# //multilevel inheritance  
  
class parent1:  
 def \_\_init\_\_(self,name):  
 self.name=name  
 print("From Parent Class:- ",self.name)  
  
 def func1(self):  
 print("using parent1 class func1 method:- ",self.name)  
  
 def \_\_str\_\_(self):  
 return "Child1 class Details:- "+"\nName:- "+self.name  
  
  
class child1:  
 def \_\_init\_\_(self,name,age):  
 parent1.\_\_init\_\_(self,name)  
 self.name=name  
 self.age = age  
 print("From child1 Class:- ", self.name)  
  
 def func2(self):  
 parent1.func1(self)  
 print("from child1 class func2 method:- ", self.name)  
  
 def \_\_str\_\_(self):  
 return "Child1 class Details:- "+"\nName:- "+str(self.name)+"\nAge:- "+str(self.age)  
  
  
class grandchild1:  
 name=None  
 age=None  
 address=None  
 def \_\_init\_\_(self,name,age,address):  
 child1.\_\_init\_\_(self,name,age)  
 self.name = name  
 self.age = age  
 self.address = address  
 print("From grandchild1 Class:- ", self.name)  
  
 def func3(self):  
 child1.func2(self)  
 print("from grandchild1 class func3 method:- ", self.name)  
  
  
 def \_\_str\_\_(self):  
 return "Grand Child1 class Details:- "+"\nName:- "+str(self.name)+"\nAge:- "+str(self.age)+"\nAddress:- "+str(self.address)  
  
  
obj=grandchild1("mahesh",19,"VNK")  
obj.func3()  
print(obj)  
  
  
  
  
# Multiple Inheritance  
  
class parent1:  
 def \_\_init\_\_(self):  
 print("From parent 1 constructor......")  
 def display(self):  
 print("From parent 1 display method...")  
  
class parent2:  
 def \_\_init\_\_(self):  
 print("From parent 2 constructor......")  
 def display(self):  
 print("From parent 2 display Method...")  
  
class child(parent1,parent2):  
 def \_\_init\_\_(self):  
 print("Calling parent 1 constructor from child class...")  
 parent1().\_\_init\_\_()  
 print("Calling parent 2 constructor from child class...")  
 parent2().\_\_init\_\_()  
 print("From child constructor.........")  
  
 # super() method will call parent class constructor by default  
 print("calling parent class constructor with super method.........")  
 super().\_\_init\_\_()  
  
 def display(self):  
 print("Calling parent 1 display method from child class...")  
 parent1().display()  
 print("Calling parent 2 display method from child class...")  
 parent2().display()  
 print("From child display method.......")  
  
 # super method will calls parent1 class display method  
 print("calling parent class display method with super method.........")  
  
 super().display()  
  
c=child()  
c.display()  
  
  
  
# Single Inheritance  
  
print("\n\n Single Inheritance \n\n")  
  
class parent:  
 def \_\_init\_\_(self, fname, lname):  
 self.firstname = fname  
 self.lastname = lname  
  
 def printname(self):  
 print("My Name:- ",self.firstname, self.lastname)  
  
class child(parent):  
 def \_\_init\_\_(self, fname, lname, age):  
 super().\_\_init\_\_(fname, lname)  
 self.age = age  
  
 def display(self):  
 super().printname()  
 print("AGE:- ",self.age)  
  
  
  
p=parent("Surya","Mahesh")  
p.printname()  
  
c = child("Mike", "Olsen", 18)  
c.display()  
  
  
  
  
  
  
  
  
  
  
# class  
class parent:  
 x=10  
  
obj=parent()  
print(obj.x)  
  
  
# class with init and str methods  
class class2:  
 def \_\_init\_\_(self,a,b,c):  
 self.a=a  
 self.b=b  
 self.c=c  
  
 print(self.a)  
 print(self.b)  
 print(self.c)  
  
 def \_\_str\_\_(self):  
 return "A value: "+str(self.a)+"\nB value:- "+str(self.b)+"\nC value:- "+str(self.c)  
  
obj=class2(1,2,3)  
print(obj)  
  
  
# class with methods  
  
class person:  
  
 def \_\_init\_\_(self):  
 print("From Person Class Constructor....")  
  
