

```
t = ('Sundeep', 25, 79.58)
```

```
s = 'set theory'
```

```
s1 = set(l)
```

```
s2 = set(t)
```

```
s3 = set(s)
```

```
print(s1)
```

```
print(s2)
```

```
print(s3)
```

Output:

```
{50, 20, 40, 10, 30}
{25, 'Sundeep', 79.58}
{'h', 'r', 'o', 's', 'y', 't', 'e', ' ' }
```

Practical – 54

What will be the output of the following program?

```
s= {1, 2, 3, 7, 6, 4)
```

```
s.discard(10)
```

```
s.remove(10)
```

```
print(s)
```

Output:

Element 10 is not present in set s.discard() would do nothing, whereas, remove() would report an error.

```
Traceback (most recent call last):
  File "C:\Users\uday0\OneDrive\Desktop\College\Python\lab\29_9_2023\ex5.py", line 3, in <module>
    s.remove(10)
  KeyError: 10
```

Practical – 55

What will be the output of the following program?

```
s1= {10, 20, 30, 40, 50}
```

```
s2 =(10, 20, 30, 40, 50}
```

```
print(id(s1), id(s2))
```

Output:

```
2366633340352 2366640225216
```

Practical - 56

What will be the output of the following program?

`s1={ 10, 20, 30, 40, 50}`

`s2 = { 10, 20, 30, 40, 50}`

`s3 = (*s1, *s2)`

`print(s3)`

Output:

```
- RESTART. C:\MSDEV\GCC
{40, 10, 50, 20, 30}
```

Practical - 57

What will be the output of the following program?

```
s= set('KanLabs')
```

```
t= s[:-1]
```

```
print(t)
```

Output:

Error: set is not a subscriptable object. In other words [] cannot be used with a set.

```
Traceback (most recent call last):
  File "C:\Users\uday0\OneDrive\Desktop\College\Python\lab\29_9_2023\ex8.py", line 2, in <module>
    t = s[:-1]
TypeError: 'set' object is not subscriptable
.
```


Practical – 58

What will be the output of the following program?

```
num = {10, 20, {30, 40}, 50}  
  
print(num)
```

Output:

Error: Nested sets are illegal.

```
Traceback (most recent call last):  
  File "C:\Users\uday0\OneDrive\Desktop\College\Python\lab\29_9_2023\ex9.py", line 1, in <module>  
    num = {10,20, {30, 40}, 50}  
TypeError: unhashable type: 'set'
```

Practical - 59

What will be the output of the following program?

```
s=(Tiger', 'Lion', 'Jackal')  
del(s)  
print(s)
```

Output:

Error: name 's' is not defined. This happens because del() deletes the set object.

```
Traceback (most recent call last):  
  File "C:\Users\uday0\OneDrive\Desktop\College\Python\lab\29_9_2023\ex10.py", 1  
line 3, in <module>  
    print(s)  
NameError: name 's' is not defined
```

Practical - 60

What will be the output of the following program?

```
fruits = ('Kiwi', 'Jack Fruit', 'Lichi')  
  
fruits.clear()  
  
print(fruits)
```

Output:

set() . After calling clear(), fruits becomes an empty set.

set ()

Practical – 61

What will be the output of the following program?

```
s = {10, 25, 4, 12, 3, 8}
```

```
s = sorted(s)
```

```
print(s)
```

Output:

```
[3, 4, 8, 10, 12, 25]
```

Practical – 62

What will be the output of the following program?

```
s = {}
```

```
t = {1,4,5,2,3}
```

```
print(type(s), type(t))
```

Output:

```
<class 'dict'> <class 'set'>
```

Practical – 63

A set contains names which begin either with A or with B. write a program to separate out the names into two sets, one containing names beginning with A and another containing names beginning with B.

