

Islington College



islington college

Information System

CC4002NA

Coursework 2

Submitted By:

Rajat Shrestha

NP01CP4A170021

Group: L1C2

Date: 12th Jan 2018

Submitted To:

Mr. Sukrit Shakya

Module Leader

Information System

Proposal:

The goal of this section is to provide overall detail on the second coursework assigned by Information System module. The assignment was given out as an individual task in week 7 and requires to be submitted within week 11. This project involves detailed documentation and program itself created to manage a billing and inventory management system in an electronic store.

1. Purpose:

This project requires student to not only develop the code but to document the procedure. Documentation is required so that it becomes more useful for future use or for any other programmer, student or association. The documentation contains detailed information on how the program was developed including the very important algorithm, flowchart and pseudocode so that it can be understood better. The program must also be properly documented by use of comments, should be created in modular manner and should discard invalid inputs.

2. Problem Statement:

This project involves a program which is developed to manage billing and inventory handling of an electronics store. After an item is bought by a customer a unique invoice should be generated for the customer in form of a text file. The inventory should also be updated according to the transactions. This procedure must be properly documented and the program should be simple, modular and should only take valid inputs. The program should run properly without errors and the process should be documented properly.

3. Objective:

The main objective is to develop a program which helps the electronic store to manage the inventory in a text file to displays and sell the available products to a customer. The aim is to develop the program which runs smoothly and interacts with the user in non-confusing manner. The program should also loop if the input is invalid and continue smoothly according to the users input. To get started with this project a valid algorithm should be produced and a flow chart will help greatly. A pseudocode was developed from the algorithm as a prototype to aid on developing the program. After the algorithm and code is written the program is filled with comments so that the code is easier to interpret it to the viewers. The report is being prepared on the development process which includes all the further details about the program. Features integrated in python is also described briefly which is included in the program.

4. Target Audience:

This project is aimed for developers, companies and others who are working on developing similar projects and inventory management programs. This project can also be helpful to students to observe how to complete certain tasks and as an example for similar projects. This project also has a good scope when it comes to developing similar program for inventory management or billing of any kind.

5. Requirements for the program:

As the program is very light weight and doesn't involve any graphical aspects, the hardware and software requirements for running this program is very basic. This program can run on any computer which can run latest version of python (3.6). I recommend using a computer with at least 2GHz of processing power 1GB of ram and 10GB free HDD space.

6. Purposed approach:

To complete the given coursework various steps were taken:

- To understand the assignment
- Gathering information
- Creating algorithm and plans
- Coding
- Documenting
- Testing

7. Progression of the project:

The given task to develop and document the billing system in python required special effort for understanding the problem, developing the solution, crafting codes, testing and solving the existing problems. This resulted the final product that had been created.

The project was started on 1st Jan 2018 and was completed and submitted on 12th Jan 2018. This timeline only shows my active engagement in the process of creating the code and its documentation. As the assignment was given out at week 7, Vital information regarding this project was actively gathered within this time doing various research and practicing python..

<u>Tasks</u>	1	2	3	4	5	6	7	8	9	10	11	12
Algorithm												
Program												
Testing												
Report												
Formatting												

Contents

1. Introduction:	1
2. Discussion and analysis:	3
3. Algorithm:	4
3.1 Flowchart:	5
3.2 Pseudocode:	6
3.2.1. Main:	6
3.2.2. Function:	7
4. Data Structures:	11
8. Program:	13
5.1. Modules:	13
5.2. Check:	14
5.3. Bill:	16
5.4. Main:	17
6. Testing:	18
6.1. Test 1:	19
6.2. Test 2:	21
6.3. Test 3:	22
6.4. Test 4:	24
6.5. Test 5:	26
7. Research:	28
8. Conclusion:	30
References	31

Table of Figures:

Figure 1 : Flowchart.....	5
Figure 2: file to dictionary	13
Figure 3: dictionary to file	13
Figure 4: check yes/no input for new customer	14
Figure 5: check yes/no input for new product	14
Figure 6: checking name validity	14
Figure 7: Checking valid product entry	15
Figure 8: check available amount.....	15
Figure 9: check the discount percent.....	15
Figure 10: Bill (i) using datetime for unique invoice generation.	16
Figure 11:Bill (ii)	16
Figure 12: Main module.....	17
Figure 13: Comparison between types of testing (Khan & Khan, 2012)	18
Figure 14: Running main.py and executing a transaction.....	19
Figure 15: Generated invoice	19
Figure 16: Triggering every invalid input loops.....	21
Figure 17: Initial contents	22
Figure 18: Transaction in progress.....	23
Figure 19: updating the inventory file in text editor and saving it.....	24
Figure 20: Program displays updated details	25
Figure 21: isolating main.py file	26
Figure 22: The program does not execute	26
Figure 23: Python official website.....	28
Figure 24: Google Books.....	29
Figure 25: Google Scholar	29

Table of tables:

Table 1: Test 1	20
Table 2: Test 2	21
Table 3: Test 3	24
Table 4: Test 4	25
Table 5: Test 5	27

1. Introduction:

As this project involves python, it contains several keywords derived from various programming languages and their derivatives. This program also includes some technical jargons which cannot be understood by the viewers in some circumstances. So, to understand this report one must first understand the meanings of these key terms used throughout the report. Some of the essential key words are as given below.

Some keywords:

Modules, function, String literals, Data structures, Python, Dictionary, List, Python Objects, Exception Handling, IDE, Debugging, Testing (Black box, Grey Box, White box), Import and loops.

This project is based on developing a program for an electronic store to manage the inventory and to generate a unique invoice for each customer. The program is created in modular way so that it can easily be modified and checked. The different modules have their respective functions. To be precise there are 3 modules with one or more function in them and a main module is used to invoke the functions in secondary modules to do the specified task. This modular method of coding could be quite useful to integrate new features, find bugs and errors and also makes the code easier to understand for other developers or student. Although this program was developed as an inventory management tool this program will be quite helpful to other students, programmers as well as a software developing company as I have documented all the required information on how this program works and how I made it. Modules are the files which contains code which can be reused. (Martelli, 2006)

The program is developed to run fluidly without errors by integrating python's Exception Handling feature and looping in case of invalid inputs. Exception handling is a method including Exceptions which are inbuilt Python tool to handle errors in a running program (Lutz, 2013). The customer name should be at least 3 characters long not including spaces, the yes/no input accepts only specified values, only products which are available are displayed and can be purchased. The functions which deals with numerical values are also looped until valid value is given and also integrates exception handling feature to loop if value error (caused by entering invalid datatype) occurs. I have used different loops in different functions so that it will loop until the input is valid. This helps the program to avoid general typos and unintended key press, but if the unintended action is somehow valid according to the code then it cannot be undone.

However, to compensate for the above problems, the text file which has inventory data can easily be modified and as the invoice has customer's name and current date and time stamp the error can be easily be traced. To do this I had used python's inbuilt functions to handle files and date/time features.

2. Discussion and analysis:

Before developing the main program, algorithm was designed to fulfill the desired expectations of the program. Then the algorithm was converted into a rough flowchart with basic features of the program. Then a rough code containing all the required feature of the program was made. However, this code was more like a pseudocode as it contained a lot of errors. Two modules were made by help of the faulty code, one handled the inventory and the other generated invoices. Then after these two modules were created the main module was usable if the input was correct. As the code was not good at handling invalid input, another module to handle the issue was created. This module had functions were designed to loop until the valid value was assigned and was invoked every time the user has to input data. Using these three secondary modules the main module was recreated. Finally, comments were used for each module and functions to describe the mechanisms of the program.

To accomplish the goals set by this project, a computer equipped following tools and software were used:

- i. Microsoft Word 2016
Word processing software to prepare the report.
- ii. JetBrains Pycharm
Python IDE for creating the code.
- iii. Python 3.6
Python development kit with libraries and other functions
- iv. Sublime Text 3
Text editor to edit texts and code
- v. notepad
Text editor to edit texts and code
- vi. Microsoft Visio 2007
Professional diagram making software to develop flowcharts

3. Algorithm:

For developing any program, creating an algorithm first will greatly help in the process. Algorithm is the exact number of steps required to solve a given problem (MacCormick, 2012). The algorithm which was created for the code is as follows:

1. Start
2. Read the data stored in inventory.txt and store it in a dictionary.
3. Ask the user if they want to start a transaction with a new client.
4. If Input is yes: go to step-5
Else: go to step-19
5. Input customer name
6. Ask the user to add product to client.
7. If Input is yes: go to step-8
Else: go to step-15
8. Display products
9. Input product name.
10. Display available amount
11. Input customer_amount
12. Subtract customer_amount from the amount in the dictionary.
13. Store product name, customer_amount and rate in customer dictionary.
14. Go to step 6.
15. Input discount percentage
16. Generate a unique invoice for each customer with customer name and current date/time as name
17. Save all the product data purchased by the customer, discounts and final amount in the invoice and save it.
18. Go to step 3
19. Update the "inventory.txt" file from the dictionary by overwriting the details.
20. Stop

3.1 Flowchart:

Flowchart helps us to understand the algorithm by viewing the loops and actions. Pictorial representation of a solution for a problem in an ordered, step by step form is called a Flowchart (Sempf, 2008). As the flowchart utilizes standardized symbols, it is the universal method for sharing algorithms to others due to its ease of use and understanding.

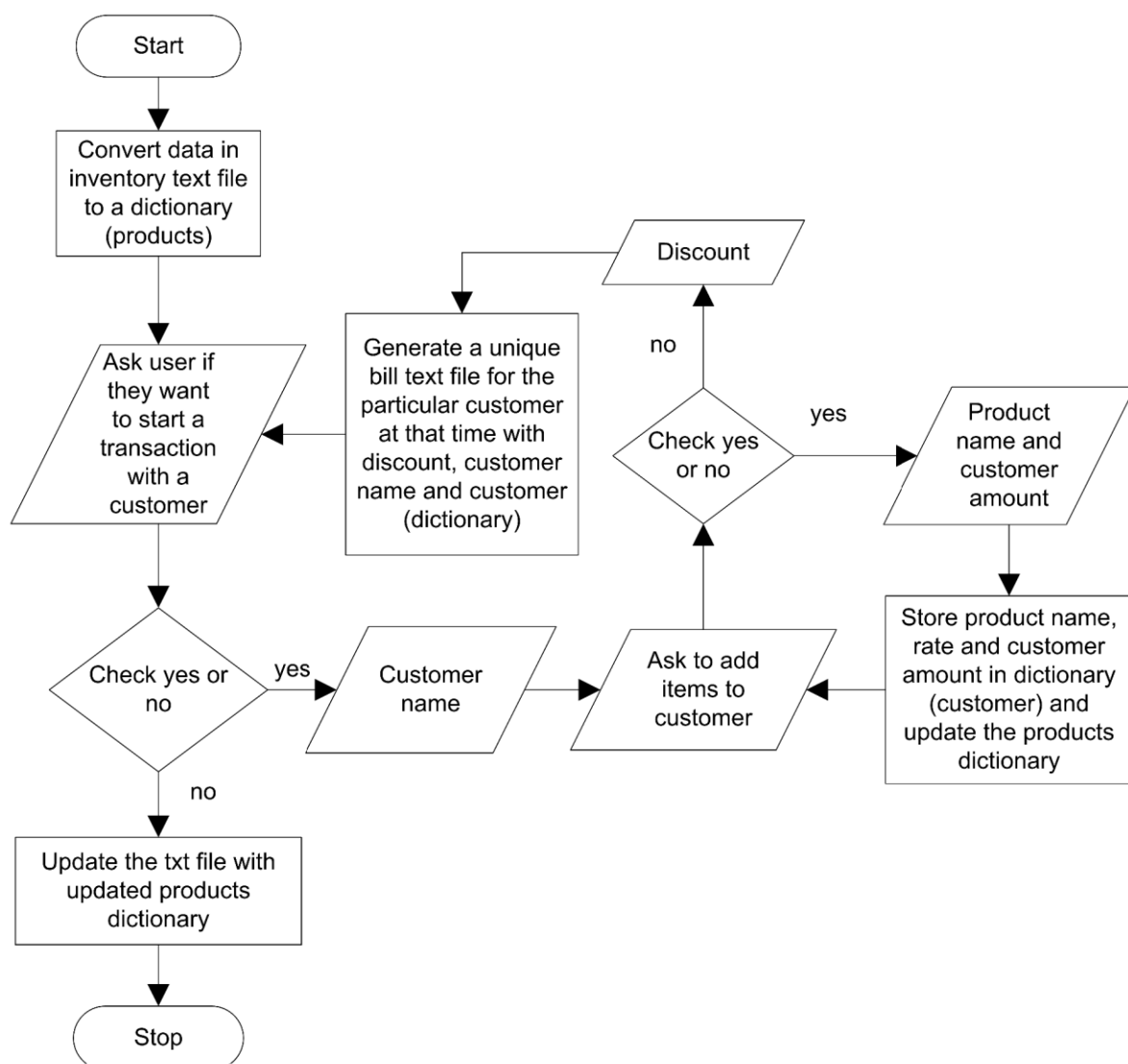


Figure 1 : Flowchart

3.2 Pseudocode:

Pseudocode looks like a code but is not a real programming code. It generic way of describing an algorithm so that it becomes easier for the developer to convert it into a real code (Farrell, 2009). So, to develop the final code, pseudocodes for each function were created and integrated to the main module. The pseudocodes for main python program and for each function are as follows:

3.2.1. Main:

The main code which has to handle all the modules and runs the program smoothly.

```
Import modules
products = modules.file_to_dict(inventory file)
While Check.yes_customer():
    input name
    Check.name(name)
    while Check.yes_product():
        item = Check.product(products)
        display product details
        customer_amount = Check.amount(amount)
        dictionary[amount] -= customer_amount
        customer_dictionary[item] = [price, customer_amount]
    input discount
    Check.discount(discount)

    Bill.bill(name, customer_dictionary, discount)

update inventory file
```

3.2.2. Function:

Each secondary function has its own specific work. 2 functions to read and update the inventory, 1 function to generate bill and 6 functions to return only valid values.

1. file_to_dict(Function to convert information in txt file to a dictionary)

```
file_to_dict(file.txt):  
    open txt file in read mode  
    construct dictionary  
    list = string.split("\n")  
    for i in list:  
        temp = i.split(" ")  
        dictionary key = temp[0]  
        dictionary values = [inventory_amount and rate]  
        display dictionary  
    end for  
    return the dictionary
```

2. dictionary_to_file (function to convert the updated dictionary to file)

```
dictionary_to_file(updated_dictionary, file.txt):  
    open txt file  
    for keys and values in dictionary:  
        write in file (key + " ", "+str(rate)+", "+str(amount)\n)  
    end_for
```

3. yes_customer (Function to loop until valid input is given for new customer)

```
yes_customer():
```

```
    While True:
```

```
        ask if the user wants to start a new transaction
```

```
        return True for 'yes' or 'y'
```

```
        return False for 'no' or 'n'
```

4. yes_product (Function to loop until valid input is given for new product)

```
yes_product():
```

```
    While True:
```

```
        ask if the customer wants to add a new product
```

```
        return True for 'yes' or 'y'
```

```
        return False for 'no' or 'n'
```

