



**Islington college**  
(इस्लिङ्टन कलेज)

**Module Code & Module Title**  
**CC4002NA Information Systems**

**Assessment Weightage & Type**  
**20% Individual Coursework**

**Year and Semester**  
**2019 Spring**

**Student name: Trishna B.C**

**Group: L1C14**

**College ID: NP01CP4S190037**

**London Met ID: 18030816**

**Assignment Due Date: 3<sup>rd</sup> May, 2019**

**Assignment Submission Date: 3<sup>rd</sup> May, 2019**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

## **Proposal**

This proposal is written to address the coursework of Information Systems which is given to us individually. It is about writing python program that handles the sales of the CD store and making bills. The coursework was given in 8<sup>th</sup> week and must be completed and submitted on 11<sup>th</sup> week.

### **Purpose:**

The purpose of this coursework is to create a python program that handles the movie rental system. We need to write algorithm and pseudocode and also make flowchart for the program in python which displays the available movies and the administrator writes the details of the customer and then making bills accordingly. The program must be continued until the administrator decides to do so.

### **Problem statement:**

The problem in this given coursework is to create a program that displays the list of the movies, quantity and price of the movies. Then borrowing and returning the movie and making the bills.

### **Aims and objectives:**

The aim of this coursework is to do the task in the best way possible. To be able to finish this coursework in a best way, lots of research and study is done so we can understand what the coursework is about. The main objective is to have a proper research and mention it and explain it in our own way so we can have better knowledge about the coursework and related topics. So, a lot of hard work is needed in order to complete the task in the given period of time.

**Proposed Approach:**

To do the tasks which is given in this coursework, following approaches will be undertaken:-

- First of all proper research should be done on the related topics such as algorithm, pseudocode, python programming and methods of testing the program.
- An algorithm is written to figure out the way of writing the program.
- Then a pseudocode is required in order to write the program in easier way.
- Flowchart would be drawn to show how the code is executed.
- Then the python program is written.
- At last, the written program would be checked to make sure it has no errors and submit the correct program.

**Scope of the project:**

This project can be very useful to the programmers and students, who are curious about python programming. It can be useful in getting knowledge about python programming and coding in python. It also makes easier to find out the errors and write the code in easy way. It would teach the way by which good algorithm, pseudocode and flowchart can be prepared and utilize to write a good program. It can also be helpful for the one who finds coding difficult.

**Target Audience:**

This coursework can be helpful for the students, programmers, teachers and the one who wants to learn about python. It can help to improve the knowledge and better understanding about python. The main target is to have a better understanding about the CD rental system using python to anyone who wants to know or learn about python which can be helpful for the other organizations as well.

**Hardware And Software Requirements:**

No hardware is required to run the program because everything under this coursework is done in soft copy. So in case of software, we need to install python 3 in our computer system in order to run the program and display the results.

**Activity Description And Timeline:**

This coursework is a bit difficult compared to the first one. Also, the task is required to be completed within 4 weeks. The coursework would be done step by step in order to complete it properly. So, a lot of task must be done from the first week itself.

- In the first week, we need to understand the main purpose of this coursework with a proper research and studies in order to gather some informations.
- In the second week, proposal, introduction and other reports would be completed along with the cover page.
- In the third week, algorithm and pseudocode would be prepared along with the flowchart.
- And lastly, the program would be written and tested to find out if there is any error.

## Contents

1. Introduction .....	8
2. Discussion And Analysis.....	10
2.1 Python: .....	10
3. Algorithm .....	12
4. Pseudocode .....	14
4.1 Main module: .....	15
4.2 Read module:.....	17
4.3 Borrow module:.....	18
4.4 Return module: .....	23
5. Flowchart.....	28
6. Data Structures.....	30
6.1 Collection data types.....	30
6.1.1 Strings: .....	30
6.1.2 Integer: .....	31
6.1.3 Boolean: .....	32
6.1.4 Dictionary: .....	32
7. Program.....	33
7.1 Trishna.py: .....	33
7.2 menu.py.....	33
7.3 borrow.py.....	33
7.4 return.py .....	33
8. Testing.....	34
9. Research .....	36
9.2 Websites:.....	37
1. <a href="https://study.com/academy/lesson/what-is-a-computer-algorithm-design-examples-optimization.html">https://study.com/academy/lesson/what-is-a-computer-algorithm-design-examples-optimization.html</a> .....	37
2. <a href="https://www.programiz.com/article/flowchart-programming">https://www.programiz.com/article/flowchart-programming</a> .....	38
3. <a href="https://www.datacamp.com/community/tutorials/data-structures-python">https://www.datacamp.com/community/tutorials/data-structures-python</a> .....	39
4. <a href="https://developer.rhino3d.com/guides/rhinopython/python-datatypes/">https://developer.rhino3d.com/guides/rhinopython/python-datatypes/</a> .....	40
5. <a href="https://www.guru99.com/black-box-testing.html">https://www.guru99.com/black-box-testing.html</a> .....	41
9.3 Books:.....	42

1. Programming for the absolute beginners by <i>Michael Dawson</i> .....	42
2. Python 3 for Absolute Beginners by <i>Tim Hall and J-P Stacey</i> .....	44
3. A Practical Introduction to Python Programming by <i>Brian Heinold</i> .....	45
4. The Hitchhiker's Guide To Python by <i>Kenneth Reitz and Tanya Schlusser</i> .....	46
9.4 Journals: .....	47
1. PYTHON: A Programming Language For Software Integration and Development by <i>Michael F Sanner</i> .....	47
2. Pseudocode, the Pseudocode Programming Process, and Alternatives to the PPP by <i>Noah Doersing</i> . ....	48
3. Introduction To Python: Data Types by <i>Kranthi Varala</i> .....	49
Bibliography.....	50

## Table Of Figure

<i>Figure 1: Flowchart of movie rental system.....</i>	<i>29</i>
<i>Figure 2: Working of Black box testing .....</i>	<i>34</i>
<i>Figure 3: Algorithm explanation.....</i>	<i>37</i>
<i>Figure 4: Flowchart explanation .....</i>	<i>38</i>
<i>Figure 5: Data Structures.....</i>	<i>39</i>
<i>Figure 6: String data type.....</i>	<i>39</i>
<i>Figure 7: Explanation on String data type .....</i>	<i>40</i>
<i>Figure 8: Black box testing.....</i>	<i>41</i>
<i>Figure 9: Programming for the absolute beginners by Michael Dawson. ....</i>	<i>42</i>
<i>Figure 10: Programming for the absolute beginners by Michael Dawson. ....</i>	<i>43</i>
<i>Figure 11: Python 3 for Absolute Beginners by Tim Hall and J-P Stacey .....</i>	<i>44</i>
<i>Figure 12: A Practical Introduction to Python Programming by Brian Heinold.....</i>	<i>45</i>
<i>Figure 13: The Hitchhiker's Guide To Python by Kenneth Reitz and Tanya Schlusser .....</i>	<i>46</i>
<i>Figure 14: A Programming Language For Software Integration and Development by Michael F Sanner.....</i>	<i>47</i>
<i>Figure 15: Pseudocode, the Pseudocode Programming Process, and Alternatives to the PPP by Noah Doersing. ....</i>	<i>48</i>
<i>Figure 16: Data Types by Kranthi Varala .....</i>	<i>49</i>

## 1. Introduction

This coursework is given to us to write a python program that displays the movie name, price and quantity of the movie and then the administrator writes the details of the costumer who borrows or returns the movie and creates the bill. For this task to be done properly, we need to create an algorithm, pseudocode and flowchart.