Code:

```
simple_set = {'Abhishek','Bunty','Ashish','Bhawna','Bimal','Aman','Bhavesb'}

A_name_set = set()

B_name_set = set()

for i in simple_set:

    if i.lower().startswith('a'):

        A_name_set.add(i)

    else:

        B_name_set.add(i)

print("Set with names begins with A: ", A_name_set)

print("Set with names begins with B: ", B_name_set)
```

Output:

```
Set with names begins with A:  {'Ashish', 'Abhishek', 'Aman'}
Set with names begins with B:  {'Bhawna', 'Bunty', 'Bhavesb', 'Bimal'}
```

Practical – 64

Create an empty set. Write a program that adds five new names to this set, modifies one existing name and deletes two names existing in it.

Code:

```
names = set()

names.add("Abhishek")

names.add("Bunty")

names.add("Manan")

names.add("Rohit")

names.add("Nidhi")

print(names)

names = set(((( " ".join(names)).replace("Abhishek","AbhishekRoka"))).split(" "))

print(names)

names.discard("Rohit")

names.discard("Nidhi")

print(names)
```

Output:

```
{'Bunty', 'Nidhi', 'Rohit', 'Abhishek', 'Manan'}
{'Bunty', 'AbhishekRoka', 'Nidhi', 'Rohit', 'Manan'}
{'Bunty', 'AbhishekRoka', 'Manan'}
```

Practical – 65

Write a program to create a set containing 10 randomly generated numbers in the range 15 to 45. Count how many of these numbers are less than 30. Delete all numbers which are greater than 35.

Code:

```
from random import randrange

nums = set([randrange(15,46) for i in range(10)])

count = 0

greaterthan35 = []

for i in nums:

    if i<30:

        count+=1

    elif i > 35:

        greaterthan35.append(i)

for i in greaterthan35:

    nums.discard(i)

print("Numbers less than 30 in given set: ", count)

print("Set after deleting all numbers greater than 35: ", nums)
```

Output:

```
Numbers less than 30 in given set:  6
Set after deleting all numbers greater than 35:  {32, 18, 21, 23, 25, 26, 27}
```


Practical – 66

What will be the output of the following program?

```
s = {'Mango', 'Banana', 'Guava', 'Kiwi'}
```

```
s.clear( )
```

```
print(s)
```

```
del(s)
```

```
print(s)
```

Output:

```
set( )
```

NameError: name 's' is not defined

```
set()  
Traceback (most recent call last):  
  File "C:\Users\uday0\OneDrive\Desktop\College\Python\lab\29_9_2023\ex17.py", 1  
line 5, in <module>  
    print(s)  
NameError: name 's' is not defined
```

Practical – 67

Create a dictionary called **students** containing names and ages. Copy the dictionary into **stud**. Empty the **students** dictionary, as **stud** continues to hold the data.

Code:

```
students= {  
    "Abhishek":19,  
    "Bunty":18,  
    "Manan":20,  
    "Mayank":18  
}  
  
stud = students.copy()  
  
print("Initially, students=",students, "\nand, stud=" ,stud)  
  
students.clear()  
  
print("After clearing students dict")  
  
print("students=",students, "\nand, stud=" ,stud)
```

Output:

```
Initially, students= {'Abhishek': 19, 'Bunty': 18, 'Manan': 20, 'Mayank': 18}  
{  
and, stud= {'Abhishek': 19, 'Bunty': 18, 'Manan': 20, 'Mayank': 18}  
After clearing students dict  
students= {}  
and, stud= {'Abhishek': 19, 'Bunty': 18, 'Manan': 20, 'Mayank': 18}
```

Practical – 68

Create a list of cricketers. Use this list to create a dictionary in which the list values become keys of the dictionary. Set the values of all keys to 50 in the dictionary created.

Code:

```
cricketers = ["Virat","Rohit","Yujvendra","Brett","Mitchell","Chris"]
```

```
cricketers_dict = dict()
```

```
for i in cricketers:
```

```
    cricketers_dict[i] = 50
```

```
print("Created cricketers dict from list is: ", cricketers_dict)
```

Output:

```
Created cricketers dict from list is: {'Virat': 50, 'Rohit': 50, 'Yujvendra': 50, 'Brett': 50, 'Mitchell': 50, 'Chris': 50}
```

Practical – 69

Write a program to sort a dictionary in ascending/descending order by key and ascending/descending order by value.

