Gusakova Alona GL.02

Projekt z Języków Symbolicznych

Temat Projektu

Automat sprzedający napoje. W tym projekcie zaprogramowałam automat z napojami tak jak działa w życiu. Nabywca, podchodząc do automatu, widzi listę napojów i ich numery, wybiera numer, wpisuje wybrany numer za pomocą przycisków, a następnie może wprowadzić kwotę dla tego napoju.

Automat przechowuje informacje o monetach znajdujących się w nim (1, 2, 5, 10, 20, 50gr, 1, 2, 5zł) oraz informacje o towarach znajdujących się w nim (przedmioty o numerach od 30 do 50), każdy o określonej cenie w określonej liczbie. Okno z przyciskami pozwala na wrzucanie monet, potem wyświetla kwotę wrzuconych monet, przyciskiem przerywa transakcję (wyrzuca wrzucone monety), przyciskami 0-9 pozwala wpisać numer wybranego towaru oraz polem wyświetla wpisany numer towaru.

Funkcjonalność

1. Są przyciski od 0 do 9 które umożliwiają wprowadzenie numeru wybranego towaru.
2. Są okienka, które pokazują numer wybranego produktu przy wpisywaniu numeru, cenę, kwotę jaką wpisujemy oraz okienko informacyjne o tym czy produkt został zakupiony lub czy należy wydać resztę.
3. Znajduje się przycisk „Add money” po naciśnięciu którego można wprowadzić pewną kwotę pieniędzy.
4. Jest przycisk "Deny transaction", po naciśnięciu którego wszystko zostaje anulowane, wracamy do pozycji wyjściowej (wpisz numer towaru) i zwracamy pieniądze, jeśli zostały wprowadzone.
5. Jest przycisk "Buy", po naciśnięciu którego kupujemy produkt.
6. Znajduje się przycisk „Give change” po naciśnięciu którego, jeśli wprowadzono kwotę większą niż cena towaru, zwracamy zmianę.

Klasy i samodzielne funkcje zawarte w projekcie

Funkcja isExistDrink(index)

```
def isExistDrink(index):
    readFile()
    global allGoods
    if index in allGoods:
        countAndPriceForOneItem = allGoods[index]
        if countAndPriceForOneItem['count'] <= 0:
            noDrink['text'] = "Lack of product"
        else:
            noDrink['text'] = "Price: " + str(int(countAndPriceForOneItem["price"])/100)
    elif index not in allGoods:
        noDrink['text'] = "Selection error"
```

Za pomocą tej funkcji możemy sprawdzić, czy mamy wprowadzony przez użytkownika numer towaru. Mamy trzy opcje na to, co może się wydarzyć:

numer wprowadzony przez użytkownika nie odpowiada zakresowi od 30 do 50 a następnie wyświetla się ostrzeżenie "Selection error" co oznacza Błąd wyboru,

gdy wprowadzony produkt istnieje, ale produkt się skończył,

lub gdy wprowadzony produkt istnieje i jest dostępny, wówczas pokazujemy jego cenę.

Funkcja showMoney()

```
def showMoney():
    global allGoods
    global sum
    button1coint.place(relx=0.5, rely=0.37)
    button2coint.place(relx=0.58, rely=0.37)
    button5coint.place(relx=0.66, rely=0.37)
    button10coint.place(relx=0.74, rely=0.37)
    button20coint.place(relx=0.82, rely=0.37)
    button50coint.place(relx=0.9, rely=0.37)
    button1papermoney.place(relx=0.6, rely=0.43)
    button2papermoney.place(relx=0.7, rely=0.43)
    button5papermoney.place(relx=0.8, rely=0.43)
    addedMoney.place(relx=0.65, rely=0.25)
```

Korzystając z tej funkcji, naciskając przycisk „Add money" zobaczymy złotę i groszę, które możemy wprowadzić.

