The Molto Bene Pizzeria ordering system

Coding Each part explains

Setting the variables and restarting

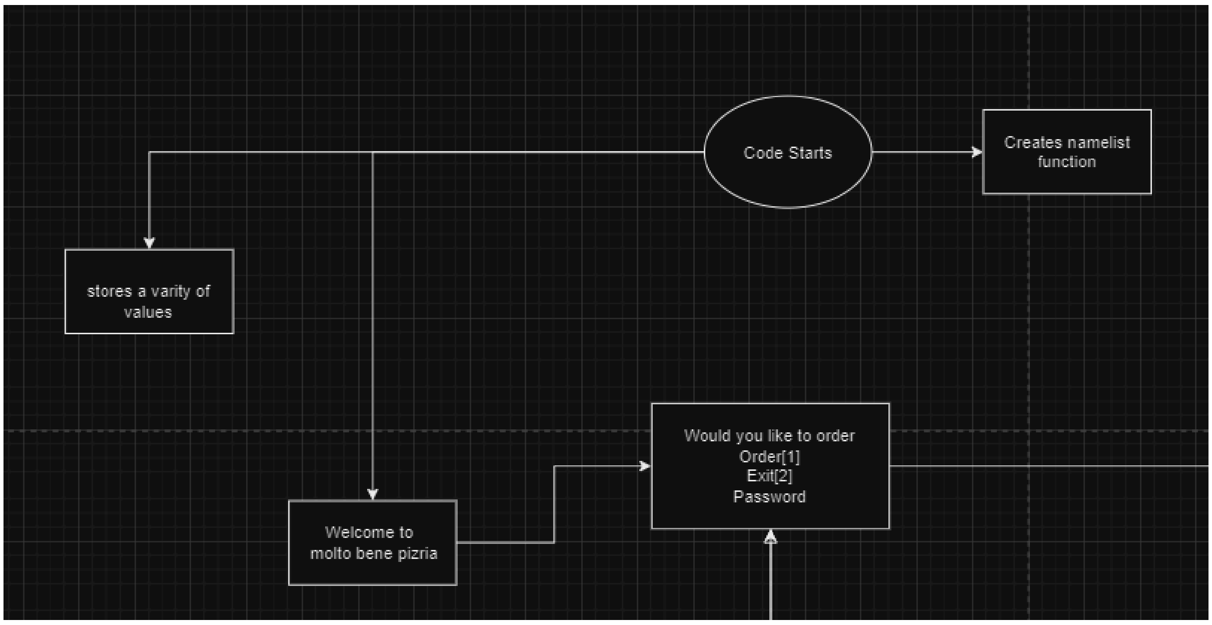
The below code shows variables and how they are set before functions are called

```python
import os  #This is for making the ui cleaner
import time#This is done to make things feel more natrual
pizza_wth_size = {} #Creates a dictonary to store all pizza and size togther
in one dictonary for printing
wordsize=[]  #This is the Size information stored in a string list for
printing
sizeList=[]  #Size list refers to the price of that pizza size
order=[]  #this stores the pizza order size
namelist=[]  #Stores all names in a list for the pickup Function
whichPizza=0  #A varible to that links to order to show which pizza they are
currently choosing a size for
whichdrink=0
check=1  #This is a varible that is used to insure that a name is in pickup
so when somone chooses to pickup a pizza if nothing is in the list of orders
it wont allow it
info=1 #This is the input for the start of the code and needs to go here as
the while loop to start the code relises on this varible not being 3
drinkorder=[] #This is for counting the order for drinks
sideorder=[] #Makes a list for the side orders
sideprice=[] #Makes a list for the price orders
drinksizelist=[] #Makes a drink size list fpr calculating price
drinkwordsize=[] #Makes a drink list for showing drink size
namelistremove=0 #This is a varible that counts how many names are in list
to stop some errors

def restarting():
#Creates a function called restarting that will restart the order when
called
    global whichPizza
#Globals to reset price for next order
    global whichdrink
    global wordsize
    global sizeList
    global order
    global check
    global info
    global namelist
    global namelistremove
    global drinkorder
    global sideorder
    global sideprice
    global drinksizelist
    global drinkwordsize
    print("\nRe Starting Order...")
#This restarts the order as the user asks
    time.sleep(1)
#Delay cuz it looks cool
    wordsize=[]
#Resets value for the restart
    sizeList=[]
    namelist.remove(namelist[namelistremove])
    order=[]
    drinkorder=[]
    drinksizelist=[]
    drinkwordsize=[]
    check=1
    sideorder=[]
    sideprice=[]
    info=1
    whichPizza=0
    whichdrink=0
    Start()
```

The first part of the code defines a variety of variables outside of the function. As well as imports the time module and os. Then creates a function called restarting which will be called on to restart the order. The variables are defined outside of the functions they are used in so that when the functions are called the values don't reset. While the restart function is made to give a cleaner design to the main menu function.