Code:

```
a_dict = {
    1:2,
    2:3,
    3:1,
    4:0
}

while True:
    print("Options:")
    print("1. Sort in ascending order by key")
    print("2. Sort in descending order by key")
    print("3. Sort in ascending order by value")
    print("4. Sort in descending order by value")
    print("5. Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        sorted_dict = dict(sorted(a_dict.items()))
        print("Ascending order by key:", sorted_dict)
    elif choice == "2":
        sorted_dict = dict(sorted(a_dict.items(), reverse=True))
        print("Descending order by key:", sorted_dict)
    elif choice == "3":
```

```
sorted_dict = dict(sorted(a_dict.items(), key=lambda item: item[1]))
print("Ascending order by value:", sorted_dict)
elif choice == "4":
    sorted_dict = dict(sorted(a_dict.items(), key=lambda item: item[1],
reverse=True))
    print("Descending order by value:", sorted_dict)
elif choice == "5":
    break
else:
    print("Invalid choice. Please enter a valid option.")

print("Program exited.")
```

Output:

```
Options:
1. Sort in ascending order by key
2. Sort in descending order by key
3. Sort in ascending order by value
4. Sort in descending order by value
5. Exit
Enter your choice: 1
Ascending order by key: {1: 2, 2: 3, 3: 1, 4: 0}
Options:
1. Sort in ascending order by key
2. Sort in descending order by key
3. Sort in ascending order by value
4. Sort in descending order by value
5. Exit
Enter your choice: 2
Descending order by key: {4: 0, 3: 1, 2: 3, 1: 2}
Options:
1. Sort in ascending order by key
2. Sort in descending order by key
3. Sort in ascending order by value
4. Sort in descending order by value
5. Exit
Enter your choice: 3
Ascending order by value: {4: 0, 3: 1, 1: 2, 2: 3}
Options:
1. Sort in ascending order by key
2. Sort in descending order by key
3. Sort in ascending order by value
4. Sort in descending order by value
5. Exit
Enter your choice: 5
Program exited.
```

Practical – 70

Write a program to create three dictionaries and concatenate them to create fourth dictionary.

Code:

```
students= {
    "Abhishek":19,
    "Bunty":18,
    "Manan":20,
    "Mayank":18
}

cricketers = {'Virat': 50, 'Rohit': 50, 'Yujvendra': 50, 'Brett': 50, 'Mitchell': 50, 'Chris': 50}

new_dict = {
    1:2,
    2:3,
    3:1,
    4:0
}

forth_dict = {**students,**cricketers,**new_dict}

print("Fourth dictionary after concatenating three dictionaries: ", forth_dict)
```

Output:

```
Fourth dictionary after concatenating three dictionaries: {'Abhishek': 19, 'Bunty': 18, 'Manan': 20, 'Mayank': 18, 'Virat': 50, 'Rohit': 50, 'Yujvendra': 50, 'Brett': 50, 'Mitchell': 50, 'Chris': 50, 1: 2, 2: 3, 3: 1, 4: 0}
```

Practical – 71

Write a program to check whether a dictionary is empty or not.

Code:

```
def isEmptyDict(a_dict):  
    if len(a_dict) <= 0:  
        return True  
    else:  
        return False  
  
new_dict = {  
    1:2,  
    2:3,  
    3:1,  
    4:0  
}  
  
empty_dict = dict()  
print("is new_dict empty? ", isEmptyDict(new_dict))  
print("is empty_dict empty? ", isEmptyDict(empty_dict))
```

Output:

```
is new_dict empty?  False  
is empty_dict empty?  True
```


Practical – 72

Suppose there are two dictionaries called boys and girls containing names as keys and ages as values. Write a program to merge the two dictionaries into a third dictionary.

Code:

Practical – 73

For the following dictionary, write a program to report the maximum and minimum salary.