Funkcja addSum()

```
def addSum(n):
    global sum
    global allGoods
    global listAllMoneyFromPerson
    listAllMoneyFromPerson.append(n)
    sum = sum + n
    addedMoney['text']=str(sum/100)
    if len(stuffNumberOfDrinks['text']) == 2:
        index = stuffNumberOfDrinks['text']
        priceAndCount = allGoods[index]
        if int(priceAndCount["price"])< sum and not checkIfChangeCanBeGiven():
            changeLable['text']="There is no change, buy-sell the goods and won't give back the change"
```

Dzięki tej funkcji pokazujemy kwotę jaką wprowadzi użytkownik oraz sprawdzamy za pomocą funkcji checkIfChangeCanBeGiven(), czy automat może wydać resztę kupującemu, jeśli wprowadzi kwotę większą od podanej ceny. Również gdy użytkownik wybrał towar, wpisał jego numer ale automat nie ma reszty to klient będzie poinformowany „There is no change, buy-sell the goods and won't give back the change" – co oznacza że użytkownik ma wybór, lub kupić wybrany towar ale nie dostać się reszty, lub anulować transakcję i zabrać swoje pieniądze.

Funkcja denyTransaction()

```
def denyTransaction ():
    global sum
    global numberDrink
    if len(stuffNumberOfDrinks['text']) == 0:
        sum = 0
    addedMoney['text']= "0"
    numberDrink = ""
    stuffNumberOfDrinks['text']=""
    noDrink['text'] = ""
    if sum > 0:
        changeLable['text'] = "Return " + str(sum/100)
        sum = 0
    else:
        changeLable['text'] = ""
```

Ta funkcja jest powiązana z przyciskiem „Deny transaction". Przy naciśnięciu tego klawisza, tak jak było to wcześniej napisane, wszystko będzie anulowane i jeśli dana osoba wprowadziła pieniądze, to będą one zwrócone.

Funkcja giveChange()

```
def giveChange():
    global sum
    addedMoney['text'] = 0
    index = stuffNumberOfDrinks['text']
    print(index)
    print(sum)
    if index == "" and sum == 0:
        changeLable['text'] = "Choose a product"
    else:
        changeLable['text'] = "Change is given"
    sum = 0
```

Ta funkcja sprawdza, czy użytkownik wpisał numer towaru. Jeśli użytkownik nie wybrał produktu, wyświetlana jest mu informacja „Choose a product" aby klient wybrał produkt.

Funkcja stuffNumber()

```
def stuffNumber(n):
    global numberDrink
    changeLable['text'] = ""
    if len(stuffNumberOfDrinks['text']) == 2:
        numberDrink = ''
    numberDrink = numberDrink + str(n)
    stuffNumberOfDrinks['text']= numberDrink
    if len(stuffNumberOfDrinks['text']) == 2:
        isExistDrink(stuffNumberOfDrinks['text'])
```

Ta funkcja pokazuje wprowadzony przez użytkownika numer towaru i jeżeli wporowadzi więcej niż dwie liczby, to numer zaczyna się wpisywać odnowa oraz dzięki funkcji isExistDrink() sprawdzamy czy napój istnieje pod takim numerem.

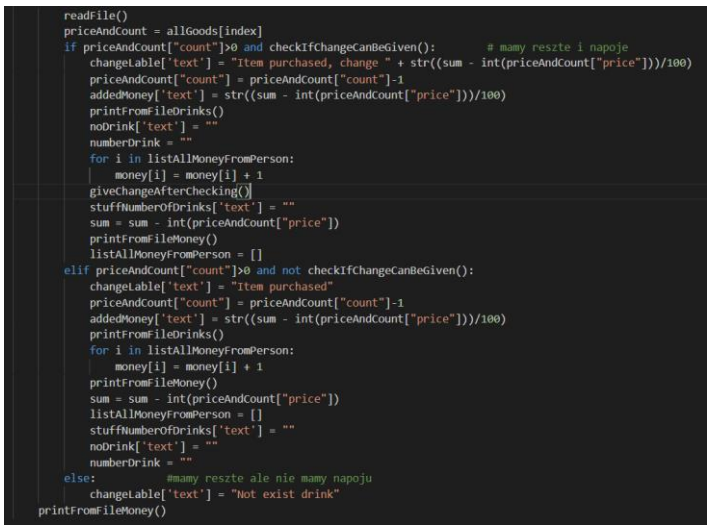
Funkcja readFile()

```
def readFile ():
    fileName = "drinks.txt"
    global allGoods
    allGoods = {}
    with open(fileName) as fileObject:
        for line in fileObject:
            splitedLine =  line.split()
            allPrices = {}
            allPrices["price"] = int(splitedLine[1])
            allPrices["count"] = int(splitedLine[2])
            allGoods[splitedLine[0]] = allPrices
    # print(allGoods)
```

