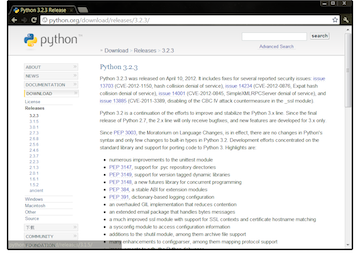
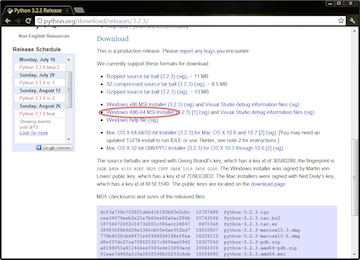
**PRACTICAL 1**

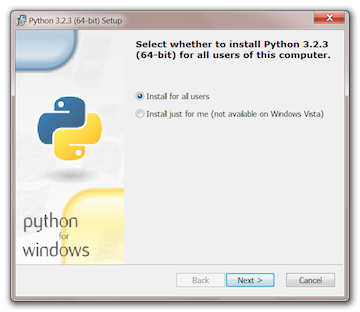
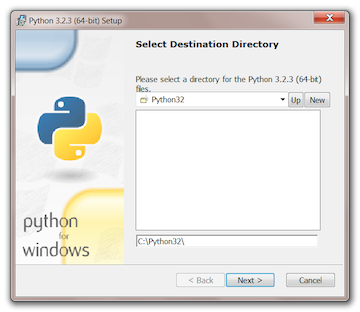
**AIM: Installation of Configuration of Python. Along with its all major editors**

Installing Python

Go to [python.org](https://python.org/downloads/) and download the version of Python that you want. In these examples, I'm downloading Python 3.2.3.

[](https://www.davidbaumgold.com/tutorials/set-up-python-windows/python-3.2.3-homepage.png) [](https://www.davidbaumgold.com/tutorials/set-up-python-windows/python-3.2.3-download.png)

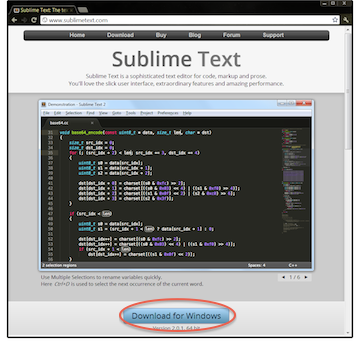
Run the installer, and let it install to the default location: in this case, C:\Python32\.

[](https://www.davidbaumgold.com/tutorials/set-up-python-windows/python-3.2.3-installer-intro.png) [](https://www.davidbaumgold.com/tutorials/set-up-python-windows/python-3.2.3-installer-location.png)

## Installing Sublime Text

Lastly, you'll need a text editor. There are many good options, but I suggest [Sublime Text](https://www.sublimetext.com/): it's very powerful and useful, and it has an unlimited free trial. Please note that Microsoft Word is a word processor, not a text editor: you cannot use Microsoft Word for writing computer programs.

Installing Sublime Text is easy: just go to the [Sublime Text](https://www.sublimetext.com/) homepage, download the installer, and install it. Sublime Text is not free software, but you can try it for as long as you like before you buy it. If you end up writing many programs using Sublime Text, I encourage you to buy a license.

[](https://www.davidbaumgold.com/tutorials/set-up-python-windows/sublime-text-homepage.png)

Verify that it all works

Now that you're all set up, let's verify that everything is working properly! The first program that any programmer writes is called "Hello, World!", and all it does is write that text out to the terminal. So, let's write "Hello, World!" in Python! If your computer is set up properly, this should work properly:

$ python -c 'print("Hello, World!")'

Hello, World!

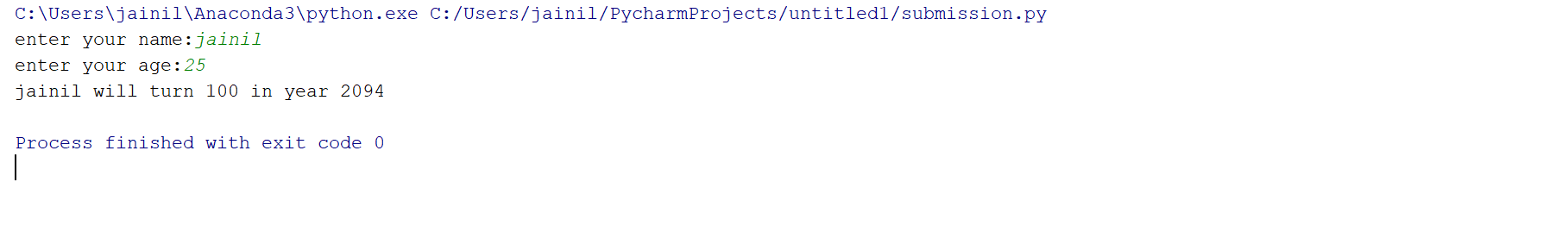
**PRACTICAL 2**

**a) Create a program that asks the user 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.**

**Code:**

**from** datetime **import** datetime  
**class** user:  
 **def** \_\_init\_\_(self,name,age):  
 self.name=name  
 self.age=age  
  
user1=user(input(**"enter your name:"**),input(**"enter your age:"**))  
x=datetime.now()  
y=x.year-int(user1.age)+100  
  
print(**"{} will turn 100 in year {}"**.format(user1.name,y))

**Output:**

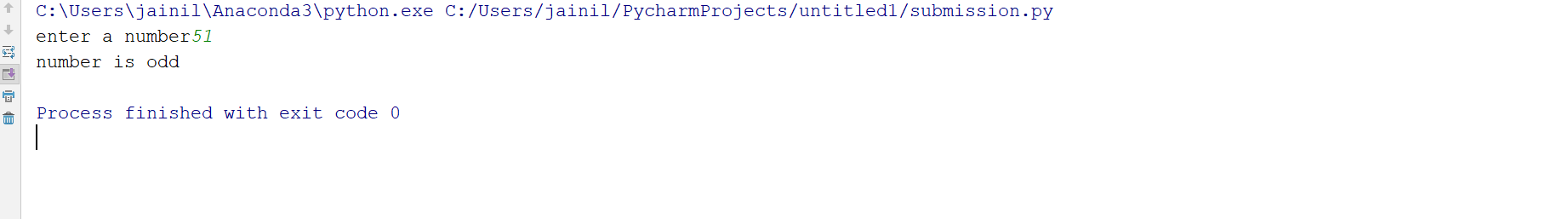


**b) Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2?**

**Code:**

x=input(**"enter a number"**)  
**if** int(x)%2==0:  
 print(**"the number is even"**)  
**else**:  
 print(**"number is odd"**)

**Output:**



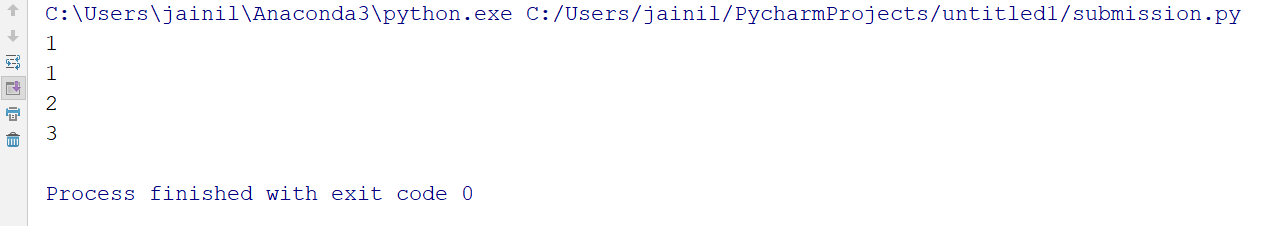
**c) Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] and write a program that prints out all the elements of the list that are less**