Code:

```
employee = {  
    "Abhishek":1000000,  
    "Bunty":80000,  
    "Manan":90000,  
    "Rohit":95000  
}  
  
print("Maximum salary: ",max(employee.values()),"\nMinimum salary:  
",min(employee.values()))
```

Output:

```
Maximum salary: 1000000  
Minimum salary: 80000
```

Practical – 74

Suppose a dictionary contains roll numbers and names of students. Write a program to receive the roll number, extract the name corresponding to the roll number and display a message congratulating the by his name. If the roll number doesn't exist in the dictionary, the message should be 'Congratulations Student!'.

Code:

```
students= {  
    "10621019":"Abhishek",  
    "10621121":"Bunty",  
    "10621001":"Manan",  
    "10621991":"Mayank"  
}  
  
rollnum = input("Enter roll number: ")  
if rollnum in students.keys():  
    print("Congratulations ",students[rollnum],",!",sep="")  
else:  
    print("Congratulations Students!")
```

Output:

```
Enter roll number: 10621019  
Congratulations Abhishek!
```

Practical – 75

Write a program that reads a string from the keyboard and creates dictionary containing frequency of each character occurring in the string. Also print these occurrences in the form of a histogram.

Code:

```
import matplotlib.pyplot as plt

def letter_frequency(user_string):

    letters_dict = {letter:user_string.count(letter) for letter in user_string if
letter.isalpha()}

    # plotting the frequency table

    plt.bar(letters_dict.keys(), letters_dict.values())

    plt.xlabel('letters')

    plt.ylabel('occurance')

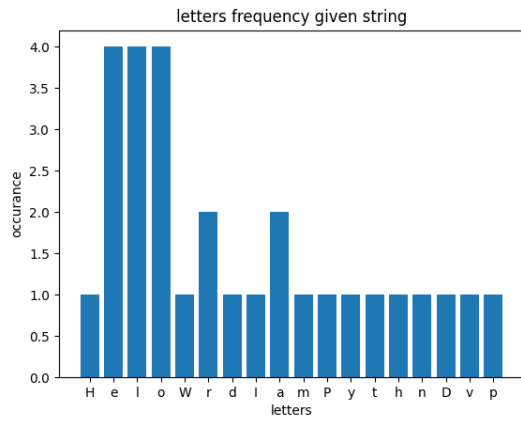
    plt.title('letters frequency given string')

    plt.show()

letter_frequency(input("Enter text: "))
```

Output:

```
Enter text: Hello,World! I am a Python Developer
```



Practical – 76

Create a dictionary containing names of students and marks obtained by them in three subjects. Write a program to replace the marks in three subjects with the total in three subjects, and average marks. Also report the topper of the class.

Code:

```
students = {  
    "abhishek":[97,98,99],  
    "amit":[78,90,67],  
    "abhinav":[67,45,90]  
}  
  
max_marks = 0  
  
name = ""  
  
for i in students.keys():  
    marks = sum(students[i])  
    if marks > max_marks:  
        max_marks = marks  
        name = i  
  
    students[i] = [marks, marks/3]  
  
print("Students performance:\n",students)  
  
print(f"Topper is: {name} with marks: {max_marks}")
```

Output:

```
Students performance:
{'abhishek': [294, 98.0], 'amit': [235, 78.33333333333333], 'abhinav': [202, 67.33333333333333]}
Topper is: abhishek with marks: 294
```

Practical – 77

Given the following dictionary:

```
portfolio = { 'accounts' : [ 'SBI', 'IOB'],  
              'shares' : ['HDFC', 'ICICI', 'TM', 'TCS'],  
              'ornaments' : ['10 gm gold', '1 kg silver']}
```

Write a program to perform the following operations:

- Add a key to portfolio called 'MF' with values 'Reliance' and 'ABSL'.
- Set the value of 'accounts' to a list containing 'Axis' and 'BOB'.
- Sort the items in the list stored under the 'shares' key.
- Delete the list stored under 'ornaments' key.