Ta funkcja odczytuje plik „drinks.txt” w którym znajdują się numery towarów, cena oraz ilość każdego produktu.

Funkcja verificationTransactionFunction()

```
def verificationTransactionFunction():
    global sum
    global numberDrink
    global allGoods
    global listAllMoneyFromPerson
    global money
    index = stuffNumberOfDrinks['text']
    if index == "":
        changeLable['text'] = "Choose a product"
    priceAndCount = allGoods[index]
    if int(priceAndCount["price"]) > sum:
        changeLable['text'] = "Not enought money"
    elif int(priceAndCount["price"]) == sum and priceAndCount["count"]>0:
        changeLable['text'] = "Item purchased"
        priceAndCount = allGoods[index]
        priceAndCount["count"] = priceAndCount["count"]-1
        addedMoney['text'] = 0
        sum = 0
        for i in listAllMoneyFromPerson:
            money[i] = money[i] + 1
        printFromFileMoney()
        stuffNumberOfDrinks['text'] = ""
        noDrink['text'] = ""
        numberDrink = ""
        printFromFileDrinks()
        readFile()
    else:     # mamy reszte
```



```
    readFile()
    priceAndCount = allGoods[index]
    if priceAndCount["count"]>0 and checkIfChangeCanBeGiven():        # mamy reszte i napoje
        changeLable['text'] = "Item purchased, change " + str((sum - int(priceAndCount["price"]))/100)
        priceAndCount["count"] = priceAndCount["count"]-1
        addedMoney['text'] = str((sum - int(priceAndCount["price"]))/100)
        printFromFileDrinks()
        noDrink['text'] = ""
        numberDrink = ""
        for i in listAllMoneyFromPerson:
            money[i] = money[i] + 1
        giveChangeAfterChecking()
        stuffNumberOfDrinks['text'] = ""
        sum = sum - int(priceAndCount["price"])
        printFromFileMoney()
        listAllMoneyFromPerson = []
    elif priceAndCount["count"]>0 and not checkIfChangeCanBeGiven():
        changeLable['text'] = "Item purchased"
        priceAndCount["count"] = priceAndCount["count"]-1
        addedMoney['text'] = str((sum - int(priceAndCount["price"]))/100)
        printFromFileDrinks()
        for i in listAllMoneyFromPerson:
            money[i] = money[i] + 1
        printFromFileMoney()
        sum = sum - int(priceAndCount["price"])
        listAllMoneyFromPerson = []
        stuffNumberOfDrinks['text'] = ""
        noDrink['text'] = ""
        numberDrink = ""
    else:           #mamy reszte ale nie mamy napoju
        changeLable['text'] = "Not exist drink"
printFromFileMoney()
```

Ta funkcja jest największa ale dzięki nią cały ten program działa poprawnie. Ona sprawdza czy użytkownik wpisał numer wybranego towaru oraz czy kwotę którą użytkownik wprowadził, wystarczy aby kupić wybrany produkt po kliknięciu na przycisk „Buy”.

Mamy kilka sytuacji do sprawdzenia:

gdy użytkownik wybrał poprawny numer towaru, zapłacił bez reszty i wybrany produkt jest dostępny w naszym wypadku ilość towaru jest więcej niż 0 w pliku „drinks.txt”. Wtedy wyskakuje informacja „Item purchased” co oznacza że towar jest kupiony, dodajemy do pliku „money.txt” pieniądzę które użytkownik wykorzystał oraz usuwamy z pliku „drinks.txt” odejmujemy 1 od ilości wybranego produktu.

gdy użytkownik wybrał poprawny numer towaru, zapłacił więcej niż kosztuje towar i wybrany produkt jest dostępny w naszym wypadku ilość towaru jest więcej niż 0 w pliku „drinks.txt”.