Another solution to the code would have been to not use as many variables and have order price and size all in one list. However, this would make it much more tedious to implement new features. The restart function could have just been put into the menu but this would make a much more clunky design.


Code Starts
Creates namelist function
stores a varity of values
Would you like to order Order[1] Exit[2] Password
Welcome to molto bene pizria

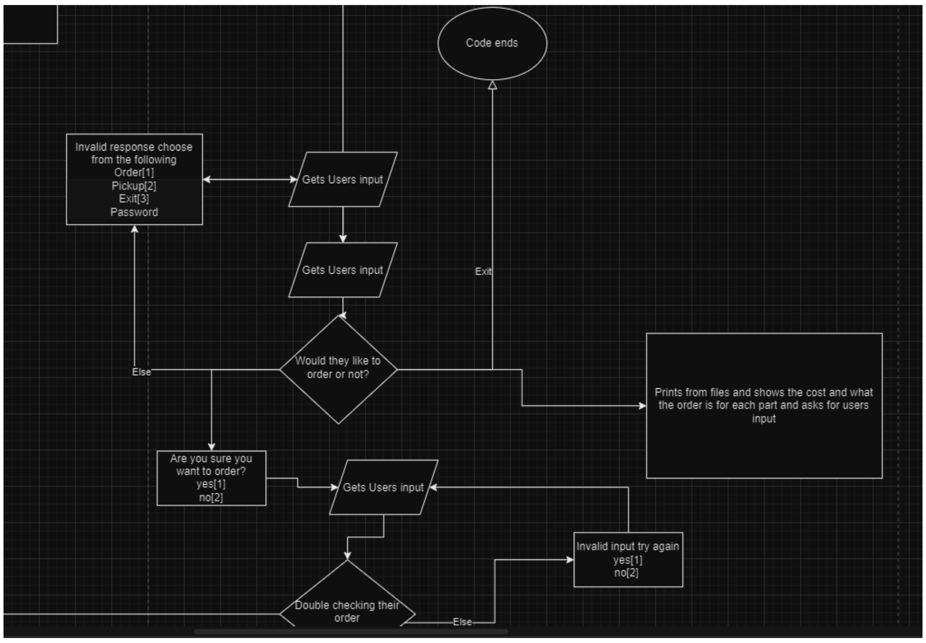
The start ordering menu

The below code is the start of the ordering process giving the user different choices to such as picking up an order or making one

```python
def Start(): #this is the start of what the user will see and will be called as a restart
    print("\n[1] Order")
    print("[2] Pickup")
    print("[3] Exit\n")
    while True: #While this loop is true witch means it repeats forever wont comment about this again unless there is a break function is used
        info=(input(""))  #Gets input and stores as a value wont talk about inputs again
        if info.isalpha(): #Uses the isalpha function to check if the input is correctos.
            system('cls')
             print("Checking...")
            time.sleep(1)
            print("\nThis is not a number input must be a number")
            print("[1] Order")
            print("[2] Pickup")
            print("[3] Exit\n")
        elif info.isdigit(): #uses the is digit function to check if its a digit if so it will procceed
            info=int(info) #Makes the input a digit so it can be comapred using if staments
            if info ==1:
                print("Starting order...")
                time.sleep(1)
                sure() #Calls a function which is the next step of the code
the is a double checking function
            elif info ==2:
                if len(namelist) >=1: #Counts the amount in namelist if
there is more then one that means an order can be picked up and will do the
indent
                    print("Sending to Pickup...")
                    time.sleep(1)
                    pickup()
                print("Checking...")
                time.sleep(1)
                print("\nCurrently we have no orders to be picked up")
#If info was equal to two but there is nothing in namlist this will print
                print("[1] Order")
                print("[2] Pickup")
                print("[3] Exit\n")
            elif info ==3:
                exit() #Exits program function
            else: #If its an invalid input it the following may occur
                print("Checking...")
                time.sleep(1)
                print("\nTry again")
                print("[1] Order")
                print("[2] Pickup")
                print("[3] Exit\n")
        else:
            print("Checking...")
            time.sleep(1)
            print("Try again")
            print("[1] Order")
            print("[2] Pickup")
            print("[3] Exit\n")
```

The code above runs through a while loop which will continue forever. This gets the input of the customer/user and then if the input is valid allows the user to continue and if invalid asks for input again. The while loops continue forever go through each possible outcome then back to the top asking for input but if an input is valid the next function is called. Note that the os.system('cls') was used in the final code but is not present here other than below as of indenting issues with the Python text box.

An alternative method to the above code would be instead of else statements to use cases. This would act very similar to the current menu function but would have made the code less readable for me as I prefer using if and else statements.