Code:

```
portfolio={  
    'accounts':['SBI','IOB'],  
    'shares':['HDFC','ICICI','TM','TCS'],  
    'ornaments':['10 gm gold','1 kg silver']  
}  
  
portfolio['MF']=['Reliance','ABSL']  
portfolio['accounts']=['Axis','BOB']  
portfolio['shares'] = sorted(portfolio['shares'])  
del portfolio['ornaments']  
  
print(portfolio)
```

Output:

```
{'accounts': ['Axis', 'BOB'], 'shares': ['HDFC', 'ICICI', 'TCS', 'TM'], 'MF': ['Reliance', 'ABSL']}
```


Practical – 78

Create two dictionaries-one containing grocery items and their prices and another containing grocery items and quantity purchased. By using the values from these two dictionaries compute the total bill.

Code:

```
grocery_prices = {  
    "onions":78.0,  
    "peas":20,  
    "palak":15,  
    "carrot":30  
}
```

```
grocery_qty = {  
    "onions":3,  
    "peas":2,  
    "palak":5,  
    "carrot":3  
}
```

```
bill=0
```

```
for i in grocery_prices.keys():
```

```
    bill += grocery_prices[i] * grocery_qty[i]
```

```
print("Total bill: Rs.", bill)
```

Output:

```
Total bill: Rs. 439.0
```

Practical – 79

Create a dictionary of 10 usernames and passwords. Receive the username and password from keyboard and search for them in the dictionary. Print appropriate message on the screen based on whether a match is found or not.

Code:

```
users={
    "user1":"helloWorld",
    "user2":"privieat",
    "user3":"isDollarGood",
    "user4":"password",
    "user5":"12345678",
    "user6":"qwerty",
    "user7":"09102003",
    "user8":"where_is_mongo",
    "user9":"hello",
    "user10":"yes_old+LEds1"
}

user = input("Enter username: ")
password = input("Enter password: ")

good = False
if user in users.keys():
    if users[user] == password:
        print("Welcome ", user)
```

```
good = not good
```

```
if not good:
```

```
    print("Bad Credentials. Try again!")
```

Ouput:

```
Enter username: user9
Enter password: hello
Welcome user9
```

Practical – 80

Given the following dictionary

```
marks = { 'Subu' : { 'Maths' : 88, 'Eng' : 60, 'SSt' : 95 },  
          'Amol' : { 'Maths' : 78, 'Eng' : 68, 'SSt' : 89 },  
          'Raka' : { 'Maths' : 56, 'Eng' : 66, 'SSt' : 77 } }
```

Write a program to perform the following operations:

- Print marks obtained by Amol in English.
- Set marks obtained by Raka in Maths to 77.
- Sort the dictionary by name.

Code:

```
marks = {  
'Subu' : { 'Maths' : 88, 'Eng' : 60, 'SSt' : 95 },  
'Amol' : { 'Maths' : 78, 'Eng' : 68, 'SSt' : 89 },  
'Raka' : { 'Maths' : 56, 'Eng' : 66, 'SSt' : 77 }  
}  
print('Marks obtained by Amol in english:', marks['Amol']['Eng'])  
marks['Raka']['Maths'] = '77'  
print(marks)  
marks = dict(sorted(marks.items( )))  
print(marks)
```

Output:

```
Marks obtained by Amol in english: 68
{'Subu': {'Maths': 88, 'Eng': 60, 'SSt': 95}, 'Amol': {'Maths': 78, 'Eng': 68, 'SSt': 89}, 'Raka': {'Maths': 77, 'Eng': 66, 'SSt': 77}}
{'Amol': {'Maths': 78, 'Eng': 68, 'SSt': 89}, 'Raka': {'Maths': 77, 'Eng': 66, 'SSt': 77}, 'Subu': {'Maths': 88, 'Eng': 60, 'SSt': 95}}
```

Practical – 81

Create a dictionary which stores the following data:

Interface	IP Address	Status
eth0	1.1.1.1	up
eth1	2.2.2.2	up
wlan0	3.3.3.3	down
wlan1	4.4.4.4	up

Write a program to perform the following operations:

- Find the status of a given interface.
- Find interface and IP of all interfaces which are up.
- Find the total number of interfaces.
- Add two new entries to the dictionary.

