**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

**Computer Science & Engineering**

**Name: Parth Niteshkumar Patel**

**ID: 19DCS098**

**Subject: Python Programming**

**Semester: 4**

**Subject Code: CE259**

**Academic year: 2020-21**

**Practical-1**

Create a program that asks the user to enter their name and their age. Printout a message addressed to them that tells them the year that they will turn 100 years old

**Program Code:**

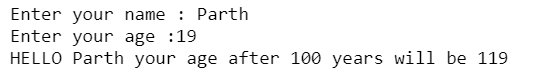
name=input("Enter your name : ")

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

age+=100

print("HELLO",name, "your age after 100 years will be",age)

**Output:**



**Practical-2.1**

Ask the user for a number. Depending on whether the number is even or 2 odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2

**Program Code:**

num=int(input("Enter the number : "))

if num%2==0:

print("The number",num,"is even")

else:

print("The number",num,"is odd")

**Output:**



**Practical-2.2**

Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89], and write a program that prints out all the elements of the list that are less than 5.

**Program Code:**

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

for i in a:

if i<5:

print(i)

**Output:**



**Practical-3.1**

Create a program that asks the user for a number and then prints out a list of all the divisors of that number. (If you don’t know what a divisor is, it is a number that divides evenly into another number. For example, 13 is a divisor of 26 because 26 / 13 has no remainder.)

**Program Code:**

num=int(input("Enter the number :"))

l1=list()

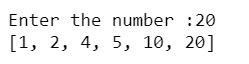
for x in range(1,num+1):

if num%x==0:

l1.append(x)

print(l1)

**Output:**



**Practical-3.2**

Take two lists, say for example these two: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

**Program Code:**

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

ans=list()

for x in a:

if x in b:

ans.append(x)

ans=list(set(ans))

print(ans)

**Output:**



**Practical-3.3**

Ask the user for a string and print out whether this string is a palindrome or not. (A palindrome is a string that reads the same forwards and backwards.)

**Program Code:**

str1=input("Enter the string")

if str1==str1[::-1]:

print(str1,"is palindrome string!!")

else:

print(str1,"is not palindrome string!!")

**Output:**



**Practical-4.1**

Let’s say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list and makes a new list that has only the even elements of this list in it

**Program Code:**

a = [1, 4, 9, 16, 25, 36,49, 64, 81, 100]

print([x for x in a if x%2==0])

**Output:**



**Practical-4.2**

Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a 4 4,5 1,3,6 1,2,4 new game) Remember the rules: Rock beats scissors, Scissors beats paper, Paper beats rock

**Program Code:**

while True:

p1=input("Enter your move :")

p2=input("Enter your move :")

if p1=="paper":

if p2=="rock":

print("p1 wins!!!")

elif p2=="scissor":

print("p2 wins!!!")

else:

print("Tied!!1")

elif p1=="rock":

if p2=="scissor":

print("p1 wins!!!")

elif p2=="paper":

print("p2 wins!!!")

else:

print("Tied!!!")

else:

if p2=="rock":

print("p2 wins!!!")

elif p2=="paper":

print("p1 wins!!!")

else:

print("Tied!!!")

x=input("Do you want to play again? (yes/no)")

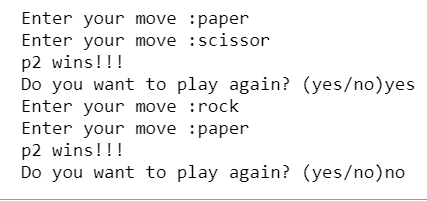
if x=="yes":

continue

else:

break

**Output:**



**Practical-4.3**

Generate a random number between 1 and 9 (including 1 and 9). Ask the user to guess the number, then tell them whether they guessed too low, too high, or exactly right.

**Program Code:**

import random as r

x=r.randint(1,10)

y=int(input("Enter the number : "))

print("random number generated :",x)

if x==y:

print("The number guessed is exactly right")

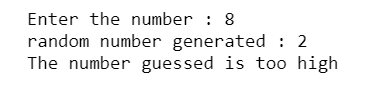
elif y>x:

print("The number guessed is too high")

else:

print("The number guessed is too low")

**Output:**



**Practical-5.1**

This week’s exercise is going to be revisiting an old exercise (see Practical 3), except require the solution in a different way. Take two lists, say for example these two: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes. Write this in one line of Python using at least one list comprehension

**Program Code:**

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

print(list(set(a).intersection(set(b))))

**Output:**



**Practical-5.2**

Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Practical 2 to help you. Take this opportunity to practice using functions, described below

**Program Code:**

def prime\_num(n):

for i in range(2,n//2):

if n%i==0:

return 1

return 0

num=int(input("Enter the number : "))

if num==1:

print("The number is neither prime or composite")

elif num==2:

print("The number is prime")

else:

tmp=prime\_num(num)

if tmp==1:

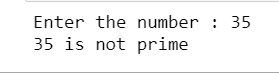
print(num,"is not prime")

else:

print(num,"is prime")

**Output:**





**Practical-5.3**

Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function.