Code ends
Invalid response choose from the following Order[1] Pickup[2] Exit[3] Password
Gets Users input
Gets Users input
Exit
Else
Would they like to order or not?
Prints from files and shows the cost and what the order is for each part and asks for users input
Are you sure you want to order? yes[1] no[2]
Gets Users input
Invalid input try again yes[1] no[2]
Double checking their order
Else

The pickup function and name list function

The next part of the code is a function for picking up pizzas which is the following. As well as how the names of orders are gotten and how they are stored in a list.

```python
def pickup(): #Creates a function called pickup
    global namelistremove #this is important for checxking things and
avoiding error down in checkout line
    for i in namelist: #Creates a for loop for in name list and shows each
current order
            print("\n")
            x=[]
            file= open("WhatOrder"+i+".txt", "r")
            readinglines=file.readlines()
            for line in readinglines:
                x.append(line.strip())
            x.pop(-1)
            for z in x: #Makes a loop called z that will loop through each
part of the lsit in x
                print(z) #prints each part pf the list this will all loop
for all the orders
    print("\nEnter the name in which your ordered under to pick up your
order:\n")
    removelistname=input()
    if removelistname in namelist: #Checks if the input is valid if so the
indented will occur
        namelist.remove(removelistname) #removes name from the orderlist
        namelistremove=namelistremove-1
        open("WhatOrder"+removelistname+".txt","w").close() #Removes from
file so if the name is ordered again there is not two orders in one

        print("Thank you for shoping at Molto Bene Pizzeria",removelistname)
    else:
        print("Checking...")
        time.sleep(1) #Else bassically if they enter an invalid order
        print("This is not an order. If want want under this name go and
order order\n")
        Start()
    Start()
def namefunc(): #Makes a function called namefunc this is for name input and
list
    global name #Globals name so it can be accessed everywhere this is used
for checkout
    global namelist #Globals namelist this is done for the above code
```

The above code and comments explain two different functions. The first function is pickup which allows users to pick up their pizza after making their order. The second is a function for storing orders and getting the name of the customer.

The pickup function prints all the current orders and if there are none sends the user back. It prints these orders by making a for loop and printing each Txt file (order) and at the top printing the name using the reading function to read the file. After this, it gets the user's input to pick up their order. If their input is not an order the code sends the user back to the start. If it is an order it will be removed from the current orders and then send the user to the start.

One idea I decided not to implement was to make a dictionary with the name and order for each order made. Though this code would have worked in a similar sense in the customer's eyes the code would have been unnecessarily long for the pigsize function.

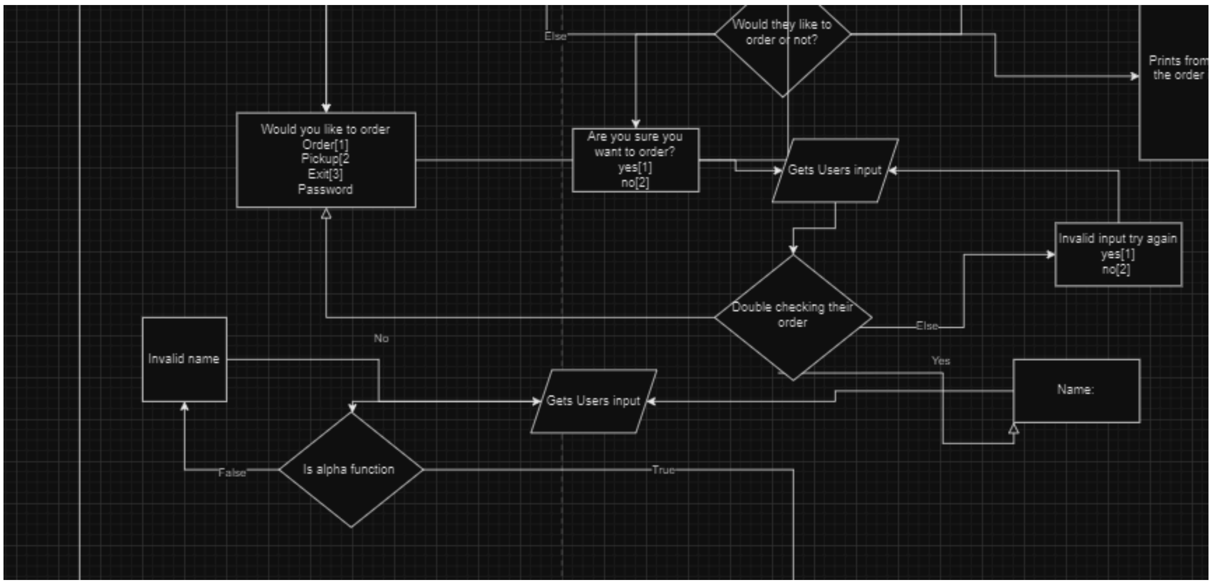
The next function name function does exactly as it says. This stores the input or name as well as appends it to a list. It does this by making a while true or a loop which will ask for input and if the inputs are invalid will loop back and ask for the input again. In the same sense as the pickup function, the order and name could have been combined into one. But as explained this would have made other parts of the code more tedious.