  
 def \_\_str\_\_(self):  
 return "My Name:- "+self.name+"\nMy age:- "+str(self.age)  
  
  
 def getdata(self):  
 name=input("Enter Name:- ")  
 age=int(input("Enter Age:- "))  
 self.name=name  
 self.age=age  
  
 password = input("Enter password:- ")  
 self.\_password=password  
  
 secretnumber = input("Enter Secret Number:- ")  
 self.\_\_secretnumber = secretnumber  
  
  
  
 def display(self):  
 print("My Name:- "+self.name+"\nMy age:- "+str(self.age))  
  
  
 def printconfidentials(self):  
 print("Password:- "+self.\_password+"\nSecret Number:- "+self.\_\_secretnumber)  
  
  
obj=person()  
obj.getdata()  
print(obj)  
obj.display()  
  
obj.printconfidentials()  
  
  
  
class child(person):  
 def \_\_init\_\_(self):  
 print("Child class constructor called")  
 super().\_\_init\_\_()  
  
obj=child()  
obj.getdata()  
print(obj)  
obj.display()  
obj.printconfidentials()

Class , list of objects , composition

class dob:  
 def \_\_init\_\_(self,day,month,year):  
 self.day=day  
 self.month=month  
 self.year=year  
  
 def \_\_str\_\_(self):  
 return "Date Of Birth:- {}/{}/{}".format(self.day,self.month,self.year)  
  
class employee:  
  
 def \_\_init\_\_(self):  
 print("From Employee class")  
  
 def getdata(self):  
 self.name=input("Enter Employee Name:- ")  
 self.age=int(input("Enter Employee Agee:- "))  
 self.salary=int(input("Enter Employee Salary:- "))  
  
 print("Give Employee date of birth details:- ")  
 day=int(input("Enter Day of Birth:- "))  
 month=int(input("Enter Month of Birth:- "))  
 year=int(input("Enter Year of Birth:- "))  
 self.empdob=dob(day,month,year)  
  
 def display(self):  
 print("\nEmployee Details:- \n")  
 print(self.name)  
 print(self.age)  
 print(self.salary)  
 print(self.empdob)  
  
  
  
l=[]  
  
n=int(input("Enter Number Of Employees:- "))  
for i in range(n):  
 empobj=employee()  
 empobj.getdata()  
 l.append(empobj)  
  
  
for i in range(n):  
 l[i].display()

exceptions

try:  
 n=int(input("Enter Text to see Value Error/integer not to see errors:- "))  
 n = int(input("Enter 0 to see error:- "))  
 print(1/n)  
   
except ValueError as e:  
 print("Value Error Raised..")  
except ZeroDivisionError as e:  
 print("Zero Division Error Raised..")  
else:  
 print("No Errors..")  
finally:  
 print("From Finally Block..")  
  
  
  
class myexception(Exception):  
 def \_\_init\_\_(self,age):  
 self.age=age  
  
try:  
 n=int(input("Enter Age for voting (<18 to see user defined exception):- "))  
 if(n<18):  
 raise myexception(n)  
  
except myexception as e:  
 print("My Exception Raised")  
 print("You Age:- ",e.age)  
 print("Allowed Age >=18")  
  
else:  
 print("U are Allowed for voting..")  
finally:  
 print("Voting Done..")  
  
  
  
class invalidageexception(Exception):  
 def \_\_init\_\_(self,age):  
 self.age=age  
  
  
try:  
 n=int(input("Enter INValid Age For Human:- to see userdefined exception.:- "))  
  
 if(n<0 or n>100):  
 raise invalidageexception(n)  
 else:  
 pass  
except invalidageexception as e:  
 print("Age is not valid for humans ingeneral.. ")  
else:  
 print("Valid Age..")  
finally:  
 print("Finally..")

csv files

import csv  
  
with open("newfile.csv","w",newline="")as f:  
 csvwriter=csv.writer(f)  
 csvwriter.writerow(["name","age","proffession"])  
 l=[["SN", "Movie", "Protagonist"], [1, "Lord of the Rings", "Frodo Baggins"],  
 [2, "Harry Potter", "Harry Potter"]]  
 csvwriter.writerows(l)  
  
with open("newfile.csv","r")as f:  
 rows=csv.reader(f)  
 for row in rows:  
 print(row)  
  
with open("file2.csv","w",newline="")as f:  
 fields=["name","age","proffession"]  
 csvwriter=csv.DictWriter(f,fieldnames=fields)  
 csvwriter.writeheader()  
  
