./

GENESIS – Advanced Python and Testing Project Summary



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| V1.1 | 11/12/2020 | Shriram M S |  |  |  |
| V1.2 | 12/12/2020 | Shriram M S |  |  |  |
| V1.3 | 13/12/2020 | Shriram M S |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc58757032)

[Stadium Ticket Booking System 4](#_Toc58757033)

[1 ACTIVITY 1 (11/12/2020) 4](#_Toc58757034)

[1.1 Python Code 4](#_Toc58757035)

[1.2 PEP8 Online 6](#_Toc58757036)

[1.3 Input Files Screenshot 7](#_Toc58757037)

[1.3.1 Match Schedule Input Excel File 7](#_Toc58757038)

[1.3.2 Booking Detail Input Excel File 8](#_Toc58757039)

[1.3.3 Login Detail Input Excel File 8](#_Toc58757040)

[1.4 Output Files Screenshot 9](#_Toc58757041)

[1.4.1 Confirmation Details Output Excel File 9](#_Toc58757042)

[2 ACTIVITY 2 (12/12/2020) 9](#_Toc58757043)

[2.1 Python Code 10](#_Toc58757044)

[2.2 PEP8 Online 15](#_Toc58757045)

[2.3 Input Files Screenshot 15](#_Toc58757046)

[2.3.1 Mumbai Stadium Schedule - Input Excel File 16](#_Toc58757047)

[2.3.2 Kolkata Stadium Schedule - Input Excel File 16](#_Toc58757048)

[2.3.3 Chennai Stadium Schedule - Input Excel File 16](#_Toc58757049)

[2.4 Output Files Screenshot 17](#_Toc58757050)

[2.4.1 Booking Detail - Output Excel File 17](#_Toc58757051)

[3 RegEx Practice 17](#_Toc58757052)

[3.1 Code 17](#_Toc58757053)

[3.2 Output 18](#_Toc58757054)

**LIST OF TABLES**

[Figure 1 : PEP8 ONLINE SCREENSHOT ACTIVITY 1 6](#_Toc58757020)

[Figure 2 : MATCH SCHEDULE SCREENSHOT ACTIVITY 1 7](#_Toc58757021)

[Figure 3 : BOOKING DETAIL SCREENSHOT ACTIVITY 1 7](#_Toc58757022)

[Figure 4 : LOGIN DETAIL SCREENSHOT ACTIVITY 1 8](#_Toc58757023)

[Figure 5 : CONFIRMATION DETAIL SCREENSHOT ACTIVITY 1 9](#_Toc58757024)

[Figure 6 :PEP8 ONLINE SCREENSHOT ACTIVITY 2 15](#_Toc58757025)

[Figure 7 : MUMBAI SCHEDULE SCREENSHOT ACTIVITY 2 15](#_Toc58757026)

[Figure 8 : KOLKATA SCHEDULE SCREENSHOT ACTIVITY 2 16](#_Toc58757027)

[Figure 9 : CHENNAI SCHEDULE SCREENSHOT ACTIVITY 2 16](#_Toc58757028)

[Figure 10 : BOOKING DETAILS SCREENSHOT ACTIVITY 2 17](#_Toc58757029)

[Figure 11 : REGEX PRACTICE CODE SCREENSHOT 18](#_Toc58757030)

[Figure 12 : REGEX PRACTICE CODE OUTPUT SCREENSHOT 19](#_Toc58757031)

# Stadium Ticket Booking System

# ACTIVITY 1 (11/12/2020)

## Python Code

import pandas as pd

pd.set\_option("display.max\_columns",None)

# File to view match schedule

df=pd.read\_csv("schedule.csv")

# Input file to see user booking details

bd=pd.read\_csv("booking\_details.csv")

# File to see authentication details

log=pd.read\_csv("login\_details.csv")

class stadium:

tot\_capacity=0

tot\_seat\_available=0

tot\_seats\_booked=0

cost=1000

tot\_cost=0

def details(self,tot\_capacity):

self.tot\_capacity=tot\_capacity

def seats\_available(self):

tot\_seat\_available=self.tot\_capacity-self.tot\_seats\_booked

return tot\_seat\_available

def book\_ticket(self,match\_booked):

print("\nTotal no. of tickets to book",bd.no\_of\_tickets[0])

bookings=bd.no\_of\_tickets[0]

if self.seats\_available() < bookings:

print("Enough tickets not available")

self.match\_schedule()

else:

self.tot\_cost=self.cost\*bookings

self.tot\_seats\_booked+=bookings

print(f"Total cost for {bookings} tickets are {self.tot\_cost}")

self.tot\_seat\_available=self.tot\_capacity-self.tot\_seats\_booked

temp\_output=[[str(bd.audience\_id[0]),str(bookings),\

str(self.tot\_cost),str("1001 plus "+str(bookings))]]

output=pd.DataFrame(temp\_output)

# Output file that shows booking confirmation details

output.to\_csv("confirmation\_details.csv",index=False,\

mode="a",header=False)

def match\_schedule(self,tot\_capacity):

self.tot\_capacity=tot\_capacity

print("\nSchedule is")

print(f"1){df.AWAY[0]} X {df.HOME[0]} {df.FORMAT[0]} on {df.DATE[0]}")

print(f"2){df.AWAY[1]} X {df.HOME[1]} {df.FORMAT[1]} on {df.DATE[1]}")

print(f"3){df.AWAY[2]} X {df.HOME[2]} {df.FORMAT[2]} on {df.DATE[2]}")

print("Match to be booked:",bd.match\_number[0])

match\_booked=bd.match\_number[0]

print("Seat Available:",self.seats\_available())

self.book\_ticket(match\_booked)

mumbai=stadium()

kolkata=stadium()

chennai=stadium()

def match():

print("\n\nWhich stadium to book tickets")

print("1)Mumbai")

print("2)Kolkata")

print("3)Chennai")

print("Choose Stadium:",bd.city[0])

selected\_stadium=bd.city[0]

if selected\_stadium == 1:

mumbai.match\_schedule(50000)

elif selected\_stadium == 2:

kolkata.match\_schedule(60000)

elif selected\_stadium == 3:

chennai.match\_schedule(70000)

def login():

if log.original\_id[0] == bd.audience\_id[0]:

match()

elif log.original\_id[1] == bd.audience\_id[0]:

match()

else:

print("No ID Found")

login()

## PEP8 Online

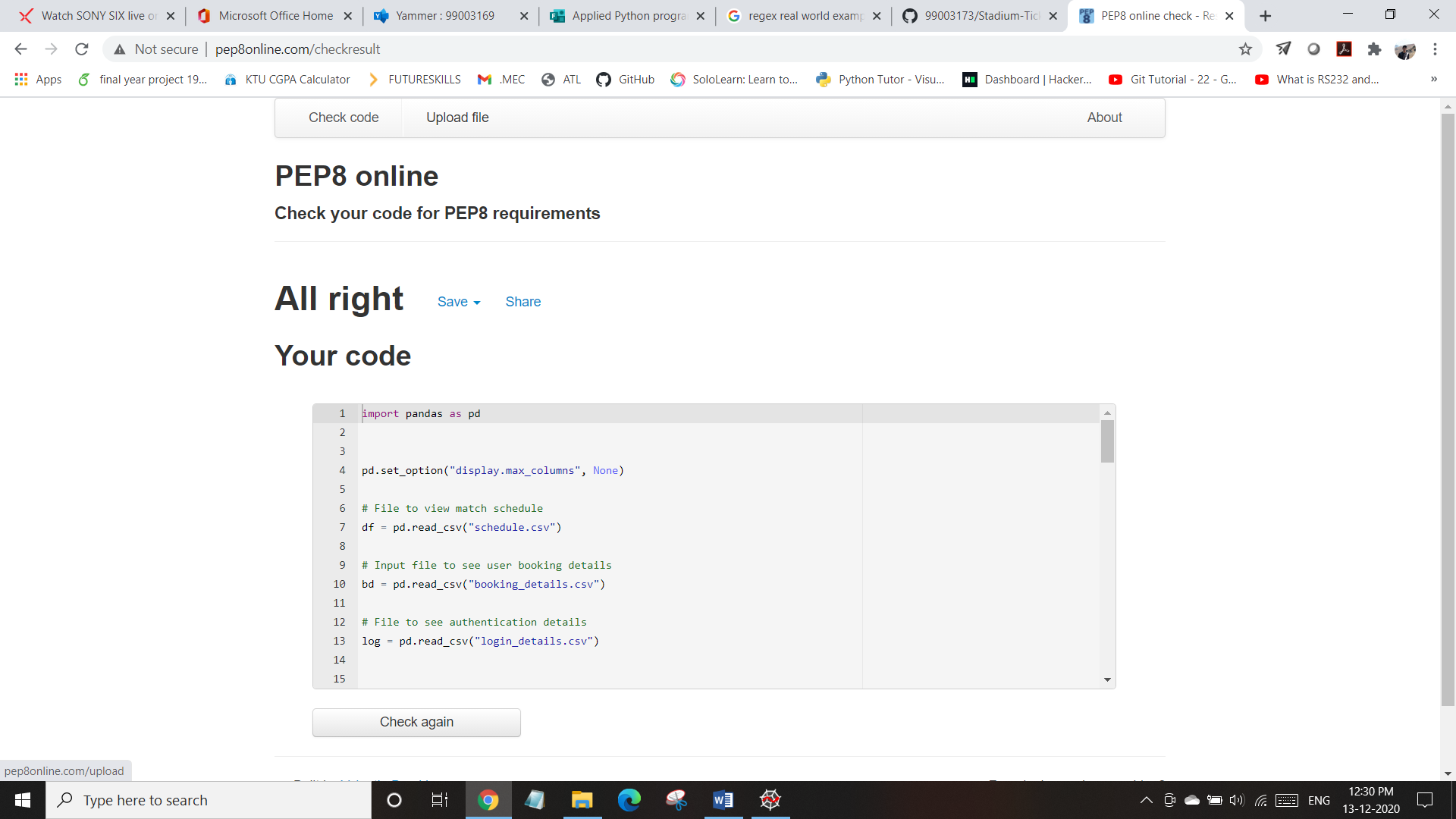


Figure : PEP8 ONLINE SCREENSHOT ACTIVITY 1

## Input Files Screenshot

### Match Schedule Input Excel File

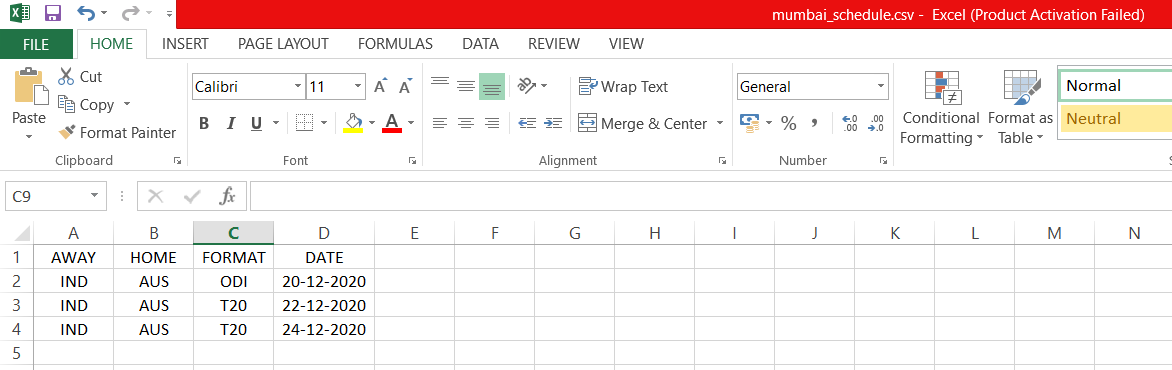


Figure : MATCH SCHEDULE SCREENSHOT ACTIVITY 1

### Booking Detail Input Excel File

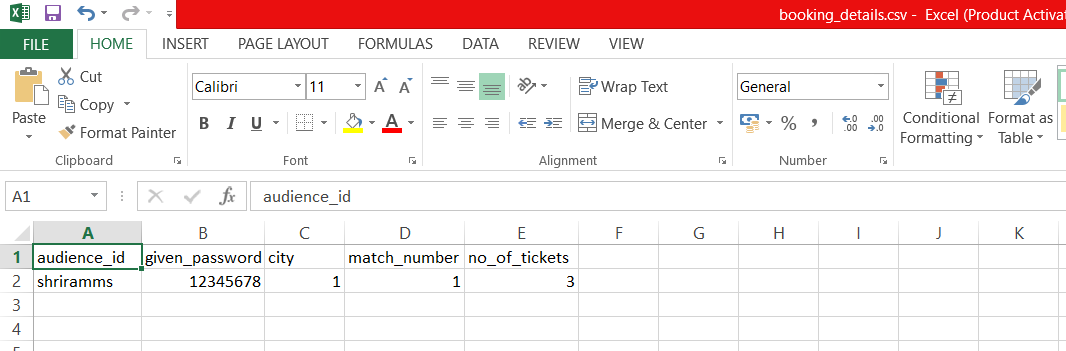


Figure : BOOKING DETAIL SCREENSHOT ACTIVITY 1

### Login Detail Input Excel File

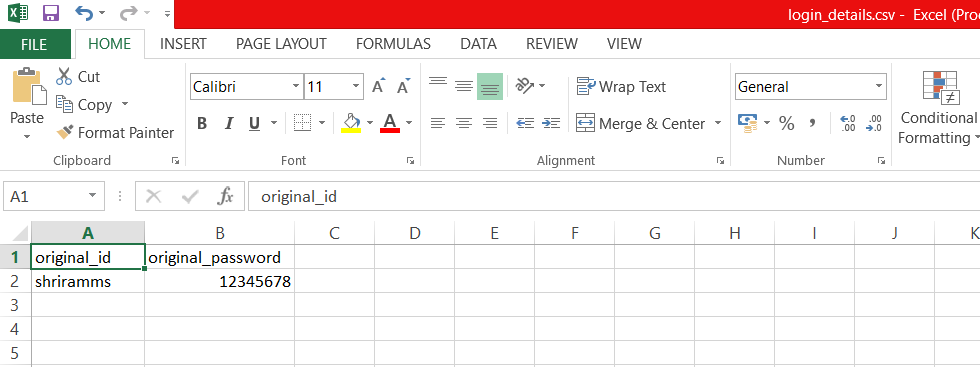


Figure : LOGIN DETAIL SCREENSHOT ACTIVITY 1

## Output Files Screenshot

### Confirmation Details Output Excel File

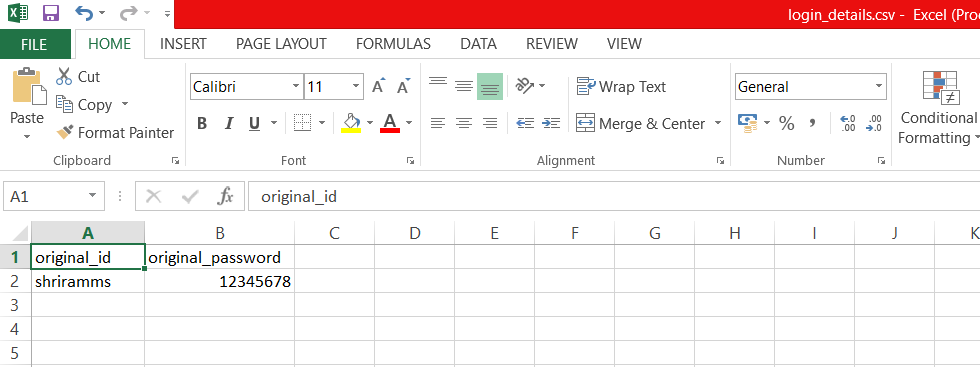


Figure : CONFIRMATION DETAIL SCREENSHOT ACTIVITY 1

# ACTIVITY 2 (12/12/2020)

## Python Code

import pandas as pd

from datetime import datetime

pd.set\_option("display.max\_columns", None)

# File to view match schedule's

mf = pd.read\_csv("mumbai\_schedule.csv")

kf = pd.read\_csv("kolkata\_schedule.csv")

cf = pd.read\_csv("chennai\_schedule.csv")

# Class to update seat availability for all the matches

class seat:

# Function to initialize attributes

def \_\_init\_\_(self):

self.tot\_seats\_available\_1 = 0

self.tot\_seats\_booked\_1 = 0

self.tot\_seats\_available\_2 = 0

self.tot\_seats\_booked\_2 = 0

self.tot\_seats\_available\_3 = 0

self.tot\_seats\_booked\_3 = 0

# Function to update seat availability

def seats\_available(self, tot\_capacity, match\_booked):

if match\_booked == 1:

self.tot\_seats\_available\_1 = tot\_capacity - self.tot\_seats\_booked\_1

return self.tot\_seats\_available\_1

elif match\_booked == 2:

self.tot\_seats\_available\_2 = tot\_capacity - self.tot\_seats\_booked\_2

return self.tot\_seats\_available\_2

elif match\_booked == 3:

self.tot\_seats\_available\_3 = tot\_capacity - self.tot\_seats\_booked\_3

return self.tot\_seats\_available\_3

# Function to display stadium seat availability details

def display\_stadium\_detail(self, tot\_capacity):

print(f"\nTotal Capacity:{tot\_capacity}")

if self.match\_booked == 1:

print(f"Total Seats Booked:{self.tot\_seats\_booked\_1}")

print(f"Total Seats Available:

{self.seats\_available(tot\_capacity, self.match\_booked)}")

elif self.match\_booked == 2:

print(f"Total Seats Booked:{self.tot\_seats\_booked\_2}")

print(f"Total Seats Available: {self.seats\_available

(tot\_capacity,

self.match\_booked)}")

elif self.match\_booked == 3:

print(f"Total Seats Booked:{self.tot\_seats\_booked\_3}")

print(f"Total Seats Available: {self.seats\_available

(tot\_capacity,

self.match\_booked)}")

# Class to update the audience member's name

class name:

# Function to initialize attributes

def \_\_init\_\_(self):

self.username = ""

def name\_entry(self, username):

self.username = username

# Class to save details about an audience member's ticket bookings

class user(seat, name):

# Function to initialize attributes

def \_\_init\_\_(self, stadium\_name):

self.stadium\_name = stadium\_name

self.bookings = 0

self.tot\_cost = 0

self.match\_booked = 0

self.date = 0

self.time = 0

super().\_\_init\_\_()

# Function to display booking details

def booking\_details(self):

print(f"\nUsername : {self.username}")

print(f"Stadium : {self.stadium\_name}")

print(f"Match : {self.match\_booked}")

print(f"Tickets : {self.bookings}")

print(f"Total Cost : {self.tot\_cost}")

print(f"Booked on : {self.date} {self.time}")

if self.stadium\_name == "Mumbai":

temp\_output = [[self.username, self.stadium\_name,

str(f"{mf.AWAY[self.match\_booked-1]} X

{mf.HOME[self.match\_booked-1]}

{mf.FORMAT[self.match\_booked-1]}"),

str(f"{mf.DATE[self.match\_booked-1]}"),

self.bookings, self.tot\_cost,

self.date, self.time]]

if self.stadium\_name == "Kolkata":

temp\_output = [[self.username, self.stadium\_name,

str(f"{kf.AWAY[self.match\_booked-1]} X

{kf.HOME[self.match\_booked-1]}

{kf.FORMAT[self.match\_booked-1]}"),

str(f"{kf.DATE[self.match\_booked-1]}"),

self.bookings, self.tot\_cost,

self.date, self.time]]

if self.stadium\_name == "Chennai":

temp\_output = [[self.username, self.stadium\_name,

str(f"{cf.AWAY[self.match\_booked-1]} X

{cf.HOME[self.match\_booked-1]}

{cf.FORMAT[self.match\_booked-1]}"),

str(f"{cf.DATE[self.match\_booked-1]}"),

self.bookings, self.tot\_cost,

self.date, self.time]]

output = pd.DataFrame(temp\_output)

# Output file that shows booking confirmation details

output.to\_csv("booking\_details.csv",

index=False, mode="a", header=False)

# Function to book tickets

def book\_ticket(self, cost, tot\_capacity):

self.bookings = int(input(f"Tickets(Rs{cost}) to Book:"))

self.date = datetime.now().date()

self.time = datetime.now().time()

print(f"Booked on {self.date} at {self.time}")

if self.match\_booked == 1:

if self.bookings <= self.seats\_available(tot\_capacity,

self.match\_booked):

self.tot\_cost = cost\*self.bookings

self.booking\_details()

self.tot\_seats\_booked\_1 += self.bookings

self.tot\_seats\_available\_1 =

tot\_capacity - self.tot\_seats\_booked\_1

elif self.match\_booked == 2:

if self.bookings <= self.seats\_available(tot\_capacity,

self.match\_booked):

self.tot\_cost = cost\*self.bookings

self.booking\_details()

self.tot\_seats\_booked\_2 += self.bookings

self.tot\_seats\_available\_2 =

tot\_capacity - self.tot\_seats\_booked\_2

elif self.match\_booked == 3:

if self.bookings <= self.seats\_available(tot\_capacity,

self.match\_booked):

self.tot\_cost = cost\*self.bookings

self.booking\_details()

self.tot\_seats\_booked\_3 += self.bookings

self.tot\_seats\_available\_3 =

tot\_capacity - self.tot\_seats\_booked\_3

# Objects of user pertaining to particular stadiums

u1 = user("Mumbai")

u2 = user("Kolkata")

u3 = user("Chennai")

# Class to update cost of each stadiums

class cost:

# Function to initialize attributes

def \_\_init\_\_(self):

self.cost = 0

def cost\_detail(self, cost):

self.cost = cost

# Class to save details about stadiums

class stadium(cost):

# Function to initialize attributes

def \_\_init\_\_(self):

self.tot\_capacity = 0

super().\_\_init\_\_()

# Function to update stadium capacity and ticket cost

def capacity\_detail(self, tot\_capacity, cost):

self.tot\_capacity = tot\_capacity

self.cost\_detail(cost)

# Function to see schedule of matches

def match\_schedule(self, u):

if u.stadium\_name == "Mumbai":

print("\nSchedule is")

print(f"1.{mf.AWAY[0]} X {mf.HOME[0]}

{mf.FORMAT[0]} on {mf.DATE[0]}")

print(f"2.{mf.AWAY[1]} X {mf.HOME[1]}

{mf.FORMAT[1]} on {mf.DATE[1]}")

print(f"3.{mf.AWAY[2]} X {mf.HOME[2]}

{mf.FORMAT[2]} on {mf.DATE[2]}")

u1.match\_booked = int(input("Match to be booked:"))

u1.display\_stadium\_detail(self.tot\_capacity)

u1.name\_entry(input("Enter Username"))

u1.book\_ticket(self.cost, self.tot\_capacity)

u1.display\_stadium\_detail(self.tot\_capacity)

if u.stadium\_name == "Kolkata":

print("\nSchedule is")

print(f"1.{kf.AWAY[0]} X {kf.HOME[0]}

{kf.FORMAT[0]} on {kf.DATE[0]}")

print(f"2.{kf.AWAY[1]} X {kf.HOME[1]}

{kf.FORMAT[1]} on {kf.DATE[1]}")

print(f"3.{kf.AWAY[2]} X {kf.HOME[2]}

{kf.FORMAT[2]} on {kf.DATE[2]}")

u2.match\_booked = int(input("Match to be booked:"))

u2.display\_stadium\_detail(self.tot\_capacity)

u2.name\_entry(input("Enter Username"))

u2.book\_ticket(self.cost, self.tot\_capacity)

u2.display\_stadium\_detail(self.tot\_capacity)

if u.stadium\_name == "Chennai":

print("\nSchedule is")

print(f"1.{cf.AWAY[0]} X {cf.HOME[0]}

{cf.FORMAT[0]} on {cf.DATE[0]}")

print(f"2.{cf.AWAY[1]} X {cf.HOME[1]}

{cf.FORMAT[1]} on {cf.DATE[1]}")

print(f"3.{cf.AWAY[2]} X {cf.HOME[2]}

{cf.FORMAT[2]} on {cf.DATE[2]}")

u3.match\_booked = int(input("Match to be booked:"))

u3.display\_stadium\_detail(self.tot\_capacity)

u3.name\_entry(input("Enter Username"))

u3.book\_ticket(self.cost, self.tot\_capacity)

u3.display\_stadium\_detail(self.tot\_capacity)

# Objects of stadium created for user access

mumbai = stadium()

kolkata = stadium()

chennai = stadium()

# Function acting as a menu to show stadiums available for booking

def match():

print("\n\nWhich stadium to book tickets")

print("1)Mumbai")

print("2)Kolkata")

print("3)Chennai")

selected\_stadium = int(input("Choose Stadium:"))

if selected\_stadium == 1:

mumbai.capacity\_detail(50000, 1200)

mumbai.match\_schedule(u1)

match()

elif selected\_stadium == 2:

kolkata.capacity\_detail(60000, 1100)

kolkata.match\_schedule(u2)

match()

elif selected\_stadium == 3:

chennai.capacity\_detail(70000, 1000)

chennai.match\_schedule(u3)

match()

else:

return 0

match()

## PEP8 Online

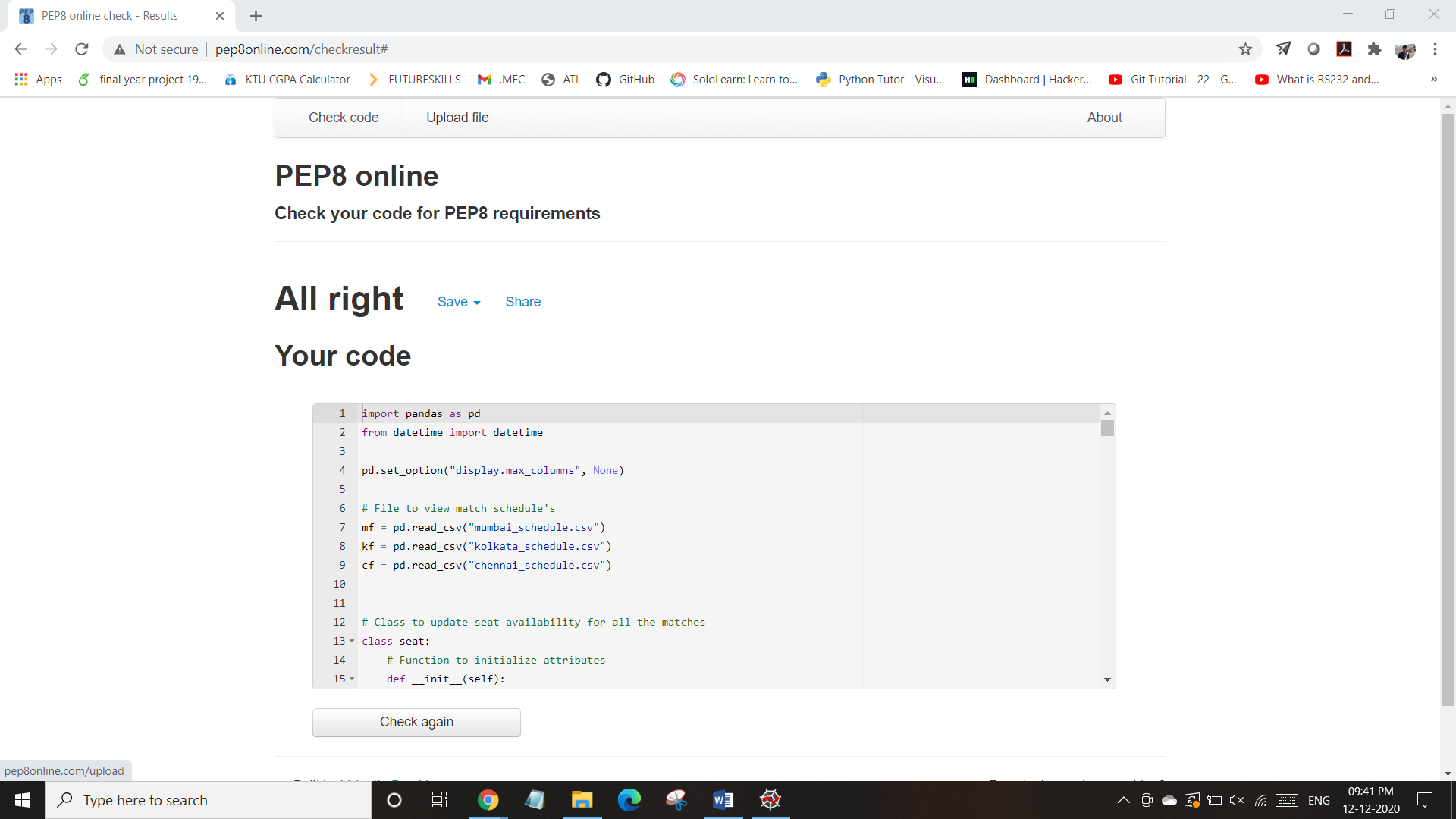


Figure :PEP8 ONLINE SCREENSHOT ACTIVITY 2

## Input Files Screenshot

### Mumbai Stadium Schedule - Input Excel File

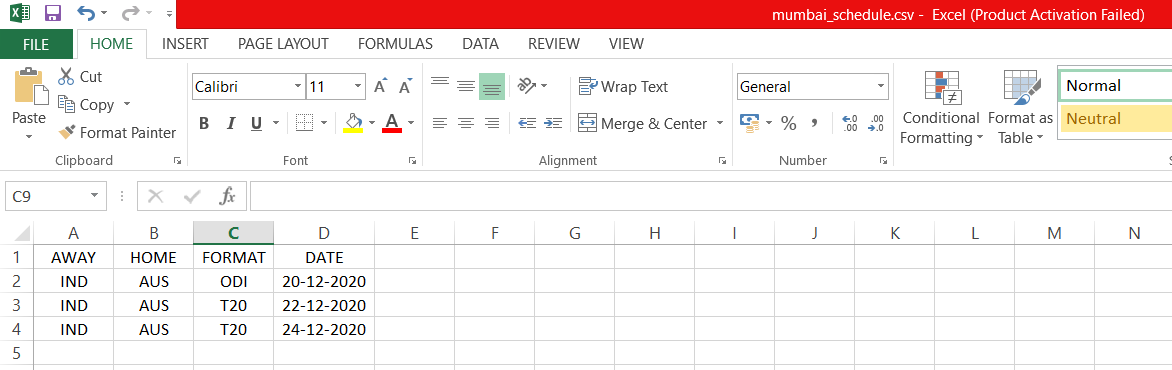


Figure : MUMBAI SCHEDULE SCREENSHOT ACTIVITY 2

### Kolkata Stadium Schedule - Input Excel File

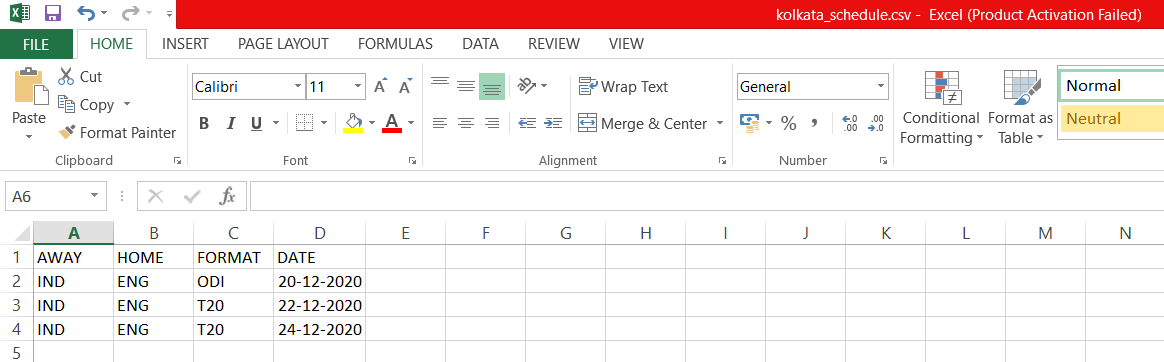


Figure : KOLKATA SCHEDULE SCREENSHOT ACTIVITY 2