The double checking/are you sure function

Double checks users' input

```python
def sure(): #Creates a function called sure which asks again if they want to order or not
    print("\nAre you sure you want to order?")
    print("[1] Yes")
    print("[2] No\n")
    while True: #While true this loops forever
        return1=(input("")) #Gets input from the user
        if return1.isalpha(): #Uses the is alpha to check if the input is invalid
            print("Try again")
            print("[1] Yes")
            print("[2] No\n")
        elif return1.isdigit(): #Uses the is alpha to check if the input is valid
            return1=int(return1) #makes the input interger for comparing
            if return1 ==1: #if the input is 1 Calls namefunction
                namefunc()
            elif return1==2: #if the input is 2 Sends back to start function
                Start()
            else: #if else invalid input the indented will occur
                print("Try again")
                print("[1] Yes")
                print("[2] No\n")
        else: #if else invalid input the indented will occur
            print("Try again")
            print("[1] Yes")
            print("[2] No\n")
```

This code above is an additional confirmation that the user is inputting what they want before sending the user to the name function and then the menu to order. This is done again using the is alpha and is digit functions through a while loop using the same structure as the start function so the alternatives would be the same.


Would they like to order or not?
Else
Prints from the order
Would you like to order Order[1] Pickup[2 Exit[3] Password
Are you sure you want to order? yes[1] no[2]
Gets Users input
Invalid input try again yes[1] no[2]
Double checking their order
Else
Yes
No
Invalid name
Gets Users input
Name:
False
Is alpha function
True

The menu function

This is the function that allows the user to navigate Molto Bene Pizzeria's different options and what they sell

```
Python
def menu(): #This menu function is for the main navigation of the code
    print("\nChoose from the menu options below")
    print("[1] Pizza")
    print("[2] Drinks")
    print("[3] Sides")
    print("[4] View current order")
    print("[5] Restart Order")
    print("[6] Checkout\n")
    while True: #This is the main forever loop to give the user their
options
        info=input("")
        if info.isalpha(): #Uses the is alpha to check if the input is
invalid if its not and is alpha  the following will occur
            print("Checking...")
            time.sleep(1)
            print("\nTry again")
            print("[1] Pizza")
            print("[2] Drinks")
            print("[3] Sides")
            print("[4] View current order")
            print("[5] Restart Order")
            print("[6] Checkout\n")
        elif info.isdigit(): #Uses the isdigit to check if the input is 
valid if it is the following will occur
            info=int(info) #Makes the intput a interger for the comparision 
if staments
            if info == 1: #checks if the input is 1 if so the indented will 
occur
                pizza() #Calls the pizza function which is for ording pizza
            elif info==2: #checks if the input is 2 if so the indented will 
occur
                drinks()
            elif info==3:
                sides()
            elif info == 5:
                if check >=2:
                    restarting()
                print("Checking...")
                time.sleep(1)
                print("\nSorry but there is currently no order to restart") 
#Invalid response for users input
                print("[1] Pizza")
                print("[2] Drinks")
                print("[3] Sides")
                print("[4] View current order")
                print("[5] Restart Order")
                print("[6] Checkout\n") #calls the drinnks fuction which is 
for ording drinks
            elif info == 6: #checks if the input is 3 if so the indented 
will occur though to get sent to checkout another if sstament will occur
                if check >=2: #checks if the input is 2 if so the indented 
will occur this is done to check if there is anything orded yet if there is 
the indented will occur
                    checkout() #calls the checkout funciton
                print("Checking...")
                time.sleep(1)
                print("\nThere is currently nothing in your cart") #Invalid 
response Asks for users input
                print("[1] Pizza")
                print("[2] Drinks")
                print("[3] Sides")
                print("[4] View current order")
                print("[5] Restart Order")
                print("[6] Checkout\n")
            elif info ==4:
                if check >1: #Checks if there is any order yet if there is 
the indented will occur which shows current oder
                    d=0
                    for k in sizeList:
                        d=d+k
                    for k in drinksizelist:
                        d=d+int(k)
                    for k in sideprice:
                        d=d+int(k)
                    print("Checking...")
                    time.sleep(1)
                    print("\nCurrent Total $"+str(d))
                    print("\nCurrent Order:") #Shows current order
                    pizsize()
                    print("\n")   #Invalid response for users input
                    print("[1] Pizza")
                    print("[2] Drinks")
                    print("[3] Sides")
                    print("[4] View current order")
                    print("[5] Restart Order")
                    print("[6] Checkout\n")
                if check==1:
                    print("Checking...")
                    time.sleep(1)
                    print("\nSorry but there is currently no order to View") 
#Invalid response for users input
                    print("[1] Pizza")
                    print("[2] Drinks")
                    print("[3] Sides")
                    print("[4] View current order")
                    print("[5] Restart Order")
                    print("[6] Checkout\n")
            else: #If none of the above are true then its invalid and the 
following will occur
                print("[1] Pizza")
                print("[2] Drinks")
                print("[3] Sides")
                print("[4] View current order")
                print("[5] Restart Order")
                print("[6] Checkout\n")
        else: #If eveerything else is invalid the following will occur
            print("[1] Pizza")
            print("[2] Drinks")
            print("[3] Sides")
            print("[4] View current order\n")
            print("[5] Restart Order")
        print("[6] Checkout")
```

Similar to the starting order function this code gets the user's input and then goes through a variety of possible outcomes giving the User their desired response though if not an option sends the User back to try and input again. But unlike the start function, lots of different functions are called depending on the user's input. This is done so that instead of the code being messy it is given a much simpler design. Though an alternative would be making the prints functions as well this would cut lines but make the code harder to navigate.