 csvwriter.writerow({"name": "aaa", "age": 10, "proffession": "aaa"})  
 csvwriter.writerow({"name": "aaa", "age": 10, "proffession": "aaa"})  
 csvwriter.writerow({"name": "aaa", "age": 10, "proffession": "aaa"})  
 csvwriter.writerow({"name": "aaa", "age": 10, "proffession": "aaa"})  
 csvwriter.writerow({"name": "aaa", "age": 10, "proffession": "aaa"})  
  
  
with open("file2.csv","r")as f:  
 rows=csv.DictReader(f)  
 for row in rows:  
 print(row)  
  
  
with open("temp.csv","w",newline="") as f:  
 writer=csv.writer(f)  
 writer.writerow(["Name","Age","Height"])  
  
 n=int(input("Enter Number of Students:- "))  
 l=[]  
 for i in range(n):  
 m=[]  
 name=input("enter name:- ")  
 m.append(name)  
 age=int(input("Enter age:- "))  
 m.append(age)  
 height=int(input("Enter Height:- "))  
 m.append(height)  
  
 l.append(m)  
  
 writer.writerows(l)  
  
with open("temp.csv","r") as f:  
 rows=csv.reader(f)  
 for row in rows:  
 print(row)  
  
with open("temp.csv","r") as f:  
 rows=csv.DictReader(f)  
 for row in rows:  
 print(row)

date,math

import math  
print(math.ceil(1.2))  
print(math.floor(1.1))  
print(math.fabs(-1.2))  
print(math.factorial(5))  
print(math.gcd(3,4))  
print(math.lcm(10,20))  
print(math.pow(2,3))  
print(math.sqrt(16))  
print(math.floor(1.1))  
  
from datetime import date  
  
today=date.today()  
print(today.year)  
print(today.month)  
print(today.day)

user defined modules

# mymodule.py  
  
def display:  
 print("Hello World..")  
  
  
from mymodule import \*  
display()

Random module

import random  
import math  
  
  
# 0.0-1.0  
x=random.random()  
print(x)  
  
# <10  
print(math.floor(x\*10))  
  
x=random.randint(1,100)  
print(x)  
  
  
x=random.randrange(1,100)  
print(x)  
  
x=random.randrange(1,10,1)  
print(x)  
  
  
x=random.randrange(1,10,3)  
print(x)  
  
print(random.choice("hello world"))  
  
print(random.choice([1,2,3,4,5,6,7,8,9]))  
print(random.choice((1,2,3,4,5,6,7,8,9)))  
  
  
l=[1,2,3,4,5,6,7,8,9]  
print(l)  
random.shuffle(l)  
print(l)

LISTS

l=[1,2,3,4,5,6,7,8,9]  
  
print(l)  
print(l[::-1])  
  
m=[1,2,3,4,5]  
l.append(m)  
  
l.append("hello")  
  
l.extend(m)  
print(l)  
  
l.insert(3,"pushpa")  
print(l)  
  
l.remove(1)  
print(l)  
  
  
l.pop()  
print(l)  
  
l.pop(3)  
print(l)  
  
print(l.index('pushpa'))  
  
  
l=[i for i in range(20) if i%2==0]  
print(l)  
  
m=[i for i in range(20) if i%2!=0]  
print(m)  
  
print(l+m)  
  
l.extend(m)  
print(l)

list of lists

student records

singleRow=[]  
studList=[]  
choice='y'  
while choice.lower()!='n':  
 studName=input('enter your name')  
 rlno=input('enter your rlno')  
 elective=input('elective :')  
 singleRow.append(studName)  
 singleRow.append(rlno)  
 singleRow.append(elective)  
 studList.append(singleRow)  
 singleRow=[]  
 choice=input('enter your choice [y/n]')  
   
   
# remove existing record  
print(studList)  
item=input("enter student's name:")  
for record in studList:  
 if item in record:  
 studList.remove(record)  
print(studList)

sort words in list

def convert(s):  
 l=s.split('-')  
 l.sort()  
 s='-'.join(l)  
 return s  
  
s=input("Enter A senntence :- ")  
print("Inputed string:- ")  
print(s)  
s=convert(s)  
print("String after sorting the words:- ")  
print(s)