5. name ((Function to loop until valid input is given for customer name)

```
name(name):
```

```
    While True:
```

```
        if name is valid (e.g. has 3 characters without spaces)
```

```
            return name
```

```
        end if
```

```
        else: ask to enter a valid name
```

6. product (Function to loop until valid input is given for new product entry)

```
product(dictionary):  
    While True:  
        if dictionary.amount > 0 :  
            display product information from dictionary  
        end if  
        input product name  
        if product is available:  
            return product name  
        end if  
        output (invalid product value)
```

7. amount (Function to loop until valid input is given for product amount with exception handling feature)

```
amount(available_amount):  
    while True:  
        display available_amount  
        ask user to enter desired_amount  
        if 0 <= desired_amount <= available_amount:  
            return desired_amount  
        end if  
        else: display "invalid amount" and loop  
        if value error occurs:  
            display "invalid amount" and loop  
        end if
```


8. discount (Function to loop until valid input is given for discount with exception handling feature)

```
discount():  
    while True  
        input discount_percent  
        if 0<= discount_percent <= 100  
            return discount_percent  
        end if  
        else display error and loop  
        if value error occurs:  
            display "invalid amount" and loop  
        end if
```

9. bill (function to generate unique bill)

```
bill(name, product_dictionary, discount_percent):  
    file = string with name and current date and time  
    new file = file.txt  
    file write date and time  
    file write name  
    total_cost = 0  
    for product_name, [amount, rate] in product_dictionary  
        total = amount*rate  
        write the product name, amount, rate and total  
        total_cost += total  
    end for  
    write discount_percent  
    write "discount_amount :" + (total_cost*discount_percent/100)  
    write "final_total : " + (total_cost - discount_amount)  
    close file
```

4. Data Structures:

Data Structures are provided by Python to store data collectively, they are all different and can be used in various scenarios for different purposes (Cokelaer, 2014). In this program I have used these data structures throughout the codes. The primitive data types and data structures are briefly explained below as it is impractical to describe it in detail.

1. Strings:

Strings are immutable sequence of characters which are indexed like list. (Dawson, 2010). Strings are constructed by quotations “ ” or ‘ ’.

Example, name = “Jack”

2. Lists:

Lists are mutable sequence of data which can consist of any datatypes as it's elements. Lists can be easily indexed and iterated through. Also known as arrays in other programming languages. Lists are constructed by square brackets [] (Bassi, 2016) .

Example: items = [“man”, 56, name]

3. Dictionaries:

Dictionaries are mutable python objects which can be used to assign certain data to its corresponding values. The keys and values of dictionary can be of any datatype and can be easily called. Dictionaries are often known as associative arrays in other languages (Lutz, 2013) and can be constructed by curly brackets { } .

Example: contacts = { “Jack” : 97798412, “Jill” : “67677688” }

5. Tuples:

Tuples are immutable sequence of data which is constructed by brackets (). In this program tuples were used to assign values for efficient code. (Scott, 2015)

Example: $(x, y) = (y, x)$

$(a, b) = (4, 5)$

6. Integers:

Numeric datatype which handles whole datatypes. It can be either positive or negative (Demirov, 2015).

Example: `number = 56`

7. Floats:

Numeric datatype which handles whole datatypes. It can be either positive or negative (Shaw, 2017). Floats are usually accurate up to 15 decimal points.

Example: `float_number = 12.456`

8. Boolean:

Boolean represents truth values. It is either "True" or "False", 0 or 1 in numeric terms (Anon., 2012).

Example: `Bunny = False`

9. Program:

The given program is developed in modular manner so that testing, modifying and debugging would be easy. The program is divided into 4 modules, and 1 module acts as a launcher which calls the functions from other 3 modules to run the overall program. The name, description and screenshots of the programs are given below.

5.1. Modules:

This module is specifically made to read and update dictionary file.

5.1.1. file_to_dict

```
def file_to_dict(filename):
    """
    function to convert text in a .txt file to dictionary

    :parameter: file: opens the specified filename should be specified in (*.txt) format
    :return: Dictionary: returns the values stored in the file in form of dictionary

    """
    dictionary = {}
    File1 = open(filename, "r")
    print("The items available are : ")
    for i in (File1.read()).split("\n"):
        if len(i) > 2:
            temp_lis = i.split(", ")
            print(temp_lis[2], temp_lis[0], " costing : ", temp_lis[1])
            dictionary[temp_lis[0]] = ([int(temp_lis[1]), int(temp_lis[2])])
    File1.close()
    return dictionary
```

Figure 2: file to dictionary

5.1.2. dictionary_to file

```
def dict_to_file(dictionary, filename):
    """
    the function to overwrite updated data from the dictionary to a file

    :parameter: dictionary: the dictionary file with updated data
    :parameter: filename: the filename to store data

    """
    file3 = open(filename, "w")
    for k, v in dictionary.items():
        v1 = v[0]
        v2 = v[1]
        file3.writelines(k + ", " + str(v1) + ", " + str(v2) + "\n")
    file3.close()
```

Figure 3: dictionary to file

5.2. Check:

This module is made for exception handling and returning only valid values.

5.2.1. yes_customer

This module contains functions which loops until a valid value is entered

```
def yes_customer():
    """function to return Boolean value if y,yes,n and no is given as input """
    print("Add new customer?")
    while True:
        input_yes_no = input("Enter yes to add new customer or no to exit program (y/n):")
        if input_yes_no == "yes" or input_yes_no == "y":
            return True

        elif input_yes_no == "no" or input_yes_no == "n":
            return False

        else:
            print("Oops! Try again")
```

Figure 4: check yes/no input for new customer

5.2.2. yes_product

```
def yes_product():
    """function to return True or False: if y,yes,n and no is given as input """
    print("Add new product to customer's bill?")
    while True:
        input_yes_no = input("Enter yes or no (y/n):")
        if input_yes_no == "yes" or input_yes_no == "y":
            return True

        elif input_yes_no == "no" or input_yes_no == "n":
            return False

        else:
            print("Oops! Try again")
```

Figure 5: check yes/no input for new product

5.2.3. name

```
def name(customer_name):
    """a function which returns valid name only (must contain 3 characters
    without space) """
    while True:
        if len(customer_name.strip()) > 2:
            return customer_name
        else:
            print("Invalid name. must contain at least 3 character")
            customer_name = input("Customer name : ")
```

Figure 6: checking name validity

5.2.4. product

```
def product(dictionary):
    """ function with exception handelling feature to display, take input and verify
    the product"""
    available_products = []
    for k, v in dictionary.items():
        if dictionary[k][1] > 0:
            available_products.append(k)
    while True:
        print("The available products are: ", available_products)
        try:
            x = input("Chose from above list :")
            if x in available_products:
                return x
            else:
                print("Oops! Try again")
        except TypeError:
            print("Oops! Try again")
```

Figure 7: Checking valid product entry

5.2.5. amount

```
def amount(available_amount):
    """Function with exception handling feature to return valid amount of
    buyable product"""
    while True:
        try:
            x = int(input("Enter desired amount: "))
            if x <= available_amount:
                if x > 0:
                    return x
                else:
                    print("Oops! invalid amount. Try again")
            else:
                print("Oops! invalid amount. Try again")
        except ValueError:
            print("Oops! invalid amount. Try again")
```

Figure 8: check available amount

```
def discount():
    """Function with exception handelling feature to return valid discount
    percent"""
    while True:
        try:
            x = int(input("Enter discount percentage: "))
            if x < 101:
                if x >= 0:
                    return x
                else:
                    print("Oops! invalid discount value. Try again")
            else:
                print("Oops! invalid discount value. Try again")
        except ValueError:
            print("Oops! invalid discount value. Try again")
```

5.2.6. discount

Figure 9: check the discount percent

5.4. Main:

This module is the main launcher which integrates the functions from above modules into a single program.

```
import Modules
import Check
import Bill

""" The main module which acts like a launcher and interacts with user by help of
    imported modules"""

products = Modules.file_to_dict("Inventory.txt")

while Check.yes_customer():
    name = Check.name(input("Enter customer name: "))
    bill_dictionary = {}

    while Check.yes_product():
        item = Check.product(products)
        (price, amount) = (products[item][0], products[item][1])
        print("the product costs : %s and %s pieces available" % (price, amount))
        customer_amount = Check.amount(amount)
        products[item][1] -= customer_amount
        bill_dictionary[item] = [price, customer_amount]

    discount = Check.discount()
    Bill.bill(name, bill_dictionary, discount)

Modules.dict_to_file(products, "inventory.txt")
```

Figure 12: Main module

The program is constructed with accurate syntax and made to be simple but efficient. It can do all of the specified tasks properly but contains some limitations.

Limitations:

1. If the same customer does two transactions within a minute unique bills would not be generated.
2. If the customer tries to buy items when the inventory is empty, the program will be stuck.

6. Testing:

To see if the program runs fluidly we must go through several tests to find out problems and solve them. Testing ensures the functionality of the program so it is a vital part of any development project. Black box testing is the method to run a program without knowing the internal mechanisms (Whittaker & Thomason, 1994).

*(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 3, No.6, 2012*

Tomorrows' tester will be professionally more educated, examine and accredited professional.

TABLE I. COMPARISON BETWEEN THREE FORMS OF TESTING TECHNIQUES [6] [7]

S. No.	Black Box Testing	Grey Box Testing	White Box Testing
1.	Analyses fundamental aspects only i.e. no proved edge of internal working	Partial knowledge of internal working	Full knowledge of internal working
2.	Granularity is low	Granularity is medium	Granularity is high
3.	Performed by end users and also by tester and developers (user acceptance testing)	Performed by end users and also by tester and developers (user acceptance testing)	It is performed by developers and testers
4.	Testing is based on external exceptions – internal behaviour of the program is ignored	Test design is based on high level database diagrams, data flow diagrams, internal states, knowledge of algorithm and architecture	Internal are fully known
5.	It is least exhaustive and time consuming	It is somewhere in between	Potentially most exhaustive and time consuming
6.	It can test only by trial and error method	Data domains and internal boundaries can be tested and over flow, if known	Test better: data domains and internal boundaries
7.	Not suited for algorithm testing	Not suited for algorithm testing	It is suited for algorithm testing (suited for all)

Figure 13: Comparison between types of testing (Khan & Khan, 2012)

Grey box testing and white box testing was also performed on the initial state of the program to find various problems and to solve them. The modular aspect of the program made testing of individual parts of code fairly easy.

Now as the program is complete, Assuming the user not knowing the inner functionality of the program black box testing was done. Various test cases were taken into account and was tested as if it was used in a real life electronic store starting a transaction with the customer.

6.1. Test 1:

Doing a simple transaction:

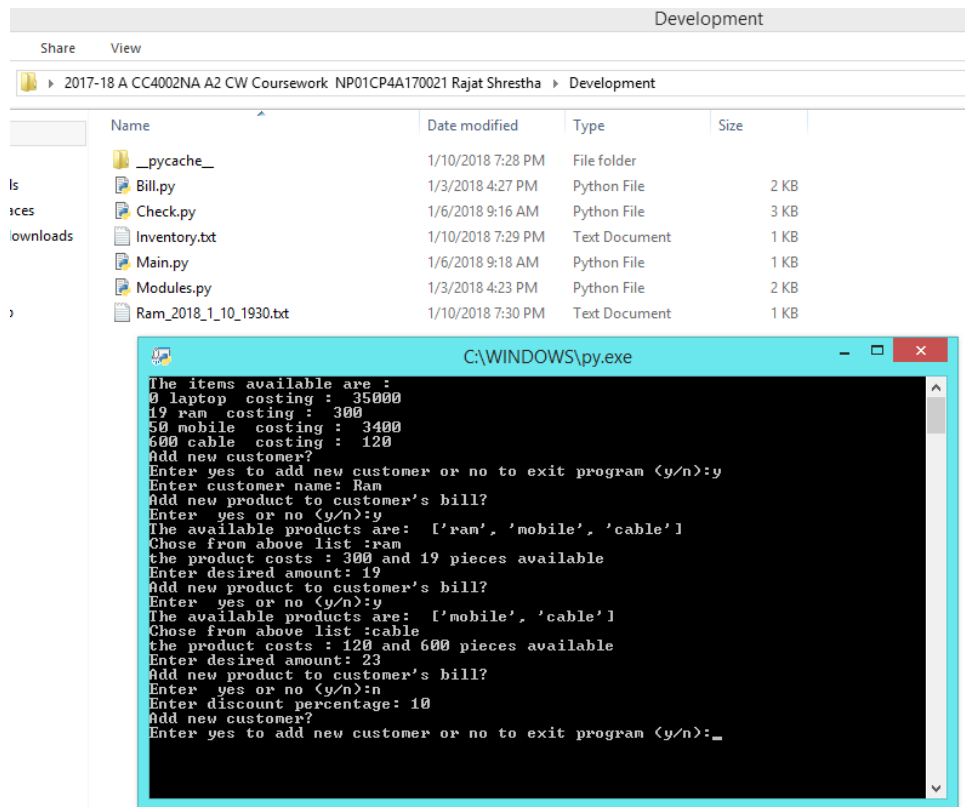


Figure 14: Running main.py and executing a transaction

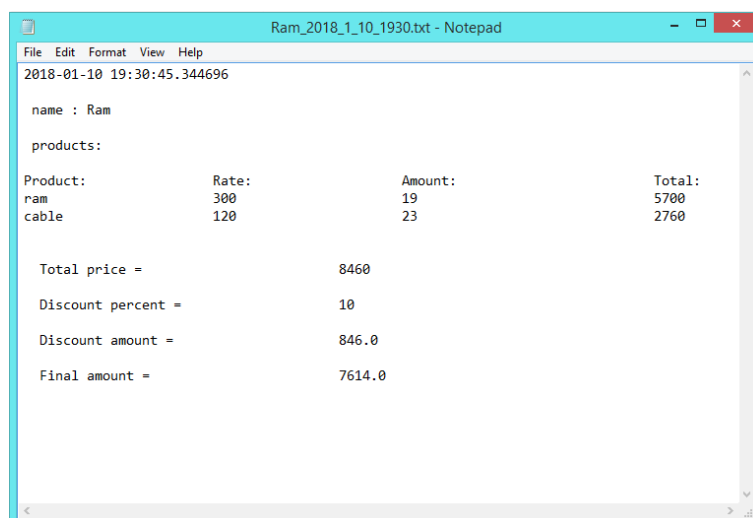


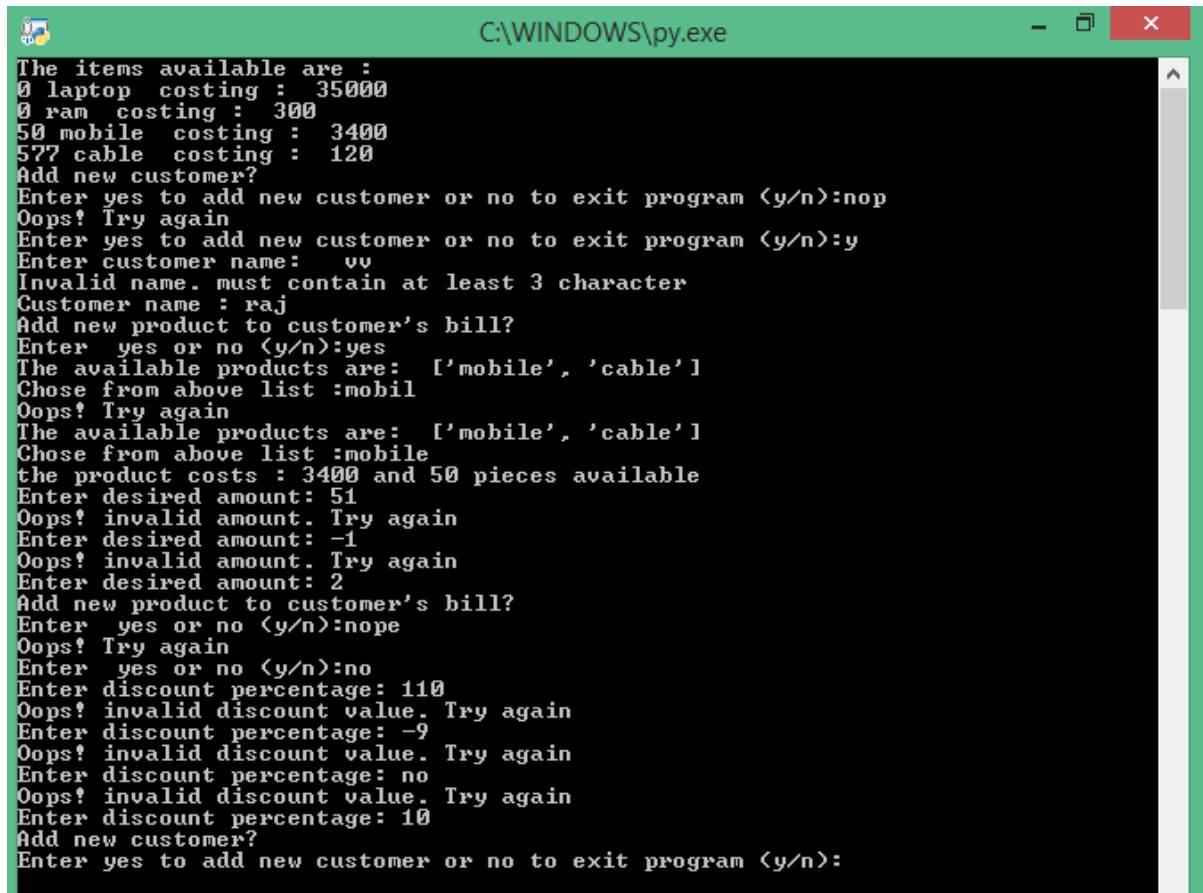
Figure 15: Generated invoice

Table 1: Test 1

Test No	1
Action	Starting the program and going through a simple transaction.
Expected Result	The program will ask for user instruction and execute fluidly without any error. Generation of a simple invoice indicating the vital information about the purchases. The invoice should be unique and should contain necessary information in the development folder.
Actual result	The program was working as expected, taking inputs from the user and doing as instructed. The invoice was also generated with a unique name to ensure none of the invoice is overwritten or updated after the transaction and was stored at the development folder
Test Result	The test was successful as the actual result was identical to expected result

6.2. Test 2:

To observe how the code handles invalid input and exceptions.



```

C:\WINDOWS\py.exe
The items available are :
0 laptop costing : 35000
0 ram costing : 300
50 mobile costing : 3400
577 cable costing : 120
Add new customer?
Enter yes to add new customer or no to exit program (y/n):nop
Oops! Try again
Enter yes to add new customer or no to exit program (y/n):y
Enter customer name: vv
Invalid name. must contain at least 3 character
Customer name : raj
Add new product to customer's bill?
Enter yes or no (y/n):yes
The available products are: ['mobile', 'cable']
Chose from above list :mobil
Oops! Try again
The available products are: ['mobile', 'cable']
Chose from above list :mobile
the product costs : 3400 and 50 pieces available
Enter desired amount: 51
Oops! invalid amount. Try again
Enter desired amount: -1
Oops! invalid amount. Try again
Enter desired amount: 2
Add new product to customer's bill?
Enter yes or no (y/n):nope
Oops! Try again
Enter yes or no (y/n):no
Enter discount percentage: 110
Oops! invalid discount value. Try again
Enter discount percentage: -9
Oops! invalid discount value. Try again
Enter discount percentage: no
Oops! invalid discount value. Try again
Enter discount percentage: 10
Add new customer?
Enter yes to add new customer or no to exit program (y/n):
  
```







Figure 16: Triggering every invalid input loops

Table 2: Test 2

Test No	2
Action	Intentional typos and invalid values were given as input to check how the program handles it.
Expected Result	It was expected to loop until valid value was given.
Actual result	The program looped till it got a valid value.
Test Result	The test was successful.

6.3. Test 3:

Observing if the txt file gets updated after a transaction.

	__pycache__	1/10/2018 7:28 PM	File folder
	Bill.py	1/3/2018 4:27 PM	Python File
	Check.py	1/6/2018 9:16 AM	Python File
	Inventory.txt	1/10/2018 7:47 PM	Text Document
	Main.py	1/6/2018 9:18 AM	Python File
	Modules.py	1/3/2018 4:23 PM	Python File

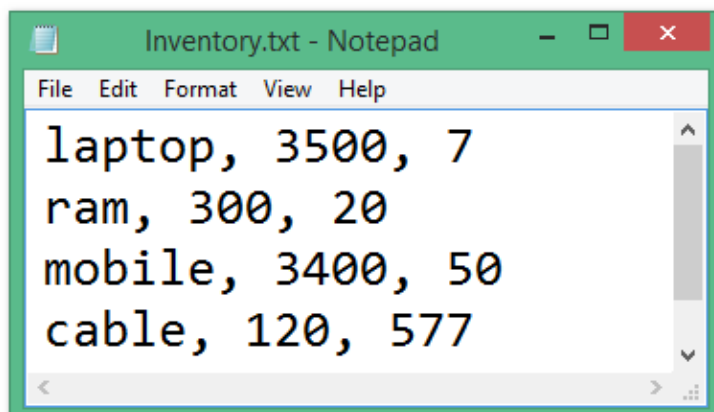


Figure 17: Initial contents

```

C:\WINDOWS\py.exe
The items available are :
7 laptop costing : 3500
20 ram costing : 300
50 mobile costing : 3400
577 cable costing : 120
Add new customer?
Enter yes to add new customer or no to exit program (y/n):y
Enter customer name: jenny
Add new product to customer's bill?
Enter yes or no (y/n):y
The available products are: ['laptop', 'ram', 'mobile', 'cable']
Chose from above list :laptop
the product costs : 3500 and 7 pieces available
Enter desired amount: 7
Add new product to customer's bill?
Enter yes or no (y/n):y
The available products are: ['ram', 'mobile', 'cable']
Chose from above list :ram
the product costs : 300 and 20 pieces available
Enter desired amount: 20
Add new product to customer's bill?
Enter yes or no (y/n):y
The available products are: ['mobile', 'cable']
Chose from above list :mobile
the product costs : 3400 and 50 pieces available
Enter desired amount: 50
Add new product to customer's bill?
Enter yes or no (y/n):y
The available products are: ['cable']
Chose from above list :cable
the product costs : 120 and 577 pieces available
Enter desired amount: 555
Add new product to customer's bill?
Enter yes or no (y/n):n
Enter discount percentage: 0
Add new customer?
Enter yes to add new customer or no to exit program (y/n):_

```

Figure 18: Transaction in progress

Bill.py	1/3/2018 4:27 PM	Python File
Check.py	1/6/2018 9:16 AM	Python File
Inventory.txt	1/10/2018 7:57 PM	Text Document
jenny_2018_1_10_1955.txt	1/10/2018 7:55 PM	Text Document
Main.py	1/6/2018 9:18 AM	Python File
Modules.py	1/3/2018 4:23 PM	Python File

```

Inventory.txt - Notepad
File Edit Format View Help
laptop, 3500, 0
ram, 300, 0
mobile, 3400, 0
cable, 120, 22

```

Figure 18(II): updated inventory file

Table 3: Test 3

Test No	3
Action	Items were bought in a simulation to observe how it affects inventory file.
Expected Result	For the displayed items and inventory file to update according to the transaction.
Actual result	The program discarded the out of stock items while taking input for available product and updated it after each purchase. The inventory file was also updated as expected.
Test Result	The test was successful.

6.4. Test 4:

Adding items in inventory file.

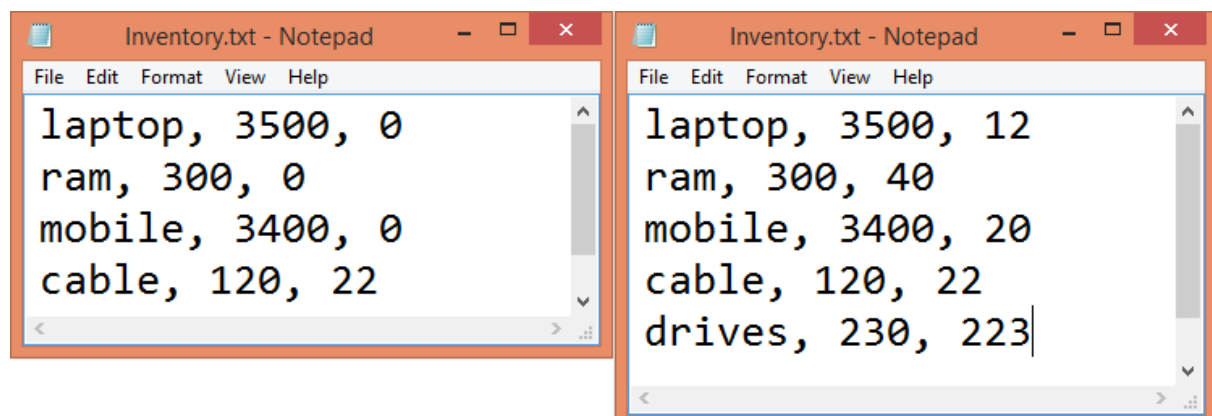
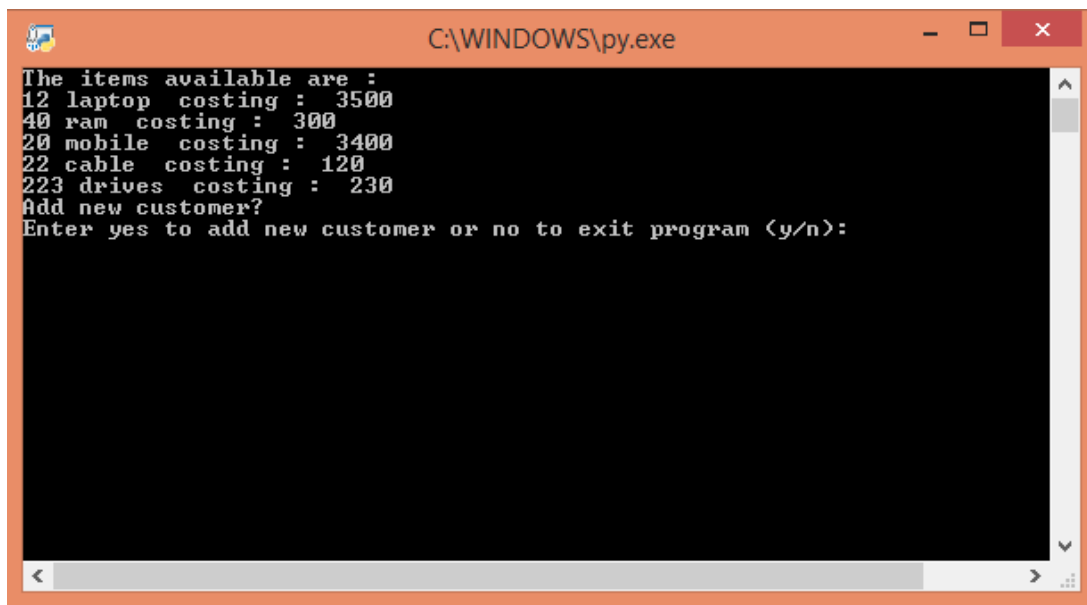


Figure 19: updating the inventory file in text editor and saving it



```

C:\WINDOWS\py.exe
The items available are :
12 laptop costing : 3500
40 ram costing : 300
20 mobile costing : 3400
22 cable costing : 120
223 drives costing : 230
Add new customer?
Enter yes to add new customer or no to exit program (y/n):

```

Figure 20: Program displays updated details

Table 4: Test 4

Test No	4
Action	Inventory file was updated by adding items
Expected Result	The program should display the updated details
Actual result	The program displayed the updated details
Test Result	The test was successful.

6.5. Test 5:

Isolating the main.py from the development folder and trying to run it.