The flowchart for this function is in the classroom as it is too big

Drinks, Drink size, Pizza, Pizza Size and Sides

Ordering food

```python
def size(): #Creates a function called size
    global whichPizza
    size={"1":4,"2":5, "3":10,"4":15,"5":20} #Creates a dictonary for calculating price
    wordsize_dic={"1":"Extra Small","2":"Small", "3":"Medium","4":"Large","5":"Extra Large"}#Creates a dictonary for representing the inputs as a word
    print("\n     What size do you want for", order[whichPizza],"?" ) #assks for users input
    print("     [1] Extra Small 4$")
    print("     [2] Small 5$")
    print("     [3] Medium 10$")
    print("     [4] Large 15$")
    print("     [5] Extra Large 20$")
    whichPizza=whichPizza+1 #Increases the varible so that next time a pizza is order it goes through to the next one
    while True: #While true basically goes forever
        size1=input() #Gets users input
        if size1 in size: # if the input is in the dictonary (is a valid input the following indented will occur)
            sizeList.append(size[size1]) #Appends to the size list which is the price list for pizza
            wordsize.append(wordsize_dic[size1]) #Appends to the word list which is the word list for pizza
            menu() #sends the user back to the main menu
        else: #if the above does not occur
            print("     Not a size") #prints the following asking the user to try again
            print("     What size do you want for", order[whichPizza-1],"?"
)
            print("     [1] Extra Small 4$")
            print("     [2] Small 5$")
            print("     [3] Medium 10$")
            print("     [4] Large 15$")
            print("     [5] Extra Large 20$")
def drinksize(): #Creates a function called size
    global whichdrink
    size_price_drink={"1":1,"2":2, "3":4}  #Creates a dictonary for
calculating price
    wordsize_dic={"1":"Small", "2":"Medium","3":"Large"} #Creates a
dictonary for representing the inputs as a word
    print("\n     What size do you want for", drinkorder[whichdrink],"? this
will add on to the drinnks normal price" ) #assks for users input
    print("     [1] Small 1$")
    print("     [2] Medium 2$")
    print("     [3] Large 4$")
    whichdrink=whichdrink+1 #Increases the varible so that next time a pizza
is order it goes through to the next one
    while True: #While loop to append values when given them if invalid asks
again
        drinksizee=input()
        if drinksizee in size_price_drink:
            drinksizelist.append(size_price_drink[drinksizee])
            drinkwordsize.append(wordsize_dic[drinksizee])
            menu() #sends the user back to the main menu
        else: #if the above does not occur
            print("     Not a size") #prints the following asking the user
to try again
            print("     What size do you want for",
drinkorder[whichdrink-1],"?" )
            print("     [1] Small 1$")
            print("     [2] Medium 2$")
            print("     [3] Large 4$")
def drinks(): #defines a function called drinks which is for ordering drinks
    global drinkorder #Globals drink order for er which is the word list for
drinks not sure why i need this but i dont have time to check
    global check #Globals check so that checkout can see and eedit if there
is an order
    coke_dict={"1":"Coca-Cola Classic", "2":"Coca-Cola light","3":"Coca-Cola
Zero","4":"Coca-Cola Vanilla"}#Creates a dictonary for representing the
inputs as a word
    print("     Choose a drink")
#Asks the user to choose between drinks
    print("     [1] Coca-Cola Classic")
    print("     [2] Coca-Cola light")
    print("     [3] Coca-Cola Zero")
    print("     [4] Coca-Cola Vanilla")
    while True: #While true repeats forever which appends values and if
invalid asks for input again
        drinks=input("")
        if drinks in coke_dict: #if the input is in the dictonary (is a
valid input the following indented will occur)
            drinkorder.append(coke_dict[drinks])
            check=check+2
            drinksize() #Sends them back to menu
        elif drinks =="Pepsei": #If the input is Pepsi with that exact
spelling the following will occur a little easter egg for what your said in
class sir
            print("\n     Dont drink Pepsei science has proven that it kills
