**Question - 1:**

Write a python program that plays a game “guess the number” as follows: The program chooses the number to be guessed by selecting an integer at random in the range 1 to 1000. The program then types:

* I have a number between 1 to 1000.
* Can you guess my number?
* Please type your first guess.

The player then types the first guess. The program responds with one of the following.

* Excellent!! You guessed the number!!! Would you like to play again (y or n)?
* Too high. Try again.
* Too low. Try again.

If the player’s guess is incorrect, you should loop (keep telling the player Too low or Too high) until the player finally gets the number right. Additionally, the program should count the number of guesses the player makes. If the player makes more than 10 guesses, then print “You should be able to do better!!!”, if he/she makes 10 guesses then print “Ahah! You know the secret!”, otherwise print “Either you know the secret or you got lucky!”

**Run Program:**

py oguz-aslanturk-dede-1.py

**Sample User Input - 1:**

*I have a number between 1 to 1000.*

*Can you guess my number?*

*Please type your guess: 500*

*Too low. Try again.*

*Please type your guess: 900*

*Too high. Try again.*

*Please type your guess: 600*

*Too low. Try again.*

**Sample Output - 1:**

*Please type your guess: 709*

*Excellent!! You guessed the number!!!*

*Either you know the secret or you got lucky*

**Sample User Input - 2:**

*I have a number between 1 to 1000.*

*Can you guess my number?*

*Please type your first guess: 500*

*Too low. Try again.*

*Please type your guess: 750*

*Too high. Try again.*

*Please type your guess: 625*

*Too high. Try again.*

*Please type your guess: 570*

*Too high. Try again.*

*Please type your guess: 550*

*Too high. Try again.*

*Please type your guess: 525*

*Too high. Try again.*

*Please type your guess: 515*

*Too high. Try again.*

*Please type your guess: 507*

*Too low. Try again.*

*Please type your guess: 510*

*Too low. Try again.*

*Please type your guess: 513*

*Too high. Try again.*

**Sample Output - 2:**

*Please type your guess: 511*

*Excellent!! You guessed the number!!!*

*You should be able to do better!!!*

**Code:**

**import random**

**def run():**

**number\_to\_guess = random.randint(1, 1000)**

**print("I have a number between 1 to 1000.")**

**print("Can you guess my number?")**

**result = False**

**guess\_count = 1**

**is\_first\_guess = True**

**while not result:**

**result = check\_guess(number\_to\_guess, is\_first\_guess)**

**is\_first\_guess = False**

**guess\_count += 1**

**print\_result(guess\_count)**

**def check\_guess(number\_to\_guess, is\_first\_guess):**

**user\_guess = get\_user\_guess(is\_first\_guess)**

**if number\_to\_guess == user\_guess:**

**print("Excellent!! You guessed the number!!!")**

**return True**

**else:**

**if number\_to\_guess > user\_guess:**

**print("Too low. Try again.")**

**else:**

**print("Too high. Try again.")**

**return False**

**def print\_result(guess\_count):**

**if guess\_count == 10:**

**print("Ahah! You know the secret!")**

**else:**

**if guess\_count > 10:**

**print("You should be able to do better!!!")**

**else:**

**print("Either you know the secret or you got lucky")**

**def get\_user\_guess(is\_first\_guess):**

**first\_guess\_text = ""**

**while True:**

**try:**

**if is\_first\_guess: first\_guess\_text = "first "**

**user\_guess = int(input(f"Please type your {first\_guess\_text}guess: "))**

**except ValueError:**

**print("Please enter a valid integer 1-1000")**

**continue**

**if 1 <= user\_guess <= 1000:**

**return user\_guess**

**else:**

**print('The integer must be in the range 1-1000')**

**run()**

**Question - 2:**

A parking garage charges a $3.00 minimum fee to park for up to two hours. The garage charges an additional $1.00 per hour for each hour or part thereof in excess of two hours. The maximum charge for any given 24-hour period is $20.00. Assume that no car parks for longer than 24 hours at a time. Write a program that will read the hours parked for each customer who parked his/her car in this garage yesterday. Then calculate and display the parking charge for each car, the average parking hours for yesterday, and the total of yesterday’s receipt. The program should use the function DetermineCharges to determine the charge for each customer. Assume that you don’t know the number of customers in advance

