1. Installation of Configuration of Python. Along with its all major editors.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

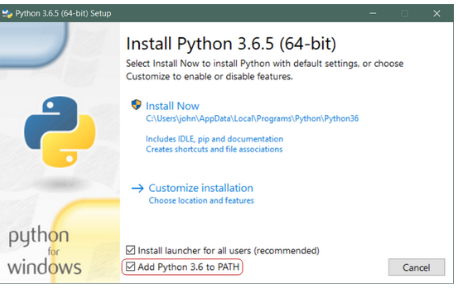
HOW TO INSTALL PYTHON ?

### Step 1: Download the Python 3 Installer

1. Open a browser window and navigate to the Download page for Windows at python.org.
2. Underneath the heading at the top that says **Python Releases for Windows**, click on the link for the **Latest Python 3 Release - Python 3.x.x**. (As of this writing, the latest is Python 3.6.5.)
3. Scroll to the bottom and select either **Windows x86-64 executable installer** for 64-bit or **Windows x86 executable installer** for 32-bit. (See below.)

### Step 2: Run the Installer

Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file. A dialog should appear that looks something like this:



Then just click **Install Now**. That should be all there is to it. A few minutes later you should have a working Python 3 installation on your system.

CONCLUSION :

In this practical we learned that what is python ,why is it needed ,and how to install Python on our PC.

1. Implement the following :
2. Create a program that asks the users to enter their name and their age. Print Out : a message addressed to them that tells them the year that they will turn 100 years old.

PROGRAM :

name = input('Enter your name :')

age = input('Enter your age : ')

age = 100 - int(age)

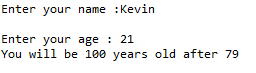
print('You will be 100 years old after {}'.format(age))

copies = int(input('Enter a number : '))

for i in range(1,int(copies+1)):

print('You will be 100 years old after {x}'.format(x=2))

OUTPUT :



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

PROGRAM :

number = int(input('Enter a number : '))

if number%2 == 0 :

print('Number is even')

else:

print('Number is odd ')

OUTPUT :





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

PROGRAM :

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

b = []

for x in a :

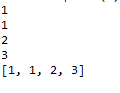
if x < 5:

print(x)

b.append(x)

print(b)

OUTPUT :



CONCLUSION :

In this practical we learned how to take input from the user and apply basic operations on that input.

1. Implement the following :
2. 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 :

number = int (input('Enter a number : '))

a=[]

for i in range(1,number+1):

if number % i == 0:

a.append(i)

print(a)

OUTPUT :



1. 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 :

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]

c=[]

"""print(set([i for i in a for j in b if i==j]))"""

for i in a:

for j in b:

if i==j:

c.append(i)

print(c)

OUTPUT :



1. 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 :

print('Enter a word for checking pallindrome : ')

word=input()

word2="".join(reversed(word))

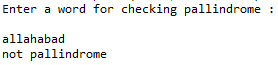
if word==word2 :

print("Pallindrome")

else :

print("not pallindrome")

OUTPUT :



CONCLUSION :

In this practical we learned the implementation of palindrome,working with given list and for loop .

1. Implement the following :
2. 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 a and makes a new list that has only the even elements of this list in it.

PROGRAM :

i=[]

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

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

OUTPUT :



1. 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 new game*)

Remember the rules:

Rock beats scissors

Scissors beats paper

Paper beats rock

PROGRAM :

stop = False

while (not stop):

answerP1 = input('Player 1: Please type your choice: Rock, Paper or Scissors:')

answerP2 = input('Player 2: Please type your choice: Rock, Paper or Scissors:')

if answerP1 == answerP2:

print('DRAW GAME')

elif answerP1 == 'Rock' and answerP2 == 'Paper':

print('PLAYER 2 WINS')

elif answerP1 == 'Rock' and answerP2 == 'Scissors':

print('PLAYER 1 WINS')

elif answerP1 == 'Paper' and answerP2 == 'Rock':

print('PLAYER 1 WINS')

elif answerP1 == 'Paper' and answerP2 == 'Scissors':

print('PLAYER 2 WINS')

elif answerP1 == 'Scissors' and answerP2 == 'Rock':

print('PLAYER 2 WINS')

elif answerP1 == 'Scissors' and answerP2 == 'Paper':

print('PLAYER 1 WINS')

else:

print('Wrong answer, please type Rock, Paper or Scissors in your next attempt!')

answer = input('Do you want to start a new game? (Yes or No) : ')

if answer == 'Yes' or 'yes':

print('New game will start')

elif answer == 'No'or 'no':

