

1)Read a directory path from user and list the contents of that directory sorted by file name.

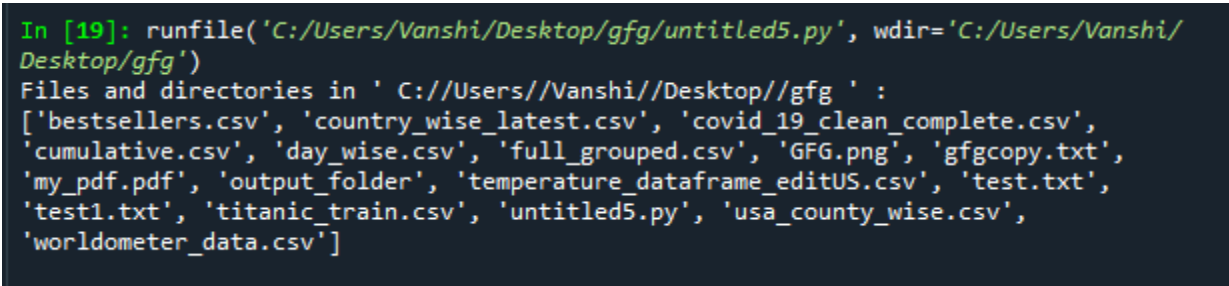
```
# import OS module
import os

# Get the list of all files and directories
path = "C://Users//Vanshi//Desktop//gfg"
dir_list = os.listdir(path)

print("Files and directories in '", path, "' :")

# prints all files
print(dir_list)
```

Output:->



```
In [19]: runfile('C:/Users/Vanshi/Desktop/gfg/untitled5.py', wdir='C:/Users/Vanshi/Desktop/gfg')
Files and directories in ' C://Users//Vanshi//Desktop//gfg ' :
['bestsellers.csv', 'country_wise_latest.csv', 'covid_19_clean_complete.csv',
'cumulative.csv', 'day_wise.csv', 'full_grouped.csv', 'GFG.png', 'gfgcopy.txt',
'my_pdf.pdf', 'output_folder', 'temperature_dataframe_editUS.csv', 'test.txt',
'test1.txt', 'titanic_train.csv', 'untitled5.py', 'usa_county_wise.csv',
'worldometer_data.csv']
```

2)Write the program to read and multiply two matrices of given size.

Code:->

```
# Program to multiply two matrices using nested loops

# take a 3x3 matrix
A = [[12, 7, 3],
      [4, 5, 6],
      [7, 8, 9]]

# take a 3x4 matrix
B = [[5, 8, 1, 2],
      [6, 7, 3, 0],
```

```
[4, 5, 9, 1]]
result = [[0, 0, 0, 0],
          [0, 0, 0, 0],
          [0, 0, 0, 0]]
# iterating by row of A
for i in range(len(A)):
    # iterating by column by B
    for j in range(len(B[0])):

        # iterating by rows of B
        for k in range(len(B)):
            result[i][j] += A[i][k] * B[k][j]

for r in result:
    print(r)
```

Output:->

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

3)Create a sample log file and demonstrate Rotating of files.

```
import logging # first of all import the module

logging.basicConfig(filename='std.log', filemode='w', format='%(name)s - %(levelname)s -
%(message)s')

logging.warning('This message will get logged on to a file')
```

Output:->

root - ERROR - This message will get logged on to a file

```
#importing the module
```

```
import logging
```

```
#now we will Create and configure logger
```

```
logging.basicConfig(filename="std.log",format='%(asctime)s %(message)s',filemode='w')
```

```
#Let us Create an object
```

```
logger=logging.getLogger()
```

```
#Now we are going to Set the threshold of logger to DEBUG
```

```
logger.setLevel(logging.DEBUG)
```

```
#some messages to test
```

```
logger.debug("This is just a harmless debug message")
```

```
logger.info("This is just an information for you")
```

```
logger.warning("OOPS!!!Its a Warning")
```

```
logger.error("Have you try to divide a number by zero")
```

```
logger.critical("The Internet is not working....")
```

Output:->

2020-06-19 12:48:00,449 - This is just harmless debug message

2020-06-19 12:48:00,449 - This is just an information for you

2020-06-19 12:48:00,449 - OOPS!!!Its a Warning

2020-06-19 12:48:00,449 - Have you try to divide a number by zero

2020-06-19 12:48:00,449 - The Internet is not working...