**than 5.**

**Code:**

a=[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  
  
**for** i **in** a:  
 **if**(i<5):  
 print(i)

**Output:**



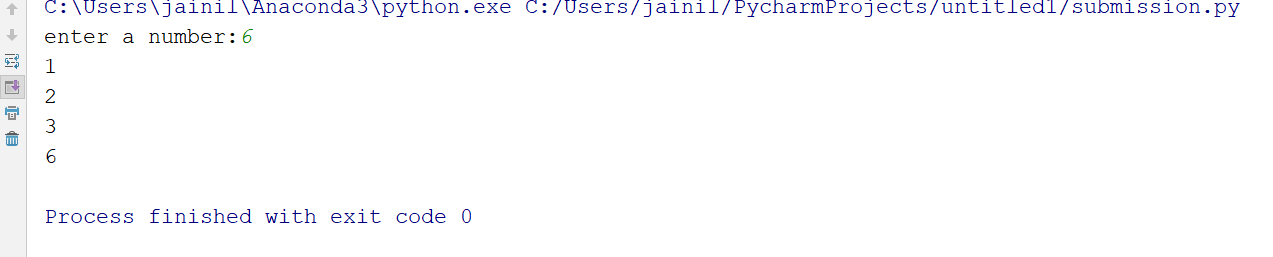
**PRACTICAL 3**

**a) 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.)**

**Code:**

x=int(input(**"enter a number:"**))  
**for** i **in** range(1,int(x/2)+1):  
 **if**(x%i==0):  
 print(i)  
print(x)

**Output:**



**b) Take two lists, say for example these two:**

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

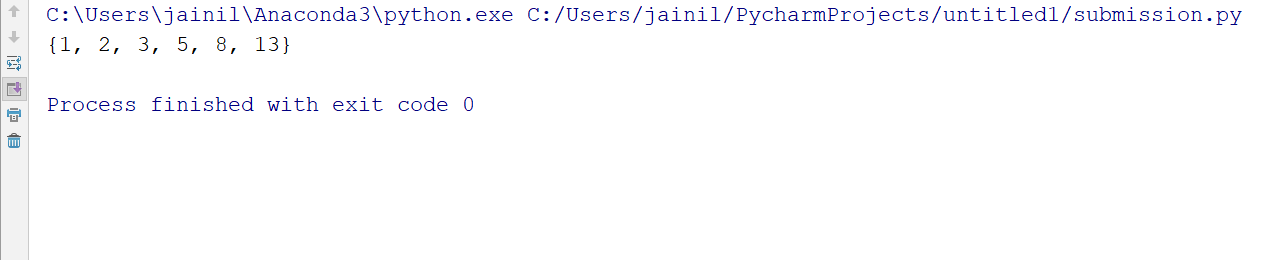
**b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]**

**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.**

**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]  
common=set()  
**for** i **in** a:  
 **for** j **in** b:  
 **if**(i==j):  
 common.add(i)  
print(common)

**Output:**

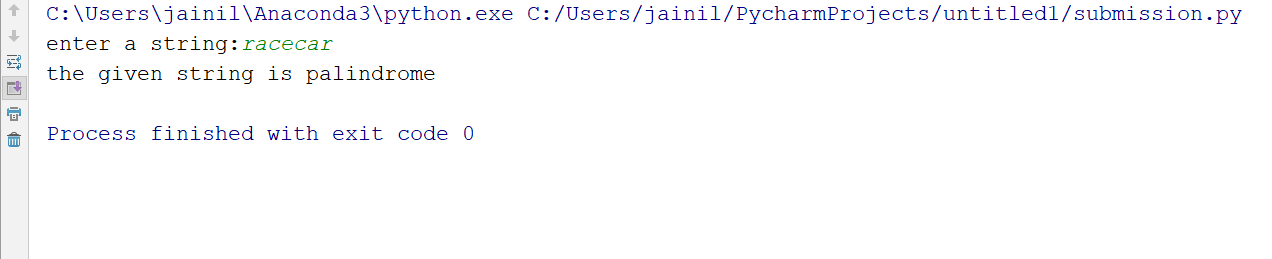


**c) Ask the user for a string and print out whether this string is a palindrome or not. (A palindrome is a string that reads the same forwards and backwards.)**

**Code:**

x=input(**"enter a string:"**)  
y=x[::-1]  
**if**(x==y):  
 print(**"the given string is palindrome"**)  
**else**:  
 print(**"not a palindrome"**)

**Output:**

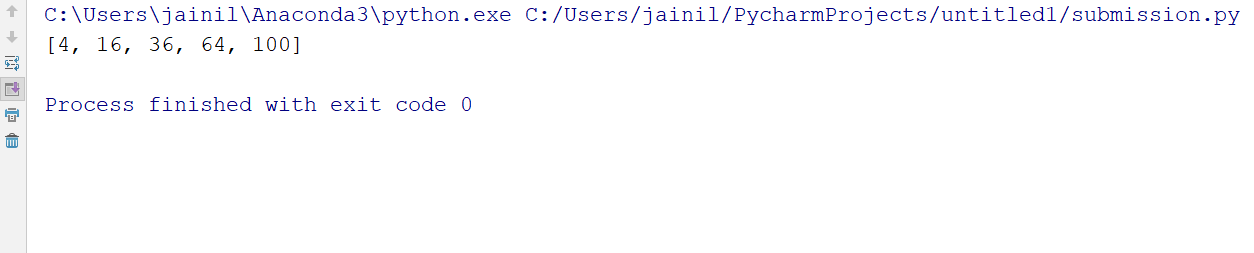


**d) Let’s say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list a and makes a new list that has only the even elements of this list in it.**

**Code:**

a = [1, 4, 9, 16, 25, 36, 49, 64,81, 100]  
b = [x **for** x **in** a **if** x%2==0]  
print(b)

**Output:**



**PRACTICAL 4**

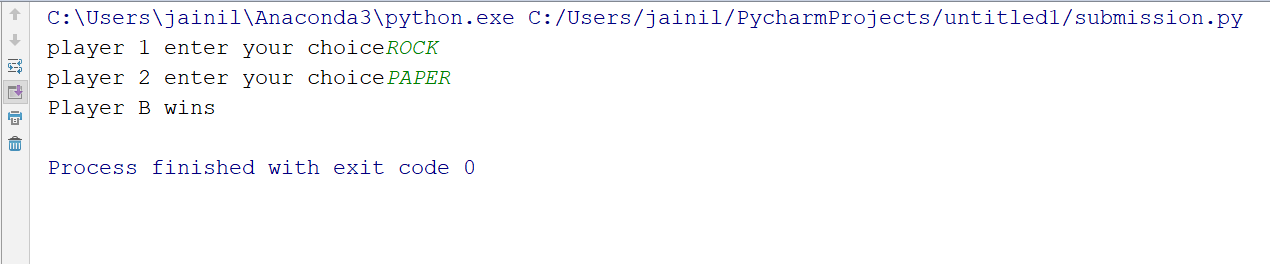
**a) Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game)**

**Remember the rules:**

**Rock beats scissors Scissors beats paper Paper beats rock**

**Code:**

**def** func (k,l):  
 **if**(k==**"ROCK" and** l==**"SCISSOR"**)**or**(k==(**"SCISSOR"**)**and** l==(**"PAPER"**))**or**(k==(**"PAPER"**)**and** l==(**"ROCK"**)):  
 **return True  
 else**:  
 **return False**x=input(**"player 1 enter your choice"**)  
y=input(**"player 2 enter your choice"**)  
  
a=x.upper()  
b=y.upper()  
  
**if not**(a==**"ROCK" or** a==**"SCISSOR" or** a==**"PAPER" or** b==**"ROCK" or** b==**"SCISSOR" or** b==**"PAPER"**):  
 print(**"INVALID INPUT BY ANYONE"**)  
  