To complete this coursework was not an easy task. Lots of hard work and determination was needed. Also, a lot of study and research was needed to do this assignment. The main objective of this coursework is to make an application which will read a text file and display the movies available and then invoice should be generated. The invoice would consist of name of the costumer, name of the movie, date and time of return when the costumer returns a movie. The duration of the movie would be 10 days and in the case of late returning of movie, a fine would be applied on a daily basis. Starting the project from the very first week itself was very challenging. In order to complete the task in the given time, the main objective of this coursework are:

1. To create a program that handles movie rental system.
2. Write algorithm and pseudocode of the program that shows the movie rental system and constructing a flowchart step by step.
3. Write a python program with the help of algorithm, pseudocode and flowchart.
4. Then run the program to check if there is any error.

In order to achieve the objectives mentioned above, lots of study and research was needed on the related topics. First of all, we need to know what this course work is



about and do the task accordingly. Without the help of teachers this course work would have been more difficult. This project was very helpful to understand python programming and how it works properly. It was quite challenging to do this course work in the given time but with the help of teachers it was a lot easier to understand what the course work is about. It took a lot of hard work and determination to complete this project in the given time. So, a lot of knowledge was needed in order to understand the related topics and complete in the given time.

## 2. Discussion And Analysis

The course work was given to write a python program to show the movie rental system. Algorithm, pseudocode and flowchart were necessary in order to write the program in a easier way. It was a difficult task so lots of hard work and research was needed to successfully complete the task in given time. In order to perform the task, python was necessary which is the required programming language in this course work.

### 2.1 Python:

Python is an interpreted, interactive, object-oriented programming language. It provides high-level data structures such as list and associative arrays(called dictionaries), dynamic typing and dynamic binding, modules, classes, exceptions, automatic memory management.etc.. It has remarkably simple and elegant syntax and general purpose programming language. Like any other scripting languages it is free, even for commercial purposes, and it can be run on practically any modern computer. A python program is compiled automatically by the interpreter into platform independent byte code which is then interpreted (*Sanner, 1999*).

Python is a powerful yet easy to use programming language developed by Guido van Rossum, first released over a decade ago in 1991. With Python, you can quickly write a small project. But Python also scales up nicely and can be used for mission-critical, commercial applications (*Dawson, 2003*). Python is an excellent language to learn programming. There are many reasons for this because it is easy to read and fast to write; it doesn't take long to come up with working code that does something meaningful. Python has a very human-friendly syntax, which makes writing elegant code easy. It can be used to write simple, larger and more complex applications with graphic interfaces indistinguishable from the programs you are used to running from your computer's main menu (*Tim Hall, 2009*).

To write a python program, first of all we need to download the latest version of python from its official website: <https://www.python.org/> and then need to install in the computer system. Python comes with a GUI-integrated development environment called IDLE. IDLE is a simple integrated development environment(IDE) that comes with python. It's a program that allows you to type in your programs and run them (*Heinold, 2012*). A development environment is a set of tools that makes writing programs easier. You can think of it as a word processor for your programs. But it's even more than a place to write, save, and edit your work. IDLE provides two modes in which to work: an interactive mode and a script mode (Dawson, Programming for the absolute beginners, 2003).

The program was written in four different modules using different python built-in functions and data structures and those modules were integrated in a single python package. The program was written and executed according to the requirements of this coursework.

### 3. Algorithm

An algorithm is a well- defined procedure that allows a computer to solve a problem. Another way to describe an algorithm is a sequence of unambiguous instructions. The use of the term "unambiguous" indicates that there is no room for subjective interpretation. Every time you ask your computer to carry out the same algorithm, it will do it in exactly the same manner with the exact same result. An algorithm can also be described as a series of logical steps in language that is easily understood (*study.com, 2019*).

An algorithm helps in writing a program in much easier way. It gives idea about how a code should be so that we can write and run the program in much easier way with less chance of bugs and errors. For this coursework, an algorithm was written so that it will help in writing a python program in much easier way. The algorithm written in this coursework to write a python program is presented below:

Step 1: Start

Step 2: Import the text file from python file.

Step 3: Show menu to choose any option: 1, 2 or 3 to borrow, return or exit the program and return back to menu.

Step 4: Press 1

Step 5: Borrow a movie.

Step 6: a) Ask movie ID.

b) Ask for the valid movie ID if movie ID is wrong.

Step 7: a) Ask the name and quantity of the movie.

b) Ask for the valid quantity if quantity of the movie is wrong.

Step 8: Decrease the quantity of the movie that has been borrowed in a new file.

Step 9: Ask the name of the customer.

Step 10: Print name of the movie, price, quantity, customer name and date & time.

Register the name of the customer, date, time, movie name and quantity of the movie.

Step 11: Then return to Step 3.

Step 12: Press option 2 to return the borrowed movie.

Step 13: Ask the name of the customer.

Step 14: Ask the movie that is going to be returned.

Step 15: Increase the quantity of the movie that is going to be returned in the registered file.

Step 16: Return to step 3.

Step 17: Press 3 to exit the program.

Step 18: End

## 4. Pseudocode

Pseudocode is a simple way of writing programming code in English. Pseudocode is not actual programming language. It uses short phrases to write code for programs before you actually create it in a specific language. Once you know what the program is about and how it will function, then you can use pseudocode to create statements to achieve the required results for your program (*study.com, 2019*).

Pseudocode makes creating programs easier. Programs can be complex and long; preparation is the key. To use pseudocode, all you do is write what you want your program to say in English. Pseudocode allows you to translate your statements into any language because there are no special commands and it is not standardized. Writing out programs before you code can enable you to better organize and see where you may have left out needed parts in your programs. All you have to do is write it in your own words in short statements (*study.com, 2019*). A good pseudocode avoids code-like statements: it tries to stay at a higher level than source code in order to increase the efficiency during the design phase and to avoid getting restricted by specific limitations or features of the target programming language. Pseudocode should describe what a design is doing, not how exactly it is going to be implemented. The level of the pseudocode should be low enough to enable the programmer to refine it into source code very quickly (*Doersing, 2019*).