your teeth")#PArts of easter egg
            print("     Buy our Coke instead they are partnered with us and
give discounts and if your lucky you may win a Prize") #PArts of easter egg
note there is no price lol
        else: #if none of the above occur it is an invalid input and the
following occurs
            print("     Not an option look from the below Choose a drink")
            print("     [1] Coca-Cola Classic")
            print("     [2] Coca-Cola light")
            print("     [3] Coca-Cola Zero")
            print("     [4] Coca-Cola Vanilla")
def pizza(): #Defines pizza for ordering pizza
    global order #globals order for the end
    global check #Globals check so that checkout can see and eedit if there
is an order
    pizza_dict={"1":"Pepperoni",
"2":"Meatlovers","3":"Cheese","4":"Chicken","5":"Margherita"}#Creates a
dictonary for representing the inputs as a word
    print("\n     Choose a pizza the prices change depending on size of
price")     #Asks the user a question
    print("     [1] Pepperoni")
    print("     [2] Meatlovers")
    print("     [3] Cheese")
    print("     [4] Chicken")
    print("     [5] Margherita")
    while True: #While true this loops forever
        pizza=input() #Gets users input
        if pizza in pizza_dict:
# if the pizza is in the options the following thing indented will occur
            order.append(pizza_dict[pizza]) #Adds to order
            check=check+2 #Allows user to go to checkout
            size() #Gets the size of pizza explained above by calling the
function
        else: #If it is not in the dictonary the response is invalid so the
following occurs
```

```
            print("\n     Not a size")
#Informs user of their error
            print("     Choose a pizza the prices change depending on size
of price")
            print("     [1] Pepperoni")
            print("     [2] Meatlovers")
            print("     [3] Cheese")
            print("     [4] Chicken")
            print("     [5] Margherita")
def sides():
    global sideorder #Globals drink order for er which is the word list for
drinks not sure why i need this but i dont have time to check
    global sideprice #Globals drink price which is the price list for drinks
not sure why i need this but i dont have time to check
    global check #Globals check so that checkout can see and eedit if there
is an order
    Sides_dict={"1":"Garlic bread", "2":"Hot chips
classic","3":"Salad","4":"6pc wings","5":"3pc wings","6":"4-Piece Molto Bene
Pizzeria McNuggets"} #Creates a dictonary for representing the inputs as a
word
    Sidesprice_dic={"1":"2","2":"3","3":"5","4":"6","5":"4","6":"8"}
#Creates a dictonary for calculating price
    print("     Choose a side")
    print("     [1] Garlic bread 2$")
    print("     [2] Hot chips classic 3$")
    print("     [3] Salad 5$")
    print("     [4] 6pc wings 6$")
    print("     [5] 3pc wings 4$")
    print("     [6] 4-Piece Molto Bene Pizzeria McNuggets 8$")
    while True: #While true repeats forever which appends the side user
wants if invalid asks again for input
        side=input("")
        if side in Sides_dict: #if the input is in the dictonary (is a valid
input the following indented will occur)
            sideorder.append( Sides_dict[side])
            sideprice.append(Sidesprice_dic[side])
            check=check+2
#Allows user to go to checkout
            menu()
        elif side =="Pepsei": #If the input is Pepsi with that exact
spelling the following will occur a little easter egg for what your said in
class sir
            print("\n     Dont drink Pepsei science has proven that it kills
your teeth")#PArts of easter egg
            print("     Buy our Coke instead they are partnered with us and
give discounts and if your lucky you may win a Prize") #PArts of easter egg
note there is no prize lol
```

```
        else: #if none of the above occur it is an invalid input and the
following occurs
            print("     Not an option look from the below Choose a side")
            print("     [1] Garlic bread 2$")
            print("     [2] Hot chips classic 3$")
            print("     [3] Salad 5$")
            print("     [4] 6pc wings 6$")
            print("     [5] 3pc wings 4$")
            print("     [6] 4-Piece Molto Bene Pizzeria McNuggets 8$")
```