Functions

# abs(), all(), any(), ascii(), bin(), bool(), bytearray(), bytes(), chr()  
# dict(), dir(), enumerate(), eval() , exec(), filter(), float(), format()  
# help(), hex(), id(), input(), int(), iter(), len(), list(), max(), min()  
# next(), object(), oct(), open(), ord(), pow(), print(), property()  
# range(), repr(), reversed(), round(), set(), slice(), sorted(), str()  
# sum(), super(), tuple(), type(), vars(), zip()  
  
  
# abs() returns absolute value of the number  
x=1.5  
print(abs(x))  
x=-1.7  
print(abs(x))  
x=3+5j  
print(abs(x))  
x=-3-5j  
print(abs(x))  
  
# all() checks whether all the elements of a list/set/tuple are True  
  
l=[True,True,True]  
print(all(l))  
  
l=[True,True,False]  
print(all(l))  
  
l=(True,True,True)  
print(all(l))  
  
l=(True,True,False)  
print(all(l))  
  
l={True,True,False,1,0}  
print(all(l))  
  
  
l={True,True,'surya',1,100}  
print(all(l))  
  
l={1:"surya",11:110,1:1.5}  
print(all(l))  
  
l={0:"surya",11:110,1:1.5}  
print(all(l))  
  
# any() checks whether any of the elements of a list/set/tuple are True  
  
l=[True,True,True]  
print(any(l))  
  
l=[True,True,False]  
print(any(l))  
  
l=(True,True,True)  
print(any(l))  
  
l=(True,True,False)  
print(any(l))  
  
l={True,True,False,1,0}  
print(any(l))  
  
  
l={True,True,'surya',1,100}  
print(any(l))  
  
l={1:"surya",11:110,1:1.5}  
print(any(l))  
  
l={0:"surya",11:110,1:1.5}  
print(any(l))  
  
  
# ascii is escapes non ascii characters  
s="sa~"  
print(ascii(s))  
  
x = ascii("S t å le")  
print(x)  
  
x=x.split(" ")  
print(ascii(x))  
  
  
  
# bin() returns binary vesrion of number  
  
x=24  
print(x)  
print(bin(x))  
  
# bool  
  
print(bool(1))  
print(bool(0))  
print(bool(True))  
print(bool("surya"))  
l=[1,0,3,5,2,2]  
print(bool(l))  
l=(1,0,3,5,2,2)  
print(bool(l))  
l=(1,0,3,5,2,2)  
print(bool(l))  
  
  
x = bytearray(4)  
print(x)  
  
x = bytes(4)  
print(x)  
  
  
def x():  
 a = 5  
  
print(callable(x))  
  
x = 5  
  
print(callable(x))  
  
x = chr(97)  
print(x)  
  
x = compile('print(55)', 'test', 'eval')  
exec(x)  
x = compile('print(55)\nprint(88)', 'test', 'exec')  
exec(x)  
  
print(complex(1))  
print(complex(1.4))  
  
  
class person:  
 age=19  
 name="surya"  
 sex="m"  
  
delattr(person,'age')  
  
d=dict(name="surya",age="20")  
print(d)  
  
d=dict(name="surya",age=10)  
print(d)  
  
x=divmod(4,2)  
print(x)  
  
x=divmod(7,2)  
print(x)  
  
x=divmod(11,5)  
print(x)  
  
class Person:  
 name = "John"  
 age = 36  
 country = "Norway"  
  
print(dir(Person))  
  
x = ('apple', 'banana', 'cherry')  
y = enumerate(x)  
print(list(y))  
  
x = ['apple', 'banana', 'cherry']  
y = enumerate(x)  
print(list(y))  
  
x = {'apple', 'banana', 'cherry'}  
y = enumerate(x)  
print(list(y))  
  
x="print('hello world')"  
print(eval(x))  
  
x = 'print(55)'  
eval(x)  
  
x = 'name = "John"\nprint(name)'  
exec(x)  
  
x = float(3)  
print(x)  
x = float("233")  
print(x)  
  
  
ages = [5, 12, 17, 18, 24, 32]  
  
def myFunc(x):  
 if x < 18:  
 return False  
 else:  
 return True  
  
adults = filter(myFunc, ages)  
  
for x in adults:  
 print(x)  
  
x = format(0.5, '%')  
print(x)  
x = format(255, 'x')  
print(x)  
  
mylist = ['apple', 'banana', 'cherry']  
x = frozenset(mylist)  
print(x)  
  
print(id(x))  
  
class Person:  
 name = "John"  
 age = 36  
 country = "Norway"  
  
x = getattr(Person, 'age')  
print(x)  
  
a = ("John", "Charles", "Mike")  
b = ("Jenny", "Christy", "Monica", "Vicky")  
  
x = zip(a, b)  
  