Hence we can say that pseudocode is a simple code which acts like an actual code which can be understood by the user. It helps in creating a program in an easier way. So the pseudocode for writing python program for four different modules which is given in this project is presented below:

**4.1 Main module:**

menu = continue

While menu = continue:

```
import read from read
```

```
print("Choose any one")
```

```
print("Enter 1 to borrow CD")
```

```
print("Enter 2 to return CD")
```

```
print("Enter 3 to exit")
```

```
x = discontinue
```

While x = discontinue

Try:

```
a = int(input())
```

```
x = continue
```

Except:

```
print("Please enter the valid number")
```

End While

If a == 1:

```
print("Borrow a CD")
```

```
import borrow from borrow
```

elif a == 2:

```
print("Return a CD")
```

```
import returnmovie from returnmovie

elif a == 3:

    print("Thank you for using movie rental system.")

    menu=discontinue

else:

    print("Please enter 1, 2 or 3")

End if

End while
```



**4.2 Read module:**

```
def read()

    print("-----")

    print("Movie ID      Movie Name      Price   Quantity")

    print("-----")

    file = open("films.txt", "r")

    b = 1

    c = {}

    d = 1

    for the in file:

        print(" ", b, "\t\t" + line.replace(" ", "\t\t"))

        b = b + 1

    End for

    file1 = open("films.txt", "r")

    for line in file1:

        line = line.replace("\n", "")

        e = line.split(",")

        c[d] = e

        d = d + 1

    End for

    return c
```

**4.3 Borrow module:**

```
import datetime from datetime
```

```
def borrow()
```

```
    Total=0
```

```
    f=""
```

```
    g={}
```

```
    now=datetime.datetime.now()
```

```
    h = False
```

```
    i=False
```

```
    while h==False:
```

```
        name=input("Enter your name")
```

```
        try:
```

```
            int(name)
```

```
            print("Enter a valid name.")
```

```
        except:
```

```
            h=True
```

```
    End while
```

```
    while i==False:
```

```
        j=False
```

```
        while h==True
```

```
try:

    film=int(input("Please enter the movie ID"))

    If 1<=film<=10:

        h=False

    else:

        print("Enter the correct movie ID.")

except:

    print("Enter a valid movie name.")

End while

while h==False

    try:

        k=int(input("Quantity of the movie you want to borrow"))

        h = True

    except:

        print("Enter a valid value.")

End while


file1=open("films.txt","r")

m=1

for line in file1:

    line=line.replace("\n","")
```

```
        g[m]=line.split(",")

        m=m+1

    End for

    o=1

    while o<=10:

        if film==o:

            p=g[o]

            If int(p[2])<=10:

                print("Sorry! The movie is out of stock.")

            elif int(p[2])<=i:

                print("Sorry! There is only",int(p[2]),"movie in stock.")

            else:

                Price=float(p[1]).replace("$","")*k

                Total=Total+Price

                remaining=int(p[2])-k

                b=1

                d=""

                File2=open("films.txt","r")

                for line in File2:

                    if film==b:
```

```
        c=line.replace(m[2]),str(remaining)

        d=d+c

    else:

        d=d+line

    End if

    b=b+1

End for

File3=open("films.txt","q")

File3.write(d)

End if


while j==False:

    s=input("Would you like to borrow more movies?(Y\N)")

    f=f+m[0]+", "

    if s.lower()=="u":

        Bill=open("borrow.txt","q")

        Bill.write(name+", "+f+", "+str(now)+", "+ "$"+str(Total))

        print("Name of the borrower = "+name)

        print("Name of the movie = "+f)

        print("Total cost = "+Total)
```

```
print("Time = "+now.strftime("%Y-%M-%D  
%H:%M:%S"))
```

```
j=True
```

```
i=True
```

```
elif s.lower()=="c":
```

```
h=True
```

```
j=True
```

```
i=False
```

```
else:
```

```
print("Please enter proper value.")
```

```
End if
```

```
End while
```

```
o=o+1
```

```
End while
```

```
End while
```

#### 4.4 Return module:

Import datetime

def returnmovie():

    now=datetime.datetime.now()

    h=False

    i=False

    f=" "

    g={}

    days=10

    fine=1

    total\_fine=0

    while h==False:

        name=input("Enter your name = ")

        try:

            int(name)

            print("Enter a valid name.")

        except:

            h=True

    End while

    while i==False:

        j=False

```
while h==True:

    try:

        film=int(input("Enter your movie ID"))

        if 1<=film<=10:

            h=False

        else:

            print("Enter a proper movie ID.")

        End if

    except:

        print("Enter a valid movie ID")

End while

while h==False:

    try:

        k=int(input("Quantity of movie you want to return"))

        h=True

    except:

        print("Enter a valid value.")

End while

file1=open("films.txt","r")

s=1

for line in file1:
```



```
        line=line.replace("\n"," ")

        g[m]=line.split(",")

        m=m+1

    End for

    p=d[film]

    remaining=int(p[2])+k

    b=""

    s=1

    File2=open("films.txt","r")

    for line in File2:

        If film==b:

            c=line.replace(p[2],str(remaining))

            d=d+c

        else:

            d=d+line

        End if

        s=s+1

    File3=open("films.txt","q")

    File3.write(d)

    End while
```

```
t=int(input("The number of days you rented this movie for"))

if t>days:

    q=t-days

    total_fine((fine)*(q))

    print("Dear customer, since you are",q,"days late, you will have to
pay a fine of $", total_fine)

else:

    print("Thank you for returning the CD in time")

End if


while j==False:

    s=input("Would you like to return any more movies?")

    f=f+m[0]+","

    if s.lower()=="u"

        return name

        return film

        return k

        return m[0]

        return now.strftime

        Return=open("return.txt","q")

        Return.write(name+","+m[0]+","+str(now)+",")
```

```
        j=True

        i=True

    elif s.lower()=="c":

        j=True

        i=False

        h=True

    else:

        print("Please enter proper value")

    End if

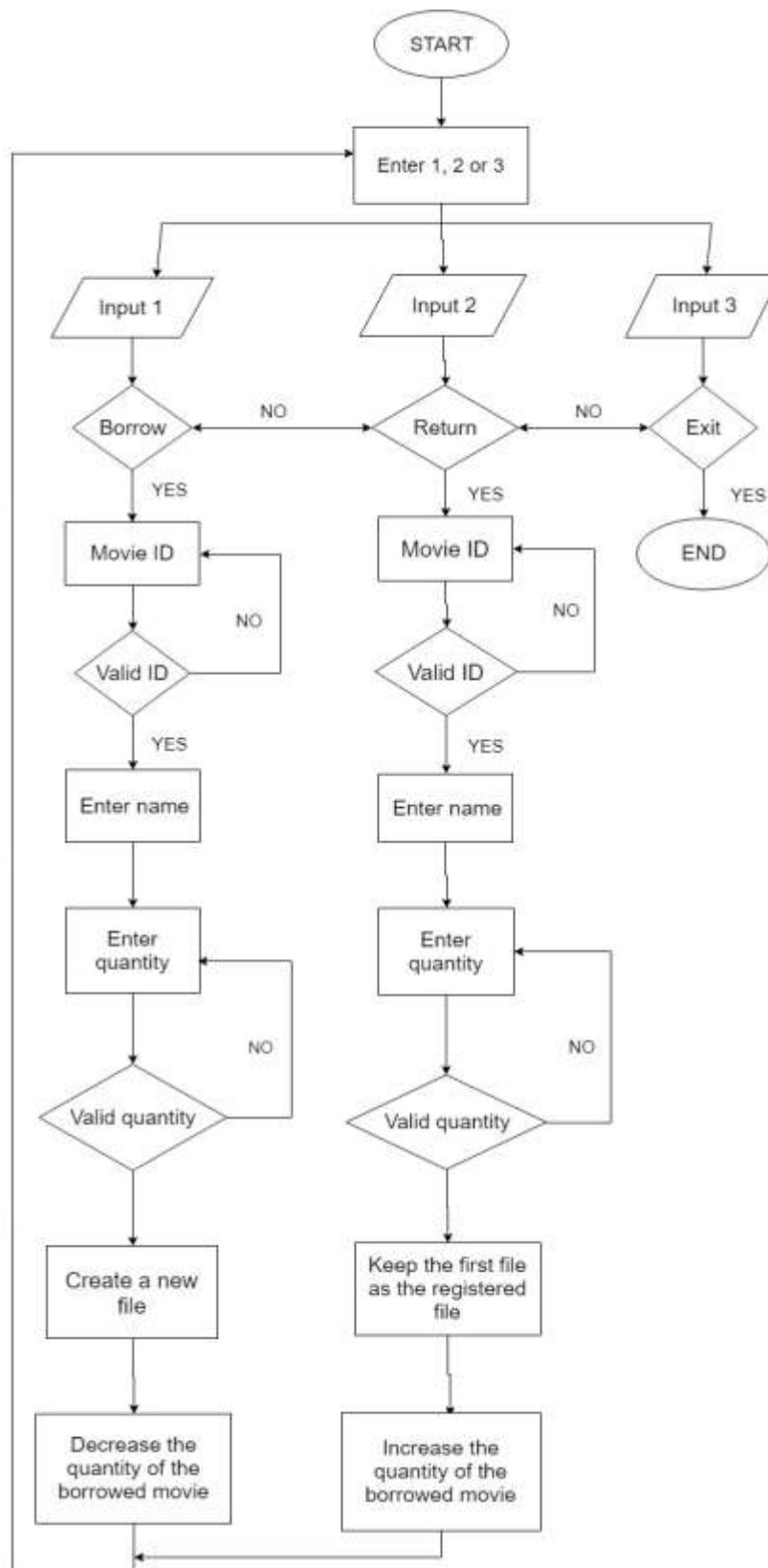
End while

End while
```

## 5. Flowchart

Flowchart is a diagrammatic representation of an algorithm. Flowchart are very helpful in writing program and explaining program to others. Different symbols are used for different states in flowchart, For example: Input / Output and decision making has different symbols (*programiz.com, 2019*).

Flowchart can simply be understood as the representation of an algorithm which can be explained with the help of a certain diagram / figure connected by arrows.



**Figure 1: Flowchart of movie rental system**

## 6. Data Structures

Data structures are a way of organizing and storing data so that they can be accessed and worked with efficiently. They define the relationship between the data, and the operations that can be performed on the data (*Jaiswal, 2017*). This helps in performing various operations easily so that we do not have to face any difficulties while performing the operation.

### 6.1 Collection data types

Data types are the classified form of data which tells us how to use the data while performing an operation. In python, there are many data types from which we can collect data. Some of the primitive data types and collection data types which are used in this project are as follows:

1. Strings
2. Integer
3. Boolean
5. Float
4. Dictionary

#### 6.1.1 Strings:

A string is a 'sequence', i.e., it's a list of items where each item has a defined position. Each character in the string can be referred, retrieved and modified by using its position. This order starts with an 'index' and always starts with 0. String objects support concatenation and repetition operations (*Varala, 2019*).

Strings are collection of alphabets, words or other characters. In Python, you can create strings by enclosing a sequence of characters within a pair of single or double quotes (Jaiswal, 2017). They are immutable which means it cannot be changed after they are created. Since strings can't be changed, we construct new strings as we go to represent computed values like if there is ('hello'+'there'), it takes in the two strings and builds a new string 'hellothere' (Google Developers, 2018). Some examples of string are:

```
x = "Hello"
```

```
y = 'Nepal'
```

```
x + x = "HelloHello"
```

```
y * 2 = 'NepalNepal'
```

### 6.1.2 Integer:

Integers are used to store numeric values. They are immutable data types which means that changing the value of a number data type results in a newly allocated object. We can use an integer represent numeric data, more specifically, whole numbers from negative to infinity, like 4, 5 or -1 (Jaiswal, 2017). Some examples of integer are:

```
x = -1
```

```
y = 0
```

```
z = 4
```

```
If a = "5" then int(a)
```

In the module **Trishna.py**, an integer  $q = 1$  &  $u = 1$  are assigned. The value assigned  $q = 1$  reads line 1 from the file **films.txt** and replaces "," and inserts tab and when the value of  $q$  increases to number 2 then it will read line 2 and vice versa. It will continue until the line is complete. The value assigned  $u = 1$

### 6.1.3 Boolean:

The built – in data type that can take up the values: True and False, which often makes them interchangeable with the integers 1 and 0. Booleans are useful in conditional and comparisons (Jaiswal, 2017). Some examples of Boolean are:

```
x = False #returns False as x is False.
```

```
y = True #returns True as y is True
```

```
x = 5 & y = 6 then print(bool(x)) #returns False as x is not equal to y.
```

### 6.1.4 Dictionary:

Dictionaries in python are list of key : values pairs. This is a very powerful datatype to hold a lot of related information that can be associated through keys. The main operation of a dictionary is to extract a value based on the key to access its members. They are also used to sort, iterate and compare data (*Fugier, 2018*). The key must be unique while inserting a value. It is denoted by curly brackets "{ }". Some examples of dictionary are:

```
A = {"Name" : Ram, "Age" : 15}
```

Here, name and age are key, and Ram and 15 are the values of dictionary.



## 7. Program

In this project, we need to create a python program which reads a text file and displays movie name, price and quantity of the movie and then the administrator writes customer name and creates a bill. It was not an easy task to write a python program for this project. There were many errors and it took a lot of time. But by debugging the errors, it was easy to understand the code and complete it successfully in the given span of time.