The above code is all of the different food options that the user can order. Each type of option has its function but they all operate on the same skeleton. This is creating dictionaries for calculating prices and storing the orders as words to show the user. Then it asks for input and if the input is not in the dictionary then the customer is told it's an invalid option and is asked to order again. As well as checking if the input is in the dictionary an else statement is made for the invalid input. This skeleton is used for all of the functions above just that it appends to different lists. An alternative method could have used a variety of if and else statements which would make the code slightly easier to read but way too long.

The flowchart for this section is in the classroom as it is too big

The pigsize function (price with sizes and other)

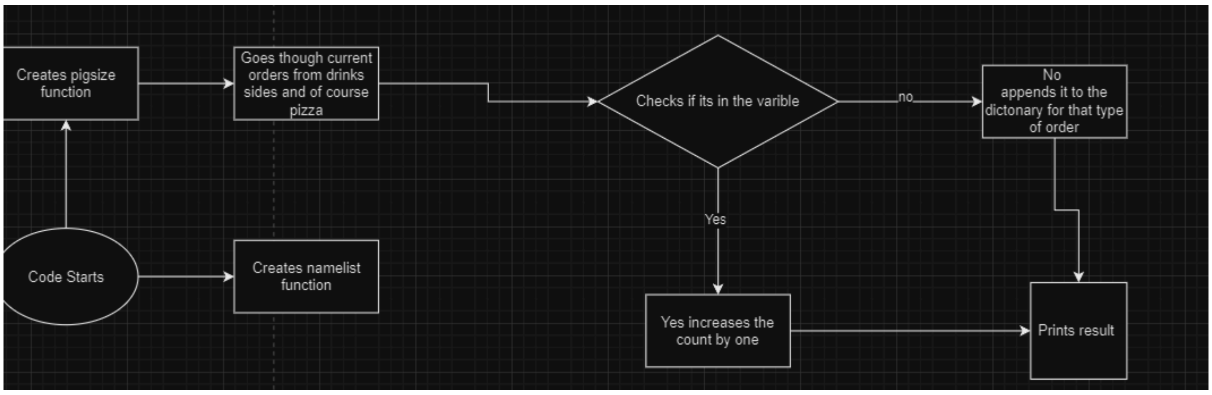
Printing the current order

```
Python
def pizsize(): #defubes a function called pizsize which combines and prints
everything
    Ranoutofidea={"Extra Small":4,"Small":5, "Medium":10,"Large":15,"Extra
Large":20}#This is for calculating price because my function was not working
correctly
    drinktoprice={"Small":1,"Medium":2, "Large":4}
    counts = {}
    counts2 = {}
    counts3={}
    for i in range(len(order)): #a for loop for all of the current pizza
orderif there are already pizzas it increases amaount if its the first its 1
        pair = (order[i], wordsize[i])
        if pair in counts: #adds if there is more than on of that
combination if not it will just be one
            counts[pair] += 1
        else:
            counts[pair] = 1
    for key, value in counts.items(): #creates a varible with key and value
for each part of counts using the item function to make a turtple with the
size pizza and amount
        pizza, size = key #pizza and then size is equal to each part in key
this splits the turple again
        print(f"{value} {pizza} Pizza {size}  $"+str(value
*int(Ranoutofidea[key[1]])))#prints the result of all the above

    for i in range(len(drinkorder)): #loops through how many drink orders
there are
        pair = (drinkorder[i],drinkwordsize[i])
        if pair in counts2:
            counts2[pair] += 1 #Add +1 same  as above count
        else:
            counts2[pair] = 1
    for key, value in counts2.items():
        drink, drinksize=key
        print(f"{value} {drink} {drinksize}   $"+str(value
*int(drinktoprice[key[1]]))) #prints it the {} are used so that i dont have
to do a bunch of ++ dont usally do it but looks nice here
    for i in range(len(sideorder)): #loops through drink order
        pair = (sideorder[i], sideprice[i])
        if pair in counts3:
            counts3[pair] += 1 #Add +1 same as above count
        else:
            counts3[pair] = 1
    for key, value in counts3.items():
        print(f"{value} {key[0]}    $"+str(value * int(key[1])))
```

The code above is a function that when called will print the current price of the order with spacing to make it easier for the customer to view. First, a dictionary if made is used to calculate the price of the pizza and another one for drinks. Then three empty dictionaries are made for the sides, pizza and drinks. These dictionaries will combine the price, size and order for the pizza. Then a for loop is made to go through each part of the order this is done for all three. It then combines each part of the order and increases the count of each part of the order if there is already one of that type in the dictionary.

An alternative solution to the above code could have been to make one dictionary to add all the values. This would have worked by combining the drinks and pizza but for sides, a variety of alters to the code would have to be made. Additionally, this would make it harder to implement other features as each part of the code would be more intertwined.


Creates pigsize function
Goes though current orders from drinks sides and of course pizza
Checks if its in the varible
no
No appends it to the dictonary for that type of order
Yes
Code Starts
Creates namelist function
Yes increases the count by one
Prints result