- Możemy wydać resztę priceAndCount["count"]>0 and checkIfChangeCanBeGiven(): w tym przypadku wyskakuje informacja "Item purchased, change " + reszta i klikając na przycisk „Give change” co oznacza Oddaj resztę, automat oddaje towar oraz resztę, lub użytkownik może dalej kontynuować zakupy i coś kupić na swoją resztę. Usuwamy z pliku „drinks.txt” odejmujemy 1 od ilości wybranego produktu.
- Nie możemy wydać resztę priceAndCount["count"]>0 and checkIfChangeCanBeGiven()ale klient mimo ostrzeżenia nadal kupuje produkt: w tym przypadku wyskakuje informacja "Item purchased ", automat oddaje towar i użytkownik może dalej kontynuować zakupy na swoją resztę lub dodać pieniądze. Usuwamy z pliku „drinks.txt” odejmujemy 1 od ilości wybranego produktu.

gdy użytkownik wybrał poprawny numer towaru, i mamy resztę ale wybrany produkt nie jest dostępny w naszym wypadku ilość towaru jest równa 0 w pliku „drinks.txt”. Wtedy wyskakuje informacja "Not exist drink" jeżeli użytkownik spróbuje go kupić.

Funkcja readFromFileHowManyMoney()

```
def readFromFileHowManyMoney():
    global money
    money = {}
    with open ("money.txt") as fileObject:
        for line in fileObject:
            splitedLine =  line.split()
            money[int(splitedLine[0])] = int(splitedLine[1])
    print(money)
```

Ta funkcja odczytuje plik „money.txt” w którym znajdują się pieniądze oraz ich ilość.

Funkcja printFromFileMoney()

```
def printFromFileMoney():
    global money
    with open("money.txt", 'w') as fileObject:
                for key in money:
                    emptyLine = ""
                    emptyLine = emptyLine + str(key) + " " + str(money[key]) + "\n"
                    fileObject.write(emptyLine)
```

Ta funkcja po każdej transakcji zapisuje ponownie plik „money.txt” w którym znajdują się pieniądze oraz ich ilość.

Funkcja printFromFileDrinks()

```
def printFromFileDrinks():
    global allGoods
    with open("drinks.txt", 'w') as fileObject:
                for key in allGoods:
                    emptyLine = ""
                    emptyLine = emptyLine + str(key) + " "
                    priceAndCount = allGoods[key]
                    emptyLine = emptyLine + str(priceAndCount["price"]) + " "
                    emptyLine = emptyLine + str(priceAndCount["count"]) + "\n"
                    fileObject.write(emptyLine)
```

Ta funkcja po każdej transakcji zapisuje ponownie plik „drinks.txt” w którym znajdują się numery drinków, cena oraz ich ilość.

Funkcja checkIfChangeCanBeGiven()

```
def checkIfChangeCanBeGiven():
    global sum
    global money
    global listAllMoneyFromPerson
    index = stuffNumberOfDrinks['text']
    priceAndCount = allGoods[index]
    countOfChange = sum - priceAndCount["price"]
    money_tmp = copy.copy(money)
    for i in listAllMoneyFromPerson:
        money_tmp[i] += 1
    billsAndCoints = list(money_tmp.keys())
    billsAndCoints.sort(reverse=True)
    for i in billsAndCoints:
        minimal = min(countOfChange//i, money_tmp[i])   # 425//500, 80  minimal 2 = 425//200, 80
        money_tmp[i] = money_tmp[i] - minimal           # 80 - 0        80 - 2 = 78
        countOfChange = countOfChange - minimal*i       # 425 - 0       425 - 2*200 = 25
    if countOfChange>0:
        return False
    else:
        return True
```

Dzięki tej funkcji my sprawdzamy czy nasz automat może wydać resztę.

