Simple Interest

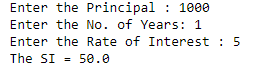
p = int(input("Enter the Principal : "))

n = float(input("Enter the No. of Years: "))

r = float(input("Enter the Rate of Interest : "))

print("The SI = {}".format(p\*n\*r/100))

Output:



IncomeTaxCalculator

def calculateIncomeTax(income):

tax = 0;

if income <= 250000: #2 Lakh 50 thousand

tax = 0

elif income <= 500000: #5 Lakh

tax = (income - 250000) \* 0.05

elif income <= 750000: #7 lakh 50 thousand

tax = (income - 500000) \* 0.10 + 12500

elif income <= 1000000: #10 Lakh

tax = (income - 750000) \* 0.15 + 37500

elif income <= 1250000: #12 lakh 50 thousand

tax = (income - 1000000) \* 0.20 + 75000

elif income <= 1500000: #15 lakh

tax = (income - 1250000) \* 0.25 + 125000

else:

tax = (income - 1500000) \* 0.30 + 187500

print("you owe", tax, "Rupees in tax!")

calculateIncomeTax(5000000)

Output:



Multiplication Tables

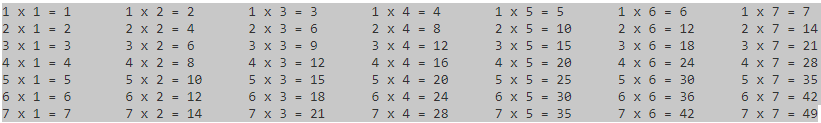
for i in range(1,8):

print(" ")

for j in range (1,8):

print(i,"x",j,"=",(i\*j),end='\t')

Output:



NumberConverter - Arrays

from array import array

a = array ('i', range(0,9))

x = int (input("enter a 3 digit number"))

a [2] = int(x%10)

a [1] = int(x/10%10)

a [0] = int(x/100)

for i in range (0,3):

if a [i] == 0:

print("zero",end=' ')

elif a [i] == 1 :

print("one",end=' ')

elif a [i] == 2 :

print("two",end=' ')

elif a [i] == 3 :

print("three",end=' ')

elif a [i] == 4 :

print("four",end=' ')

elif a [i] == 5 :

print("five",end=' ')

elif a [i] == 6 :

print("six",end=' ')

elif a [i] == 7 :

print("seven",end=' ')

elif a [i] == 8 :

print("eight",end=' ')

else:

print("nine",end=' ')

Output:



NumberToTextConverter – User Defined Functions

Words={0:"zero",1:"one",2:"two",3:"three",4:"four",5:"five",6:"six",7:"seven",8:"eight",9:"nine",10:"ten"}

def word\_converter(value):

try:

if(isinstance(value,int)):

print(words[value])

elif(isinstance(value,str)):

for key, val in words.items():

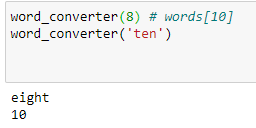
if(val == value):

print(key)

except:

print("Some error occurred while executing the code....... :(")

Output:



StudentInformation - lists

def getStudentDetails():

b = []

dno = input("Enter your Department Number: ")

name = input("Enter your Name:")

dob = input("Enter Date of Birth: ")

fname = input("Enter Father's name: ")

score = int(input("Enter your score: "))

b.extend([dno,name,dob,fname,score])

return b

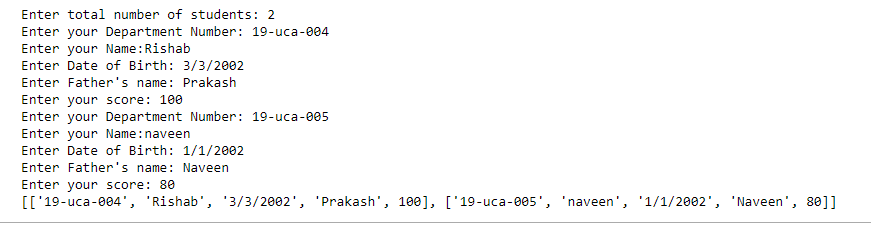
a=[]

n = int(input("Enter total number of students: "))

for i in range(0,n):

a.append(getStudentDetails())

print(a)



Student Information - Dictionary

dno={1:"19-UCA-013",2:"19-UCA-014",3:"19-UCA-015"}

name={1:"Carmel",2:"Francis",3:"Divagar"}

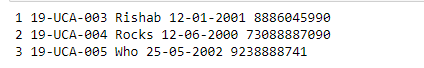
dob={1:"12-01-2001",2:"12-06-2000",3:"25-05-2002"}

mobile={1:"8056045990",2:"7305487090",3:"9234568741"}

for key in dno:

print(key, dno[key],name[key],dob[key],mobile[key])

Output:



StandardDeviation

import math

n = int(input("Enter the number of Values: "))

weight = []

s = 0

for i in range(n):

w=0

weight.append(int(input("Enter the weight")))

s+=w

print(weight)

average = int(s/n)

d =0

for i in weight:

d += (i - average)\*\*2

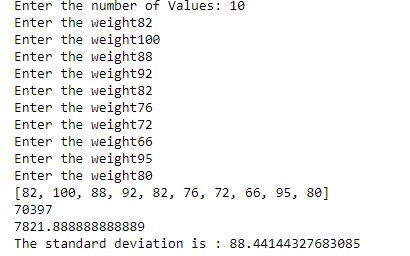
print(d)

d = d/(n-1)

print(d)

sd = math.sqrt(d)

print("The standard deviation is : {}".format(sd))



Stack Implementation – Modules

stack = []

while(True):

x = int(input("Enter your choice: [1]-Insert [2]-Delete [3]-Display"))

if(x==1):

v = int(input("Enter any integer value: "))

stack.append(v)

print(stack)

elif(x==3):

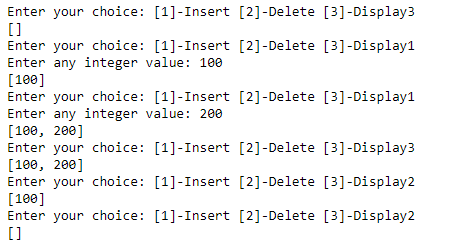
print(stack)

elif x ==2 :

stack.pop()

print(stack)

Output:



Age Calculator

def ageCalculator(y, m, d):

import datetime

today = datetime.datetime.now().date()

dob = datetime.date(y, m, d)

age = int((today-dob).days / 365.25)

print(age ,"YEARS" )

ageCalculator(1988, 9, 3)

Output:



Person\_Student\_Professor – Inheritance

class Parent:

def myfun\_p(self):

print("I'm the Parent")

class Child(Parent):

def myfun\_c(self):

print("I'm the Child")

obj = Child()

obj.myfun\_p()

obj.myfun\_c()

class Name:

name = ""

def myfun1(self):

print(self.name)

class Surname:

surname = ""

def myfun2(self):

print(self.surname)

class Student(Name, Surname):

def parents(self):

print("Name :", self.name)

print("Surname :", self.surname)

stud = Student()

stud.name = "Lily"

stud.surname = "Jones"

stud.parents()

class Principal:

def principal(self):

print("I’m the Principal")

class Teacher(Principal):

def teacher(self):

print("I’m a Teacher")

class Student(Teacher):

def student(self):

print("I’m a student")

d = Student()

d.principal()

d.teacher()

d.student()

class HOD:

def func1(self):

print("I'm the HOD")

class CO\_ORDINATOR(HOD):

def func2(self):

print("I'm the co-ordinator under HOD")

class staff(HOD):

def func3(self):

print("I'm a STAFF under HOD")

o1 = CO\_ORDINATOR()

o2 = staff()

o1.func1()

o1.func2()

o2.func3()

class College:

def func1(self):

print("Loyola College")

class Student1(College):

def func2(self):

print("This function is in student 1. ")

class Student2(College):

def func3(self):

print("This function is in student 2.")

class Student3(Student1, College):

def func4(self):

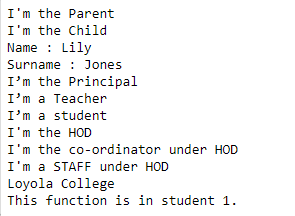
print("This function is in student 3.")

o = Student3()

o.func1()

o.func2()

Output:



Cart\_Updating – Operator Overriding

class Order:

def \_\_init\_\_(self, cart, customer):

self.cart = list(cart)

self.customer = customer

def \_\_add\_\_(self, other):

new\_cart = self.cart.copy()

new\_cart.append(other)

return Order(new\_cart, self.customer)

def \_\_radd\_\_(self, other):

new\_cart = self.cart.copy()

new\_cart.insert(0, other)

return Order(new\_cart, self.customer)

order = Order(['DogFood', 'CatFood'], 'petStore')

order.cart

order = order + 'Fish'

order.cart

order = 'Bones' + order

order.cart

Output:

**['Bones', 'DogFood', 'CatFood', 'Fish']**

ShapeSquare – Overriding

**class** Shape:

data1 **=** "abc"

**def** no\_of\_sides(self):

print("My sides need to be defined. I am from shape class.")

**def** two\_dimensional(self):

print("I am a 2D object. I am from shape class")

**class** Square(Shape):

data2 **=** "xyz"

**def** no\_of\_sides(self):

print("I have 4 sides. I am from Square class")

**def** color(self):

print("I have teal color. I am from Square class.")

sq **=** Square()

sq**.**no\_of\_sides()

sq**.**two\_dimensional()

sq**.**color()

Output:

