BAN401: Mid-term assignment 2018, group 132

This PDF is created from a Jupyter Notebook, output is shown after code chunk

Problem 1

In [5]:

```
new_list = [11, 2, 4, 5, 6, 7, 8, 9, 10] #list to check

def odd_printer(list):
    # go through array, using len(list) as the stopping point
    for i in range(len(list)):
        # first check if element is odd, then check if index is odd
        if new_list[i]%2 != 0 and i%2 != 0:
            # print element if both element and index is odd
            print(new_list[i])




odd_printer(new_list)  # test the odd_printer on the given list
```

```
5
7
9
```

Problem 2

In [5]:

```
def tax():
    # prompt user for input
    annual_income = input("Enter your annual income in numbers: ")
    # use the tax_calculator helper function to calculate the tax
    tax = tax_calculator(int(annual_income))
    print("Your tax is: " + str(tax)) #cast tax to string and print it
    # calc after tax income and cast it to string
    print("Your after tax income is: " + str(int(annual_income)-int(tax)))


def tax_calculator(income):
    #use variables instead of numbers in the ifelif statement
    #This way it's easier to change the tax classes if later needed
    taxClass1 = 55000
    taxClass2 = 90000
    taxClass3 = 190000
    taxClass4 = 390000

    #this ifelif statement calculates the tax using a progressive tax system
    if(income < taxClass1):
        return income * 0.00
    elif(income <= taxClass2):
        return (income-taxClass2) * 0.1
    elif(income <= taxClass3):
        return (income-taxClass2) * 0.2+(taxClass2-taxClass1) * 0.1
    elif(income <= taxClass4):
        return (income-taxClass3) * 0.3 + (taxClass3 - taxClass2) * 0.2 + (taxClass
    elif(income >= taxClass4):
        return (income - taxClass4)*0.4 + (taxClass4-taxClass3) * 0.3 + (taxClass3 -
    (taxClass2-taxClass1) * 0.1


tax() # call the tax function
```

```
Enter your annual income in numbers: 190000
Your tax is: 23500.0
Your after tax income is: 166500
```

Problem 3

In [7]:

```
def print_x(n):
        for i in range(n):
            for j in range(n):
                if(j == i):  # print first x in the row
                    print("#", end = '')
                # print last x in row, decremented by i each loop
                elif(j == n-1-i):
                    print("#", end= '')
                else:
                    print(" ", end = '')  # print space
            print("")  # print nothing so we get next line

print_x(5)
print('\n')
print_x(10)
```

```
#   #
 # #
  #
 # #
#   #


#         #
 #       #
  #     #
   #   #
    ##
    ##
   #   #
  #     #
 #       #
#         #
```

Problem 4

In [9]:

```
from math import sqrt

list = [53.21, 50.73, 75.34, 45.18, 51.95, 47.47,
52.85, 49.93, 49.66, 46.09, 50.16, 47.21, 46.02,
47.87, 49.40, 51.47, 22.80, 46.80, 49.66, 53.24] #the input data

def check_if_normal_distr(observations):

    mean = calc_mean(observations) #call helper function to calculate mean
    #call helper function to calculate standard deviation
    stdev = calc_st_dev(observations)

    for observation in observations:
        print("Observation value "+ str(observation) + " " + is_withing_three(stdev
        # using the is_within_three function prints out if observation
        # value is within control limits




def calc_mean(observations):
    return (1/len(observations)) * sum(observations) #returns the mean


def calc_st_dev(observations):
    temp = 0
    mean = calc_mean(observations)
    for observation in observations: #loop throug list to find sum of (x-u)^2
        temp += (observation-mean)**2 #this is the :(x-u)^2
    return sqrt((1/len(observations)) * temp) #returns standard deviation

def is_withing_three(stdev, mean, observation):
    upperLimit = (mean+3*stdev) #set the upper limit
    lowerLimit = (mean-3*stdev) #set the lower limit
    #ifelse statement to print correct output
    if(observation >= lowerLimit and observation <= upperLimit ):
        return "is within the control limits"
    elif(observation>upperLimit):
        return "is above the upper control limit"
    elif(observation<lowerLimit):
        return "is under the lower control limit"
    else:
        "error"


check_if_normal_distr(list)
```

```
Observation value 53.21 is within the control limits
Observation value 50.73 is within the control limits
Observation value 75.34 is above the upper control limit
Observation value 45.18 is within the control limits
Observation value 51.95 is within the control limits
Observation value 47.47 is within the control limits
Observation value 52.85 is within the control limits
Observation value 49.93 is within the control limits
Observation value 49.66 is within the control limits
Observation value 46.09 is within the control limits
Observation value 50.16 is within the control limits
Observation value 47.21 is within the control limits
```

```
Observation value 46.02 is within the control limits
Observation value 47.87 is within the control limits
Observation value 49.4 is within the control limits
Observation value 51.47 is within the control limits
Observation value 22.8 is under the lower control limit
Observation value 46.8 is within the control limits
Observation value 49.66 is within the control limits
Observation value 53.24 is within the control limits
```

