PYTHON & AUTOMATION

A PROJECT REPORT

Submitted by

Dhruvi Vimalbhai Virani

190320111070

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS & COMMUNICATION ENGINEERING

L. J.INSTITUTE OF ENGINEERING AND TECHNOLOGY, AHMEDABAD

Gujarat Technological University, Ahmedabad

[MAY , 2023]

L. J.INSTITUTE OF ENGINEERING AND TECHNOLOGY, AHMEDABAD

LJ Campus, LJ College Rd, near Sanand - Sarkhej Road, Makarba, Sarkhej-Okaf, Gujarat 382210

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **PYTHON & AUTOMATION** has been carried out by **Dhruvi Vimalbhai Virani** under my guidance in partial fulfillment for the degree of Bachelor of Engineering in Electronics and Communication , 8th Semester of Gujarat Technological University, Ahmadabad during the academic year 2021-22.

Mosam Pandya	Mosam Pandya
Internal Guide	Head of the Department

EINFOCHIPS OFFER LETTER

ACKNOWLEDGEMENT

The internship opportunity I am having with Einfochips-An Arrow Company was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the JIMMY PATEL, Technical lead at the Company who in spite of being extraordinarily busy with her duties, took time out to hear, guide, and keep me on the correct path and allowing me During My training at Einfochips-An Arrow Company.

Abstract

Industrial training is an important phase of a student’s life. A well-planned, properly executed, and evaluated industrial training helps a lot in developing a professional attitude. It develops an awareness of the industrial approach to problem-solving, based on a broad understanding of the process and mode of operation of an organization.

The aim and motivation of this industrial training are to receive discipline, skills, teamwork, and technical knowledge through a proper training environment, which will help me, as a student in the field of IT, to develop responsiveness to the self-disciplinary nature of problems.

During 12 weeks of training at Einfochips-An Arrow Company, I was assigned as a technical supporter to help the company in the field of IT. Throughout this industrial training, I have learned new software development languages used in automation and software such as Python, Python in Selenium, Robot Framework. I am also able to implement what I have learned over the past year as an **Electronics and Communication** student at **L.J. INSTITUTE OF ENGINEERING & TECHNOLOGY**, Ahmedabad.

Table of Contents

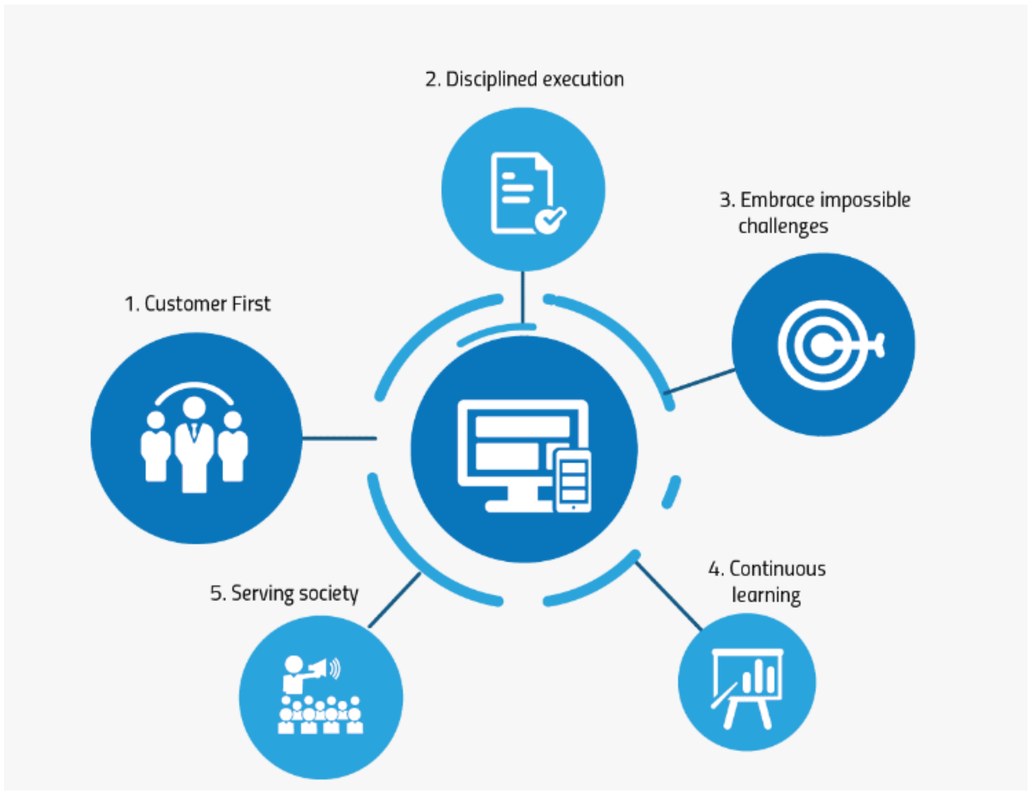
--> EInfochips, an Arrow company, is a leading global provider of product engineering and semiconductor design services. With over 500+ products developed and 40M deployments in 140 countries, eInfochips continues to fuel technological innovations in multiple verticals. The company's service offerings include digital transformation and connected IoT solutions across various cloud platforms, including AWS and Azure.