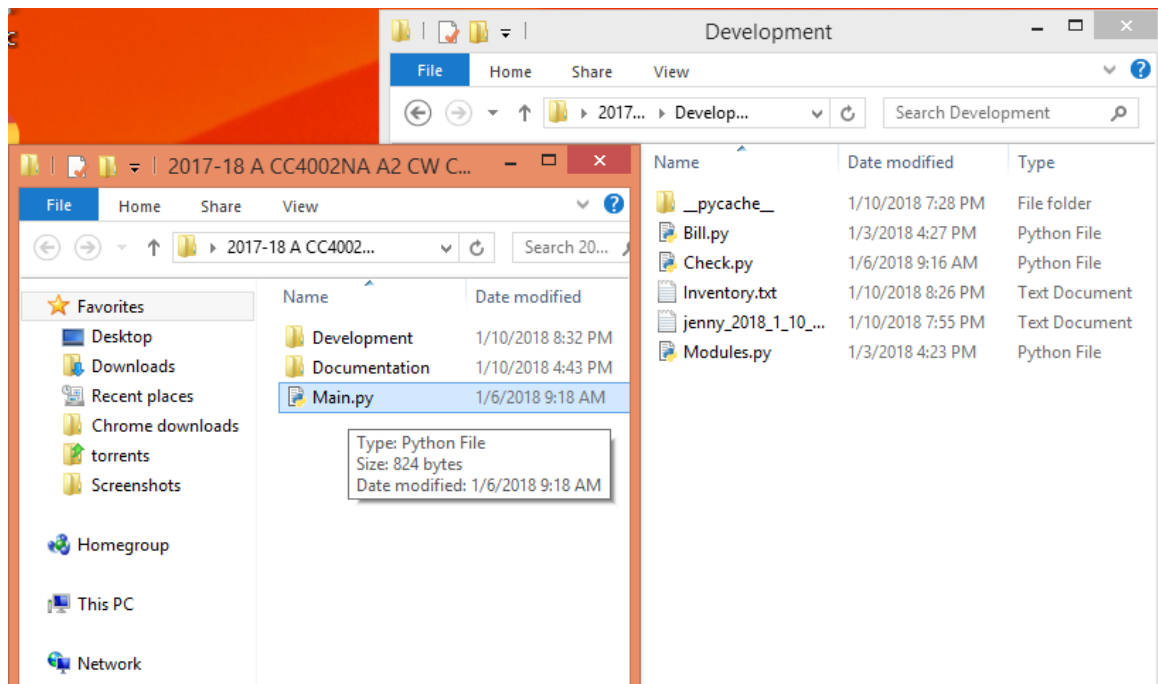


Figure 21: isolating main.py file

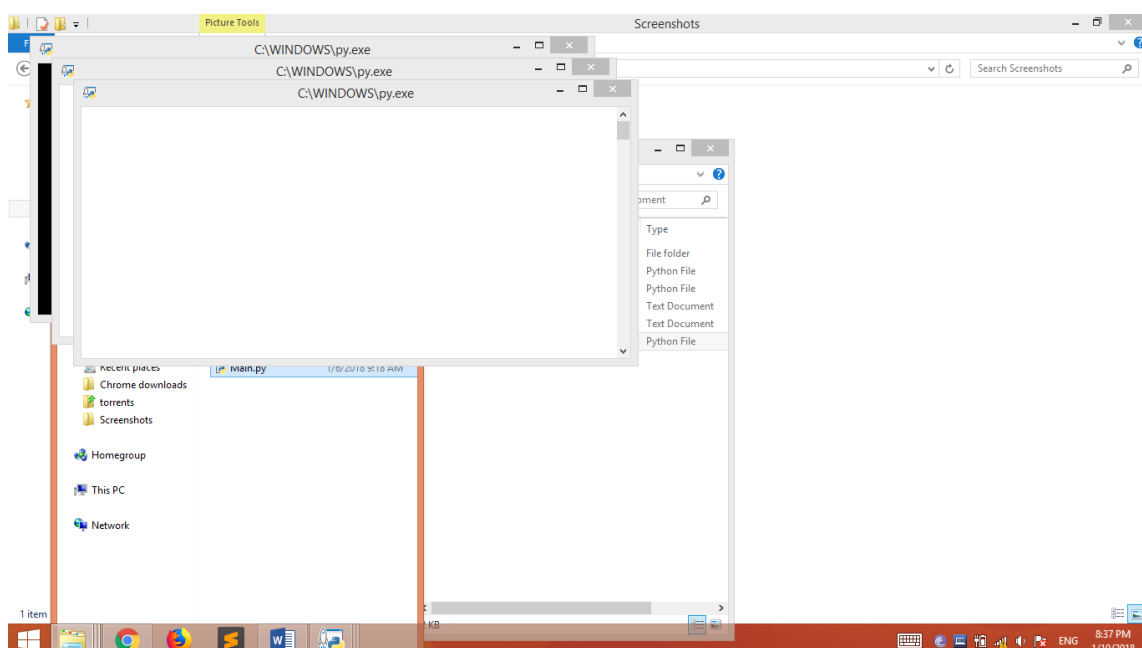


Figure 22: The program does not execute

Table 5: Test 5

Test No	5
Action	The main.py file was moved from the development folder and forced to run.
Expected Result	It was expected for the program to crash
Actual result	The program failed to execute due to missing modules needed for the execution of the program
Test Result	The test was as expected.

The testing of the program was done which met the expectation how the program works. We can conclude that the program works as expected and can be used in various aspects.

7. Research:

This project would be impossible to complete without the help of books, websites and journals containing vital information about the respective subjects. Several books and websites contained information that was required to write the program and journals was also quite helpful. Google Scholar and Google Books were used to provide legit information supporting this project. Providing right information is quite hard nowadays due to the vastness of the internet. As people can express their feelings on the internet it is quite difficult to find reasonable works supported by evidence so the search engines were strictly limited to google scholar and books.

The books, websites and journals are properly referenced at the end of the project to honor the authors for their work which was very helpful. Every idea and works extracted from the referenced materials are properly cited to their respective topics. The previous coursework was also quite useful as a reference material for the formatting and some information on data structures.