Problem 5

In [10]:

```
from random import randint

def game():
    correctGuess = randint(1,10) #make an random integer between 1 and 10
    guess = input("Please enter a number between 1 and 10 to play the game, or ente

    #while loop that continues until we either
    # get the right guess or if the user presses 0
    while int(guess) != correctGuess:
        if (int(guess) == 0): #if the user presses 0 we break
            break
        #if the guess is too high we tell the user
        elif(int(guess) > correctGuess):
            guess = input("Your guess is too high. Please guess a lower number\n")
        elif(int(guess) < correctGuess): #if the guess is too low we tell the user
            guess = input("Your guess is to low. Please guess a higher number\n")

    #if the user guessed the right number we congratulate the user
    if(int(guess)== correctGuess):
        print("Congratulations you guessed the right number, the number was: " + st
    #if the user has not guessed the right input but hit 0 we tell them that they q
    else:
        print("You quit! Better luck next time")



game()
```

```
Please enter a number between 1 and 10 to play the game, or enter 0 to
quit
7
Your guess is too high. Please guess a lower number
1
Your guess is to low. Please guess a higher number
4
Your guess is to low. Please guess a higher number
5
Your guess is to low. Please guess a higher number
6
Congratulations you guessed the right number, the number was: 6
```

Problem 6

In [18]:

```
def write_to_file(n):
    # open the file test.txt,
    # if it doesnt exist then it makes a file names test.txt.
    # We then tell the computer to open it
    # for reading and writing using w+ as the mode
    with open("test.txt", mode="w+") as file:
        #call the asterisks function to print the asterisks
        file.write(write_asterisks())
        for i in range(1,n+1): #outer loop for the value which we multiply by
            #inner loop for the value which is the 1-10 for each outer loop number
            for j in range(1,n+1):
                #write to the file using .format for a good output
                file.write("{0:>2} times {1:>2} is {2:>3} \n"
                           .format(str(j),   str(i), str(j * i)))
            #write asterisks at end of each times table
            file.write(write_asterisks())


def write_asterisks():
    #returns the right amount of asterisks for the times table
    return ("*"*18+"\n")


write_to_file(10)

#  display file in console
file = open('test.txt', 'r')
for line in file.readlines():
    print(line)
```

```
 8 times  9 is  72

 9 times  9 is  81

10 times  9 is  90

******************

 1 times 10 is  10

 2 times 10 is  20

 3 times 10 is  30

 4 times 10 is  40

 5 times 10 is  50

 6 times 10 is  60
```