--> Vision of Einfochips:
"We want to make eInfochips a leading global innovative technology company that will have a transformational impact on the society by creating leaders and generating stakeholder value"
– Sumit Sethi, COO.

Founded in 1994, our work culture is built over years of experience in providing innovative solutions to our clients and our indomitable spirit to excel in all aspects of our engagement. We are geographically spread across India, Japan, and the USA with 10 design centers and offices.

The fuel to our growth is driven by our passion and strong commitment to our core values:
--Customer First
--Disciplined Execution
--Embrace Impossible Challenges
--Continuous Learning
--Serving Society

eInfochips An Arrow Company | The Solutions People



--Digital Engineering

~Cloud Services
~DevOps Services
~IoT Solutions & Services
~Mobility Solutions & Services
~Machine Learning Services
~Data Analytics Services
~Remote Device Managemen

--Device Engineering

~Hardware Design Services
~Design to Manufacturing
~Embedded Systems & Software Development
~Multimedia & Digital Solutions
~Image Tuning Service
~Product Sustenance Engineering

--Quality Engineering

~Product Testing Services
~IoT Testing Services
~Device-to-Cloud QA Automation Services
~Cognitive QA Services
~Mobile and Web Testing Services
~Quality Process Consulting
~Wireless Testing Services

--Silicon Engineering

~ASIC/FPGA Design & Development
~Design Verification & Validation
~Physical Design and DFT

Chapter 2: My Role in my ongoing internship at the Company

--> My role in Einfochips- An Arrow Company is an engineer in Quality assurance(QA).

--> I have been placed in the department of Intelligence Automation(IA).

--> Under the mentorship of Jimmy Patel who is the Technical Guide- Level 1 of the IA department, I have been assigned in automation field in robot framework with python which is one of the most important framework in the field of automation.

--> I have been given a project of Python to make certain games using different functions and classes at first.

--> Python with selenium was also taught to us to understand the working of how automation can be done.

--> Further, using robot framework, I have been given several tasks to automate different websites.

3.1 Introduction: *Python*

Python was designed for readability, and has some similarities to the English language with influence from ma
Python uses new lines to complete a command, as opposed to other programming languages which often use
Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and clas
It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python can be used on a server to create web applications.
Python can be used alongside software to create workflows.
Python can connect to database systems. It can also read and modify files.
Python can be used to handle big data and perform complex mathematics.
Python can be used for rapid prototyping, or for production-ready software development

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
Python has a simple syntax similar to the English language.
Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
Python can be treated in a procedural way, an object-oriented way or a functional way.

Python was designed for readability, and has some similarities to the English language with influence from mathematics.
Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Automation

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it’s possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

Test Automation is the best way to increase the effectiveness, test coverage, and execution speed in software testing. Automated software testing is important due to the following reasons:

~ Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming
~It is difficult to test for multilingual sites manually
~Test Automation in software testing does not require Human intervention. You can run automated test unatte
~Test Automation increases the speed of test execution
~Automation helps increase Test Coverage
~Manual Testing can become boring and hence error-prone.