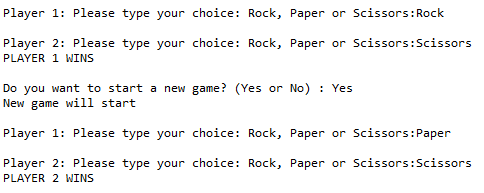
stop = True

print('GAME OVER')

else:

print('Wrong answer, please type Yes or No in your next attempt! : ')

OUTPUT :



1. 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. (Hint: remember to use the user input lessons from the very first exercise)

PROGRAM :

import random

rand\_int = random.randint(1,9)

user\_int= int(input('Enter your guessed number : '))

if rand\_int == user\_int :

print('Equals')

elif rand\_int > user\_int :

print('Less')

elif rand\_int < user\_int :

print('Greater ')

print('The random number was {}'.format(rand\_int))

OUTPUT :



CONCLUSION :

In this practical we learned how to use if with elif and how to find factors of a number.

1. Implement the following :

a. This week’s exercise is going to be revisiting an old exercise (see [Exercise 5](https://www.practicepython.org/exercise/2014/03/05/05-list-overlap.html)), 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 :

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(set([i for i in a for j in b if i==j]))

OUTPUT :



b. 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 Exercise 4 to help you. Take this opportunity to practice using functions, described below.

PROGRAM :

num = int(input('Enter your number : '))

c=[]

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

if num % i == 0:

c.append(i)

if len(c) >= 3:

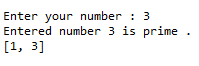
print('Entered number {} is not prime .'.format(num))

else:

print('Entered number {} is prime .'.format(num))

print(c)

OUTPUT :





c. 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 :

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

print(a[0],a[len(a)-1])

li = input("Enter list : ")

new = li.split()

def func(li):

return li[0],li[len(li)-1]

print(func(li))

OUTPUT :



CONCLUSION :

In this practical we learned about how to split in a list,how to append inside a list and write a comprehensive command in one line.

1. Implement the following :
2. Write a program that asks the user how many Fibonnaci numbers to generate and then generates them. Take this opportunity to think about how you can use functions. Make sure to ask the user to enter the number of numbers in the sequence to generate.

PROGRAM :

fibo = int(input('Enter a number for fibo limit : '))

def fib(fibo):

a = 0

b = 1

for i in range(1,fibo+1):

c=a+b

a=b

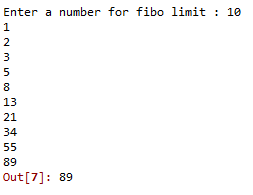
b=c

print(c)

return c

fib(fibo)

OUTPUT :



1. 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 :

def dep(a):

return list(set(a))

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

print(a)

print(dep(a))

OUTPUT :



1. Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. For example, say I type the string:

My name **is** Michele

Then I would see the string:

Michele **is** name My is shown back to me.

PROGRAM :

s=input("ENter a string : ")

def rev(s):

l=s.split()

l.reverse()

s=' '.join(l)

print(s)

rev(s)

OUTPUT :



CONCLUSION :

In this practical we learned how to how to work with different methods of list, and how to implement the functions by creating them.

1. Implement the following :
2. 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. Include your code in a main method.

PROGRAM :

import string

import random

def passwordGenerator(b):

everything = ' '

upper = string.ascii\_uppercase

lower = string.ascii\_lowercase

special = string.punctuation

number = string.digits

everything = upper + lower + special + number

# print(everything)

password = random.sample(everything, b)

#print(password)

random.shuffle(password)

for i in password:

password = "".join(password)

return password

a = int(input("Enter the desied length of the password "))

print("The password is ", passwordGenerator(a))

OUTPUT :



1. Use the BeautifulSoup and requests Python packages to print out a list of all the article titles on the New York Times homepage.

PROGRAM :

import requests

from bs4 import BeautifulSoup

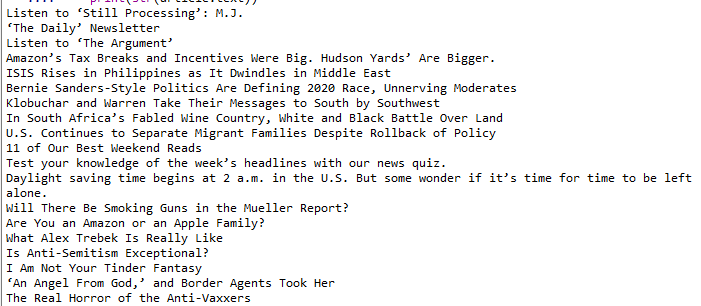
source = requests.get("https://www.nytimes.com").text

soup = BeautifulSoup(source, 'lxml')

for article in soup.find\_all('h2'):

print(str(article.text))

OUTPUT :



1. Create a program that will play the “cows and bulls” game with the user. The game works like this:

Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. For every digit that the user guessed correctly *in the correct place*, they have a “cow”. For every digit the user guessed correctly *in the wrong place* is a “bull.” Every time the user makes a guess, tell them how many “cows” and “bulls” they have. Once the user guesses the correct number, the game is over. Keep track of the number of guesses the user makes throughout teh game and tell the user at the end.