**Run Program:**

py oguz-aslanturk-dede-2.py

**Sample User Input - 1:**

*Please type parking hour (0 for exit): 20*

*Charge for 20 hour is : 20*

**Sample Output - 1:**

*Total receipt for yesterday is : 20*

*Average parking hour is : 20.00*

**Sample User Input - 2:**

*Please type parking hour (0 for exit): 5*

*Charge for 5 hour is : 6*

**Sample Output - 2:**

*Total receipt for yesterday is : 26*

*Average parking hour is : 12.50*

**Code:**

**MIN\_PARKING\_HOUR = 2**

**UP\_TO\_TWO\_HOUR\_CHARGE = 3**

**PER\_HOUR\_CHARGE = 1**

**MAX\_CHARGE = 20**

**def Run():**

**charge\_list = []**

**parking\_hour\_list = []**

**total\_receipt = 0**

**average\_parking\_hour = 0**

**while True:**

**parking\_hour = GetParkingHour()**

**if parking\_hour == 0:**

**PrintResult(total\_receipt, average\_parking\_hour)**

**return**

**charge = DetermineCharges(parking\_hour)**

**print(f"Charge for {parking\_hour} hour is : {charge}")**

**parking\_hour\_list.append(parking\_hour)**

**charge\_list.append(charge)**

**total\_receipt = TotalReceipt(charge\_list)**

**average\_parking\_hour = CalculateAverageParkingHour(parking\_hour\_list)**

**PrintResult(total\_receipt, average\_parking\_hour)**

**def DetermineCharges(hours\_parked):**

**if hours\_parked <= MIN\_PARKING\_HOUR:**

**charge = UP\_TO\_TWO\_HOUR\_CHARGE**

**if hours\_parked > MIN\_PARKING\_HOUR:**

**charge = (hours\_parked - MIN\_PARKING\_HOUR) \* PER\_HOUR\_CHARGE + UP\_TO\_TWO\_HOUR\_CHARGE**

**if charge > MAX\_CHARGE:**

**return MAX\_CHARGE**

**else:**

**return charge**

**def GetParkingHour():**

**while True:**

**try:**

**parking\_hour = int(input("Please type parking hour (0 for exit): "))**

**except ValueError:**

**print("Please enter a valid integer 0-24")**

**continue**

**if 0 <= parking\_hour <= 24:**

**return parking\_hour**

**else:**

**print('Parking hour should be in the range 1-24')**

**def TotalReceipt(charge\_list):**

**total\_receipt = 0**

**for charge in charge\_list:**

**total\_receipt += charge**

**return total\_receipt**

**def CalculateAverageParkingHour(parking\_hour\_list):**

**total\_parkig\_hour = 0**

**for hour in parking\_hour\_list:**

**total\_parkig\_hour += hour**

**return total\_parkig\_hour / len(parking\_hour\_list)**

**def PrintResult(total\_receipt, average\_parking\_hour):**

**print("####### CUMULATIVE RESULT ######")**

**print(f"Total receipt for yesterday is : {total\_receipt}")**

**print(f"Average parking hour is : {average\_parking\_hour:.2f}")**

**print("################################")**

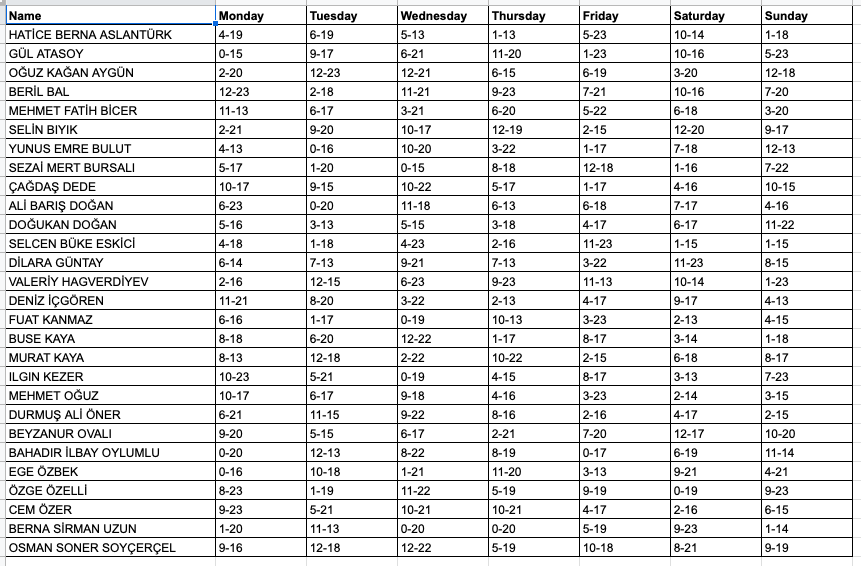
**Run()**

**Question - 3:**

We have a company working with shifts, so they assign shifts to staff. Company also cares about any excuses staff have. So they do not assign a shift to one that has any excuse on that shift period.

Here we have an excuse list from a company(This list is downloaded as CSV and saved as txt). Program users choose a day and decide the shift beginning hour and end hour. Program finds available users to assign the shift.(Employee do not have any excuse in that time period).