**elif** a==b:  
 print(**"it's a tie"**)  
  
  
**elif** func(a,b):  
 print(**"Player A wins"**)  
  
**else**:  
 print(**"Player B wins"**)  
  
**Output:**

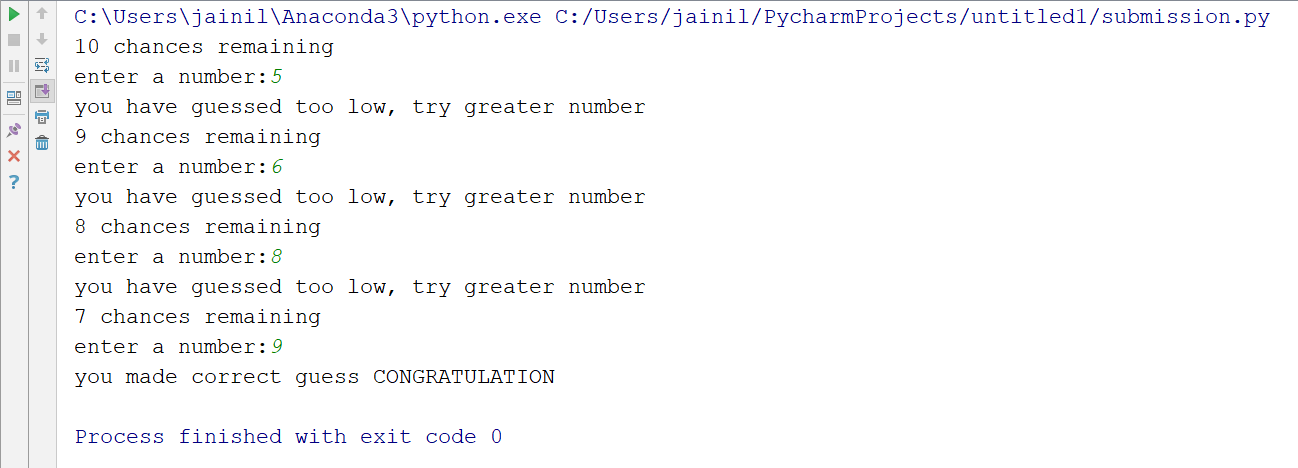


**b) Generate a random number between 1 and 9 (including 1 and 9). Ask the user to guess the number, then tell them whether they guessed too low, too high, or exactly right. (Hint: remember to use the user input lessons from the very first practical)**

**Code:**

**import** random  
x=random.randrange(1,10)  
i=0;  
flag=**True  
while**(flag **and** (i<10)):  
 print(str(10-i)+**" chances remaining"**)  
 i=i+1  
 y=input(**"enter a number:"**)  
 **if** (int(y)==x):  
 flag=**False**;  
 print(**"you made correct guess CONGRATULATION"**)  
  
 **elif**(int(y)>x):  
 print(**"you have guessed too high, try lower number"**)  
  
 **else**:  
 print(**"you have guessed too low, try greater number"**)

**Output:**



**PRACTICAL 5**

**a) This week’s exercise is going to be revisiting an old exercise (see Practical 2), 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]**

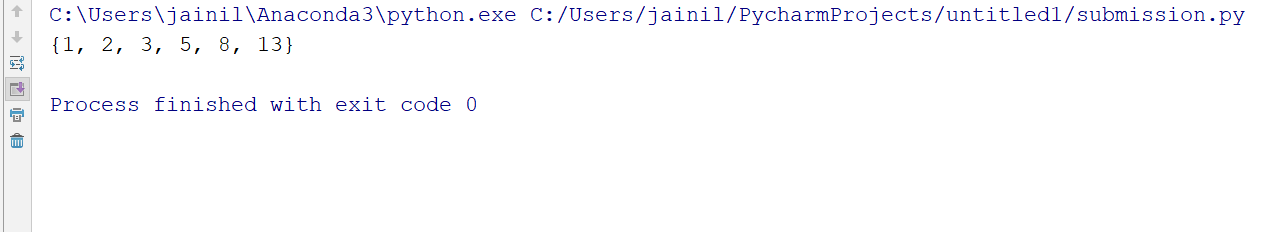
**b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]**

**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**

**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(set(i **for** i **in** a **for** j **in** b **if** i==j))  
**"""LIST COMPREHENSIVE """**

**Output:**



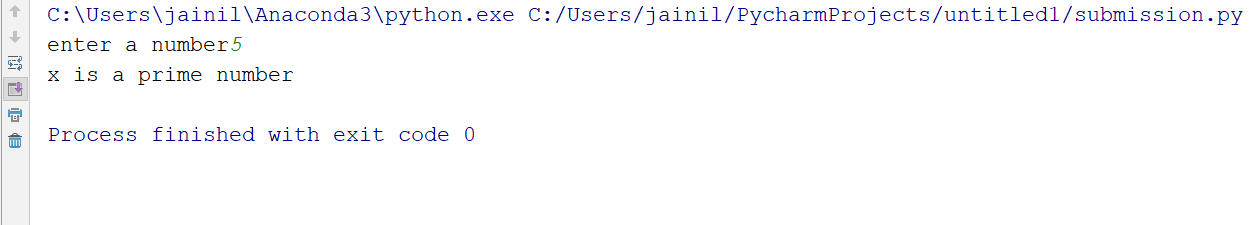
**b) Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Practical 2 to help you.**

**Take this opportunity to practice using functions, described below.**

**Code:**

x=input(**"enter a number"**)  
  
flag=**True  
  
  
for** i **in** range (2,int(int(x)/2)):  
 **if**(int(x)%i==0):  
 flag=**False  
 break  
  
if** flag==**True**:  
 print(**"x is a prime number"**)  
**else**:  
 print(**"x is not a prime number"**)

**Output:**

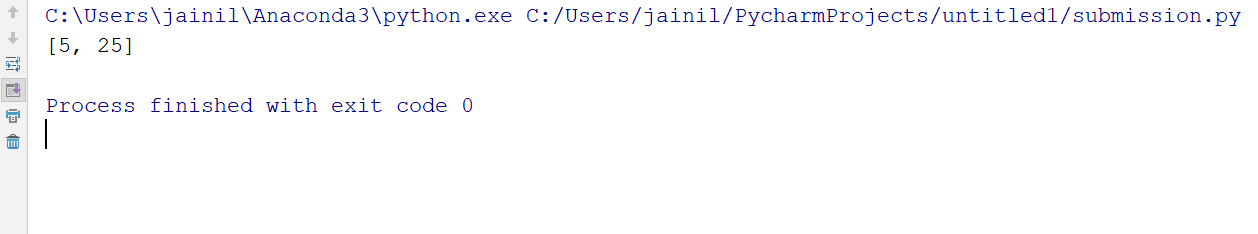


**c) Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function.**

**Code:**

**def** firstandlast(x):  
 **return**([x[0],x[len(x)-1]])  
a = [5, 10, 15, 20,25]  
print(firstandlast(a))

**Output:**



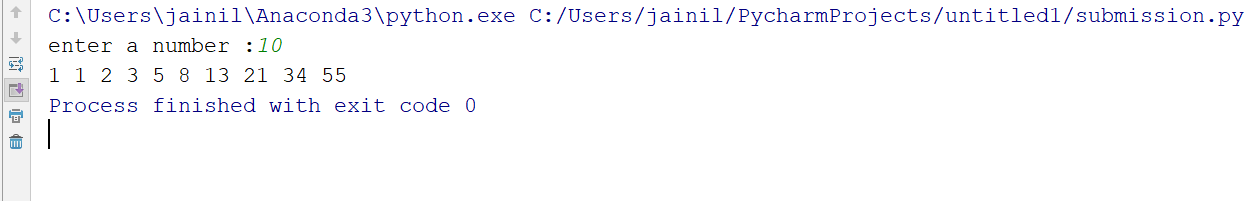
**PRACTICAL 6**

**a) 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.(Hint: The Fibonnaci seqence is a sequence of numbers where the next number in the sequence is the sum of the previous two numbers in the sequence. The sequence looks like this: 1, 1, 2, 3, 5, 8, 13, …)**