Say the number generated by the computer is 1038. An example interaction could look like this:

Welcome to the Cows and Bulls Game!

Enter a number:

>>> 1234

2 cows, 0 bulls

>>> 1256

1 cow, 1 bull

...

Until the user guesses the number.

PROGRAM :

import random

n = str(random.randint(1000,9999))

print(n)

nlist = []

cow = 0

for i in n:

nlist.append(i)

while cow < 4 and exit !="x":

x = str(input("Choose a 4 digit number, x to exit: "))

xlist = []

cow = 0

bull = 0

if x!= "x":

for i in x:

xlist.append(i)

for i in nlist:

if i in xlist and nlist.index(i) == xlist.index(i):

cow +=1

if i in xlist and nlist.index(i) != xlist.index(i):

bull +=1

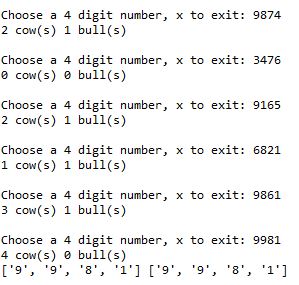
print(cow, "cow(s)", bull, "bull(s)")

else:

exit = "x"

print(nlist, xlist)

OUTPUT :



CONCLUSION :

In this practical we learned how to generate random numbers and use them for different purposes as well as how to get text from URL using Beautiful Soap.

1. Implement the following :
2. Using the requests and BeautifulSoup Python libraries, print to the screen the full text of the article on this website: https://in.mashable.com/tech/2135/honor-view-20-review-i-dont-miss-my-oneplus-6t-anymoreThe article is long, so it is split up between 4 pages. Your task is to print out the text to the screen so that you can read the full article without having to click any buttons. This will just print the full text of the article to the screen. It will not make it easy to read, so next exercise we will learn how to write this text to a .txt file.

PROGRAM :

import urllib.request

from bs4 import BeautifulSoup

url = "https://in.mashable.com/tech/2135/honor-view-20-review-i-dont-miss-my-oneplus-6t-anymore"

with urllib.request.urlopen(url) as uri:

html = uri.read()

soup = BeautifulSoup(html)

# kill all script and style elements

for script in soup(["script", "style"]):

script.extract() # rip it out

# get text

text = soup.get\_text()

# break into lines and remove leading and trailing space on each

lines = (line.strip() for line in text.splitlines())

# break multi-headlines into a line each

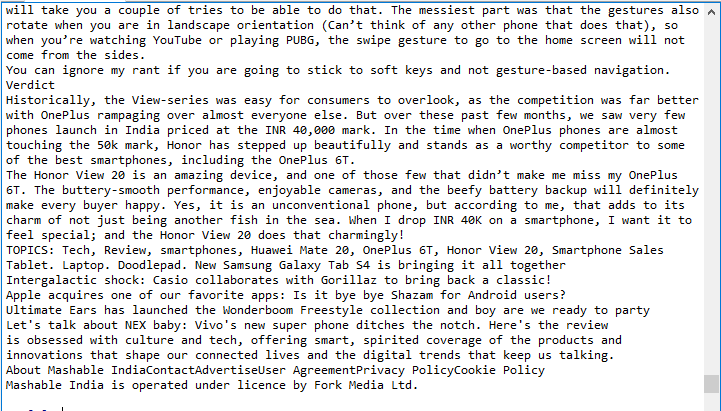
chunks = (phrase.strip() for line in lines for phrase in line.split(" "))

# drop blank lines

text = '\n'.join(chunk for chunk in chunks if chunk)

print(text)

OUTPUT :



1. 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.

Extras:

Use binary search.

PROGRAM :

def in\_list(list,s):

min=0

max=len(list)-1

while(min<=max):

mid = int((min+max) / 2)

if(list[mid] == s):

return True

if list[mid] < s:

min = mid+1

else:

max = mid-1

return False

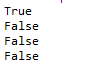
print (in\_list([1,2,3,4,5,8],4))

print (in\_list([1,2,3,4,5,8],10))

print (in\_list([1,2,3,4,5,8],0))

print (in\_list([1,2,3,4,5,8],7))

OUTPUT :



CONCLUSION :

In this practical we learned different concepts regarding the scrapping of text from the website ,and use of function for finding the elements in a list.