Code:

```
ifs = {  
'eth0': {'IP' : '1.1.1.1', 'Status' : 'up'},  
'eth1': {'IP' : '2.2.2.2', 'Status' : 'up'},  
'wlan0': {'IP' : '3.3.3.3', 'Status' : 'down'},  
'wlan1': {'IP' : '4.4.4.4', 'Status' : 'up'}  
}  
  
test = input('Enter interface: ')  
print(ifs[test]['Status'])  
  
for k, v in ifs.items() :  
    if v['Status'] == 'up' :  
        print(k, v['IP'])  
  
print('Total interfaces = ', len(ifs))  
  
ifs['eth2'] = {'IP' : '5.5.5.5', 'Status' : 'down'}  
ifs['wlan2'] = {'IP' : '6.6.6.6', 'Status' : 'up'}
```

```
for k, v in ifs.items() :
```

```
    print(k, v)
```

Output:

```
Enter interface: wlan0
down
eth0 1.1.1.1
eth1 2.2.2.2
wlan1 4.4.4.4
Total interfaces = 4
eth0 {'IP': '1.1.1.1', 'Status': 'up'}
eth1 {'IP': '2.2.2.2', 'Status': 'up'}
wlan0 {'IP': '3.3.3.3', 'Status': 'down'}
wlan1 {'IP': '4.4.4.4', 'Status': 'up'}
eth2 {'IP': '5.5.5.5', 'Status': 'down'}
wlan2 {'IP': '6.6.6.6', 'Status': 'up'}
```

Practical – 82

Suppose a dictionary contains 5 key-value pairs of name and marks. Write a program to print them from last pair to first pair. Keep deleting every pair printed, such that the end of printing the dictionary falls empty.

Code:

```
marks = { 'Subu' : 88, 'Amol' : 78, 'Raka' : 56, 'Dinesh' : 68, 'Ranjit' : 88 }
```

```
l = len(marks)-1
```

```
for i in range(l) :
```

```
    print(marks.popitem( ))
```

```
print(marks)
```

output:

```
('Ranjit', 88)
('Dinesh', 68)
('Raka', 56)
('Amol', 78)
{'Subu': 88}
```


Practical – 83

Given the following dictionary:

```
d = { 'd1': {'Fruitname' : 'Mango', 'Season' : 'Summer'}, 'd2': {'Fruitname' : 'Orange', 'Season' : 'Winter'}}
```

How will you access and print Mango and Winter?

Code:

```
d = { 'd1': {'Fruitname' : 'Mango', 'Season' : 'Summer'}, 'd2': {'Fruitname' : 'Orange', 'Season' : 'Winter'}}  
print(d['d1']['Fruitname'])  
print(d['d2']['Season'])
```

output:

```
Mango  
Winter
```

Practical – 84

Write a program where dataset is given as python dictionary, and program interacts with user and print output as per user need on basis of UserID.

Code:

```
relation = {  
    "U123": {  
        "rollnum": "10621019",  
        "name": "Abhishek Roka",  
        "city": "New Delhi"  
    },  
    "U124": {  
        "rollnum": "10621009",  
        "name": "Bunty",  
        "city": "Patna"  
    },  
    "U125": {  
        "rollnum": "10621897",  
        "name": "Manan",  
        "city": "NSP"  
    }  
}  
  
while True:
```

```

print("Options:")

print("1. Print data for a specific user")

print("2. Exit")

choice = input("Enter your choice: ")


if choice == "1":

    user_id = input("Enter user ID (e.g., U123, U124, U125): ")

    if user_id in relation:

        user_data = relation[user_id]

        print(f"User ID: {user_id}")

        print(f"Roll Number: {user_data['rollnum']}")

        print(f"Name: {user_data['name']}")

        print(f"City: {user_data['city']}")

    else:

        print("User ID not found in the dataset.")

elif choice == "2":

    break

else:

    print("Invalid choice. Please enter a valid option.")


print("Program exited.")

```

Output:

```
Options:
1. Print data for a specific user
2. Exit
Enter your choice: 1
Enter user ID (e.g., U123, U124, U125): U123
User ID: U123
Roll Number: 10621019
Name: Abhishek Roka
City: New Delhi
Options:
1. Print data for a specific user
2. Exit
Enter your choice: 2
Program exited.
```