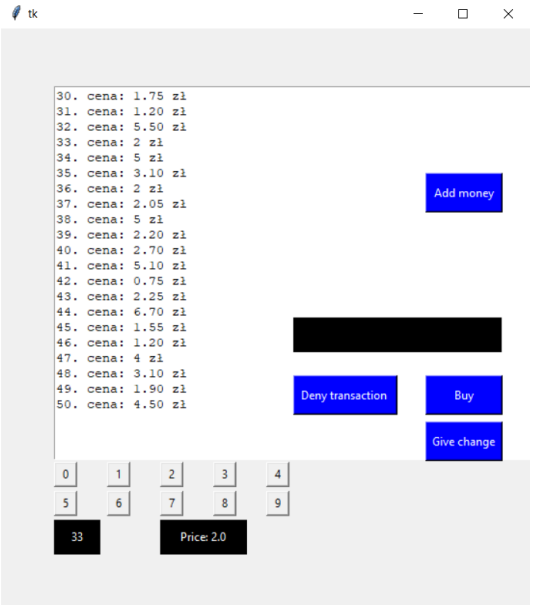
Funkcja giveChangeAfterChecking()

```
def giveChangeAfterChecking():
    global sum
    global money
    index = stuffNumberOfDrinks['text']
    priceAndCount = allGoods[index]
    countOfChange = sum - priceAndCount["price"]
    billsAndCoints = list(money.keys())
    billsAndCoints.sort(reverse=True)
    for i in billsAndCoints:
        minimal = min(countOfChange//i, money[i])   # 425//500, 80  minimal 2 = 425//200, 80
        money[i] = money[i] - minimal               # 80 - 0        80 - 2 = 78
        countOfChange = countOfChange - minimal*i   # 425 - 0       425 - 2*200 = 25
```

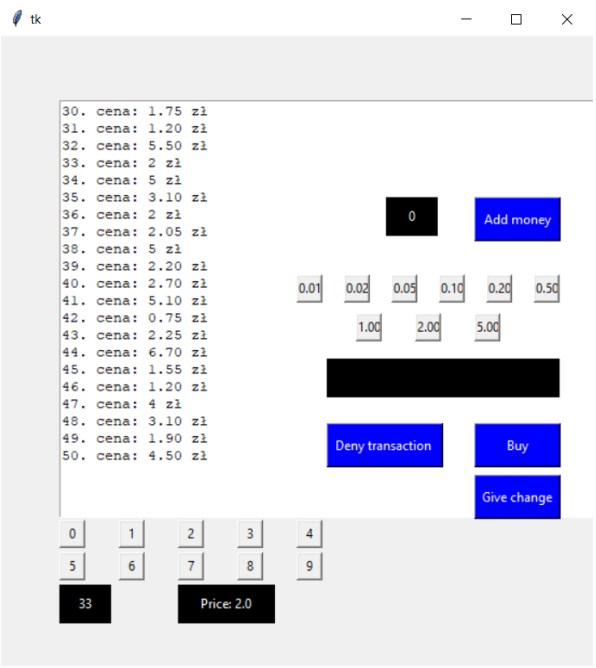
Po sprawdzeniu czy możemy wydać resztę dzięki funkcji checkIfChangeCanBeGiven(), my rachujemy jaką kupiurą możemy oddać resztę.