1. Implement the following :
2. Take the code from the How To Decode A Website exercise , and instead of printing the results to a screen, write the results to a txt file. In your code, just make up a name for the file you are saving to.

PROGRAM :

import requests

from bs4 import BeautifulSoup

source = requests.get("https://www.nytimes.com").text

def get\_title(text):

n=input(text)

return str(n)

soup = BeautifulSoup(source, 'lxml')

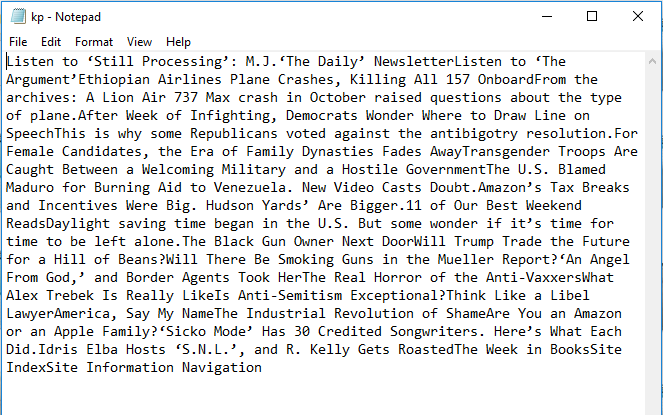
with open(get\_title('What do you want to name the file?'), 'w') as open\_file:

for article in soup.find\_all('h2'):

open\_file.write(str(article.text))

OUTPUT :





1. 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 :

count = dict()

with open("demo.txt",'r') as f :

x=f.read()

y=x.split()

for i in y :

count[i]=0

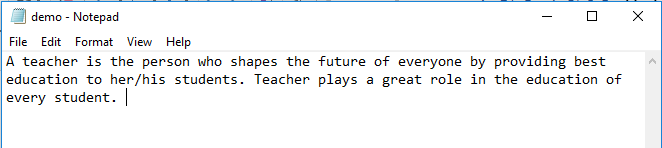
for i in y :

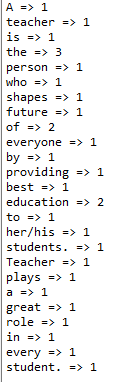
count[i]+=1

for key,val in count.items():

print (key, "=>", val)

OUTPUT :





CONCLUSION :

In this practical we learned the different concepts of how to write the contents of a website into a text file and also how to calculate the number of letter of a text.

1. Implement the following :
2. Develop programs to understand the control structures of python.

PROGRAM :

for item in range(5):

print(item\*\*2)

score =86

if score >= 90 :

print('A')

elif score >=80:

print('B')

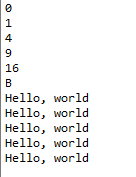
counter = 1

while counter <= 5:

print("Hello, world")

counter = counter + 1

OUTPUT :



1. Develop programs to learn different types of structures (list, dictionary, tuples) in python.

PROGRAM :

thislist = ["apple", "banana", "cherry"]

print(thislist[1])

thislist[1] = "blackcurrant"

print(thislist)

thistuple = ("apple", "banana", "cherry")

print(thistuple)

thistuple = ("apple", "banana", "cherry")

print(thistuple[1])

thisset = {"apple", "banana", "cherry"}

print(thisset)

thisset = {"apple", "banana", "cherry"}

for x in thisset:

print(x)

thisdict = {

"brand": "Ford",

"model": "Mustang",

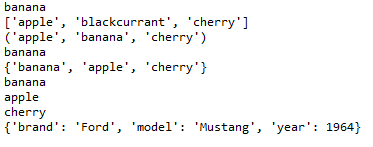
"year": 1964

}

print(thisdict)

x = thisdict["model"]

OUTPUT :



1. Develop programs to understand working of exception handling and assertions.

PROGRAM :

try:

fh = open("testfile", "r")

fh.write("This is my test file for exception handling!!")

except IOError:

print ("\nError: can\'t find file or read data")

else:

print ("Written content in the file successfully")

def KelvinToFahrenheit(Temperature):

assert (Temperature >= 0),"Colder than absolute zero!"