**Program Code:**

def lst\_mode(tmp):

ans=[]

ans.append(tmp[0])

ans.append(tmp[-1])

return ans

a = [5, 10,15, 20, 25]

print(lst\_mode(a))

**Output:**



**Practical-6.1**

Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function

**Program Code:**

def fibo(n):

a=1

b=1

print(a,b,end=" ")

while n>2:

ans=a+b

a=b

b=ans

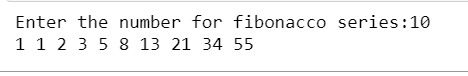
print(ans,end=" ")

n-=1

num=int(input("Enter the number for fibonacco series:"))

fibo(num)

**Output:**



**Practical-6.2**

Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates

**Program Code:**

def duplicate(lst):

new\_lst=list(set(lst))

return new\_lst

lst=[1,1,1,2,2,3,3,3,5,4,4,4,5,5,5,5]

print(duplicate(lst))

**Output:**



**Practical-6.3**

Write a program (using functions!) that asks the user for a long string function. Containing multiple words. Print back to the user the same string, except with the words in backwards order.

**Program Code:**

def reverse\_list(lst):

tmp\_lst=lst.split()

tmp\_lst.reverse()

ans=" ".join(tmp\_lst)

return ans

str=input("Enter the string :")

print(reverse\_list(str))

**Output:**



**Practical-7.1**

Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password.

**Program Code:**

import random as r

digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

lowerCase= ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'p', 'q','r', 's', 't', 'u', 'v',

'w', 'x', 'y','z']

upperCase = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'M', 'N', 'O', 'p', 'Q','R', 'S', 'T', 'U',

'V', 'W', 'X', 'Y','Z']

symbols= ['@', '#', '$', '%', ':', '.', '/','>','<', '\*', '(', ')']

combinedList=upperCase+digits+symbols+lowerCase+symbols

maxLength=int(input("Enter the length of password you need:"))

password=""

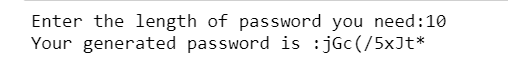
while(maxLength>0):

password+=r.choice(combinedList)

maxLength-=1

print("Your generated password is :"+password)

**Output:**



**Practical-7.2**

Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle

**Program Code:**

class Circle:

def \_\_init\_\_(self,radius):

self.radius=radius

self.area=0

self.peri=0

def Area(self):

area=4\*3.14\*self.radius\*self.radius;

return area

def Perimeter(self):

peri=2\*3.14\*radius

return peri

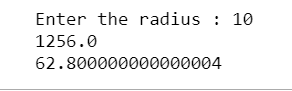
radius=int(input("Enter the radius : "))

Circle\_1=Circle(radius)

print(Circle\_1.Area())

print(Circle\_1.Perimeter())

**Output:**



**Practical-8.1**

Python supports classes inheriting from other classes. The class being inherited is called the Parent or Superclass, while the class that inherits is called the Child or Subclass. How can we define the order in which the base classes are searched when executing a method?

**Program Code:**

class Family:

def showFamily(self):

print("family:")

class Father(Family):

fatherName = ""

def showFather(self):

print(self.fatherName)

class Mother(Family):

motherName = ""

def showMother(self):

print(self.motherName)

class Son(Father, Mother):

def showParent(self):

print("Father name : ",self.fatherName)

print("Mother name : ",self.motherName)

obj=Son()

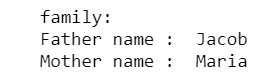
obj.fatherName = "Jacob"

obj.motherName = "Maria"

obj.showFamily()

obj.showParent()

**Output:**



**Practical-8.2**

Write a function that takes an ordered list of numbers (a list where the elements are in order from smallest to largest) and another number. The function decides whether or not the given number is inside the list and returns (then prints) an appropriate boolean.

**Program Code:**

def list\_search(lst,num):

if num in lst:

return True

else:

return False

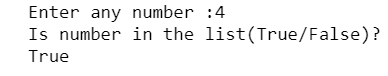
lst=[1,2,3,4,5,6,7,8,9]

number=int(input("Enter any number :"))

print("Is number in the list(True/False)? ")

print(list\_search(lst,number))

**Output:**



**Practical-8.3**

Given a .txt file that has a list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen.

**Program Code:**

fname=open("C:/Users/Parth Patel/Downloads/name.txt","r")

fileDict=dict()

fileList=fname.readlines()

for name in fileList:

name=name.strip()

if name in fileDict.keys():

fileDict[name]+=1

else:

fileDict[name]=1

for name in fileDict.keys():

print(name, " : " ,fileDict[name])

fname.close()

**Output:**



**Practical-9.1**

Develop programs to learn regular expressions using python.