To write python code, it took a lot of time to successfully complete. So, a lot of research and study was done in the related topic in order to understand more clearly to complete the coursework without any errors. First of all, an algorithm was written to understand and write a code in easier way. Then a flowchart was made on the basis of the algorithm to show how the program is going to run and the pseudocode was created to take a look at rough version of the actual program which is going to help in writing the program easily. Finally, a python program was written with the help of algorithm, flowchart and pseudocode. The code was very lengthy so it was divided into different modules to understand the code in a better way.

The python program was divided in four different modules which are listed below:

1. Trishna.py
2. menu.py
3. borrow.py
4. return.py

### 7.1 Trishna.py:

### 7.2 menu.py

### 7.3 borrow.py

### 7.4 return.py

## 8. Testing

Testing code is very important. Getting used to writing testing code and running this code in parallel is now considered a good habit. Used widely, this method helps you define more precisely your code's intent and have a more decoupled architecture (*Kenneth Reitz, 2018*). There are various types of testing such as gray box, black box, white box, etc. Since, this coursework asks about black box so some description about black box is given below:

Black box testing is defined as a testing technique in which functionality of the Application Under Test(AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications.



**Figure 2: Working of Black box testing**

In black box testing, we just focus on inputs and output of the software system without bothering about internal knowledge of the software program. The above black-box can be any software system you want to test. For example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under black box testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation (*guru99.com, 2019*).

Using black box testing, various test cases were designed and carried out which are presented below:

## 9. Research

To complete all the tasks given in this course work, lots of difficulties were seen. But, due to the constant effort and research on various related topics made the task a lot easier. This coursework would not have been completed without various research and studies. It was quite difficult to do this task without proper research because we get to know a lot of knowledge about the related topics. So, research plays an important role while doing the projects which will help in gaining knowledge and creativity. To complete this project, a lot of research was done to make it better.

So, a lot of websites, books and journals were consulted for getting information about the related topics. Some of them are presented below:

## 9.2 Websites:

1. <https://study.com/academy/lesson/what-is-a-computer-algorithm-design-examples-optimization.html>

## What Is an Algorithm?

Consider how you use a computer in a typical day. For example, you start working on a report, and once you have completed a paragraph, you perform a spell check. You open up a spreadsheet application to do some financial projections to see if you can afford a new car loan. You use a web browser to search online for a kind of car you want to buy.

You may not think about this very consciously, but all of these operations performed by your computer consist of algorithms. An **algorithm** is a well-defined procedure that allows a computer to solve a problem. Another way to describe an algorithm is a sequence of unambiguous instructions. The use of the term 'unambiguous' indicates that there is no room for subjective interpretation. Every time you ask your computer to carry out the same algorithm, it will do it in exactly the same manner with the exact same result.

Consider the earlier examples again. Spell checking uses algorithms. Financial calculations use algorithms. A search engine uses algorithms. In fact, it is difficult to think of a task performed by your computer that does not use algorithms.

## How Do Algorithms Work?

Let's take a closer look at an example.

A very simple example of an algorithm would be to find the largest number in an unsorted list of numbers. If you were given a list of five different numbers, you would have this figured out in no time, no computer needed. Now, how about five million different numbers? Clearly, you are going to need a computer to do this, and a computer needs an algorithm.

Below is what the algorithm could look like. Let's say the input consists of a list of numbers, and this list is called L. The number L1 would be the first number in the list, L2 the second number, etc. And we know the list is not sorted - otherwise, the answer would be really easy. So, the input to the algorithm is a list of numbers, and the output should be the largest number in the list.

The algorithm would look something like this:

**Figure 3: Algorithm explanation**



Through this website, some knowledge about algorithm and how it works was gained. The explanation is simple because it can be easily understood.

## 2. <https://www.programiz.com/article/flowchart-programming>

Flowchart is a diagrammatic representation of an algorithm. Flowchart are very helpful in writing program and explaining program to others.

### Symbols Used In Flowchart

Different symbols are used for different states in flowchart, For example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.

**Figure 4: Flowchart explanation**

From this website, I got gain some information about flowchart more precisely. The explanation is simple and easily understood. It gives some basic knowledge about flowchart, what kind of diagram should be used for different purposes and its functions. Also, it provides some knowledge about how a flowchart should be done.

### 3. <https://www.datacamp.com/community/tutorials/data-structures-python>

Data structures are a way of organizing and storing data so that they can be accessed and worked with efficiently. They define the relationship between the data, and the operations that can be performed on the data. There are many various kinds of data structures defined that make it easier for the data scientists and the computer engineers, alike to concentrate on the main picture of solving larger problems rather than getting lost in the details of data description and access.

In this tutorial, you'll learn about the various Python data structures and see how they are implemented:

- [Abstract Data Type and Data Structures](#)

**Figure 5: Data Structures**

## String

Strings are collections of alphabets, words or other characters. In Python, you can create strings by enclosing a sequence of characters within a pair of single or double quotes. For example:

```
'cake', "cookie", etc.
```

You can also apply the `+` operations on two or more strings to concatenate them, just like in the example below:

```
x = 'Cake'
y = 'Cookie'
x + ' & ' + y
```

```
'Cake & Cookie'
```

**Figure 6: String data type**

From this website, I got to know some detailed information about different data types such as string, integer, dictionary, etc. It gives some details about data types and how it works while coding with its functions.

4. <https://developer.rhino3d.com/guides/rhinopython/python-datatypes/>

## String

Create string variables by enclosing characters in quotes. Python uses single quotes `' '` double quotes `" "` and triple quotes `"""` to denote literal strings. Only the triple quoted strings `"""` also will automatically continue across the end of line statement.

```
firstName = 'john'
lastName = "smith"
message = """This is a string that will span across multiple lines. Using newline characters
and no spaces for the next lines. The end of lines within this string also count as a newline when
printed"""
```

Strings can be accessed as a whole string, or a substring of the complete variable using brackets `[]`. Here are a couple examples:

```
var1 = 'Hello World!'
var2 = 'RhinoPython'

print var1[0] # this will print the first character in the string an `H`
print var2[1:5] # this will print the substring 'hinoP'
```

Python can use a special syntax to format multiple strings and numbers. The string formatter is quickly covered here because it is seen often and it is important to recognize the syntax.

**Figure 7: Explanation on String data type**

This website provides the detailed explanations of various data types. It provides with different examples which makes it easier to understand better.



5. <https://www.guru99.com/black-box-testing.html>

## What is BLACK Box Testing? Techniques, Example & Types



### What is Black Box Testing?

Black box testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications.