The knowledge of a programming language can help a QA specialist in different ways.
Besides providing access to test automation tools, the ability to understand code also advances manual testing. One way or another, it enhances a person’s competencies and makes a software testing company better prepared for the market challenges. The question is what programming language to learn and use for writing tests. Usually, tech specialists face a “Python vs Java” dilemma. Let’s focus on the former in today’s post.

Python test automation is simply using Python for test automation.

Python is one of testing teams’ favorite programming languages. It has multiple features that make it suitable for automated testing. For example, you can benefit from Python’s dynamic typing but also make use of checkers like Pyre for static typing. Python is easy to learn and can also be used for complex problems. Therefore, users of different skill levels can use Python.

Python is well-known for having a library available for almost everything so you don’t need to waste time writing code from scratch. And if you find something that’s not exactly what you wanted, you can also make changes to the libraries and use them as per your use case. And in addition to all this, Python has an amazing community to help you when you’re stuck.

PyTest is a native Python test library with a superset of PyUnit’s features. Rather than modeling JUnit’s architecture, it has a distinctly Python flavor. It makes heavy use of Python decorators and assertions.

PyTest also supports parameterized testing (without the aid of plugins like Nose) that improves code reuse and simplifies code coverage.

You can create a program using Python for your web automation tasks like gathering data from websites and putting the data into an Excel spreadsheet. Most repeated workflows involving actions on the web with defined steps can be automated by writing a Python program. Once you have your scripts written, it’s simple to reuse them to make your routine tasks easier.

For an all-around better web automation solution, look to robotic process automation (RPA). With RPA, website automation is made easy for even the most basic business users. Plus, with a great RPA solution, you can easily integrate any existing Python scripts while moving forward with automation for other departments in your enterprise-wide workflows so you don’t have to start completely from scratch.
The best RPA software should be easy-to-use with simple drag-and-drop features. That way, anyone on staff can become an automation expert without having to become a programmer.

3.2 Disadvantages of Python & Automation:

The downside of using Python for web automation is the programming. You’re going to need some pretty extensive programming skills, and if you’re not a trained programmer—or don’t have one on staff—setting up automated tasks with Python can be a daunting task. If you don’t have a Python expert, you shouldn’t rely on Python for automating your workflows. And if you happen to have legacy Python programs already implemented that break, you’ll need that expert around to put the pieces back together.

Chapter 4: Python

4.1 Movie Guessing Game:

My first code of the project was to make a movie guessing game which is a one user game.

```
def A():

   balance = balance - 30
   print(balance)
   movie = ["3idiots", "Shershah", "Pathaan", "Yeh jawani Hai deewani"]
   movie = random.choice(movie)
   movie = movie.replace(" ", "")
   movie = movie.lower()
   print(movie)
   movie = list(movie)
   movie1 = ["*"] * len(movie)
   print(movie1)
   chance = 5
   while (movie != movie1 and chance > 0):
        c = input("enter a character/digit:")
        if c in movie:
            for i in range(len(movie)):
                if movie[i] == c:
                    movie1[i] = c
        else:
            chance = chance - 1
            print("wrong answer you have", chance, "chances left")
        print(movie1)
```

4.2 Rock,Paper,scissor Game:

My second game of the project was to make a rock, paper scissor game which is a two player game.

```
def B():
    # rock paper scissor game

    chance = 0
    while chance < 3:
        turn = ["r", "p", "s"]
        turn = random.choice(turn)
        user = input("enter your r for rock,p for paper and s for scissor: ")
        print(turn)
        # turn=list(turn)
        b = turn
        if user == "r" and b == "p":
            print("computer is winner")
        elif user == "s" and b == "r":
            print("computer is winner")
        elif user == "s" and b == "p":
            print("computer is winner")
        elif user == b:
            print("tie")
```