**Program Code:**

import re # importing regular expression

y = "python is one of the most important language"

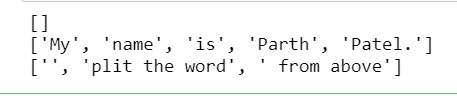
y1 = re.findall(r"^\W+", y)

print(y1)

print((re.split(r'\s','My name is Parth Patel.')))

print((re.split(r's','split the words from above')))

**Output:**



**Practical-9.2**

Develop programs for data structure algorithms using python – sorting (Bubble sort and Insertion sort)

**Program Code:**

def insertionSort(lst):

for i in range(1, len(lst)):

key = lst[i]

j = i-1

while j >=0 and key < lst[j] :

lst[j+1] = lst[j]

j -= 1

lst[j+1] = key

return lst

def bubbleSort(lst):

n = len(lst)

for i in range(n-1):

for j in range(0, n-1-i):

if lst[j] > lst[j+1] :

lst[j], lst[j+1] = lst[j+1], lst[j]

return lst

lst = [ 2,10,30,23,66,4,1,67,89,90,100,32,112]

is\_ans = insertionSort(lst)

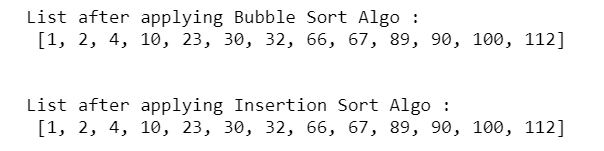
bs\_ans = bubbleSort(lst)

print ("List after applying Bubble Sort Algo :\n",bs\_ans)

print("\n")

print("List after applying Insertion Sort Algo :\n",is\_ans)

**Output:**



**Practical-9.3**

Develop programs to understand working of exception handling and assertions

**Program Code:**

import sys

ranList = ['p', 0, 5,2]

for item in ranList:

try:

print("Item from the list :",item)

ans = 1/int(item)

break

except:

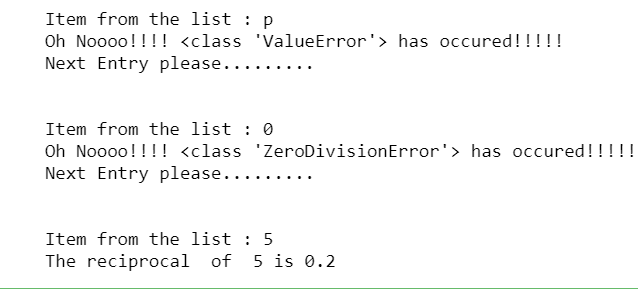
print("Oh Noooo!!!!",sys.exc\_info()[0],"has occured!!!!!")

print("Next Entry please.........")

print("\n")

print("The reciprocal of ",item,"is",ans)

**Output:**



**Practial-10**

Introduction to Django- Python based free and open-source web framework and Flask- Python based micro web framework.

**DJANGO**

* Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

* Django is a free, open-source web framework written in the Python programming language.

* A “web framework” is software that abstracts away many of the common challenges related to building a website, such as connecting to a database, handling security, user accounts, and so on. These days most developers rely on web frameworks rather than trying to build a website truly from scratch. Django was first released in 2005 and has been in continuous development since then.
* Ridiculously fast.
* Django was designed to help developers take applications from concept to completion as quickly as possible.
* Reassuringly secure.
* Django takes security seriously and helps developers avoid many common security mistakes.
* Exceedingly scalable.
* Some of the busiest sites on the Web leverage Django’s ability to quickly and flexibly scale.

**FLASK**

* [Flask](http://flask.pocoo.org/) ([source code](https://github.com/pallets/flask)) is a Python [web framework](https://www.fullstackpython.com/web-frameworks.html) built with a [small core and easy-to-extend philosophy](http://flask.pocoo.org/docs/design/).
* Flask is considered more [Pythonic](http://blog.startifact.com/posts/older/what-is-pythonic.html) than the [Django](https://www.fullstackpython.com/django.html) web framework because in common situations the equivalent Flask web application is more explicit.
* Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

**OPEN-SOURCE WEB FRAMEWORK**

* Python gives a wide scope of frameworks to developers.
* There are two types of [Python frameworks](https://www.netsolutions.com/insights/flask-vs-django/) – Full Stack Framework and Non-Full Stack Framework.
* The full-stack Python frameworks give full support to developers including basic components like form generators, form validation, and template layouts.

**MICROFRAMEWORK:**

* A microframework is a term used to refer to minimalistic web application frameworks.
* It is contrasted with full-stack frameworks.
* It lacks most of the functionality which is common to expect in a full-fledged web application framework