4)Write a function to print prime number in the given range n1 to n2 (use default 1 for parameter n1).

Python program to display all the prime numbers within an interval

```
lower = 900
```

```
upper = 1000
```

```
print("Prime numbers between", lower, "and", upper, "are:")
```

```
for num in range(lower, upper + 1):
```

```
    # all prime numbers are greater than 1
```

```
    if num > 1:
```

```
        for i in range(2, num):
```

```
            if (num % i) == 0:
```

```
                break
```

```
        else:
```

```
            print(num)
```

Output:->

Prime numbers between 900 and 1000 are:

907

911

919

929

937

941

947

953

967

971

977

983

991

997

5)Write the program to demonstrate lambda and filter.

```
# Python program to demonstrate
# lambda functions
string ='GeeksforGeeks'
# lambda returns a function object
print(lambda string : string)
```

Output

```
<function <lambda> at 0x7f65e6bbce18>
```

6)Demonstrate List and Dictionary with its important function (minimum 4).

```
# Python3 code to demonstrate working of
# Finding min value keys in dictionary
# Using min() + list comprehension + values()

# initializing dictionary
test_dict = {'Gfg' : 11, 'for' : 2, 'CS' : 11, 'geeks':8, 'nerd':2}

# printing original dictionary
print("The original dictionary is : " + str(test_dict))

# Using min() + list comprehension + values()
# Finding min value keys in dictionary
temp = min(test_dict.values())
```

```
res = [key for key in test_dict if test_dict[key] == temp]
```

```
# printing result
```

```
print("Keys with minimum values are : " + str(res))
```

Output:

The original dictionary is : {'nerd': 2, 'Gfg': 11, 'geeks': 8, 'CS': 11, 'for': 2}

Keys with minimum values are : ['nerd', 'for']

7)Write the program to demonstrate lambda and map.

```
sequences = [10,2,8,7,5,4,3,11,0, 1]
filtered_result = map (lambda x: x*x, sequences)
print(list(filtered_result))
```

Output:

```
[100, 4, 64, 49, 25, 16, 9, 121, 0, 1]
```

8)Write a function to find Sum of digits a given number which produce single number (e.g. 99721 = 1)

```
// C++ program to find sum of
```

```
// digits of a number until
```

```
// sum becomes single digit.
```

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int digSum(int n)
```

```
{
```

```
int sum = 0;

// Loop to do sum while
// sum is not less than
// or equal to 9
while(n > 0 || sum > 9)
{
    if(n == 0)
    {
        n = sum;
        sum = 0;
    }
    sum += n % 10;
    n /= 10;
}
return sum;
}

// Driver program to test the above function
int main()
{
    int n = 1234;
    cout << digSum(n);
    return 0;
}
```

Output :

1

9)Write a program to read list of n elements (Strings). Remove duplicate elements from list

Python code to remove duplicate elements

def Remove(duplicate):

 final_list = []

 for num in duplicate:

 if num not in final_list:

 final_list.append(num)

 return final_list

Driver Code

duplicate = [2, 4, 10, 20, 5, 2, 20, 4]

print(Remove(duplicate))

Output:

[2, 4, 10, 20, 5]

10)Demonstrate exception handling with minimum 3 types of exception

try:

 a=5

 b=0

 print (a/b)

except TypeError:

 print('Unsupported operation')

except ZeroDivisionError:

 print ('Division by zero not allowed')

print ('Out of try except blocks')

Output

Division by zero not allowed

Out of try except blocks

11) Create a dictionary by adding the key value pair from user. Check for duplicate before adding. Display the value of key given by user.