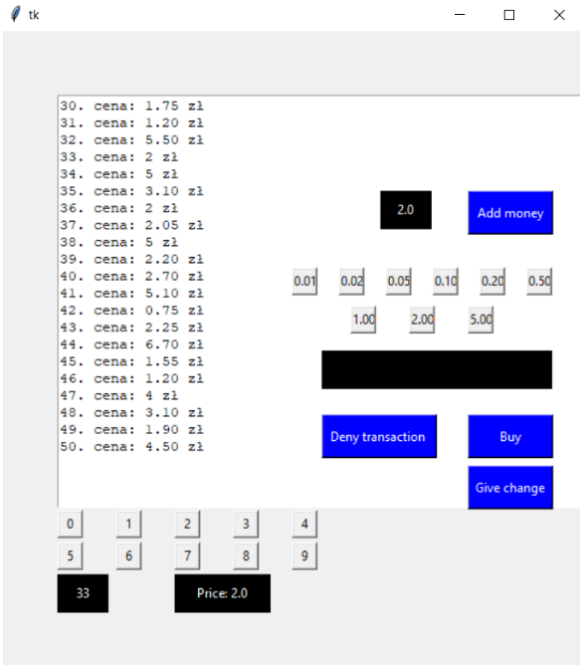
Testy

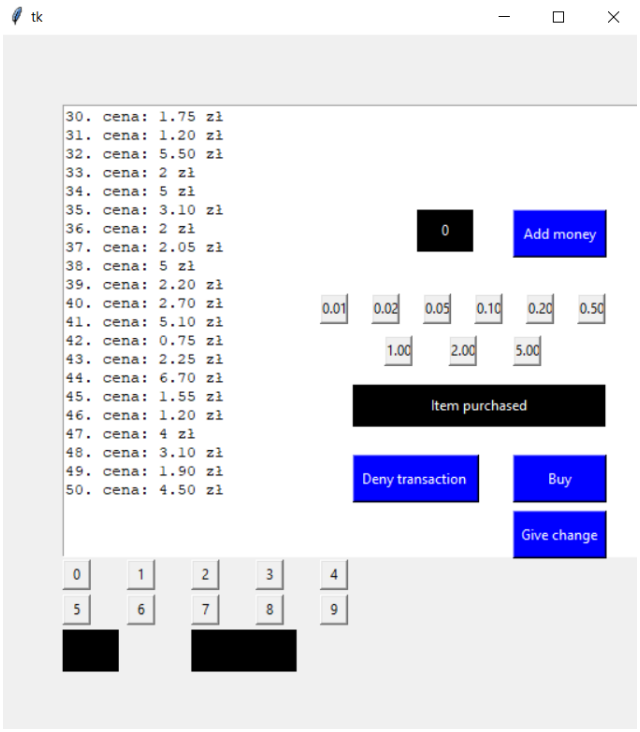
1. Sprawdzenie ceny jednego towaru - oczekiwana informacja o cenie.



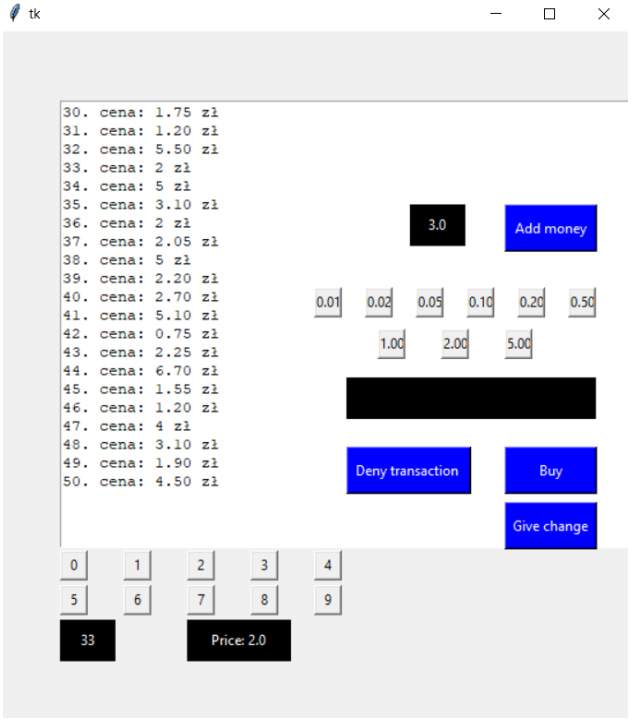
2. Wrzucenie odliczonej kwoty, zakup towaru - oczekiwany brak reszty.