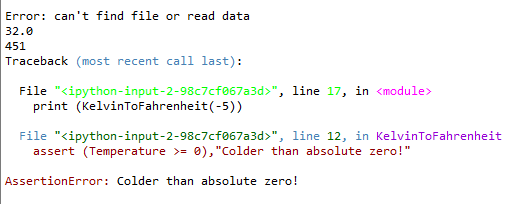
return ((Temperature-273)\*1.8)+32

print (KelvinToFahrenheit(273))

print (int((KelvinToFahrenheit(505.78))))

print (KelvinToFahrenheit(-5))

OUTPUT :



1. Develop programs to learn concept of functions scoping, recursion and list mutability.

PROGRAM :

#recursion

def calc\_factorial(x):

"""This is a recursive function

to find the factorial of an integer"""

if x == 1:

return 1

else:

return (x \* calc\_factorial(x-1))

num = 4

print("The factorial of", num, "is", calc\_factorial(num))

#scopes

def f():

s = "Me too."

print (s)

s = "I love Geeksforgeeks"

f()

print (s)

#list mutability

color = ["red", "blue", "green"]

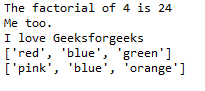
print(color)

color[0] = "pink"

color[-1] = "orange"

print(color)

OUTPUT :



CONCLUSION :

In this practical we learned the different concepts regarding exceptions , assertions, recursion , list mutability and also about the list ,tuples and dictionary.

1. Implement the following :
2. Develop programs to understand working of exception handling and assertions.

PROGRAM :

try:

fh = open("testfile", "r")

fh.write("This is my test file for exception handling!!")

except IOError:

print ("\nError: can\'t find file or read data")

else:

print ("Written content in the file successfully")

def KelvinToFahrenheit(Temperature):

assert (Temperature >= 0),"Colder than absolute zero!"

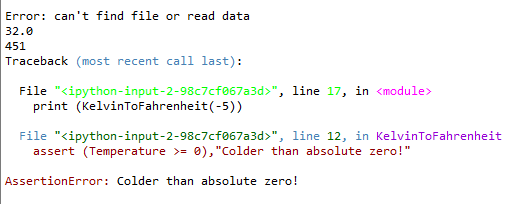
return ((Temperature-273)\*1.8)+32

print (KelvinToFahrenheit(273))

print (int((KelvinToFahrenheit(505.78))))

print (KelvinToFahrenheit(-5))

OUTPUT :



1. Develop programs for data structure algorithms using python – searching, sorting and hash tables.

PROGRAM :

#SEARCHING

def linear\_search(values, search\_for):

search\_at = 0

search\_res = False

# Match the value with each data element

while search\_at < len(values) and search\_res is False:

if values[search\_at] == search\_for:

search\_res = True

else:

search\_at = search\_at + 1

return search\_res

l = [64, 34, 25, 12, 22, 11, 90]

print("12 is in the list : ",linear\_search(l, 12))

print("91 is in the list : ",linear\_search(l, 91))

#SORTING

def mergeSort(arr):

if len(arr) > 1:

mid = len(arr) // 2 # Finding the mid of the array

L = arr[:mid] # Dividing the array elements

R = arr[mid:] # into 2 halves

mergeSort(L) # Sorting the first half

mergeSort(R) # Sorting the second half

i = j = k = 0

# Copy data to temp arrays L[] and R[]

while i < len(L) and j < len(R):

if L[i] < R[j]:

arr[k] = L[i]

i += 1

else:

arr[k] = R[j]

j += 1

k += 1

# Checking if any element was left

while i < len(L):

arr[k] = L[i]

i += 1

k += 1

while j < len(R):

arr[k] = R[j]

j += 1

k += 1

# Code to print the list

def printList(arr):

for i in range(len(arr)):

print(arr[i], end=" ")

print()

# driver code to test the above code

if \_\_name\_\_ == '\_\_main\_\_':

arr = [12, 11, 13, 5, 6, 7]

print("Given array is", end="\n")

printList(arr)

mergeSort(arr)

print("Sorted array is: ", end="\n")

printList(arr)

#HASHTABLE

hash\_table = [[] for \_ in range(10)]

print (hash\_table)

def hashing\_func(key):

return key % len(hash\_table)

def insert(hash\_table, key, value):

hash\_key = hashing\_func(key)

hash\_table[hash\_key].append(value)

print("1.insert")

print("2.view")

while(True):

choice=int(input())

if(choice==1):

print("enter data")

data=int(input())

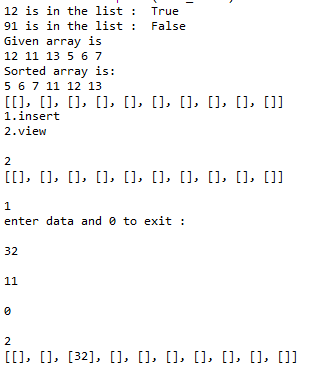
key=hashing\_func(data)

insert(hash\_table,key,data)

if(choice==2):

print(hash\_table)

OUTPUT :



1. Develop programs to learn regular expressions using python.

PROGRAM :

import re

import requests

the\_idiot\_url = 'https://www.gutenberg.org/files/2638/2638-0.txt'

def get\_book(url):