The checkout function

The checkout function is used in checkout and resets a variety of values of purchase

```python
global whichPizza
#Globals to reset price for next order
    global whichdrink
    global wordsize
    global sizeList
    global order
    global check
    global info
    global namelist
    global namelistremove
    global drinkorder
    global sideorder
    global sideprice
    Ranoutofidea={"Extra Small":4,"Small":5, "Medium":10,"Large":15,"Extra
Large":20} #same as in pigsize but this time is needed for appending to txt
file
    drinktoprice={"Small":1,"Medium":2, "Large":4}
    d=0
    if check >1:
        for k in sizeList:
            d=d+k
        for k in drinksizelist:
            d=d+int(k)
        for k in sideprice:
            d=d+int(k)
    print("Loading order...")
    time.sleep(1)
    print("\nOrder:")
    print("\nTotal Price $"+str(d))
    time.sleep(0.5)
    pizsize()
    print("\nAre you sure you want to Procceed to checkout? If not go back
to order from the menu") #Asks the user a question
    print("\n[1] Proceed ")
    print("[2] Back to menu")
    while True: #This loop confirms the order and then another confirmation
below for credit card
        z=(input())
        if z.isdigit():
            z=int(z)
            if z==1:
                break #This breaks the while true loop and countiues to the
next part of the function
            elif z ==2:
                menu() #Calls the start function and the code loops
            else: #if it is an interger but not one or two the indentded
occurs
                print("\n Thats not an option") #States this is not an
option
                print(" [1] Proceed ")
                print(" [2] Back to menu")
        elif z.isalpha(): #Uses the is alpha to check if the input is
invalid
            print("\nThats not an option") #States this is not an option
            print("[1] Proceed ")
            print("[2] Back to menu")
        else: #If anything else like an empty space occurs then the indented
will occur
            print("\nThats not an option")
            print("[1] Proceed ")
            print("[2] Back to menu")
    print("\nThanks for confirming to proceed!! Please enter in your credit
card note all credit cards have 10 digits")
    while True: #while true repeats until break which is used again here
            z=input("")
            if len(z)==10 and z.isdigit(): #Checks if the input is valid
with is digit and if a valid credit card
                print("Checking...")
                time.sleep(1)
                print("Thats a unique credit card! Thanks for shopping dont
forget to pickup your order")
                break
#Ends the while true loop
            elif z.isalpha(): #informs the user that the input is invalid
                print("Checking...")
                time.sleep(1)
                print("Thats not even a number! Note all credit cards must
have 10 digits/numbers")
            else:
                print("Checking...")
                time.sleep(1)
                print("Please enter a valid credit card it must have 10
digits")
    with open("WhatOrder"+name+".txt", "a") as file: #opens a file or
creates one to append the indented
        file.write("This is ")
        file.write(name)
        file.write("'s Order\n")
        file.write("\nTotal Price $"+str(d))
    with open("WhatOrder"+name+".txt","a") as file: #opens a file or creates
one to append the indented
        counts = {}
        counts2 = {}
        counts3={}
        for i in range(len(order)): #A loop combining pizza and size of it
increases the nymeber if there is more than one
            pair = (order[i], wordsize[i])
            if pair in counts:
                counts[pair] += 1
            else:
                counts[pair] = 1
        for key, value in counts.items():
            pizza, size = key
            file.write(f"\n{value} {pizza} Pizza {size}  $"+str(value
*int(Ranoutofidea[key[1]])))#Writes the results to the file
        for i in range(len(drinkorder)):
            pair = (drinkorder[i],drinkwordsize[i])
            if pair in counts2:
                counts2[pair] += 1 #Add +1 san as above count
            else:
                counts2[pair] = 1
        for key, value in counts2.items():
            drink, drinksize=key
            file.write(f"\n{value} {drink} {drinksize}   $"+str(value
*int(drinktoprice[key[1]])))#prints it the {} are used so that i dont have
to do a bunch of ++ dont usally do it but looks nice here
        for i in range(len(sideorder)):
            pair = (sideorder[i], sideprice[i])
            if pair in counts3:
                counts3[pair] += 1
#Add +1 san as above count
            else:
                counts3[pair] = 1
        for key, value in counts3.items():
            file.write(f"\n{value} {key[0]}    $"+str(value * int(key[1])))
    with open("WhatOrder"+name+".txt","a") as file: #opens file again last
time
```

```
            file.write("\nThank you for choosing Molto Bene Pizzeria ")
#Classic end of reciet moment
    wordsize=[] #Resets value for the next order
    sizeList=[]
    order=[]
    check=1
    info=1
    drinkorder=[]
    whichPizza=0
    whichdrink=0
    sideorder=[]
    sideprice=[]
    namelistremove=namelistremove+1
    Start()
#Goes to next order
print("\nWelcome to Molto Bene Pizzeria we sell many pizzas and drinks")
#This is the actual start of what the user sees welcomes them to pizza place
Start()
```

The final step for the customer is confirming the order and showing an overview as per assignment requirements and some polish like getting a credit card before printing the text file of the order. Confirming the order is done similarly to the double-checking function using the alpha and digit functions to check the values.

After the order is confirmed they are asked for a 10-digit credit card number. If invalid they are valid it will write the order to a txt file under the username. This is done using the file. write function along with the pig size functions format and some extra writing lines to make the receipt easier to view. Then the values are reset for the next order. T

The checkout is done like this as it is a simple process of implementing already used code. Though a variety of parts in the code could be written as functions implementing already existing code makes it simpler to put in one function instead. Alternatively, different functions could have been made and all put into the checkout but this works just as well.

This is also too large so I will put it on the flow chart attached to the classroom.

Evaluation of the code

The above code has made a pizza ordering system that gets the user name/order and a combination of pizza sizes and flavours. When the order is placed and during the ordering process it shows the user the current order. Then allows the user to pick up the pizza from the display of current orders removing it from current orders. Additionally, they were given different drink options, drink sizes, and sides to purchase as well as a summary of orders with the price of each product shown. There is also an easter egg put into the code to see if you can find it. Finally, it allows for there to be more than one order or another order under a different name. If this pizza ordering system were to be implemented at a real pizza shop a few adjustments would be made. This includes printing the txt files as a receipt, changing some of the prices and adding different drinks and some optional desserts as well as solving a price bug that occasionally occurs. In conclusion, I am happy with my work on this assignment as well as my planning through spreading out the workload would have been wiser.