#use the tuple() function to display a readable version of the result:  
  
print(tuple(x))  
  
a = ("John", "Charles", "Mike")  
b = ("Jenny", "Christy", "Monica")  
  
x = zip(a, b)  
  
print(tuple(x))  
  
a = ('apple', 'banana', 'cherry')  
b = "Hello World"  
c = 33  
  
x = type(a)  
y = type(b)  
z = type(c)  
x = tuple(('apple', 'banana', 'cherry'))  
print(x)  
  
a = (1, 2, 3, 4, 5)  
x = sum(a)  
print(x)  
  
a = (1, 2, 3, 4, 5)  
x = sum(a, 7)  
print(x)  
  
x = str(3.5)  
print(x)  
print(type(x))  
  
a = ("b", "g", "a", "d", "f", "c", "h", "e")  
x = sorted(a)  
print(x)  
print(a)  
  
a = (1, 11, 2)  
x = sorted(a)  
print(x)  
print(a)  
  
# Create a tuple and a slice object. Use the slice object to get only the two first items of the tuple:  
a = ("a", "b", "c", "d", "e", "f", "g", "h")  
  
x = slice(2)  
  
print(a[x])  
  
# Create a tuple and a slice object. Start the slice object at position 3, and slice to position 5, and return the result:  
  
a = ("a", "b", "c", "d", "e", "f", "g", "h")  
x = slice(3, 5)  
print(a[x])  
  
# Create a tuple and a slice object. Use the step parameter to return every third item:  
  
a = ("a", "b", "c", "d", "e", "f", "g", "h")  
x = slice(0, 8, 3)  
print(a[x])  
  
x = set(('apple', 'banana', 'cherry'))  
print(x)  
  
x = round(5.76543)  
print(x)  
  
x = round(5.76543, 2)  
print(x)  
  
x = round(5.76543, 3)  
print(x)  
  
alph = ["a", "b", "c", "d"]  
for x in alph:  
 print(x,end="")  
print()  
ralph = reversed(alph)  
for x in ralph:  
 print(x,end="")  
print()  
  
x = range(6)  
for n in x:  
 print(n)  
print(type(x))  
  
x = range(3, 20)  
for n in x:  
 print(n)  
  
x = range(3, 20, 2)  
for n in x:  
 print(n)  
  
x = pow(4, 3)  
print(x)  
  
# Return the value of 4 to the power of 3, modulus 5 (same as (4 \* 4 \* 4) % 5):  
  
x = pow(4, 3, 5)  
print(x)  
  
  
x = ord("h")  
print(x)  
  
print(chr(x))  
  
mylist = iter(["apple", "banana", "cherry"])  
x = next(mylist)  
print(x)  
x = next(mylist)  
print(x)  
x = next(mylist)  
print(x)  
  
  
mylist = iter(["apple", "banana", "cherry"])  
x = next(mylist, "orange")  
print(x)  
x = next(mylist, "orange")  
print(x)  
x = next(mylist, "orange")  
print(x)  
x = next(mylist, "orange")  
print(x)  
  
  
x = min(5, 10,1828)  
print(x)  
  
x = max(5, 10,10,109,287)  
print(x)  
  
  
def myfunc(n):  
 return len(n)  
  
x = map(myfunc, ('apple', 'banana', 'cherry'))  
print(x)  
  
  
mylist = ["apple", "banana", "cherry"]  
x = len(mylist)  
  
  
x = int(3.5)  
print(x)  
  
x = int(3)  
print(x)  
  
x = int("3")  
print(x)  
  
x = hex(255)  
print(x)  
  
  
print('Enter your name:')  
x = input()  
print('Hello, ' + x)