4.3 Flames Game:

My third game was to make Flames game in which user will enter two inputs of different names and the relation between that two names will be shown in the form of flames (Friend, Love, Affection, Marriage, Enemy, Sibling)

```
def C():
    # FLAMES GAME

    string1 = "FLAMES"
    lst1 = list(string1)
    name1 = input("Enter the first name:").lower()
    name2 = input("Enter the second name :").lower()
    lsname1 = list(name1)
    lsname2 = list(name2)
    for x in lsname1[:]:
        if x in lsname2:
            lsname1.remove(x)
            lsname2.remove(x)
    total = len(lsname1) + len(lsname2)
    len_lst1 = len(lst1)
    count = 0
    # To define the range how many times the game need to be played
    while ((len(lst1)) != 1):  # While length is =1 then the game is not possible
        for x in range(len(lst1)):
            count += 1
            if (count == total):
                count = 0
                lst1.remove(lst1[x])
        # print("The list size ",len_lst1)
        # len_lst1-=1
    print(lst1)
```

4.4 Tic Tac Toe Game:

My fourth game of the project was to make a tic tac toe game which is a two player game.

```
def G():
    global balance
    balance = balance - 30
    print(balance)
    import numpy as np
    mat = np.array([["_", "_", "_"], ["_", "_", "_"], ["_", "_", "_"]])

    def print_board(mat_copy):
        global mat
        print(mat_copy)

    def check_winner():
        global cwin
        for i in range(0, 3):
            if mat[i][0] == mat[i][1] and mat[i][0] == mat[i][2]:
                if mat[i][0] == "x":
                    print("X won the match")
                    cwin = "win"
                    break
                else:
                    cwin = "win"
                    break
            else:
                cwin = "loss"
        for i in range(0, 3):
            if mat[0][i] == mat[1][i] and mat[0][i] == mat[2][i]:
                if mat[i][0] == "x":
                    print("X won the match")
                    cwin = "win"
                    break
                else:
                    cwin = "win"
                    break
            else:
                cwin = "loss"

    turn = 1
    cwin = "loss"
    print_board(mat)
    # 1 for x and 0 for O
    while (cwin != "win"):
        if (turn == 1):
            print("x turn")
            row = int(input("please enter a value of row: "))
            column = int(input("please enter a value of column: "))
            mat[row][column] = "x"
            # print(x)
            print_board(mat)
            print(cwin)
            cwin = check_winner()
            print(cwin)
            turn = 0
        else:
            print("y turn")
            row = int(input("please enter a value of row: "))
            column = int(input("please enter a value of column: "))
            mat[row][column] = "O"
            # print(x)
            print_board(mat)
            cwin = check_winner()
            turn = 1

    if (cwin == "win"):
        print("match over")
```

4.5 Snake and Ladder game:

My Fifth game of my project was to create a snake and ladder game which is a two player game.

```
import random
ladder={1:38,4:14,8:30,21:42,28:76,50:67,71:92,80:99}
snake={97:78,95:56,88:24,62:18,48:26,36:6,32:10}
i=0
position=0
b="y"
while position<100 and b=="y":
  c=0
  b=input("Do you want to play: ")
  Dice=random.randint(1,6)
  print(Dice)
  for i,j in ladder.items():
    if i==position and c==0:
      position=j
      print("position is: ",position)
      position=position+Dice
      print("position is: ",position)
      c=1
      if position>100:
        position=position-Dice
        print("position is: ",position)
  for k,n in snake.items():
    if k==position and c==0:
      position=n
      print("position is: ",position)
      position=position+Dice
      print("position is: ",position)
      c=1
      if position>100:
        position=position-Dice
  if c==0:
    position=position+Dice
    if position>100:
      position=position-Dice
  if position==100:
    print("you won")
```