This list includes staff full name, days of week and time period the user has an excuse so not available for shift. Time period is formatted as start of excuse and end hour of excuse in 24 hour format.

****

**Run Program:**

py oguz-aslanturk-dede-3.py

**Sample User Input - 1:**

*Monday : 1*

*Tuesday : 2*

*Wednesday : 3*

*Thursday : 4*

*Friday : 5*

*Saturday : 6*

*Sunday : 7*

*Exit : 0*

*Please type shift day: 1*

*Please type shift beginning hour: 16*

*Please type shift end hour: 21*

**Sample Output - 1:**

*Available personnel for this shift:*

*GÜL ATASOY*

*MEHMET FATİH BİCER*

*YUNUS EMRE BULUT*

*DOĞUKAN DOĞAN*

*DİLARA GÜNTAY*

*VALERİY HAGVERDİYEV*

*FUAT KANMAZ*

*MURAT KAYA*

*EGE ÖZBEK*

*OSMAN SONER SOYÇERÇEL*

*OSMAN SONER SOYÇERÇEL*

**Sample User Input - 2:**

*Monday : 1*

*Tuesday : 2*

*Wednesday : 3*

*Thursday : 4*

*Friday : 5*

*Saturday : 6*

*Sunday : 7*

*Exit : 0*

*Please type shift day: 5*

*Please type shift beginning hour: 9*

*Please type shift end hour: 12*

**Sample Output - 2:**

*Available personnel for this shift:*

*SEZAİ MERT BURSALI*

**Code:**

**def run():**

**excuse\_file = open('personnel\_excuse\_list.txt', 'r')**

**line = excuse\_file.readline()**

**excuse\_list = []**

**while line != '':**

**line = excuse\_file.readline()**

**if line != '':**

**personnel\_excuses = line.split(',')**

**excuse\_list.append(personnel\_excuses)**

**excuse\_file.close()**

**day = get\_working\_day()**

**if day == 0:**

**return**

**working\_hour\_begins = get\_shift\_start()**

**working\_hour\_ends = get\_shift\_end(working\_hour\_begins)**

**available\_personnel = find\_available\_personnel(excuse\_list, day, working\_hour\_begins, working\_hour\_ends)**

**print\_result(available\_personnel)**

**def get\_working\_day():**

**print\_days()**

**while True:**

**try:**

**shift\_day = int(input("Please type shift day: "))**

**except ValueError:**

**print("Please enter a valid day 0-7")**

**continue**

**if 0 <= shift\_day <= 7:**

**return shift\_day**

**else:**

**print('Shift day should be in the range 1-7')**

**def get\_shift\_start():**

**while True:**

**try:**

**shift\_start = int(input("Please type shift beginning hour: "))**

**except ValueError:**

**print("Please enter a valid a valid shift 0-23")**

**continue**

**if 0 <= shift\_start <= 23:**

**return shift\_start**

**else:**

**print('Shift start should be in the range 0-23')**

**def get\_shift\_end(working\_hour\_begins):**

**while True:**

**try:**

**shift\_end = int(input("Please type shift end hour: "))**

**except ValueError:**

**print(f"Please enter a valid shift end hour {working\_hour\_begins}-23")**

**continue**

**if working\_hour\_begins < shift\_end <= 23:**

**return shift\_end**

**else:**

**print(f'Shift end should be in the range {working\_hour\_begins}-23')**

**def print\_result(available\_personnel):**

**print("Available personnel for this shift: ")**

**for personnel in available\_personnel:**

**print(personnel)**

**def print\_days():**

**print("Monday : 1")**

**print("Tuesday : 2")**

**print("Wednesday : 3")**

**print("Thursday : 4")**

**print("Friday : 5")**

**print("Saturday : 6")**

**print("Sunday : 7")**

**print("Exit : 0")**

**def find\_available\_personnel(excuse\_list, day, working\_hour\_begins, working\_hour\_ends):**

**available\_personnel = []**

**for personnel\_excuse\_list in excuse\_list:**

**if check\_personnel\_status(personnel\_excuse\_list[day], working\_hour\_begins, working\_hour\_ends ):**

**available\_personnel.append(personnel\_excuse\_list[0])**

**return available\_personnel**

**def check\_personnel\_status(personnel\_excuse\_hours, working\_hour\_begins, working\_hour\_ends):**

**hours = personnel\_excuse\_hours.split('-')**

**excuse\_hour\_begins = int(hours[0])**

**excuse\_hour\_ends = int(hours[1])**

**if excuse\_hour\_ends <= working\_hour\_begins:**

**return True**

**if working\_hour\_ends <= excuse\_hour\_begins:**

**return True**

**return False**

**run()**