### Chennai Stadium Schedule - Input Excel File

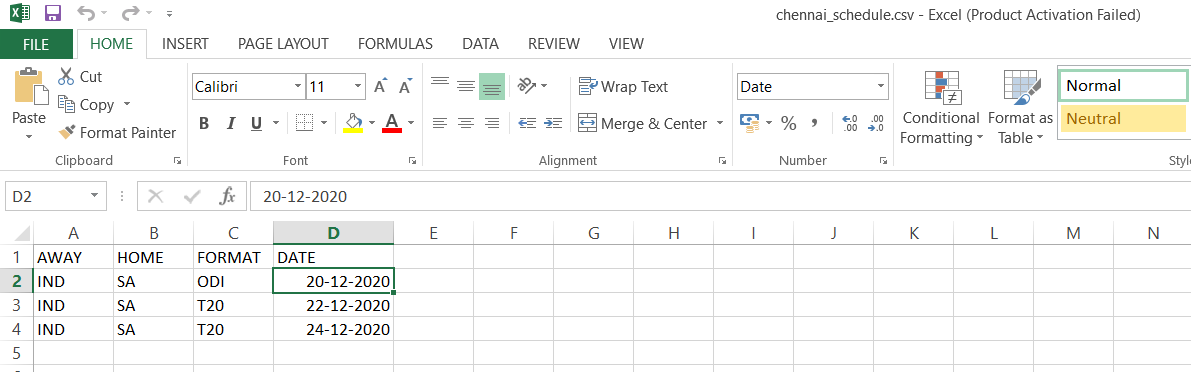


Figure : CHENNAI SCHEDULE SCREENSHOT ACTIVITY 2

## Output Files Screenshot

### Booking Detail - Output Excel File

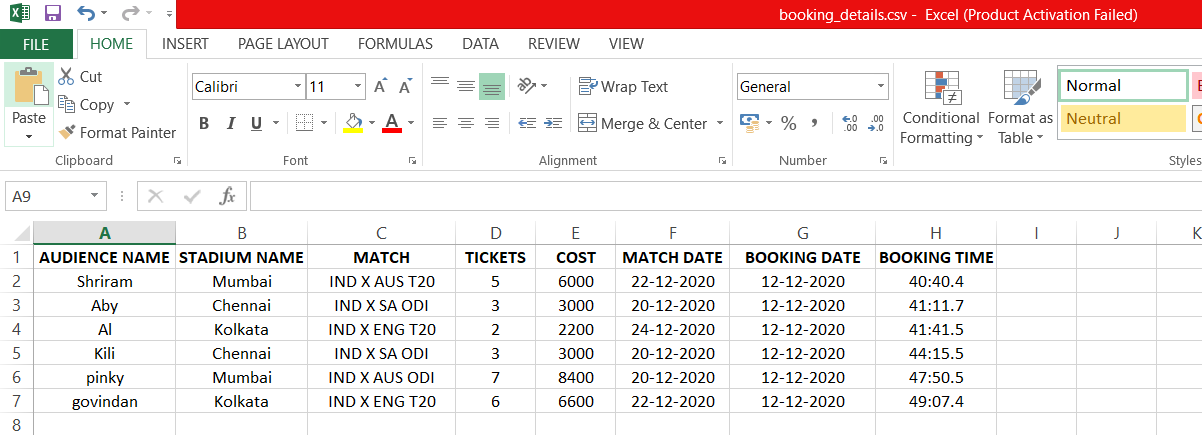


Figure : BOOKING DETAILS SCREENSHOT ACTIVITY 2

# RegEx Practice

## Code

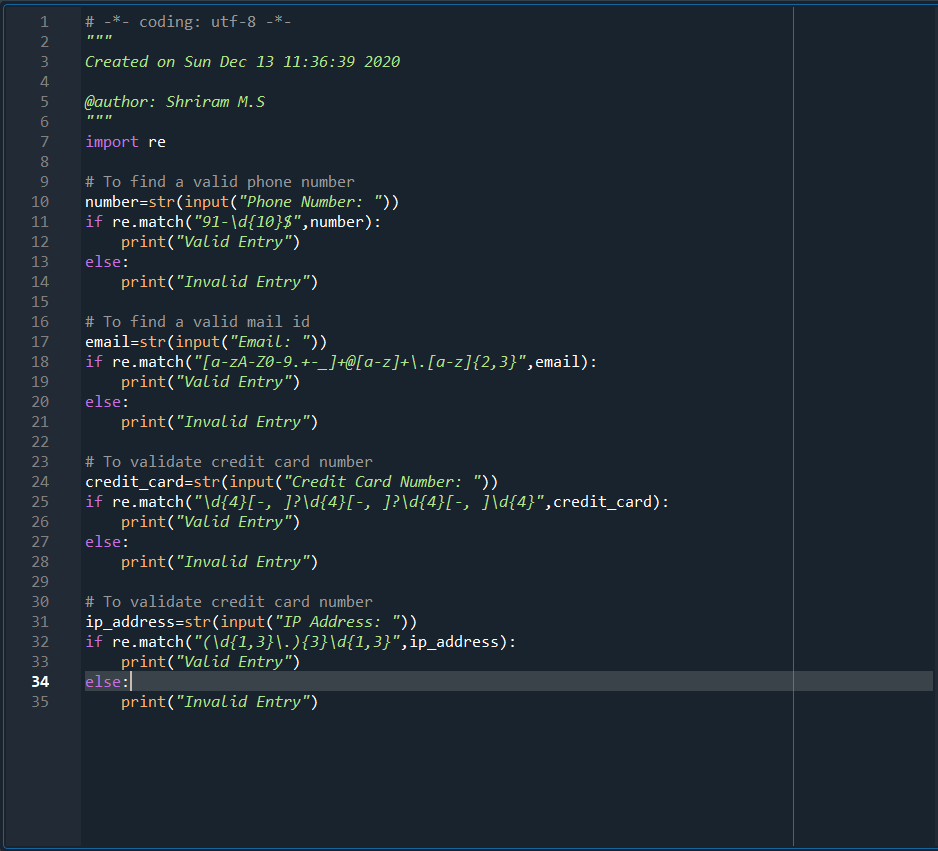


Figure : REGEX PRACTICE CODE SCREENSHOT

## Output

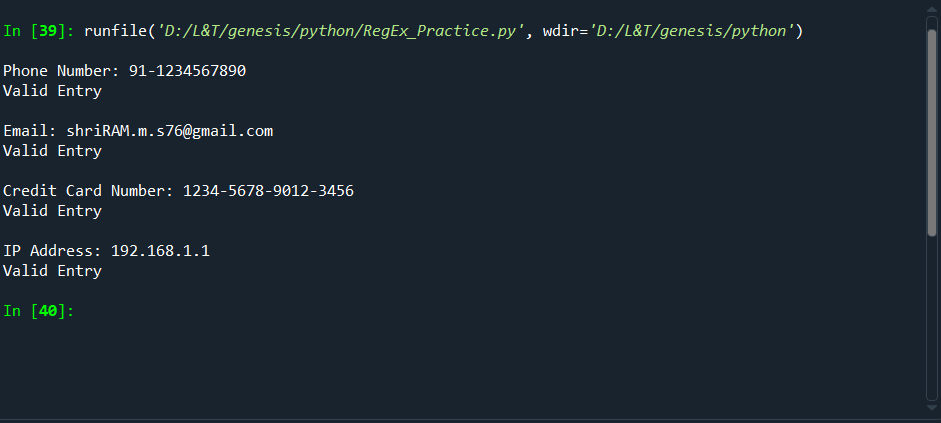


Figure : REGEX PRACTICE CODE OUTPUT SCREENSHOT