**Code:**

**def** fib(a,b,n):  
 c=a+b  
 print(c,end=**" "**)  
 **if**(n>0):  
 fib(b,c,n-1)  
  
  
x=input(**"enter a number :"**)  
print(1,end=**" "**)  
**if**(int(x)>1):  
 print(1,end=**" "**)  
**if**(int(x)>2):  
 fib(1,1,int(x)-3)

**Output:**

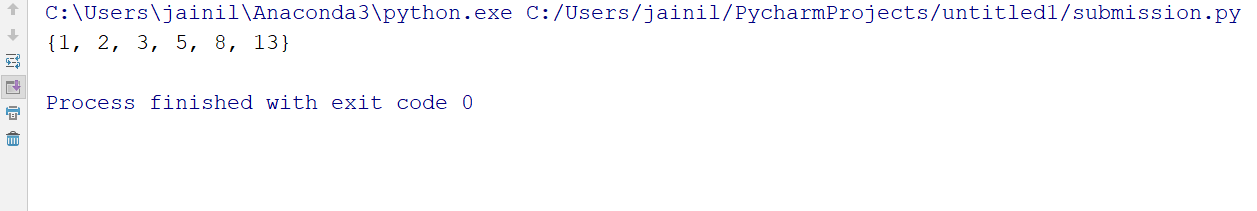


**b) Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.**

**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]  
customlist = set(i **for** i **in** a **for** j **in** b **if** i==j)  
print(customlist)

**Output:**



**c) 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**

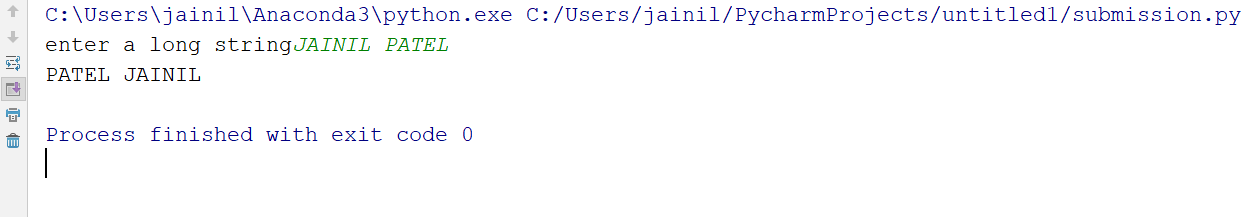
**Then I would see the string:**

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

**Code:**

**def** reverse(s):  
 y=s.split()  
 k=**""  
 for** i **in** reversed(y):  
 k=k+(i+**" "**)  
  
 **return** k  
  
  
x=input(**"enter a long string"**)  
print(reverse(x))

**Output:**

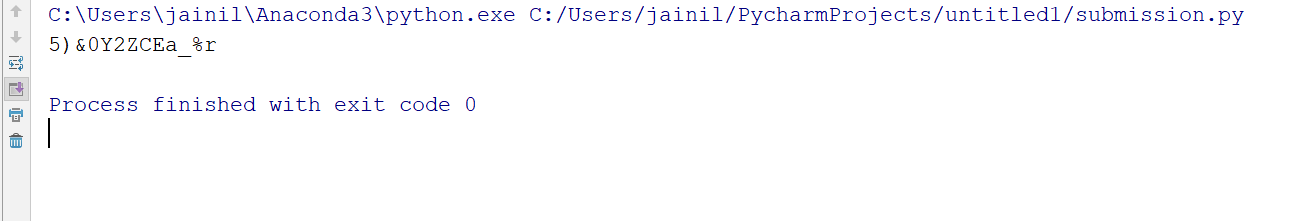


**d) Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your run-time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

**Code:**

**import** string  
**from** random **import** \*  
characters = string.ascii\_letters + string.punctuation + string.digits  
password = **""**.join(choice(characters) **for** x **in** range(randint(8, 16)))  
print(password)

**Output:**



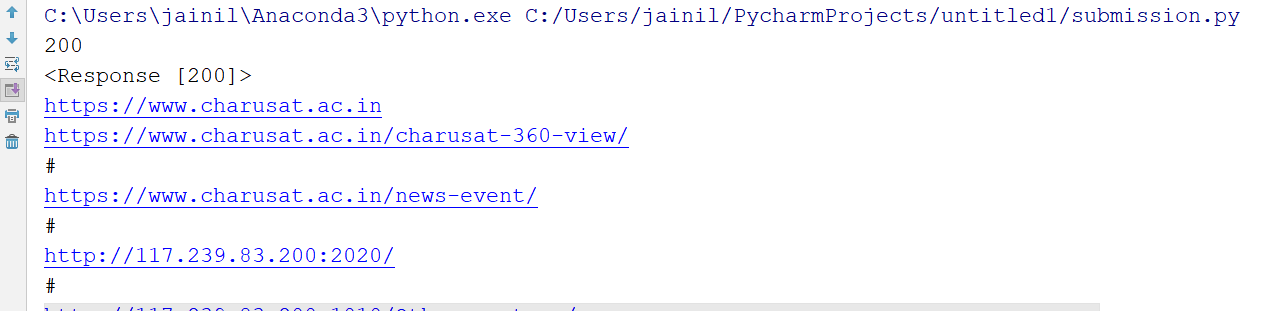
**e) Use the BeautifulSoup and requests Python packages to print out a list of all**

**the article titles on the New York Times homepage.**

**Code:**

*#web scrapping with python***import** re  
**from** bs4 **import** BeautifulSoup  
**import** requests  
  
result=requests.get(**"http://www.charusat.ac.in"**)  
print(result.status\_code)  
print(result)  
  
c=result.content  
  
  
soup=BeautifulSoup(c,features=**"lxml"**)  
  
*#print (soup.prettify())***for** link **in** soup.find\_all(**'a'**):  
 print(link.get(**'href'**))

**Output:**



**PRACTICAL 7**

**a) 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**

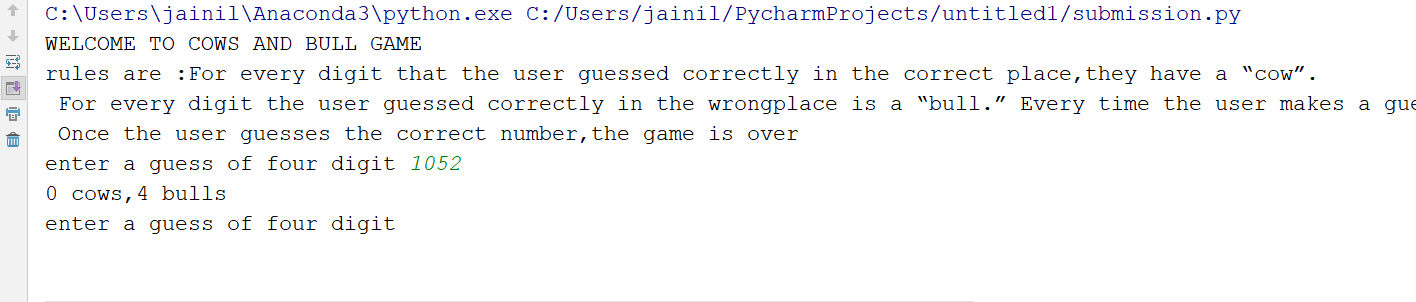
**1 cow, 1 bull**

**...**

**Until the user guesses the number.**

**Code:**

**import** random  
**def** numberofmatch(randomnumber,x):  
 firstdigit = (randomnumber // 1000) % 10  
 seconddigit = (randomnumber // 100) % 10  
 thirddigit = (randomnumber // 10) % 10  
 fourthdigit = (randomnumber // 1) % 10  
 firstdigitguessed = (x // 1000) % 10  
 seconddigitguessed = (x // 100) % 10  
 thirddigitguessed = (x // 10) % 10  
 fourthdigitguessed = (x // 1) % 10  
 count=0  
 **if**(firstdigit==firstdigitguessed):  
 count+=1  
 **if**(seconddigit==seconddigitguessed):  
 count+=1  
 **if**(thirddigit==thirddigitguessed):  
 count+=1  
 **if**(fourthdigit==fourthdigitguessed):  
 count+=1  
 **return** count  
  
print(**"WELCOME TO COWS AND BULL GAME"**)  
print(**'rules are :For every digit that the user guessed correctly in the correct place,they have a “cow”.\n For every digit the user guessed correctly in the wrongplace is a “bull.” Every time the user makes a guess, tell them how many“cows” and “bulls” they have.\n Once the user guesses the correct number,the game is over'**)  
randomnumber=random.randrange(1000,9999)  
  
flag=**True  
while** flag:  
 x=int(input(**"enter a guess of four digit "**))  
 c=numberofmatch(randomnumber,x)  
 print(**"{} cows,{} bulls"**.format(c,4-c))  
 **if**(c==4):  
 flag=**False** print(**"congratulation you guessed correctly."**)  
  