# Sends a http request to get the text from project Gutenberg

raw = requests.get(url).text

# Discards the metadata from the beginning of the book

start = re.search(r"\\*\\*\\* START OF THIS PROJECT GUTENBERG EBOOK .\* \\*\\*\\*",raw ).end()

# Discards the metadata from the end of the book

stop = re.search(r"II", raw).start()

# Keeps the relevant text

text = raw[start:stop]

return text

def preprocess(sentence):

return re.sub('[^A-Za-z0-9.]+' , ' ', sentence).lower()

book = get\_book(the\_idiot\_url)

processed\_book = preprocess(book)

print("our book")

print(processed\_book)

print("capitalize all i to I")

processed\_book = re.sub(r'\si\s', " I ", processed\_book)

print(processed\_book)

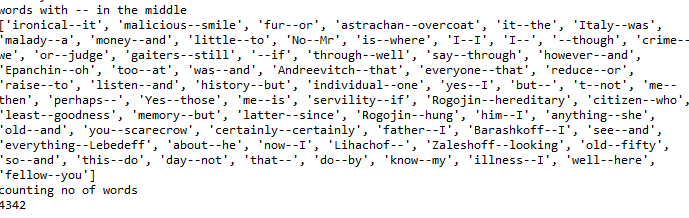
print("words with -- in the middle")

print(re.findall(r'[a-zA-Z0-9]\*--[a-zA-Z0-9]\*', book))

print("counting no of words")

print(len(re.split(" ",processed\_book)))

OUTPUT :



CONCLUSION :

In this practical we learned different implementation of searching,sorting ,about exceptions and assertions and also about the regular expressions.

1. Implement the following :
2. Learn to plot different types of graphs using PyPlot.

PROGRAM :

import pandas as pd

import matplotlib.pyplot as plt

# create 2D array of table given above

data = [['E001', 'M', 34, 123, 'Normal', 350],

['E002', 'F', 40, 114, 'Overweight', 450],

['E003', 'F', 37, 135, 'Obesity', 169],

['E004', 'M', 30, 139, 'Underweight', 189],

['E005', 'F', 44, 117, 'Underweight', 183],

['E006', 'M', 36, 121, 'Normal', 80],

['E007', 'M', 32, 133, 'Obesity', 166],

['E008', 'F', 26, 140, 'Normal', 120],

['E009', 'M', 32, 133, 'Normal', 75],

['E010', 'M', 36, 133, 'Underweight', 40]]

# dataframe created with

# the above data array

df = pd.DataFrame(data, columns=['EMPID', 'Gender',

'Age', 'Sales',

'BMI', 'Income'])

# create histogram for numeric data

df.hist()

# show plot

plt.show()

df.plot.bar()

# plot between 2 attributes

plt.bar(df['Age'], df['Sales'])

plt.xlabel("Age")

plt.ylabel("Sales")

plt.show()

df.plot.box()

# individual attribute box plot

plt.boxplot(df['Income'])

plt.show()

plt.pie(df['Age'], labels={"A", "B", "C",

"D", "E", "F",

"G", "H", "I", "J"},

autopct='% 1.1f %%', shadow=True)

plt.show()

plt.pie(df['Income'], labels={"A", "B", "C",

"D", "E", "F",

"G", "H", "I", "J"},

autopct='% 1.1f %%', shadow=True)

plt.show()

plt.pie(df['Sales'], labels={"A", "B", "C",

"D", "E", "F",

"G", "H", "I", "J"},

autopct='% 1.1f %%', shadow=True)

plt.show()

# scatter plot between income and age

plt.scatter(df['income'], df['age'])

plt.show()

# scatter plot between income and sales

plt.scatter(df['income'], df['sales'])

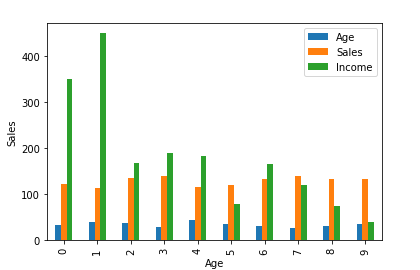
plt.show()

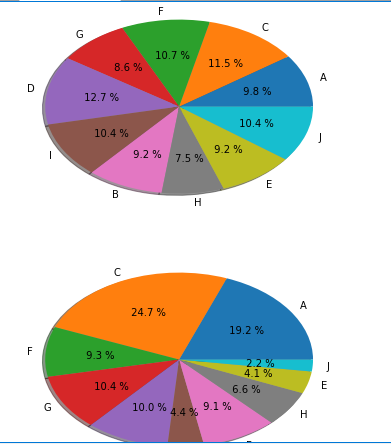
# scatter plot between sales and age

plt.scatter(df['sales'], df['age'])

plt.show()