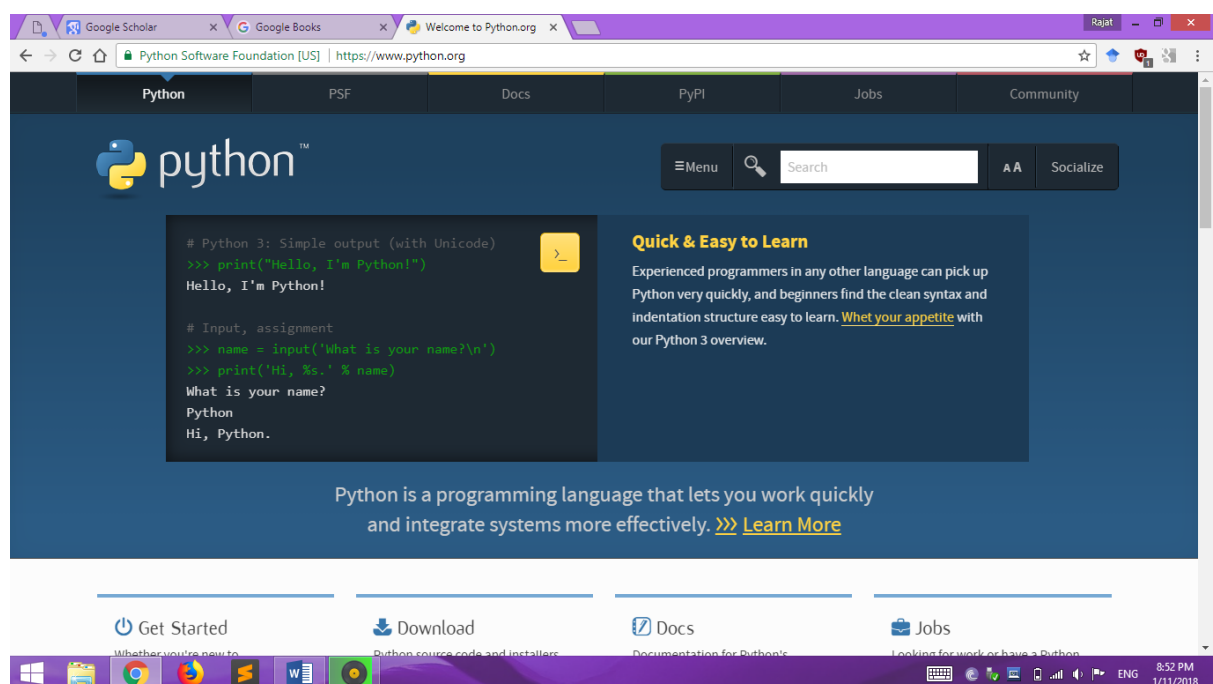


Figure 23: Python official website

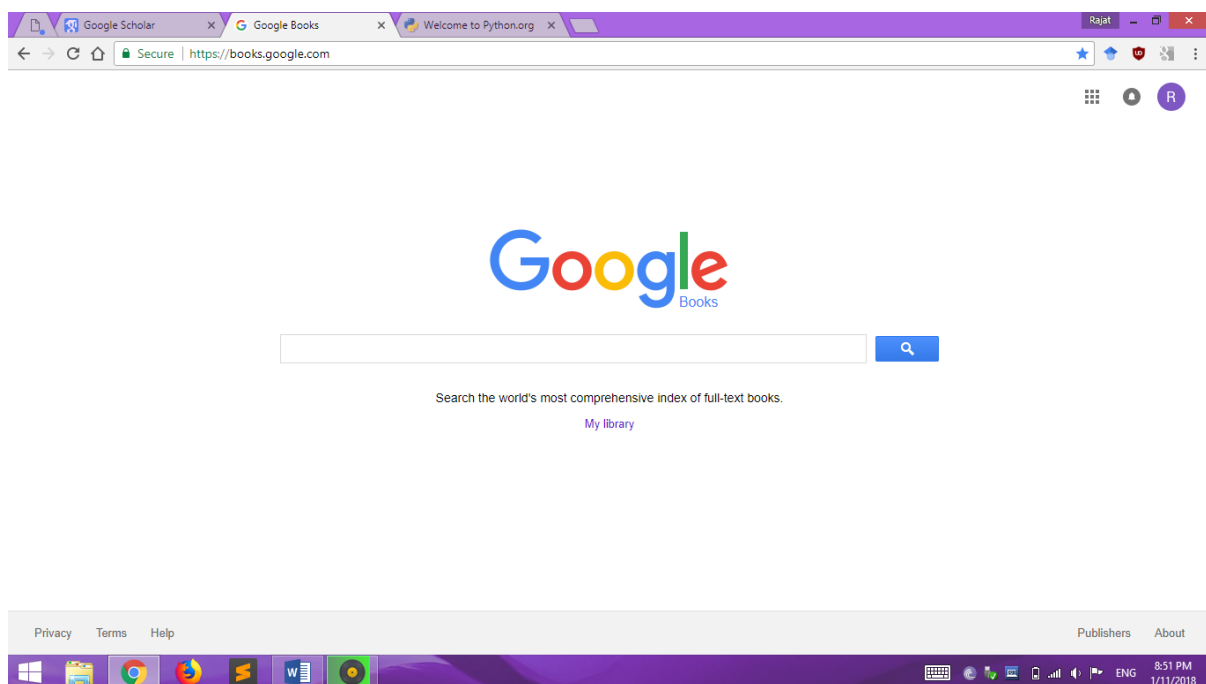


Figure 24: Google Books

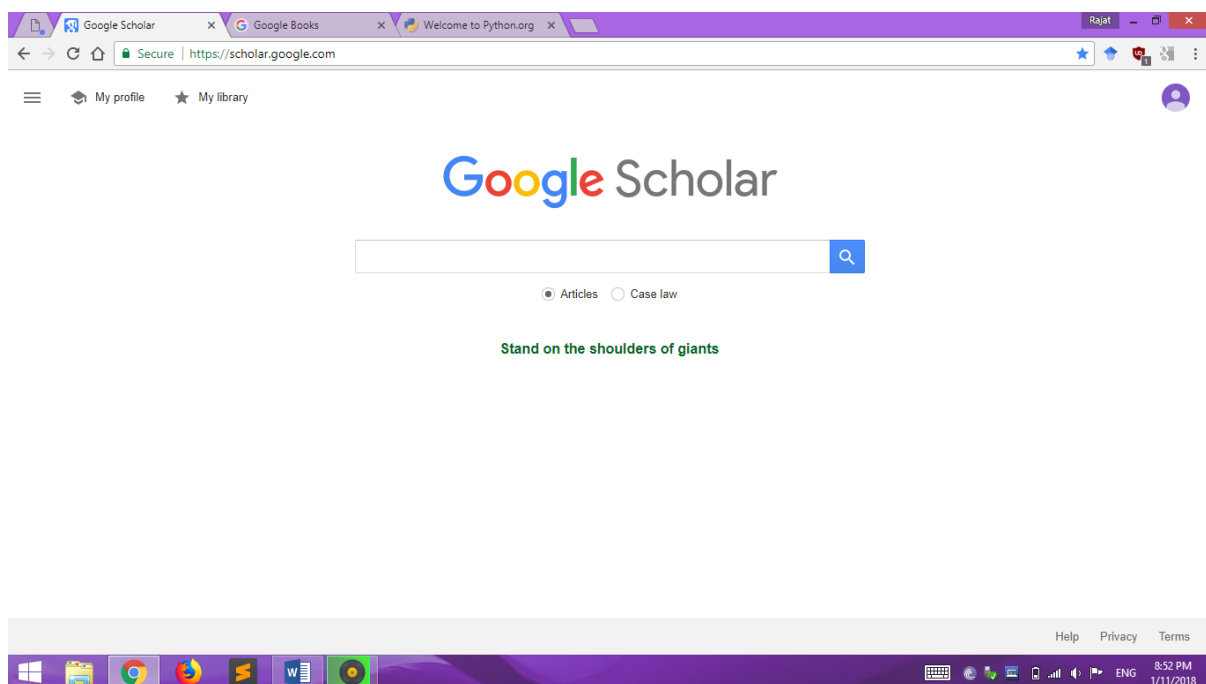


Figure 25: Google Scholar

8. **Conclusion:**

For developing any type of program, small or big, complex or simple; deeper knowledge must be acquired at first. This will help develop the program faster and makes the procedure simpler. Algorithm must be created for simplifying and for management purposes. Flowcharts helps to give the information about the working of the algorithm in depth and development of pseudocode helps greatly when creating a program (Hahn & Valentine, 2007). The development of the program should be well documented as it can help people to understand the mechanisms of the program. The program should not be very complex and messy, must contain comments to indicate their function. The use of comments helps greatly to understand the workings. Coding in modular manner helps to maintain overall structure of the code by making it simpler and easier to debug. Modular functions can also be reused in the codes for better functionality.

To develop the code proper tools should be used for better efficiency. Sublime text is a great text editor for editing simple texts and codes. PyCharm is an IDE made by JetBrains which provides decent interface for writing and testing python code and also points out errors. Finally, python should be installed so that the program can run. Visio was quite helpful on creating flowchart. Therefore, good tools and right information leads to success.

The program was tested extensively to find errors and imperfections and corrected accordingly. This resulted in creation of better code which was gradual and consistent. The modular structure of the code helped very much on white box testing. At the completion of the program black box testing was done.

To complete the project several books, journals and websites were consulted for information on related topics.

References

- Anon., 2012. *Booleans, True or False in Python*. [Online]
Available at: <http://www.pythonforbeginners.com/basics/boolean>
[Accessed 10 1 1018].
- Bassi, S., 2016. *Python for Bioinformatics*. s.l.:CRC Press.
- Cokelaer, T., 2014. 4. *Data Structures (list, dict, tuples, sets, strings) — Python Notes (0.14.0)*. [Online]
Available at: http://www.thomas-cokelaer.info/tutorials/python/data_structures.html
[Accessed 26 12 2017].
- Dawson, M., 2010. *Python Programming for the Absolute Beginner*. 3rd ed. Boston: Cengage Learning.
- Demirov, I., 2015. *Primitive data types — Python book 1.0 documentation*. [Online]
Available at: <http://pythonvisually.com/ebook/primitive-data-types.html>
[Accessed 10 1 2018].
- Farrell, J., 2009. *Just Enough Programming Logic and Design*. 1 ed. Boston: Cengage Learning.
- Hahn, B. D. & Valentine, . D. T., 2007. *Essential MATLAB for Engineers and Scientists*. 3rd ed. Boston: Elsevier.
- Khan, F. & Khan, M. . E., 2012. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 3(6), p. 15.
- Lutz, M., 2013. *Learning Python*. 5th ed. Sebastopol: O'Reilly Media, Inc..
- MacCormick, J., 2012. *Nine algorithms that changed the future : the ingenious ideas that drive today's computers*. 1st ed. Princeton, New Jersey, : Princeton University Press.
- Martelli, A., 2006. *Python in a Nutshell: A Desktop Quick Reference*. 2nd ed. Sebastopol: O'Reilly Media, Inc..

Rossum, v. . G., 1995. *Python Language Document*. [Online]

Available at: <http://www.python.org/>

[Accessed 10 1 2018].

Rossum, v. G., 2018. 8.1. *datetime — Basic date and time types — Python 2.7.14 documentation*. [Online]

Available at: <https://docs.python.org/2/library/datetime.html>

[Accessed 6 january 2018].

Scott, B., 2015. *python for kids for dummies*. 1st ed. Hoboken, New Jersey: John Wiley & sons.

Sempf, B., 2008. *Visual Basic 2008 For Dummies*. 1st ed. Indianapolis: Wiley publishing, Inc..

Shaw, Z. A., 2017. *Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code*. 1st ed. Boston: Addison-Wesley Professional.

Whittaker, J. . A. & Thomason, . M. G., 1994. A Markov Chain Model for Statistical Software Testing. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 20(10), pp. 1,2.