String functions

# String Function Sample Program  
str1 = "This is a test string"  
str2 = " Next string "  
  
print("str1 is \"This is a test string\"")  
  
print("isaplha()")  
if str1.isalpha():  
 print("True")  
else:  
 print("False")  
  
print("islower()")  
if str1.islower():  
 print("True")  
else:  
 print("False")  
  
print("isupper()")  
if str1.isupper():  
 print("True")  
else:  
 print("False")  
  
print("isdigit()")  
if str1.isdigit():  
 print("True")  
else:  
 print("False")  
  
print("startswith('is')")  
if str1.startswith("is"):  
 print("True")  
else:  
 print("False")  
  
print("endsswith(\"string\")")  
if str1.startswith("string"):  
 print("True")  
else:  
 print("False")  
  
print("lower()")  
print(str1.lower())  
  
print("lstrip()")  
print(str2)  
print(str2.lstrip())  
  
print("find()")  
print(str1.find("is", 4))  
  
print("find()")  
print(str1.find("is"))  
  
print("find()")  
print(str1.find("is", 4))  
  
print("replace()")  
print(str1.replace("a", "the"))  
  
print("replace()")  
print(str1.replace("is", "at", 1))  
  
print("split()")  
print(str1.split())  
  
x = ["SASTRA", "University"]  
y = " "

File I/O

lst=["SASTRA", "University","Tanjore"]  
Ofile=open("SampTxtFile.Txt","w")  
for item in lst:  
 Ofile.write(item+"\n")  
Ofile.close()  
  
Out\_lst=[]  
with open("SampTxtFile.Txt","r") as Ifile:  
 for item in Ifile:  
 Out\_lst.append(item)  
  
print(Out\_lst)  
  
  
l=[]  
with open("SampTxtFile.Txt","r") as f:  
 rows=f.readlines()  
 for row in rows:  
 l.append(row)  
  
print(l)  
  
  
  
with open("SampTxtFile.Txt","r") as f:  
 rows=f.read()  
 print(rows)  
  
  
with open("SampTxtFile.Txt","r") as f:  
 rows=f.readlines()  
 for row in rows:  
 row.replace("\n","")  
 print(row)

NIGHT PREPARATION

import csv  
import datetime  
  
with open("mycsvfile.csv","w",newline="")as file:  
 csvwriter=csv.writer(file)  
 csvwriter.writerow(['aaa','bbb','ccc'])  
 for i in range(5):  
 csvwriter.writerow([i,i,i])  
  
  
with open('mycsvfile.csv','r')as file:  
 rows=csv.reader(file)  
 for row in rows:  
 print(row)  
  
  
  
with open('myfile.txt',"w") as file:  
 file.write("hello\n")  
 for i in range(5):  
 file.write("hello {}".format(i+1)+"\n")  
  
with open('myfile.txt','r')as file:  
 rows=file.readlines()  
 for row in rows:  
 print(row)  
  
  
  
class myexception(Exception):  
 pass  
  
try:  
 n=10  
 if(n<18):  
 raise myexception  
except myexception as e:  
 print("Not Allowed For VOTING...")  
  
  
  
import random  
  
x=random.random()  
print(x)  
  
x=random.randint(1,10)  
print(x)  
  
x=random.randrange(1,10,2)  
print(x)  
  
  
  
from datetime import date,time,datetime  
  
today=date.today()  
print(today)  
  
print(today.year)  
print(today.month)  
print(today.day)  
  
  
t=datetime.now()  
print(t)  
  
print(t.hour)  
print(t.minute)  
print(t.second)  
  
print(t.year)  
print(t.month)  
print(t.day)  
  
  
t=(1,2,3)  
  
t=t+(1,)  
print(t)  
  
  
l=[1,2,3,4,5,4,3,2,1]  
print(l)  
  
l.append(6)  
  
print(l.count(1))  
print(l.index(1))  
print(l.reverse())  
l.sort()  
print(l)  
  
m=sorted(l)  
print(m)  
  
l=l[::-1]  
print(l)  
  
s="hello"  
print(s)  
ss=s[::-1]  
print(ss)  
  
  
s='abcde'  
l=list(s)  
print(l)  
  
s=str(l)  
print(s)  
  
s='abc,water,hello'  
print(s.split(','))

import csv  
import datetime  
  
with open("mycsvfile.csv","w",newline="")as file:  
 csvwriter=csv.writer(file)  
 csvwriter.writerow(['aaa','bbb','ccc'])  
 for i in range(5):  
 csvwriter.writerow([i,i,i])  
  
  
with open('mycsvfile.csv','r')as file:  
 rows=csv.reader(file)  
 for row in rows:  
 print(row)  
  
  
  
with open('myfile.txt',"w") as file:  
 file.write("hello\n")  
 for i in range(5):  
 file.write("hello {}".format(i+1)+"\n")  
  
with open('myfile.txt','r')as file:  
 rows=file.readlines()  
 for row in rows:  
 print(row)  
  
  
  
class myexception(Exception):  
 pass  
  
try:  
 n=10  
 if(n<18):  
 raise myexception  
except myexception as e:  
 print("Not Allowed For VOTING...")  
  