OUTPUT :





1. Implement classical ciphers using python.

PROGRAM :

#A python program to illustrate Caesar Cipher Technique

def encrypt(text,s):

result = ""

for i in range(len(text)):

char = text[i]

if (char.isupper()):

result += chr((ord(char) + s-65) % 26 + 65)

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

#check the above function

text = "ATTACKATONCE"

s = 4

print ("Text : " ,text)

print ("Shift : ",str(s))

print ("Cipher: " ,encrypt(text,s))

OUTPUT :



CONCLUSION :

In this practical we learned different ciphers and how to encrypt them.

1. Implement the following :
2. Draw graphics using Turtle.

PROGRAM :

import turtle

ninja = turtle.Turtle()

ninja.speed(10)

for i in range(180):

ninja.forward(100)

ninja.right(30)

ninja.forward(20)

ninja.left(60)

ninja.forward(50)

ninja.right(30)

ninja.penup()

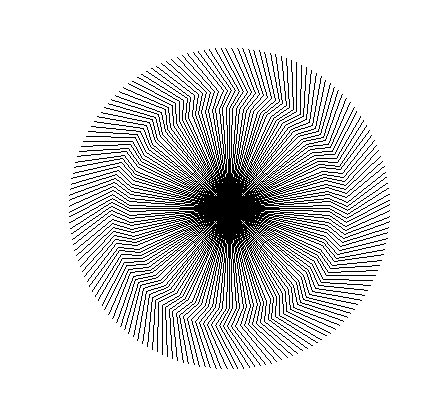
ninja.setposition(0, 0)

ninja.pendown()

ninja.right(2)

turtle.done()

OUTPUT :



1. Develop programs to learn GUI programming using Tkinter.

PROGRAM :

from tkinter import \*

import os

# Designing window for registration

def register():

global register\_screen

register\_screen = Toplevel(main\_screen)

register\_screen.title("Register")

register\_screen.geometry("300x250")

global username

global password

global username\_entry

global password\_entry

username = StringVar()

password = StringVar()

Label(register\_screen, text="Please enter details below", bg="blue").pack()

Label(register\_screen, text="").pack()

username\_lable = Label(register\_screen, text="Username \* ")

username\_lable.pack()

username\_entry = Entry(register\_screen, textvariable=username)

username\_entry.pack()

password\_lable = Label(register\_screen, text="Password \* ")

password\_lable.pack()

password\_entry = Entry(register\_screen, textvariable=password, show='\*')

password\_entry.pack()

Label(register\_screen, text="").pack()

Button(register\_screen, text="Register", width=10, height=1, bg="blue", command=register\_user).pack()

# Designing window for login

def login():

global login\_screen

login\_screen = Toplevel(main\_screen)

login\_screen.title("Login")

login\_screen.geometry("300x250")

Label(login\_screen, text="Please enter details below to login").pack()

Label(login\_screen, text="").pack()

global username\_verify

global password\_verify

username\_verify = StringVar()

password\_verify = StringVar()

global username\_login\_entry

global password\_login\_entry

Label(login\_screen, text="Username \* ").pack()

username\_login\_entry = Entry(login\_screen, textvariable=username\_verify)

username\_login\_entry.pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Password \* ").pack()

password\_login\_entry = Entry(login\_screen, textvariable=password\_verify, show='\*')

password\_login\_entry.pack()

Label(login\_screen, text="").pack()

Button(login\_screen, text="Login", width=10, height=1, command=login\_verify).pack()

# Implementing event on register button

def register\_user():

username\_info = username.get()

password\_info = password.get()

file = open(username\_info, "w")

file.write(username\_info + "\n")

file.write(password\_info)

file.close()

username\_entry.delete(0, END)

password\_entry.delete(0, END)

Label(register\_screen, text="Registration Success", fg="green", font=("calibri", 11)).pack()

# Implementing event on login button

def login\_verify():

username1 = username\_verify.get()

password1 = password\_verify.get()

username\_login\_entry.delete(0, END)

password\_login\_entry.delete(0, END)

list\_of\_files = os.listdir()

if username1 in list\_of\_files:

file1 = open(username1, "r")

verify = file1.read().splitlines()

if password1 in verify:

login\_sucess()

else:

password\_not\_recognised()

else:

user\_not\_found()

# Designing popup for login success

def login\_sucess():

global login\_success\_screen

login\_success\_screen = Toplevel(login\_screen)

login\_success\_screen.title("Success")

login\_success\_screen.geometry("150x100")