**Output:**



**PRACTICAL 8**

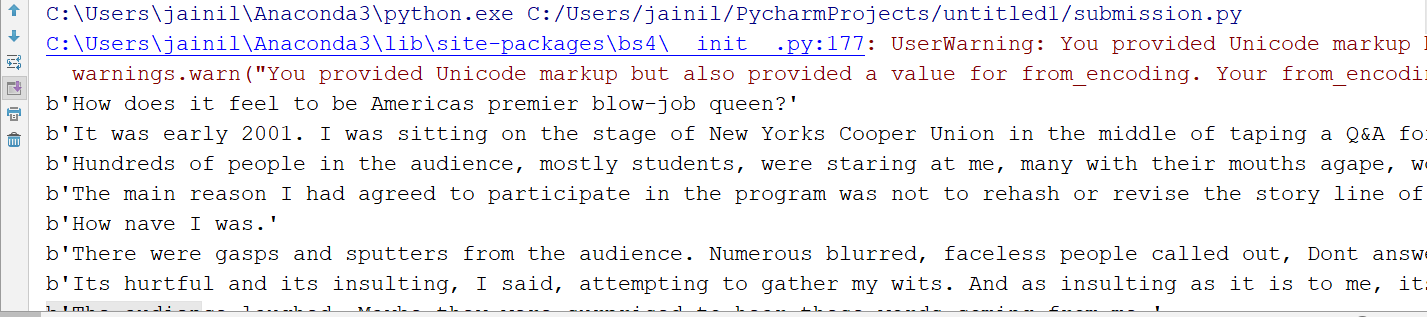
**a) Using the requests and BeautifulSoup Python libraries, print to the screen the full text of the article on this website: http://www.vanityfair. com/society/2014/06/monica-lewinsky-humiliation-culture.**

**The 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.**

**Code:**

**import** requests  
**from** bs4 **import** BeautifulSoup  
**import** codecs  
**import** textwrap  
  
  
**def** get\_web\_page(url):  
 r = requests.get(url)  
 **return** r.text  
  
**def** format\_text\_write(content, fileName):  
 dedented\_text = textwrap.dedent(content).strip()  
 writeBuffer = textwrap.fill(dedented\_text, width = 100)  
 write\_text(fileName, writeBuffer)  
  
**def** write\_text(text\_name, content):  
 f = codecs.open(text\_name,**'a'**)  
 f.write(content)  
  
  
url = **'http://www.vanityfair.com/society/2014/06/monica-lewinsky-humiliation-culture'**fileName = **'VanityFair.txt'**html\_fileName = **'VanityFair\_html.txt'**FormattedFileName = **'VanityFair\_Formatted.txt'**html = get\_web\_page(url)  
soup = BeautifulSoup(html, **'html.parser'**, from\_encoding=**'utf-8'**)  
article = soup.find\_all(class\_=**"content-section"**)  
  
**for** section **in** article:  
 paragraph = section.find\_all(**'p'**)  
 **for** sentence **in** paragraph:  
 **if** sentence **is not None**:  
 sentence\_text = sentence.get\_text()  
 **if** len(sentence\_text):  
 print(sentence\_text.encode(**'ascii'**, **'ignore'**))  
 write\_buffer = sentence\_text + **'\n'** write\_text(fileName, write\_buffer)  
  
**with** open(fileName, **'r'**) **as** content\_file:  
 content = content\_file.read()  
 format\_text\_write(content, FormattedFileName)

**Output:**



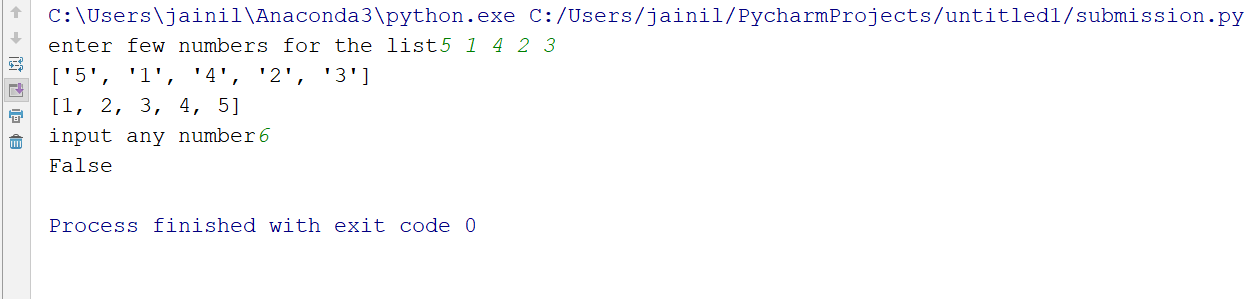
**b) Write a function that takes an ordered list of numbers (a list where the elements are in order from smallest to largest) and another number. The function decides whether or not the given number is inside the list and**

**returns (then prints) an appropriate boolean.**

**Code:**

**def** contains(x,y):  
 **for** i **in** x:  
 **if**(i==y):  
 **return True  
 return False**x=input(**"enter few numbers for the list"**)  
y=list(x.split(**" "**))  
z=[]  
print(y)  
**for** i **in** y:  
 z.append(int(i))  
z.sort()  
print(z)  
inp=input(**"input any number"**)  
print(contains(x,inp))

**Output:**



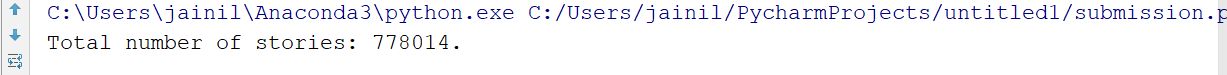
**PRACTICAL 9**

**a) 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.**

**Code:**

**from** bs4 **import** BeautifulSoup **as** soup  
**import** requests  
**import** re  
  
my\_url = **'https://www.nytimes.com'***# opening connection, grabbing the page.*r = requests.get(my\_url)  
page\_html = r.text  
  
*# html parsing*page\_soup = soup(page\_html, **'html.parser'**)  
  
  
*# grabs each story header and stores into a list*stories = page\_html  
  
print(**"Total number of stories: "** + str(len(stories)) + **". \n"**)  
open\_file = open(**'file\_to\_save.txt'**, **'w'**)  
  
*# Regex to eliminate tags and characters. Lstrip and rstrip to remove whitespaces/newlines.***for** b **in** stories:  
 content = str(re.sub(**"<.\*?>"**, **""**, str(b))).lstrip().rstrip()  
 open\_file.write(**'\n'** + content)  
  
open\_file.close()