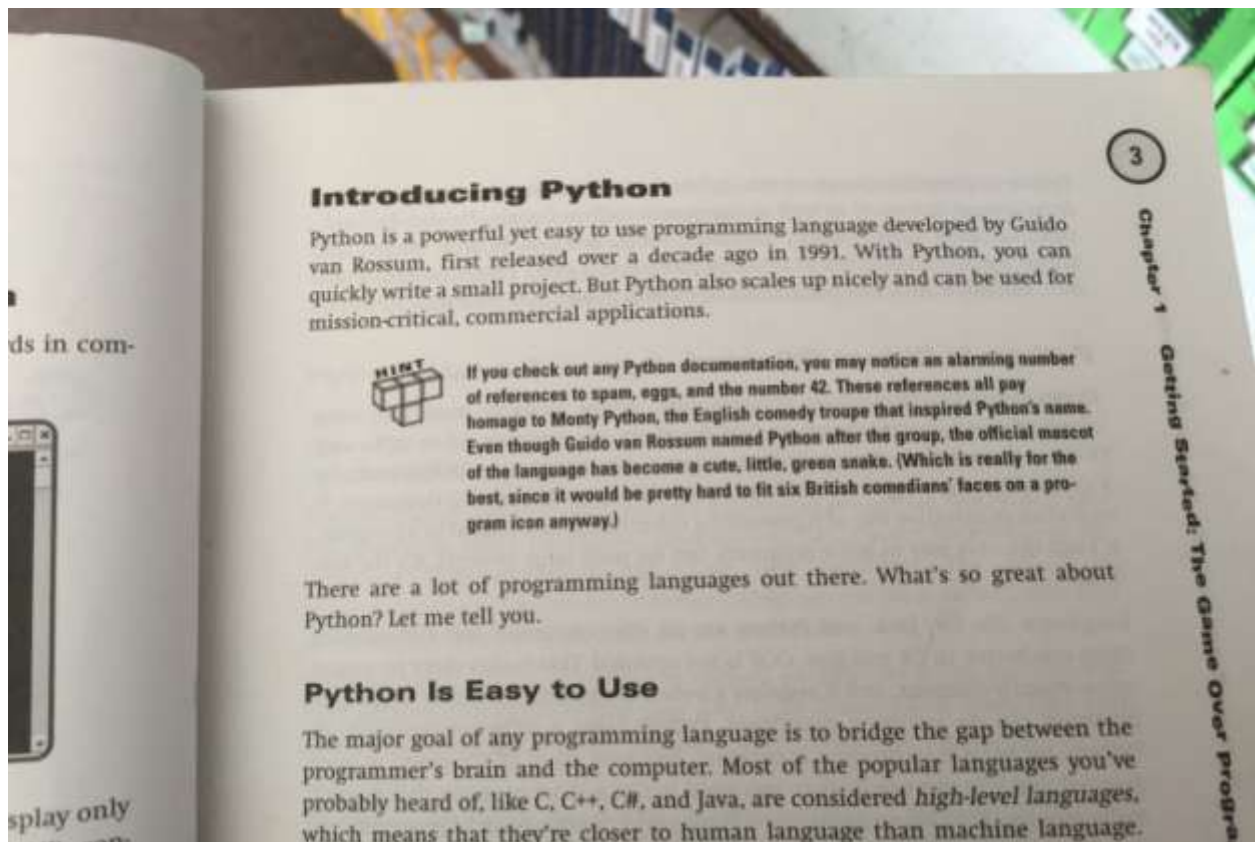
In BlackBox Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

**Figure 8: Black box testing**

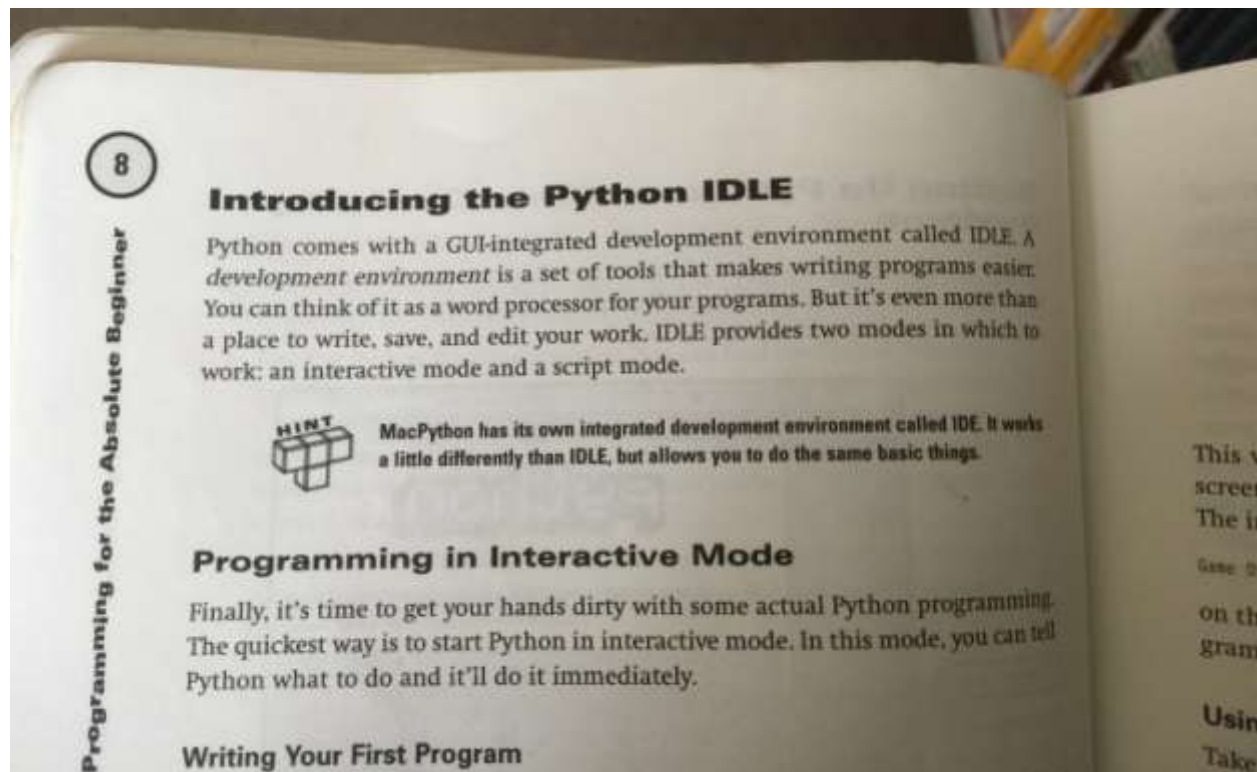
This website gives information about balck box testing.