4.6 Auction: My Sixth task of the project was to create an auction between buyers and sellers.

```
S1_vm = {"S1":7,"S2":6,"S3":5,"S4":9}
S1_bid = {"S1":20,"S2":23,"S3":19,"S4":30}
U1_vm={"B1":4 ,"B2":3,"B3":2,"B4":5 }
U1_bid={"B1":25 ,"B2":22,"B3":17,"B4":21 }

sorted_s1 = dict(sorted(S1_bid.items(), key=lambda kv: kv[1]))
sorted_u1 = dict(sorted(U1_bid.items(), key=lambda kv: kv[1], reverse=True))
print(sorted_s1)
print(sorted_u1)
dict1={}
for i in sorted_u1.keys():
    for j in sorted_s1.keys():
        if S1_vm[j]>=U1_vm[i]:
            dict1[i]=j
            S1_vm[j]=S1_vm[j]-U1_vm[i]
            #del U1_vm[i]
            break
print("Allocation: ",dict1)
lst_seller=[]
lst_user=[]
for i,j in dict1.items():
    lst_seller.append(S1_bid[j])
    lst_user.append(U1_bid[i])
print("Price to be paid by each buyer will be: ",max(lst_seller))
print("Payment received to all sellers will be: ",min(lst_user))
a1={}
a2={}
for i,j in dict1.items():
    p=max(lst_seller)-S1_bid[j]
    q=U1_bid[i]-min(lst_user)
    a1[j]=p
    # print(a1)
    a2[i]=q
    # print(a2)
print("profit of each seller is ",a1)
print("profit of each user is ",a2)

sum1=0
for i, j in a1.items():
    for k, m in dict1.items():
        if (i==m):
            sum1=sum1+(j*U1_vm[k])
print(sum1)
print("the average of seller after selling the machines: ",sum1/len(a1))

sum2=0
for i,j in a2.items():
    for k,m in U1_vm.items():
        if(i==k):
            sum2=sum2 + (j*m)
print(sum2)
print("the average profit of seller after selling the machines: ",sum2/len(a2))

b1=len(a1)
b2=len(a2)
print("the average profit of buyer per machine is ",(sum(a1.values())/b1))
print("the average profit of user per machine is ",(sum(a2.values())/b2))
```

STEP1:

```
S1_vm = {"S1":7,"S2":6,"S3":5,"S4":9}
S1_bid = {"S1":20,"S2":23,"S3":19,"S4":30}
U1_vm={"B1":4 ,"B2":3,"B3":2,"B4":5 }
U1_bid={"B1":25 ,"B2":22,"B3":17,"B4":21 }
```

~ S1_vm is the dictionary for available VMs, S1_bid is the dictionary for cost of each VM.
~ U1_vm is the dictionary for required VMs of buyer, U1_bid is the dictionary for bid of each VM.
~ We have 4 sellers(CSP=CLOUD SERVICE PROVIDER) and 4 buyers/users here.
~ We are taking availability of virtual machines(VMs) with each seller and the cost per VM.
~ Similarly we are taking requirement of each buyer of VM and the bid that they are keeping.

STEP2:

```
sorted_s1 = dict(sorted(S1_bid.items(), key=lambda kv: kv[1]))
sorted_u1 = dict(sorted(U1_bid.items(), key=lambda kv: kv[1], reverse=True))
print(sorted_s1)
print(sorted_u1)
```

~ Now we are sorting the number of sellers in ascending order of their bid
~ Also we are sorting the number of buyers in descending order of their bid.

STEP3:

```
dict1={}
for i in sorted_u1.keys():
    for j in sorted_s1.keys():
        if S1_vm[j]>=U1_vm[i]:
            dict1[i]=j
            S1_vm[j]=S1_vm[j]-U1_vm[i]
            #del U1_vm[i]
            break
print("Allocation: ",dict1)
```

~ In next step, we are allocating the buyer to the seller if seller has sufficient resources then allocated.
~ We have stored the allocation in an empty dictionary named dict1.

STEP4:

```
lst_seller=[]
lst_user=[]
for i,j in dict1.items():
    lst_seller.append(S1_bid[j])
    lst_user.append(U1_bid[i])
print("Price to be paid by each buyer will be: ",max(lst_seller))
print("Payment received to all sellers will be: ",min(lst_user))
```

~ In this step, we are creating two empty list to store maximum bid of seller and minimum bid of buyer/user respectively.

STEP5:

```
a1={}
a2={}
for i,j in dict1.items():
    p=max(lst_seller)-S1_bid[j]
    q=U1_bid[i]-min(lst_user)
    a1[j]=p
    # print(a1)
    a2[i]=q
    # print(a2)
print("profit of each seller is ",a1)
print("profit of each user is ",a2)
```

~ In this step, we are creating another two empty lists to store profit of each seller and each buyer in those empty l respectively.
~ For calculating profit of each seller, we are considering maximum value of cost and subtracting remaining values from that maximum
Here in my code, it is max(lst_seller)-S1_bid[j]
~ Similarly, for calculating profit of each buyer/user, we are considering minimum value of bid and subtracting remaining values to that
minimum value. Here in my code, it is U1_bid[i]-min(lst_user)