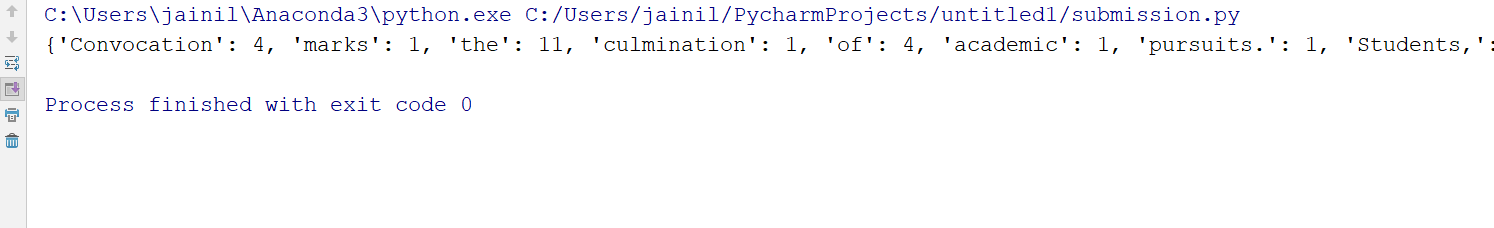
**Output:**



**b) 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.**

**Code:**

count=dict()  
  
**with** open(**"jainil"**,**'r'**) **as** f:  
 x=f.read()  
 y=x.split()  
 **for** i **in** y:  
 count[i]=0  
 **for** i **in** y:  
 count[i]+=1  
  
print(count)

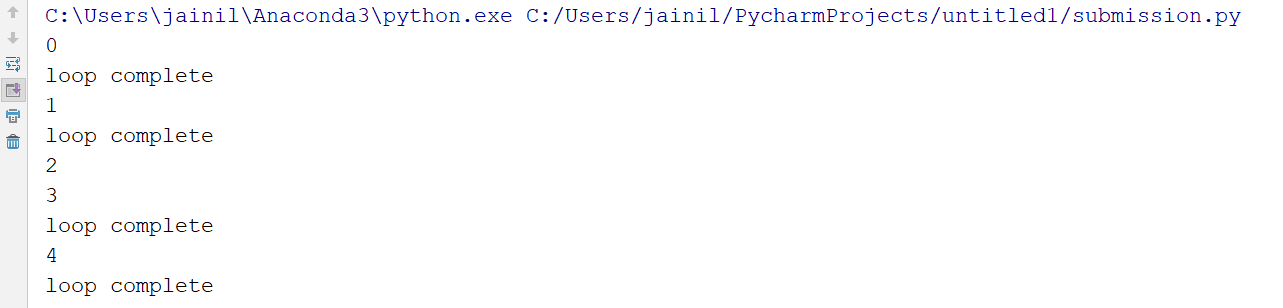
**Output:** 

**PRACTICAL 10**

**a) Develop programs to understand the control structures of python**

**Code:**

**for** i **in** range(10):  
 print(i)  
 **if**(i==2):  
 **continue  
 if**(i==8):  
 **break  
 else**:  
 **pass** print(**"loop complete"**)  
print()  
print()  
k=0  
**while True**:  
 k+=1  
 print(k)  
 **if**(k==3):  
 **continue  
 elif**(k==5):  
 **pass  
 elif**(k==9):  
 **break** print(**"loop complete"**)  
**Output:**

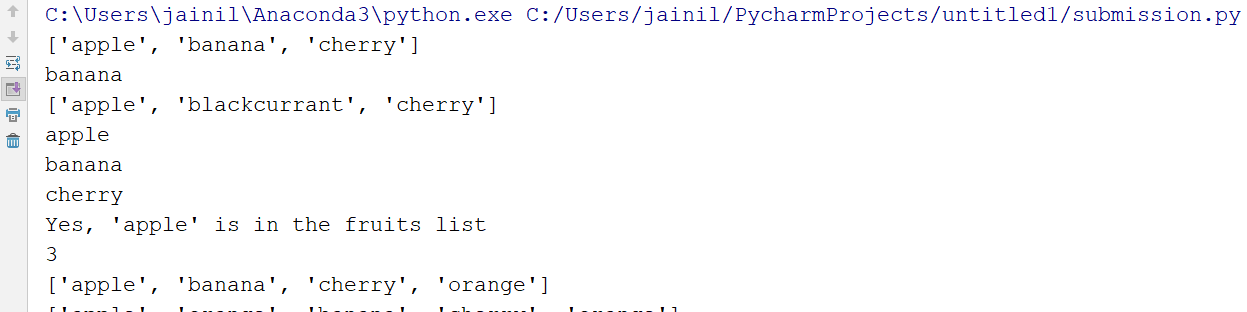


**b) Develop programs to learn different types of structures (list, dictionary, tuples)**

**Code:**

thislist = [**"apple"**, **"banana"**, **"cherry"**]  
print(thislist)  
print(thislist[1])  
thislist[1] = **"blackcurrant"**print(thislist)  
thislist = [**"apple"**, **"banana"**, **"cherry"**]  
**for** x **in** thislist:  
 print(x)  
**if "apple" in** thislist:  
 print(**"Yes, 'apple' is in the fruits list"**)  
print(len(thislist))  
thislist.append(**"orange"**)  
print(thislist)  
thislist.insert(1, **"orange"**)  
print(thislist)  
thislist.remove(**"banana"**)  
print(thislist)  
thislist.pop()  
print(thislist)  
**del** thislist[0]  
print(thislist)  
thislist.clear()  
print(thislist)  
thislist = list((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets*print(thislist)  
  
thistuple = (**"apple"**, **"banana"**, **"cherry"**)  
print(thistuple)  
print(thistuple[1])  
*# The values will remain the same:*print(thistuple)  
**for** x **in** thistuple:  
 print(x)  
**if "apple" in** thistuple:  
 print(**"Yes, 'apple' is in the fruits tuple"**)  
print(len(thistuple))  
thistuple = tuple((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets*print(thistuple)  
thisset = {**"apple"**, **"banana"**, **"cherry"**}  
print(thisset)  
**for** x **in** thisset:  
 print(x)  
thisset = {**"apple"**, **"banana"**, **"cherry"**}  
print(**"banana" in** thisset)  
thisset.add(**"orange"**)  
print(thisset)  
thisset.update([**"orange"**, **"mango"**, **"grapes"**])  
print(thisset)  
print(len(thisset))  
thisset.remove(**"banana"**)  
print(thisset)  
thisset.discard(**"banana"**)  
print(thisset)  
thisset = {**"apple"**, **"banana"**, **"cherry"**}  
x = thisset.pop()  
print(x)  
print(thisset)  
thisset.clear()  
print(thisset)  
thisset = set((**"apple"**, **"banana"**, **"cherry"**)) *# note the double round-brackets*print(thisset)  
thisdict = {  
 **"brand"**: **"Ford"**,  
 **"model"**: **"Mustang"**,  
 **"year"**: 1964  
}  
print(thisdict)  
x = thisdict[**"model"**]  
thisdict[**"year"**] = 2018  
**for** x **in** thisdict:  
 print(x)  
**for** x **in** thisdict:  
 print(thisdict[x])  
**for** x **in** thisdict.values():  
 print(x)  
**for** x, y **in** thisdict.items():  
 print(x, y)  
**if "model" in** thisdict:  
 print(**"Yes, 'model' is one of the keys in the thisdict dictionary"**)  
print(len(thisdict))  
thisdict[**"color"**] = **"red"**print(thisdict)  
thisdict.pop(**"model"**)  
print(thisdict)