```
# Python code to demonstrate
# finding duplicate values from a dictionary

# initialising dictionary
ini_dict = {'a':1, 'b':2, 'c':3, 'd':2}

# printing initial_dictionary
print("initial_dictionary", str(ini_dict))

# finding duplicate values
# from dictionary
# using a naive approach
rev_dict = {}

for key, value in ini_dict.items():
    rev_dict.setdefault(value, set()).add(key)

result = [key for key, values in rev_dict.items() if len(values) > 1]

# printing result
print("duplicate values", str(result))
```

Output:

```
initial_dictionary {'c': 3, 'b': 2, 'd': 2, 'a': 1}
duplicate values [2]
```

```
# Python code to demonstrate
# finding duplicate values from dictionary
```

```
# initialising dictionary

ini_dict = {'a':1, 'b':2, 'c':3, 'd':2}

# printing initial_dictionary
print("initial_dictionary", str(ini_dict))
```

```
# finding duplicate values
# from dictionary using flip
flipped = {}

for key, value in ini_dict.items():
    if value not in flipped:
        flipped[value] = [key]
    else:
        flipped[value].append(key)

# printing result
print("final_dictionary", str(flipped))
```

Output:->

```
initial_dictionary {'a': 1, 'c': 3, 'd': 2, 'b': 2}
final_dictionary {1: ['a'], 2: ['d', 'b'], 3: ['c']}
```

```
# Python code to demonstrate
```

finding duplicate values from dictionary

```
from itertools import chain
```

```
# initialising dictionary
```

```
ini_dict = {'a':1, 'b':2, 'c':3, 'd':2}
```

```
# printing initial_dictionary
```

```
print("initial_dictionary", str(ini_dict))
```

```
# finding duplicate values
```

```
# from dictionary using set
rev_dict = {}
for key, value in ini_dict.items():
    rev_dict.setdefault(value, set()).add(key)

result = set(chain.from_iterable(
    values for key, values in rev_dict.items()
    if len(values) > 1))

# printing result
print("resultant key", str(result))
```

Output:->

```
initial_dictionary {'b': 2, 'd': 2, 'c': 3, 'a': 1}
resultant key {'d', 'b'}
```

12)Write and demonstrate program to read an integer and functions to check given number is Armstrong or not.

Python program to check if the number is an Armstrong number or not

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

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
```

```
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

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

Output 1

Enter a number: 663

663 is not an Armstrong number

Output 2

Enter a number: 407

407 is an Armstrong number

13)Write a program to read a filename along with its path and create it (directories and file) if it does not exists.

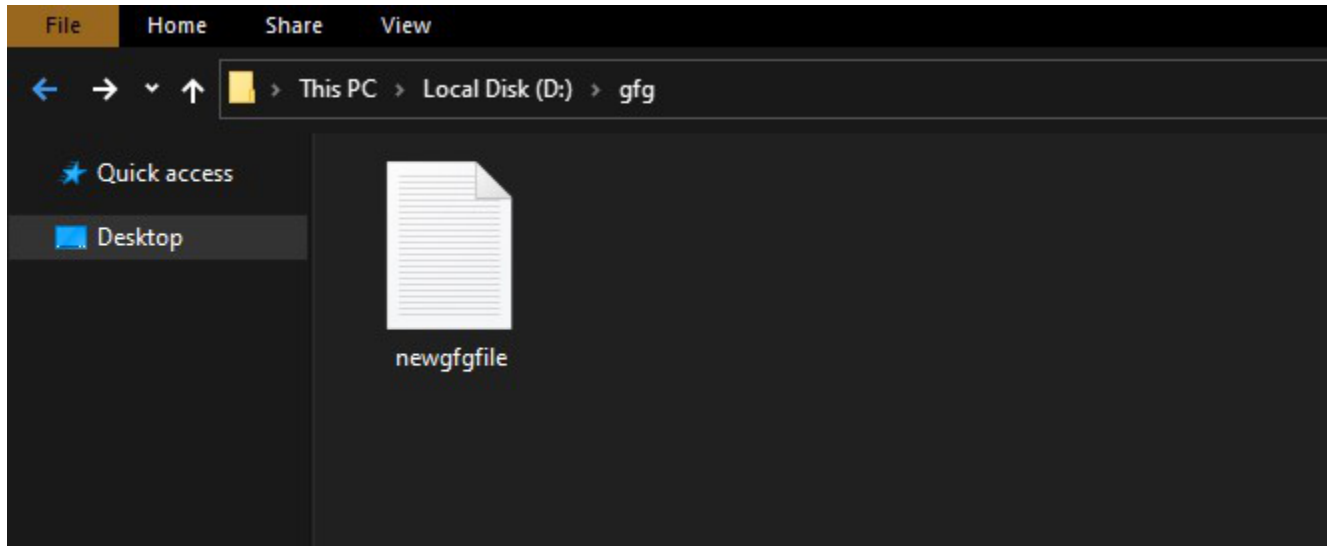
```
# path of this script
directory = "D:\gfg\\"