STEP6:

```
b1=len(a1)
b2=len(a2)
print("the average profit of seller is ",(sum(a1)/b1))
print("the average profit of user is ",(sum(a2)/b2))
```

~ In this step, we are calculating the average of profit of seller and buyer/user.
~ So, here sum(a1) is addition of profits stored in a1 and sum(a2) is addition of profits stored in a2.

user will have a bonus card with a limit of 500 points and each game will cost 30 points.
Also there will be cafe from which user can buy food from the available balance.

```
import random
print("Welcome to GameZone\n","\U0001f600")
games={"A":"movie_guessing game","B":"rock_paper_scissor","C":"Flames","D":"cafe",
"G":"TicTacToe","F":"snake_ladder","E":"Exit Zone"}
print(games,"\n")
for i, j in games.items():
    print("Enter", i,"to play", j)
userValue = input("Enter Your choice: ").lower()
balance = 500 #total balance is 500 when entering into gamezone and per game costs it 30 rupees
Flag=0
def A():
    #game = 30
   global balance
   balance = balance - 30
   print(balance)
   movie = ["3idiots", "Shershah", "Pathaan", "Yeh jawani Hai deewani"]
   movie = random.choice(movie)
   movie = movie.replace(" ", "")
   movie = movie.lower()
   print(movie)
   movie = list(movie)
   movie1 = ["*"] * len(movie)
   print(movie1)
   chance = 5
   while (movie != movie1 and chance > 0):
        c = input("enter a character/digit:")
        if c in movie:
            for i in range(len(movie)):
                if movie[i] == c:
                    movie1[i] = c
        else:
            chance = chance - 1
            print("wrong answer you have", chance, "chances left")
        print(movie1)

   if chance > 0:
    balance = balance + 20
    print("Your current balance is: ", balance)
def B():
    # rock paper scissor game
    global balance
    balance = balance - 30
    print(balance)
    chance = 0
    while chance < 3:
        turn = ["r", "p", "s"]
        turn = random.choice(turn)
        user = input("enter your r for rock,p for paper and s for scissor: ")
        print(turn)
        # turn=list(turn)
        b = turn
        if user == "r" and b == "p":
            print("computer is winner")
        elif user == "s" and b == "r":
            print("computer is winner")
        elif user == "s" and b == "p":
            print("computer is winner")
        elif user == b:
            print("tie")
        else:
            balance = balance + 20
            print("Your current balance is: ",balance)
            print("user is winner")
        chance += 1
def C():
    # FLAMES GAME
    print("This game is just for fun so no bonus points will be given here")
    global balance
    balance = balance - 30
    print(balance)
    string1 = "FLAMES"
    lst1 = list(string1)
    name1 = input("Enter the first name:").lower()
    name2 = input("Enter the second name :").lower()
    lsname2 = list(name2)
    for x in lsname1[:]:
        if x in lsname2:
            lsname1.remove(x)
            lsname2.remove(x)
    total = len(lsname1) + len(lsname2)
    len_lst1 = len(lst1)
    count = 0
    # To define the range how many times the game need to be played
    while ((len(lst1)) != 1):  # While length is =1 then the game is not possible
        for x in range(len(lst1)):
            count += 1
            if (count == total):
                count = 0
                lst1.remove(lst1[x])
        # print("The list size ",len_lst1)
        # len_lst1-=1
    print(lst1)
def D():
    # exercise of restaurant
    global balance
    # balance = balance - 100
    # print(balance)
    menu = {"pizza": 90, "burger": 50, "sandwich": 60, "pasta": 75, "buttermilk": 30}
    print("menu list\n")
    a = "".lower() # a is the key
    sum = 0
    lst_order = {}
    for i, j in menu.items():
        print(i, "price", j)
    while (a != "no"):
        order = input("\nenter your order: ").lower()
        Quantity = input("enter the quantity: ")
        lst_order[order] = Quantity  # the order and quantity is getting stored in lst_order
        dictionary
        print("\n'press ok to order more and no to stop'\n")
        a = input("do you want to order more?").lower()
    print("\norder confirmation: ", order)
    for i, j in lst_order.items():
        for k, m in menu.items():
            if (i == k):
                sum = sum + int(j) * int(m)  # here j is the quantity which is stored in lst_order an
    print("your total bill is: ", sum)
    add = input("do you want to add items in your order?: ").lower()
    if add == "yes":
        order = input("please enter your order: ").lower()
        quantity = input("please enter the quantity: ")
        lst_order[order] = Quantity
    sum = 0
    for i, j in lst_order.items():
        for k, m in menu.items():
            if (i == k):