**Output:**

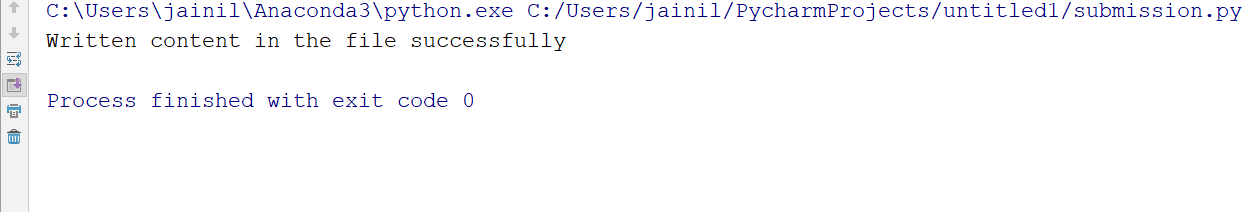


**c) Develop programs to understand working of exception handling and assertions**

**Code:**

fh=open(**"testfile"**, **"w"**)  
**try**:  
 fh = open(**"testfile"**, **"w"**)  
 fh.write(**"This is my test file for exception handling!!"**)  
**except** IOError:  
 print (**"Error: can\'t find file or read data"**)  
**else**:  
 print (**"Written content in the file successfully"**)  
**finally**:  
 fh.close()

**Output:**

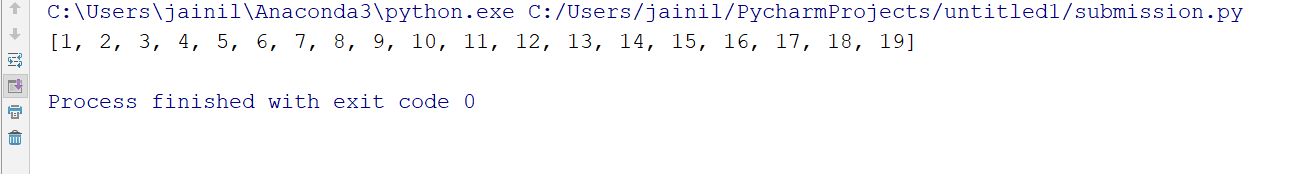


**d) Develop programs to learn concept of functions scoping, recursion and list mut**

**Code:**

l=[]  
**def** a(n):  
 **if**(n<20):  
  
 l.append(n)  
 n+=1  
 a(n)  
 **return** l  
 **else**:  
 **return**print(a(1))

**Output:**



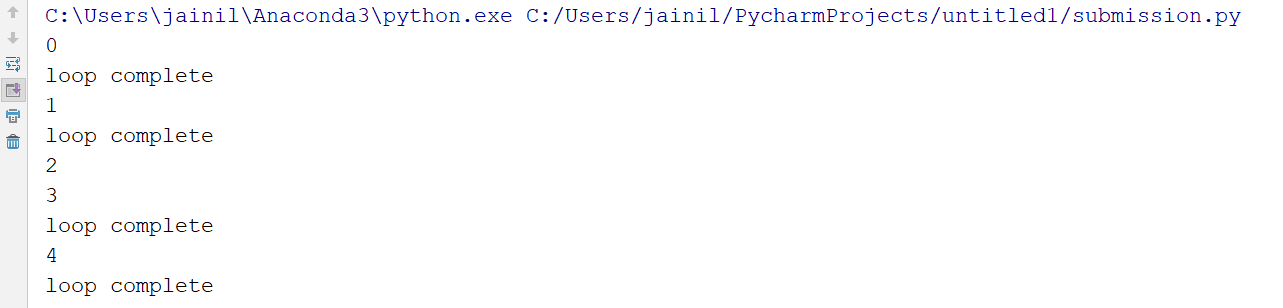
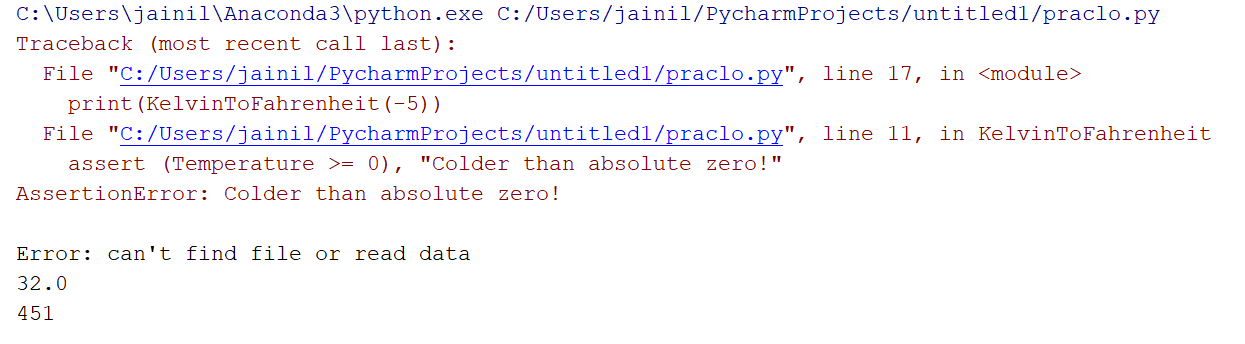
**PRACTICAL 11**