### 9.3 Books:

#### 1. Programming for the absolute beginners by *Michael Dawson*.



*Figure 9: Programming for the absolute beginners by Michael Dawson.*



*Figure 10: Programming for the absolute beginners by Michael Dawson.*

This book provides some information about python and how it works. It has good explanations in a simple way which makes learners easy to understand what it is talking about. Also, I searched for some knowledge about IDLE which is an IDE which works in python.

## 2. Python 3 for Absolute Beginners by *Tim Hall and J-P Stacey*

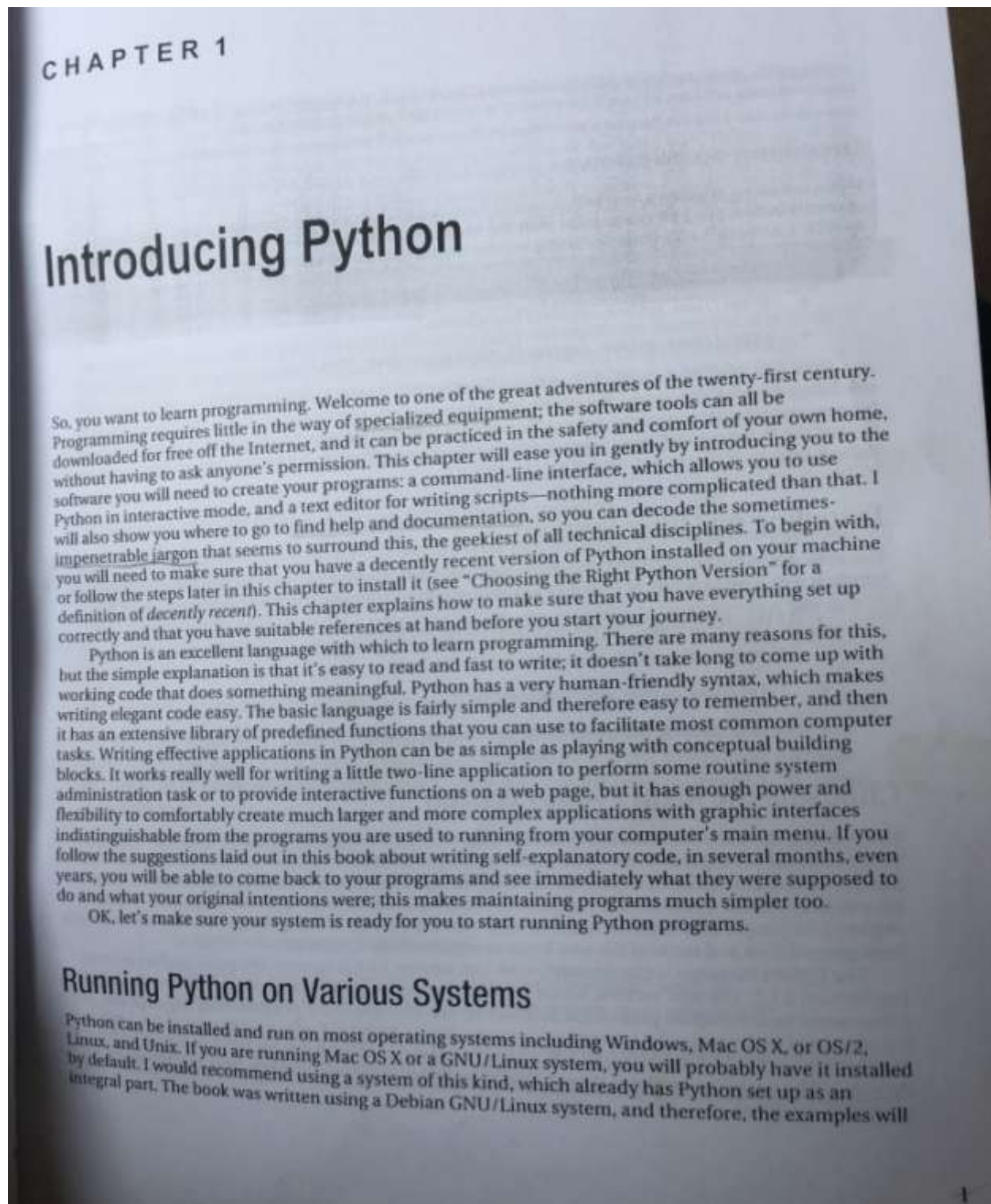


Figure 11: Python 3 for Absolute Beginners by Tim Hall and J-P Stacey

This book gives some introduction about Python programming language and describes how it works through different sub headings.

### 3. A Practical Introduction to Python Programming by *Brian Heinold*

#### 1.2 IDLE

IDLE is a simple integrated development environment (IDE) that comes with Python. It's a program that allows you to type in your programs and run them. There are other IDEs for Python, but for now I would suggest sticking with IDLE as it is simple to use. You can find IDLE in the Python 3.4 folder on your computer.

When you first start IDLE, it starts up in the shell, which is an interactive window where you can type in Python code and see the output in the same window. I often use the shell in place of my calculator or to try out small pieces of code. But most of the time you will want to open up a new window and type the program in there.

**Note** At least on Windows, if you click on a Python file on your desktop, your system will run the program, but not show the code, which is probably not what you want. Instead, if you right-click on the file, there should be an option called `Edit with Idle`. To edit an existing Python file,

*Figure 12: A Practical Introduction to Python Programming by Brian Heinold*

I took some information about IDLE which is used in python programming language as an IDE. It explains very easily how IDLE works in python.



#### 4. The Hitchhiker's Guide To Python by *Kenneth Reitz and Tanya Schlusser*

Testing your code is very important.

Getting used to writing testing code and running this code in parallel is now considered a good habit. Used wisely, this method helps you define more precisely your code's intent and have a more decoupled architecture.

Some general rules of testing:

- A testing unit should focus on one tiny bit of functionality and prove it correct.
- Each test unit must be fully independent. Each test must be able to run alone, and also within the test suite, regardless of the order that they are called. The implication of this rule is that each test must be loaded with a fresh dataset and may have to do some cleanup afterwards. This is usually handled by `setUp()` and `tearDown()` methods.
- Try hard to make tests that run fast. If one single test needs more than a few milliseconds to run, development will be slowed down or the tests will not be run as often as is desirable. In some cases, tests can't be fast because they need a complex data structure to work on, and this data structure must be loaded every time the test runs. Keep these heavier tests in a separate test suite that is run by some scheduled task, and run all other tests as often as needed.
- Learn your tools and learn how to run a single test or a test case. Then, when developing a function inside a module, run this function's tests frequently, ideally automatically when you save the code.

*Figure 13: The Hitchhiker's Guide To Python by Kenneth Reitz and Tanya Schlusser*

This book provides with information about various topics related to Python programming language. Here, I searched some information about testing in python and it's various types.