Label(login\_success\_screen, text="Login Success").pack()

Button(login\_success\_screen, text="OK", command=delete\_login\_success).pack()

# Designing popup for login invalid password

def password\_not\_recognised():

global password\_not\_recog\_screen

password\_not\_recog\_screen = Toplevel(login\_screen)

password\_not\_recog\_screen.title("Success")

password\_not\_recog\_screen.geometry("150x100")

Label(password\_not\_recog\_screen, text="Invalid Password ").pack()

Button(password\_not\_recog\_screen, text="OK", command=delete\_password\_not\_recognised).pack()

# Designing popup for user not found

def user\_not\_found():

global user\_not\_found\_screen

user\_not\_found\_screen = Toplevel(login\_screen)

user\_not\_found\_screen.title("Success")

user\_not\_found\_screen.geometry("150x100")

Label(user\_not\_found\_screen, text="User Not Found").pack()

Button(user\_not\_found\_screen, text="OK", command=delete\_user\_not\_found\_screen).pack()

# Deleting popups

def delete\_login\_success():

login\_success\_screen.destroy()

def delete\_password\_not\_recognised():

password\_not\_recog\_screen.destroy()

def delete\_user\_not\_found\_screen():

user\_not\_found\_screen.destroy()

# Designing Main(first) window

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

main\_screen.geometry("300x250")

main\_screen.title("Account Login")

Label(text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13)).pack()

Label(text="").pack()

Button(text="Login", height="2", width="30", command=login).pack()

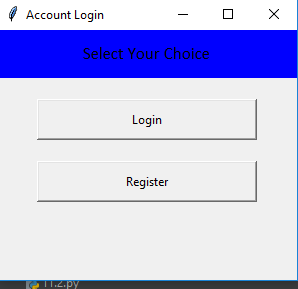
Label(text="").pack()

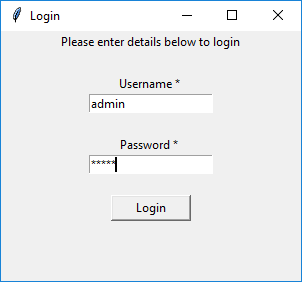
Button(text="Register", height="2", width="30", command=register).pack()

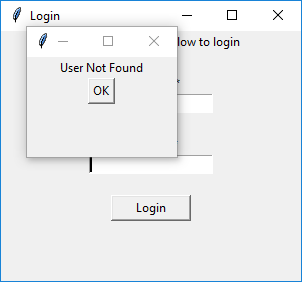
main\_screen.mainloop()

main\_account\_screen()

OUTPUT :







CONCLUSION :

In this practical we learned different concepts about tkinter and turtle.

1. Implement the following :
2. Django Framework

WHAT IS DJANGO?

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch. Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

WHY DJANGO?

Django is a Web framework written in Python.A Web framework is a software that supports the development of dynamic Web sites, applications, and services. It provides a set of tools and functionalities that solves many common problems associated with Web development, such as security features, database access, sessions, template processing, URL routing, internationalization, localization, and much more.

INSTALLATION

The basic setup consists of installing Python, Virtualenv, and Django.In the Command Prompt, execute the command below:

pip install virtualenv

mkdir myproject

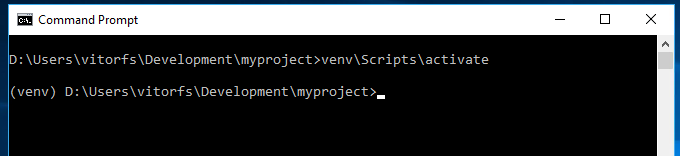
cd myproject

virtualenv venv

Our virtual environment is created. Now before we start using it, we need to activate:

venv\Scripts\activate

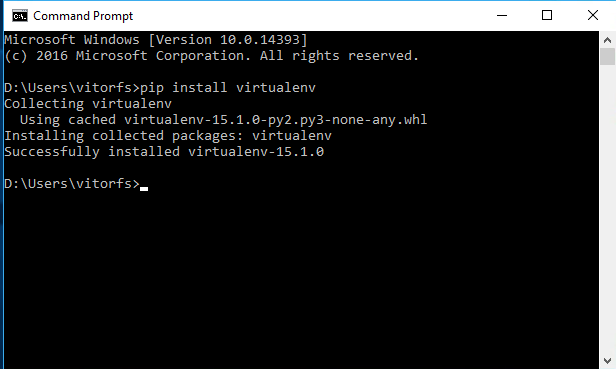
venv\Scripts\deactivate.bat



INSTALLNG VIRTUALENV

In the Command Prompt, execute the command below:

**pip install virtualenv**



INSTALLING DJANGO

**pip install django**