**a) Develop programs to understand working of exception handling and assertions.**

**Code:**

**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:**

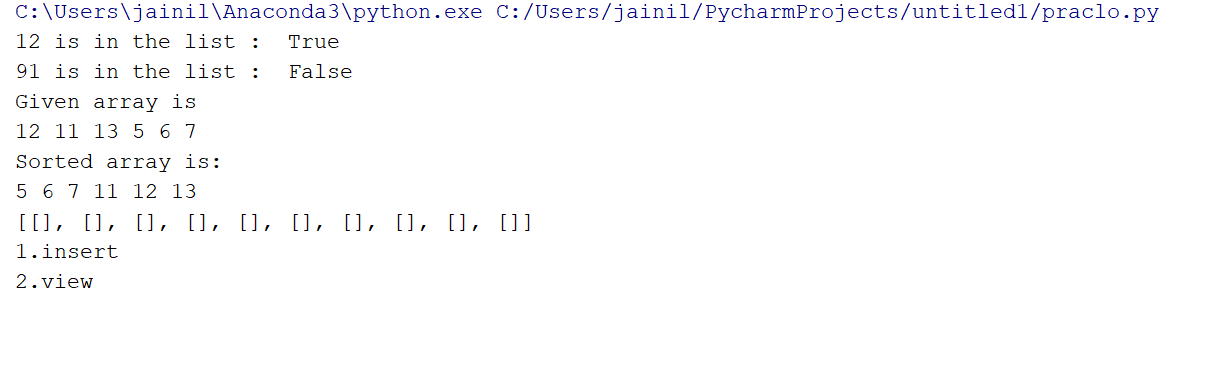


**b) Develop programs for data structure algorithms using python – searching, sorting and hash tables**

**Code:**

*# 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:**

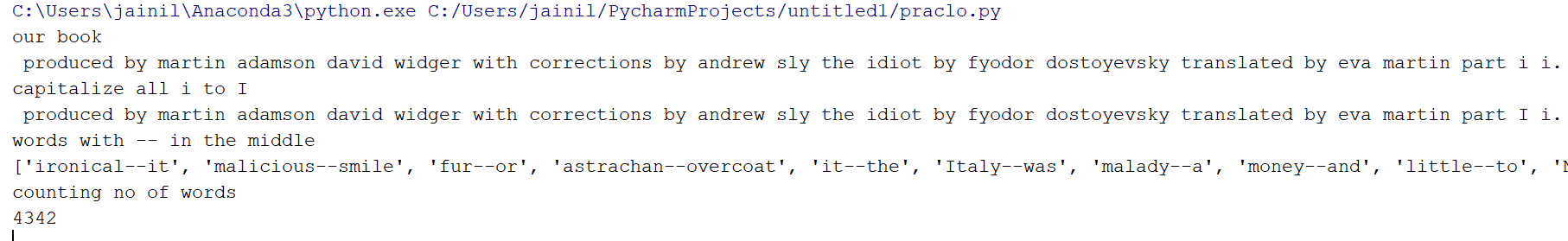


**c) Develop programs to learn regular expressions using python.**

**Code:**

**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:**

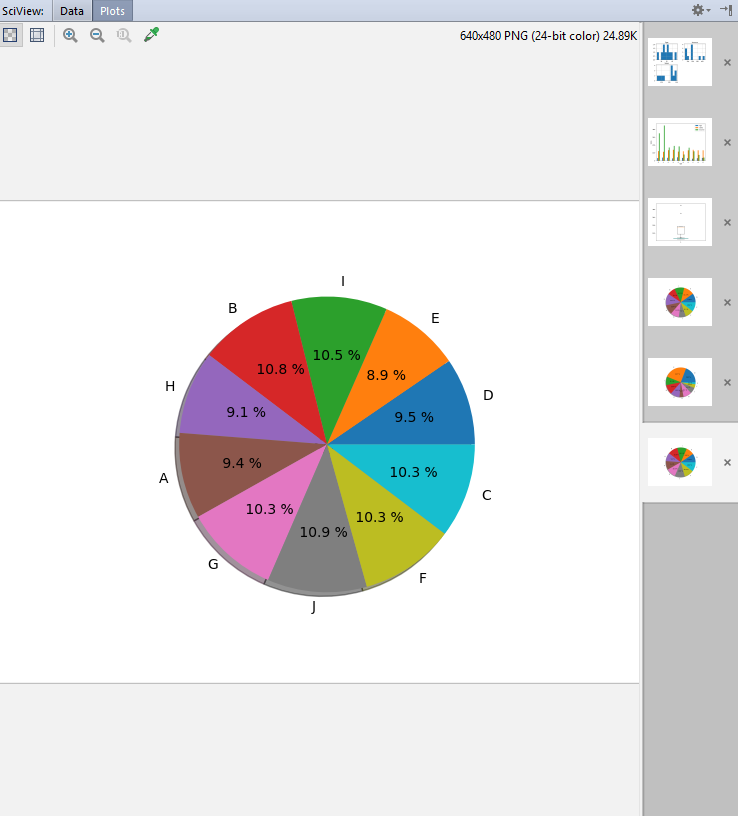


**PRACTICAL 12**

**a) Learn to plot different types of graphs using PyPlot.**

**Code:**

**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()

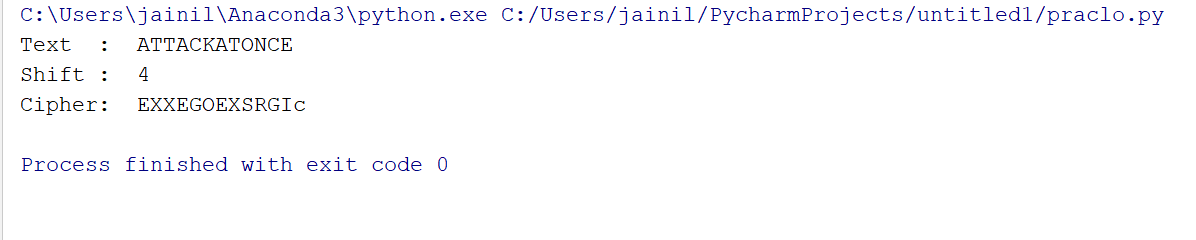
**Output:**

**b) Implement classical ciphers using python.**

**Code:**

*# 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.

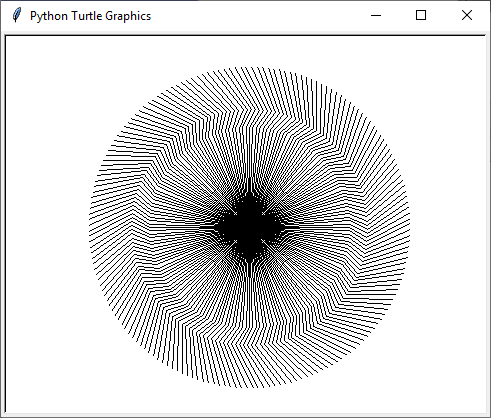
**PRACTICAL 13**

**a) Draw graphics using Turtle.**

**Code:**

**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:**

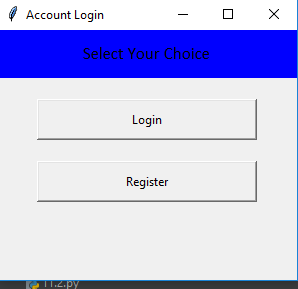


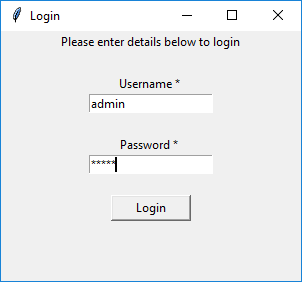
**b) Develop programs to learn GUI programming using Tkinter.**

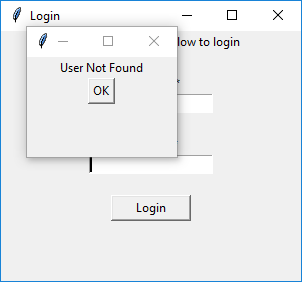
**Code:**

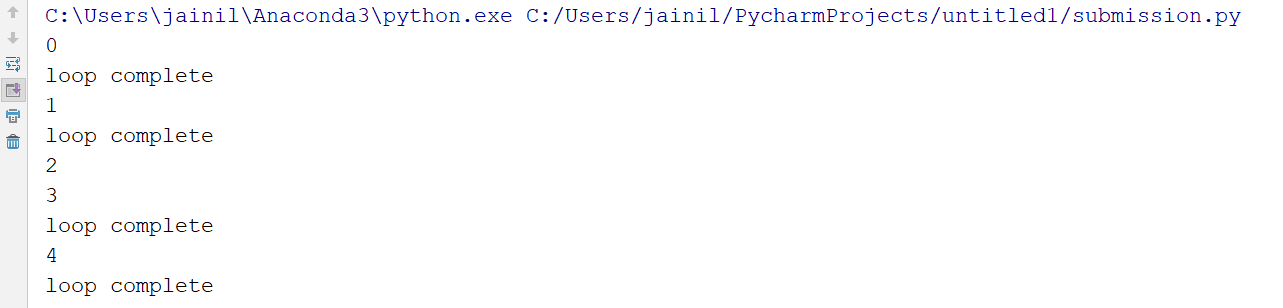
**from** tkinter **import** \*  
**import** os  
  
  
*# Designing window for registration***def** register():  
 **global** register\_screen  
 register\_screen = Toplevel(main\_screen)  
 register\_screen.title(**"Register"**)  
 register\_screen.geometry(**"300x250"**)  
  
 **global** username  
 **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.

**PRACTICAL 14**

Implement the following : 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

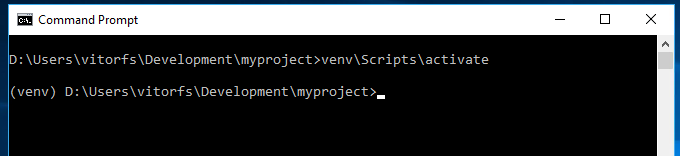
cd myproject

virtualenv venv

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

venv\Scripts\activate

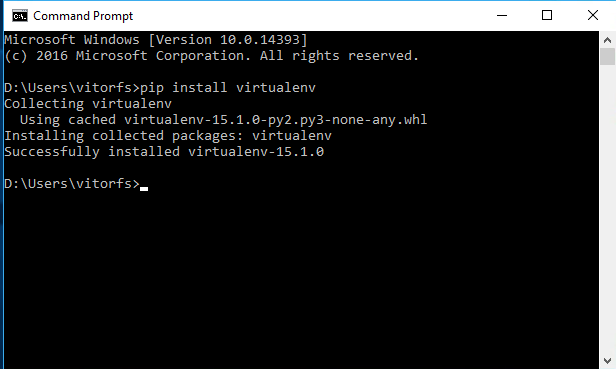
venv\Scripts\deactivate.bat



INSTALLNG VIRTUALENV

In the Command Prompt, execute the command below:

**pip install virtualenv**



INSTALLING DJANGO

**pip install django**