                sum = sum + int(j) * int(m)
    print("your total bill is: ", sum)
    r = input("do you want to remove any item?: ").lower()
    if r == "yes":
        remove_item = input("enter the item you want to remove: ").lower()
        del lst_order[remove_item]
    sum = 0
    for i, j in lst_order.items():
        for k, m in menu.items():
            if (i == k):
                sum = sum + int(j) * int(m)
    print("your total bill is: ", sum)
    print("Thank you for ordering")
    balance=balance-sum
    print(balance)

def F():
    global balance
    balance = balance - 30
    print(balance)
    ladder = {1: 38, 4: 14, 8: 30, 21: 42, 28: 76, 50: 67, 71: 92, 80: 99}
    snake = {97: 78, 95: 56, 88: 24, 62: 18, 48: 26, 36: 6, 32: 10}
    i = 0
    position = 0
    b = "y"
```

```
        b = input("Do you want to play: ")
        Dice = random.randint(1, 6)
        print("position is: ", position)
        print(Dice)
        # for ladder part
        # j is the position of user after dice
        for i, j in ladder.items():
            if i == position and c == 0:
                position = j
                print("position is: ", position)
                position = position + Dice
                print("current position is: ", position)
                c = 1
                if position > 100:
                    position = position - Dice
                    print("position is: ", position)
        # for snake part
        # n is the psoition of player after dice
        for k, n in snake.items():
            if k == position and c == 0:
                position = n
                print("position is: ", position)
                position = position + Dice
                print("position is: ", position)
                c = 1
                if position > 100:
                    position = position - Dice
        if c == 0:
            position = position + Dice
            if position > 100:
                position = position - Dice
        if position == 100:
            print("you won")
def G():
    global balance
    balance = balance - 30
    print(balance)
    import numpy as np
    mat = np.array([["_", "_", "_"], ["_", "_", "_"], ["_", "_", "_"]])

    def print_board(mat_copy):
        global mat
        print(mat_copy)

    def check_winner():
        global cwin
        for i in range(0, 3):
            if mat[i][0] == mat[i][1] and mat[i][0] == mat[i][2]:
                if mat[i][0] == "x":
                    print("X won the match")
                    cwin = "win"
                    break
                else:
                    cwin = "win"
                    break
            else:
                cwin = "loss"
        for i in range(0, 3):
            if mat[0][i] == mat[1][i] and mat[0][i] == mat[2][i]:
                if mat[i][0] == "x":
                    print("X won the match")
                    cwin = "win"
                    break
                else:
                    cwin = "win"
                    break
            else:
                cwin = "loss"
    turn = 1
      cwin = "loss"
      print_board(mat)
    # 1 for x and 0 for O
```

```
            print("x turn")
            row = int(input("please enter a value of row: "))
            column = int(input("please enter a value of column: "))
            mat[row][column] = "x"
            # print(x)
            print_board(mat)
            print(cwin)
            cwin = check_winner()
            print(cwin)
            turn = 0
        else:
            print("y turn")
            row = int(input("please enter a value of row: "))
            column = int(input("please enter a value of column: "))
            mat[row][column] = "O"
            # print(x)
            print_board(mat)
            cwin = check_winner()
            turn = 1

    if (cwin == "win"):
        print("match over")

print(userValue)
while(userValue!='e'):
    if(userValue == 'a'):
        A()
    elif(userValue == 'b'):
        B()
    elif(userValue == 'c'):
        C()
    elif(userValue == 'd'):
        D()
    elif(userValue=='f'):
        F()
    elif(userValue=='g'):
        G()
    elif (userValue == 'e'):
        break
    # elif(userValue==
    # 'G'):
    #     G()
    else:
        print("Invalid game option")
    print(games)
    userValue = input("Enter Your choice: ").lower()

print("Your current balance is ", balance)
```

Chapter [illegible]: Python with Selenium

-->Selenium Python bindings provide APIs using which you can write functional tests using the Selenium WebDriver. Like other Selenium language bindings, Selenium Python APIs can be leveraged to develop highly efficient tests that let you interact with the WebElements on the AUT (Application Under Test).