Problem 7

In [19]:

```
revenueProductA = [700, 10000, 200, 33333, 4000, 666, 777, 150203] # from month 1 t
revenueProductB = [3333, 38586, 190293, 38383,70999, 13868, 113969, 190293] # from 1



def max_revenues(list1, list2):
    max_of_list1 = max(list1) #get the max value(s) of list 1
    max_of_list2 = max(list2) #get the max value(s) of list 2
    if max_of_list1 > max_of_list2: #compare the max of each list
        #use.format to get a nice output
        print( '${:,.2f}'.format(max_of_list1) +
              " is the highest revenue over months")
        print("That was the revenue for the following months: " + '{}'.format(
            # format the list from the find_max_months function
            # using format_custom_string
            str(format_custom_string(find_max_months(list1, max_of_list1)))))
    else:
        print('${:,.2f}'.format(max_of_list2) +
              " is the highest revenue over months") #same as above
        print("That was the revenue for the following months: "
              '{}'.format(str(format_custom_string(find_max_months(list2,
                                                                   max_of_list2)))))



#use this instead of .index() to find multiple values
def find_max_months(list, max_rev):
    indexes = [] #empty list
    for i in range(len(list)):
        #if we find the value append it to the empty list of indexes
        if (list[i] == max_rev):
            indexes.append(i+1)
    return indexes #return the index(es)


def format_custom_string(list):
    temp_str = "" #make an empty string
    for item in list:
        temp_str += (str(item)) #add item name
        temp_str += (" and ") #add and to separate the items
    temp_str = temp_str[:-5] #remove the extra and the for loop adds
    return temp_str #returns the string giving us a nice output


max_revenues(revenueProductA,revenueProductB)
```

```
$190,293.00 is the highest revenue over months
That was the revenue for the following months: 3 and 8
```

Problem 8

In [21]:

```
def char_counter():
    dict = {} #create empty dictionary
    sentence = input("Please enter a sentence\n") #get user input
    for char in sentence: #loop trough each character in the sentence
        if char == char in dict: #if the char is equal to a key in the dictionary,
            # We increment the value of the key
            dict[char] += 1
        # if the char is not already in the dictionary
        # we add the char to the dictionary
        else:
            dict[char] = 1
    print(dict)



char_counter()
```

```
Please enter a sentence
I am here
{'I': 1, ' ': 2, 'a': 1, 'm': 1, 'h': 1, 'e': 2, 'r': 1}
```

Problem 9

In [22]:

```
def int_number_game():
    correct_input = False #boolean for checking if correct input is given
    while correct_input != True:
        user_input = input("Please enter the positive number: ") #get user input
        if int(user_input) > 0: #if the number is positive
            # call helper function sum_up to get the sum from 1 to user_input,
            # cast it to a string and print it
            print("The total sum from 1 to " +
                  user_input + " is: "+str(sum_up(int(user_input))))
            #same as above except we call the sum_up_squares function
            print("The total sum of squares from 1 to "+
                  user_input + " is: " + str(sum_up_squares(int(user_input))))
            correct_input = True #end game
        #if the user input is negative or 0 we prompt the user for a new number
        else:
            print("Could you please enter positive number once again")


def sum_up(n):
    total = 0 #initialize int
    for i in range(n+1): #increment total with i until we get to the parameter n
        total += i
    return total #return total


def sum_up_squares(n):
    total = 0 #initialize int
    #increment total with square of i until we get to the paramter n
    for i in range(n+1):
        total += i**2
    return total #return total



int_number_game()
```

```
Please enter the positive number: -8
Could you please enter positive number once again
Please enter the positive number: 0
Could you please enter positive number once again
Please enter the positive number: 6
The total sum from 1 to 6 is: 21
The total sum of squares from 1 to 6 is: 91
```

Problem 10

In [24]:

```
num = [[2, 3, 4], [3, 4, 5], [5, 6, 8], [45, 67, 4]]



def sum_sublist(list):
    total = 0 #initalize an int
    for i in list: #loop through a sublist and add each number to total
        total += i
    return total #return total

def max_of_lists(list):
    max = 0 #initialize an int
    for i in list: #loop through the whole list
        temp = sum_sublist(i) #get the sum of sublist
        #if the sum of the sublist is bigger than what we have found before
        # we set max to it
        if temp > max:
            max = temp
    return max

def print_max_of_lists(list):
    print("The list provided to the function is" + str(list)) #print first list
    #print the max of the list using max_of_lists function
    print("The maximum sum of the list is: " + str(max_of_lists(list)))



print_max_of_lists(num)
```

```
The list provided to the function is[[2, 3, 4], [3, 4, 5], [5, 6, 8],
[45, 67, 4]]
The maximum sum of the list is: 116
```