# get fileName from user
filepath = directory + input("Enter filename: ")

# Creates a new file
with open(filepath, 'w+') as fp:
    pass
```

Output:

Enter filename: newgfgfile.txt

**14) Write and demonstrate program to read an integer and functions to find factorial of a given number.**

```
# Python program to find the factorial of a number provided by the user.
```

```
# change the value for a different result
```

```
num = 7
```

```
# 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)
```

Output

The factorial of 7 is 5040

15)Read a file and findout- i)Total Number of lines and display all lines.

```
# Python program to count the  
# number of lines in a text file
```

```
# Opening a file  
file = open("gfg.txt", "r")  
Counter = 0
```

```
# Reading from file  
Content = file.read()  
CoList = Content.split("\n")
```

```
for i in CoList:  
    if i:  
        Counter += 1
```

```
print("This is the number of lines in the file")  
print(Counter)
```

Output:->

This is the number of lines in the file

4

ii)Total Number of words and display all words.

```
file = open("C:\data.txt", "rt")
```

```
data = file.read()
```

```
words = data.split()
```

```
print('Number of words in text file :', len(words))
```

Output:->

Number of words in text file : 14

iii)Separate and display each sentence from the file.

```
# Python program to read
```

```
# file word by word
```

```
# opening the text file
```

```
with open('GFG.txt','r') as file:
```

```
    # reading each line
```

```
    for line in file:
```

```
        # reading each word
```

```
        for word in line.split():
```

```
            # displaying the words
```

```
            print(word)
```

Output:->

Geeks

4

geeks

16)Write a program to demonstrate class, object, Inheritance

```
# A Python program to demonstrate inheritance
```

```
# Base or Super class. Note object in bracket.
```

```
# (Generally, object is made ancestor of all classes)
```

```
# In Python 3.x "class Person" is
```

```
# equivalent to "class Person(object)"
```

```
class Person(object):
```

```
    # Constructor
```

```
    def __init__(self, name):
        self.name = name
```

```
    # To get name
```

```
    def getName(self):
        return self.name
```

```
    # To check if this person is an employee
```

```
    def isEmployee(self):
        return False
```

```
# Inherited or Subclass (Note Person in bracket)
```

```
class Employee(Person):
```

```
    # Here we return true
```

```
    def isEmployee(self):
        return True
```

```
# Driver code
```

```
emp = Person("Geek1") # An Object of Person
```

```
print(emp.getName(), emp.isEmployee())
```

```
emp = Employee("Geek2") # An Object of Employee
```

```
print(emp.getName(), emp.isEmployee())
```


Output:

Geek1 False

Geek2 True

17) Write and demonstrate program to read an integer and functions to display first n terms of Fibonacci series

Program to display the Fibonacci sequence up to n-th term

```
nterms = int(input("How many terms? "))
```

first two terms

```
n1, n2 = 0, 1
```

```
count = 0
```

check if the number of terms is valid

```
if nterms <= 0:
```

```
    print("Please enter a positive integer")
```

if there is only one term, return n1

```
elif nterms == 1:
```

```
    print("Fibonacci sequence upto",nterms,":")
```

```
    print(n1)
```

generate fibonacci sequence

```
else:
```

```
    print("Fibonacci sequence:")
```

```
    while count < nterms:
```

```
        print(n1)
```

```
        nth = n1 + n2
```

```
        # update values
```

```
        n1 = n2
```

```
n2 = nth
```

```
count += 1
```

Output

How many terms? 7

Fibonacci sequence:

0

1

1

2

3

5

8