The bindings support local browser drivers (e.g., Chrome, Firefox, Internet Explorer, etc.) and provide a remote WebDriver that lets you connect to a remote cloud Selenium Grid.

--> Different websites such as kubernetes, webucator, docker, pizza hut has been automated by me.

--> Using different actions such as .click(), .is_displayed(), .send_keys(), .scroll_location_into_view(), .is_selected(), driver.find_elements(), action chains, etc, we can automate any type of website.

-->To automate a website, we need to inspect the particular element by Xpath, css locators, id, class,etc.

-->Several libraries and keys needs to be install and import at the beginning of the python file. such as

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from Reusable import common
from selenium.webdriver.support.select import Select
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.alert import Alert your text
```

Code of automation to validate sign in of webucator website:

```
def Sign_in(self):
    driver.get("https://www.webucator.com/account/login/")
    driver.implicitly_wait(10)
    time.sleep(2)
    driver.find_element(By.NAME, "login").click()
    driver.find_element(By.NAME, "login").send_keys(common.resdXMLAsPerNode("email"))
    driver.find_element(By.NAME, "password").click()
    driver.find_element(By.NAME, "password").send_keys(common.resdXMLAsPerNode("passc"))
    time.sleep(3)
    time.sleep(3)
    driver.find_element(By.XPATH, "(//button[text()='Sign In'])").click()
    # if driver.find_element(By.XPATH, "(//div[@class='text-danger']/strong)").get_attribute('value')!="" :
    if driver.find_element(By.XPATH,
       "(//div[@class='text-danger']/strong)").text==common.resdXMLAsPerNode("passc"):
        print(common.resdXMLAsPerNode("error"))

    time.sleep(3)
```

Code of automation to validate the search box and scrolling of flipkart website:

```
def practice_website2(self):
    global driver
    driver = webdriver.Chrome()
    driver.get("https://www.flipkart.com/")
    driver.implicitly_wait(10)
    driver.maximize_window()
    driver.find_element(By.XPATH,"//button[@class='_2KpZ6l _2doB4z']").click()
    time.sleep(2)
    driver.find_element(By.XPATH,
        "//form[@class='_2M8cLY header-form-search']").is_displayed()
    time.sleep(2)
    driver.find_element(By.XPATH,
        "//input[@class='_3704LK']").send_keys("i phone 12 128bg")
    driver.find_element(By.XPATH,
        "//button[@class='L0Z3Pu']").click()
    time.sleep(2)
    ele = driver.find_element(By.XPATH,
        "//div[text()='APPLE iPhone 12 Pro (Gold, 128 GB)']")
    actionChains = ActionChains(driver)
    actionChains.move_to_element(ele).perform()
    #driver.find_element(By.XPATH,
    "//div[text()='APPLE iPhone 12 Pro (Gold, 128 GB)']").location_once_scrolled_into_view
    #driver.find_element(By.XPATH,
        "//div[text()='APPLE iPhone 12 Pro (Gold, 128 GB)']").is_displayed()
    time.sleep(2)
    driver.find_element(By.XPATH,
        "//div[text()='APPLE iPhone 12 Pro (Gold, 128 GB)']").click()
    time.sleep(3)
```

Code of automation to drag and drop:

```
    def test_DemoWebsite(self):
        global driver
        driver = webdriver.Chrome()
        driver.get("http://www.dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-3.html")
        driver.implicitly_wait(5)
        driver.maximize_window()
        src=driver.find_element(By.XPATH,"//div[@id='box5']")
        dest=driver.find_element(By.XPATH,"//div[@id='box105']")
        actionChains=ActionChains(driver)
        actionChains.drag_and_drop(src,dest).perform()
```