  
  
import random  
  
x=random.random()  
print(x)  
  
x=random.randint(1,10)  
print(x)  
  
x=random.randrange(1,10,2)  
print(x)  
  
  
  
from datetime import date,time,datetime  
  
today=date.today()  
print(today)  
  
print(today.year)  
print(today.month)  
print(today.day)  
  
  
t=datetime.now()  
print(t)  
  
print(t.hour)  
print(t.minute)  
print(t.second)  
  
print(t.year)  
print(t.month)  
print(t.day)  
  
  
t=(1,2,3)  
  
t=t+(1,)  
print(t)  
  
  
l=[1,2,3,4,5,4,3,2,1]  
print(l)  
  
l.append(6)  
  
print(l.count(1))  
print(l.index(1))  
print(l.reverse())  
l.sort()  
print(l)  
  
m=sorted(l)  
print(m)  
  
l=l[::-1]  
print(l)  
  
s="hello"  
print(s)  
ss=s[::-1]  
print(ss)  
  
  
s='abcde'  
l=list(s)  
print(l)  
  
s=str(l)  
print(s)  
  
s='abc,water,hello'  
print(s.split(','))

PANDAS

import pandas as pd  
  
# d=pd.read\_csv("F:\\pythonPractice\\dataset.csv")  
# df=pd.DataFrame(d)  
# print(df)  
  
df = pd.DataFrame({'points': [25, 12, 15, 14],'assists': [5, 7, 13, 12]})  
print(df)  
  
print("\n\n")  
  
data={'NAME':["mahesh","surya","pemma"],'rollnum':[1,2,3],"perc":[100,90,89]}  
df=pd.DataFrame(data)  
print(df)  
  
print("\n\n")  
  
  
l=[("name","mahesh","surya","pemma"),("age",10,20,30),("perc",100,90,89),("marks",100,90,89),("avg",100,90,89)]  
df=pd.DataFrame(l)  
print(df)  
print("\n\n")  
  
  
df=pd.DataFrame({'NAME':["mahesh","surya","pemma","aaa","bbb","ccc","ddd"],'rollnum':[1,2,3,4,5,6,7],"perc":[100,90,89,100,100,100,100],"age":[10,20,30,40,50,60,70],"avg":[1,2,3,4,5,6,7],"rank":[10,20,30,40,50,60,70]})  
print(df)  
  
print("\n\n")  
  
  
print(df.head())  
print("\n\n")  
  
  
print(df.head(1))  
print("\n\n")  
  
print(df.tail())  
print("\n\n")  
  
print(df.tail(1))  
print("\n\n")  
  
print(df.describe())  
print("\n\n")  
  
print(df.shape)  
print("\n\n")  
  
print(df[0:5:1])  
print("\n\n")  
  
print(df[::2])  
print("\n\n")  
  
print(df[::-1])  
print("\n\n")  
  
print(df[["NAME","age"]])  
print("\n\n")  
  
  
print(df[["NAME","age"]][::-1])  
print("\n\n")  
  
print(df[["NAME","age"]][::2])  
print("\n\n")  
  
for rec in df.iterrows():  
 print(rec)  
 print("\n")  
  
print("\n\n")  
  
# access by row\_number in loc  
print(df.loc[0])  
print("\n\n")  
  
print(df.loc[0,["NAME","age"]])  
print("\n\n")  
  
print(df.loc[0:5])  
print("\n\n")  
  
  
print(df.loc[0])  
print("\n\n")  
  
  
print(df.iloc[0,[0]])  
print("\n\n")  
  
  
print(df.iloc[[1,3,5 ]])  
print("\n\n")  
  
print(df.iloc[0:5,[1,3,5 ]])  
print("\n\n")  
  
print(df['perc'])  
  
print(df[df['perc']<100])  
print("\n\n")  
  
  
print(df)  
print("\n\n")  
  
print(df.sort\_values("perc"))  
print("\n\n")  
  
print(df.sort\_values("perc",ascending=False))  
print("\n\n")  
  
  
print(df.sort\_values("age",ascending=True))  
print("\n\n")  
  
print(df.sort\_values("age",ascending=False))  
print("\n\n")  
  