Problem 11

In [26]:

```
nestedDict= {
    'APL': {'startTime': 0, 'endTime':1, 'startValue':1500, 'endValue':2000,},
    'TRVX': {'startTime': 1, 'endTime':2, 'startValue':750, 'endValue':800},
    'AKA': {'startTime': 3, 'endTime':5, 'startValue':4500, 'endValue':5300},
    'NAS': {'startTime': 0, 'endTime':2, 'startValue':250, 'endValue':400},
    'TEL': {'startTime': 1, 'endTime':2, 'startValue':900, 'endValue':1300},
              } #the dictionary for the assignment

def CAGR(startTime, endTime, startValue, endValue):
    pow = (1/(endTime-startTime)) #calculate the power to use
    #calculate the change in value from start to end
    valueChange = endValue/startValue
    # #return the valuechange with ther power we found and substract 1, ie calc CAG
    return (valueChange**pow)-1

#this adds cagr for each company in the dictionary
def add_cagr(nestedDict):
    for dict in nestedDict: #loop trough the nested dictionary
        #add cagr to each dictionary using cagr function
        nestedDict[dict]['CAGR'] =
        CAGR(nestedDict[dict]['startTime'],
             nestedDict[dict]['endTime'],
             nestedDict[dict]['startValue'],
             nestedDict[dict]['endValue'])

def print_highest_cagr(nestedDict):
    add_cagr(nestedDict)#calculate CAGR and add it to each companies dictionary
    max = 0 #initialize max for holding the max value
    dicts = [] #initialize an empty dictionary
    #the string we will use for storing the name of the company with the highest CA
    nameOfMax = "None"
    for name, dict in nestedDict.items(): #loop throug each company name
        for key, value in sorted(dict.items()): #loop through each key in the dicti
            if key == 'CAGR' and value > max: #if the key found is CAGR and the val
                max = value #set max to the value
                nameOfMax = name #set the nameOFMax to the name of the company
                #print the company with the highest CAGR and format it
    print("The company with the highest CAGR is: {0} with a CAGR of: {1}"
          .format(nameOfMax,round(max,2)))



print_highest_cagr(nestedDict)
```

```
The company with the highest CAGR is: TEL with a CAGR of: 0.44
```

Problem 12

In [28]:

```
#initialize the empty game board using a nested list
game_board = [['-', '-', '-'],
              ['-', '-', '-'],
              ['-', '-', '-']]

def insert_pieces(row, column,shape):
    #set the given spot of the game_board to the shape
    game_board[row][column] = shape

def print_game_board():
    for i in game_board:#loop through game_board and print it
        #the asteriks makes it so we dont get the '' around each -
        print(*i)
    print("") #get the next line



def game():
    gameover = False #gamestatus
    while gameover != True:
        print_game_board()#each loop we print the current game board
        row = input("Enter row index (type q to quit): ") #get row as input
        if row == 'q': #if the user types q we quit the game
            gameover = True
            break
        column = input("Enter col index: ") #get column as input
        shape = input("Enter shape: ") #get the shape the user wants to add
        insert_pieces(int(row),int(column),shape) #insert piece into game board

game() #run the game
```

```
- - -
- - -
- - -

Enter row index (type q to quit): 0
Enter col index: 0
Enter shape: X
X - -
- - -
- - -

Enter row index (type q to quit): 2
Enter col index: 1
Enter shape: O
X - -
- - -
- O -

Enter row index (type q to quit): 2
Enter col index: 2
Enter shape: X
X - -
- - -
- O X

Enter row index (type q to quit): q
```