## 9.4 Journals:

### 1. PYTHON: A Programming Language For Software Integration and Development by *Michael F Sanner*.

#### Python

Python is an interpreted, interactive, object-oriented programming language. It provides high-level data structures such as list and associative arrays (called dictionaries), dynamic typing and dynamic binding, modules,

---

classes, exceptions, automatic memory management, etc.. It has a remarkably simple and elegant syntax and yet is a powerful and general purpose programming language. It was designed in 1990 by Guido van Rossum. Like many other scripting languages it is free, even for commercial purposes, and it can be run on practically any modern computer. A python program is compiled automatically by the interpreter into platform independent byte code which is then interpreted. We are running unmodified components written in Python under linux, Windows NT, 98, 95, IRIX, SunOS, OSF.

Python is modular by nature. The kernel is very small and can be extended by importing extension modules. The Python distribution includes a diverse library of standard extensions (some written in Python, others in C or C++) for operations ranging from string manipulations and Perl-like regular expressions, to Graphical User Interface (GUI) generators and including web-related utilities, operating system services, debugging and profiling tools, etc. New extension modules can be created to extend the language with new or legacy code. We describe these extension capabilities below. There are a substantial number of extension modules that have been developed and are distributed by members of the Python user community. These extension modules, sometimes referred to as "packages" or components include as GADFLY, an SQL database manager written in Python; PIL, the Python imaging library; FNORB and OmniBorker, CORBA compliant Object Request Brokers (ORB) written in Python; Gendoc, an automated documentation tool; and Numeric Python, just to name a few.

The best ressource for Python along with the books that are available is probably the Python web site (<http://www.python.org>). It provides access to code, documentation, packages, articles, mailing lists etc. It is also worth mentioning the recent creation of the biopython.org web site, a collaborative software effort for computational biology and chemistry very much like bioperl.

**Figure 14: A Programming Language For Software Integration and Development by Michael F Sanner.**

From this journal, I get to know some ideas about Python and gives some information about Python related yopics.

## 2. Pseudocode, the Pseudocode Programming Process, and Alternatives to the PPP by *Noah Doersing*.

### 2 Contribution

#### 2.1 Good Pseudocode

In this section, outline some criteria regarding the quality of pseudocode will be outlined, and what characterizes good pseudocode will be defined.

Pseudocode in the context of this paper consists of precise, natural-language descriptions of specific actions. This sets it apart from other kinds of pseudocode commonly used in algorithm papers or classes, where pseudocode is understood as a way of writing down programs at a relatively low level without adhering to the specific rules of any programming language. Of course, when implementing a formula or anything that is more conventionally expressed in a textual representation other than natural language, this guideline loses some validity.

Good pseudocode avoids code-like statements: it tries to stay at a higher level than source code in order to increase the efficiency during the design phase and to avoid getting restricted by specific limitations or features of the target programming language. This allows the programmer to capture the intent of an action inside a routine: Pseudocode should describe what a design is doing, not how exactly it is going to be implemented.

However the level of the pseudocode should be low enough to enable the programmer to refine it into source code very quickly (almost automatically).

Fig. 1 provides an example of pseudocode following these guidelines. Its purpose (creating a dialog box and adding it to some data structure) can be easily guessed without further knowledge about any implementation details.

```

Keep track of current number of resources in use
If another resource is available
    Allocate a dialog box structure
    If a dialog box structure could be allocated
        Note that one more resource is in use
        Initialize the resource
        Store the resource number at the
            location provided by the caller
    Endif
Endif
Return true if a new resource was created;
else return false

increment resource number by 1
allocate a dlg struct using malloc
if malloc() returns NULL then return 1
invoke OSrsrc_init to initialize a resource
for the operating system
*hrsrcPtr = resource number
return 0

```

**Figure 15: Pseudocode, the Pseudocode Programming Process, and Alternatives to the PPP by Noah Doersing.**

This journal gives some information about pseudocode and mainly about good pseudocose and why it is important.



3. Introduction To Python: Data Types by *Kranthi Varala*

# Strings

---

- A string object is a 'sequence', i.e., it's a list of items where each item has a defined position.
- Each character in the string can be referred, retrieved and modified by using its position.
- This order is called the 'index' and always starts with 0.

```
>>> S = 'Hello'
>>> len(S)
5
>>> S[0]
'H'
>>> S[4]
'o'
>>> S[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

```
>>> S[-1]
'o'
>>> S[3:]
'lo'
>>> S[2:5]
'llo'
```

*Figure 16: Data Types by Kranthi Varala*

This journal gives some knowledge about various data types and its objectives.

## Bibliography

(2019, April 29). Retrieved April 29, 2019, from programiz.com:

<https://www.programiz.com/article/flowchart-programming>

(2019, April 29). Retrieved April 29, 2019, from study.com: <https://study.com/academy/lesson/what-is-a-computer-algorithm-design-examples-optimization.html>

(2019, May 2). Retrieved May 2, 2019, from guru99.com: <https://www.guru99.com/black-box-testing.html>

Dawson, M. (2003). In M. Dawson, *Python programming for the absolute beginners* (p. 3). Boston: Premier press.

Dawson, M. (2003). In M. Dawson, *Programming for the absolute beginners* (p. 8). Boston: Premier press.

Doersing, N. (2019). *Pseudocode, the Pseudocode Programming Process, and Alternatives to PPP* , 10.

Fugier, D. (2018, December 5). *Rhino Developer Docs*. Retrieved April 3, 2019, from developer.rhino3d.com: <https://developer.rhino3d.com/guides/rhinopython/python-datatypes/>

Google Developers. (2018, 11 20). Retrieved 4 28, 2019, from developers.google.com: <https://developers.google.com/edu/python/strings>

Heinold, B. (2012). *A Practical Introduction to Python Programming*. BrianHeinold.net.

Jaiswal, S. (2017, December 8). *DataCamp*. Retrieved April 3, 2019, from datacamp.com: [https://www.datacamp.com/community/tutorials/data-structures-python?utm\\_source=adwords\\_ppc&utm\\_campaignid=1455363063&utm\\_adgroupid=65083631748&utm\\_device=c&utm\\_keyword=&utm\\_matchtype=b&utm\\_network=g&utm\\_adpostion=1t1&utm\\_creative=278443377095&utm\\_targetid](https://www.datacamp.com/community/tutorials/data-structures-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=1t1&utm_creative=278443377095&utm_targetid)

Kenneth Reitz, T. S. (2018). *The Hitchhiker's Guide to Python*.

Sanner, M. F. (1999). *Python: a programming language for software integration and development* , 2-3.

study.com. (2019, April 29). Retrieved April 29, 2019, from study.com: <https://study.com/academy/lesson/pseudocode-definition-examples-quiz.html>

Tim Hall, J.-P. S. (2009). In J.-P. S. Tim hall, *Python 3 for Absolute Beginners* (p. 1). New York: Apress.

Varala, K. (2019). *Introduction to Python: Data Types* , 20.