# sort by two columns  
print(df.sort\_values(["age","perc"]))  
print("\n\n")  
  
print(df.sort\_values(["age","perc"],ascending=[1,1]))  
print("\n\n")  
  
df["sex"]="M"  
print(df)  
print("\n\n")  
  
df["grade"]=df["perc"]  
print(df)  
print("\n\n")  
  
df["result"]="PASS/FAIL"  
print(df)  
print("\n\n")  
  
df.drop(columns="result")  
print(df)  
print("\n\n")  
  
print(df.duplicated())

NUMPY

import numpy as np  
  
a=np.array([2,4,6])  
b=np.array([1,2,3])  
print(a)  
print(b)  
print(a+b)  
print(a-b)  
print(a\*b)  
print(a/b)  
  
  
a=np.array([1,2,3])  
print(a)  
  
b=np.array([[1,2,3],[4,5,6]])  
print(b)  
  
print(a.ndim)  
  
print(a.shape)  
  
print(a.size)  
  
print(a.dtype)  
  
print(a.itemsize)  
print(a.nbytes)  
  
a=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])  
print(a)  
  
print(a[0,5])  
print(a[1,5])  
  
print(a[0,])  
print(a[1,])  
  
print(a[:,2])  
  
print(a[0,2:7])  
  
a[1,5]=100  
print(a)  
  
  
u=np.array([1,2])  
v=np.array([3,2])  
print(u\*v)  
  
print(np.dot(u,v))  
  
print(a)  
print(a.shape)  
print(a.size)  
print(a.ndim)  
  
  
  
X=np.array([[1,0],[0,1]])  
Y=np.array([[2,2],[2,2]])  
Z=np.dot(X,Y)  
  
print(Z)

COURSERA FINAL QUIZ

# [0:2] is putting the stop sign at index 2, which is the third element. Therefore, the elements before the stop sign are returned.  
  
var='01234567'  
print(var[::2])  
  
name="EMILY"  
print(name.find('L'))  
  
print(type(1.0))  
  
print(float(3))  
  
print(7.2//2)  
  
# This shows you that the // operator rounds down the result of the division of two numbers to the nearest whole number. Even if the decimal point is 9, the // operator would still round the result down to the nearest whole number.  
  
  
print(1/2)  
print(2//3)  
  
# Question 8  
# How many identical keys can a dictionary have  
#  
# No, each key in a dictionary should be unique. You can't have two keys with the same value.  
# Attempting to use the same key again will just overwrite the previous value stored.  
# If a key needs to store multiple values, then the value associated with the key should be a list or another dictionary.  
#  
  
  
  
# what is tuple  
#  
# Tuples are used to store multiple items in a single variable.  
# Tuple is one of 4 built-in data types in Python used to store  
# collections of data, the other 3 are List, Set, and Dictionary,  
# all with different qualities and usage. A tuple is a collection which is ordered and unchangeable.  
#  
  
  
  
l=["abc","def","ghi"]  
  
# .  
# Question 10  
# What does the split() method return from a list of words?  
# The list of words in a string separated by a delimiter  
  
  
  
# Lists are mutable tuples are not  
  
import numpy as np  
a=np.array([1,1,1,1,1])  
print(a+1)  
  
a=np.array([0,1,0,1,0])  
b=np.array([1,0,1,0,1])  
print(a\*b)  
print(a-b)  
print(a+b)  
  
# The sorted() function returns a sorted list of the specified iterable object. You can specify ascending or descending order.  
  
x="7"  
if(x!=1):  
 print('How are you?')  
else:  
 print('Hi')  
  
  
def add(x):  
 return (x + x)  
  
print(add(1))  
  
# You can specify ascending or descending order. Strings are sorted alphabetically, and numbers are sorted numerically. Note: You cannot sort a list that contains BOTH string values AND numeric values.

Prime numbers between a range

a= int(input("Enter a:- "))  
b=int(input("Enter b:- "))  
  
  
flag=0  
  
while(a<b):  
 flag=1  
 i=2  
 while(i<a):  
 if(a%i==0):  
 flag=0  
 break  
 else:  
 i=i+1  
 if(flag==1):  
 print(a)  
  
 a=a+1

PERMUTATIONS OF A STRING

vector<string>find\_permutation(string str)

{

vector<string>v;

sort(str.begin(),str.end());

do

{

v.push\_back(str);

}while(next\_permutation(str.begin(),str.end()));

return v;

}