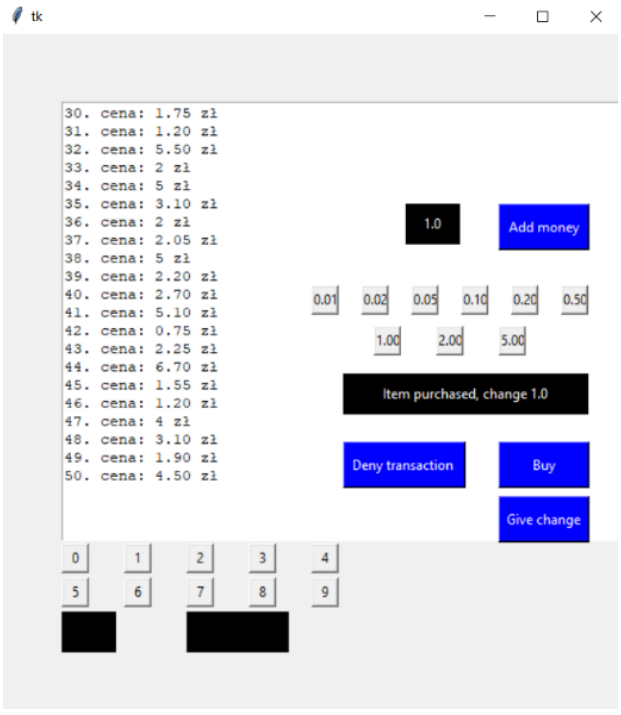


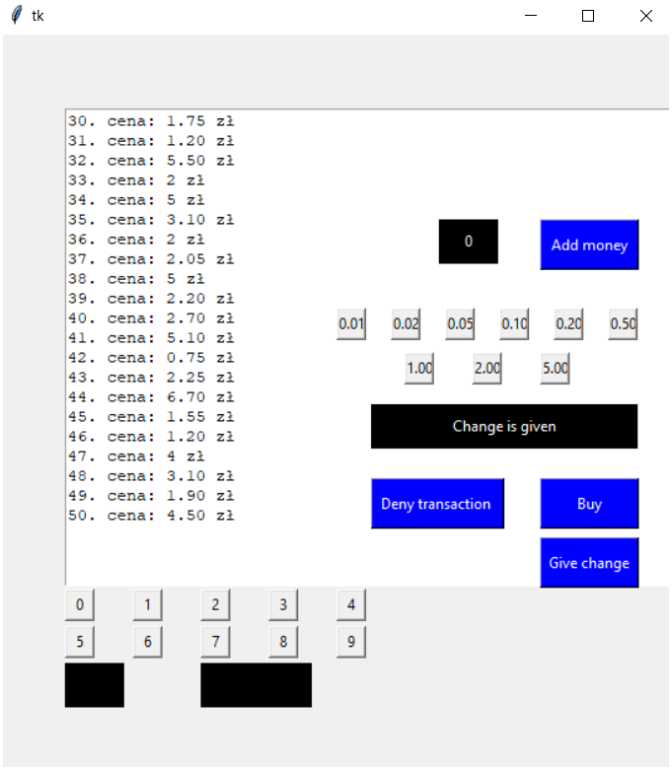




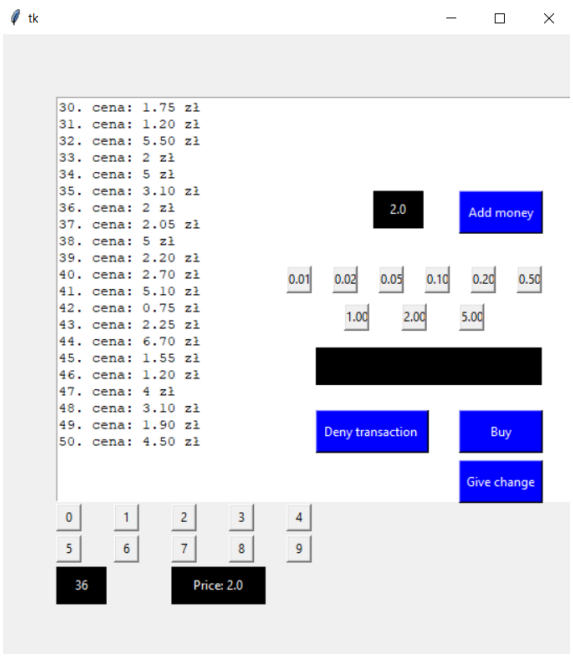
3. Wrzucenie większej kwoty, zakup towaru - oczekiwana reszta.

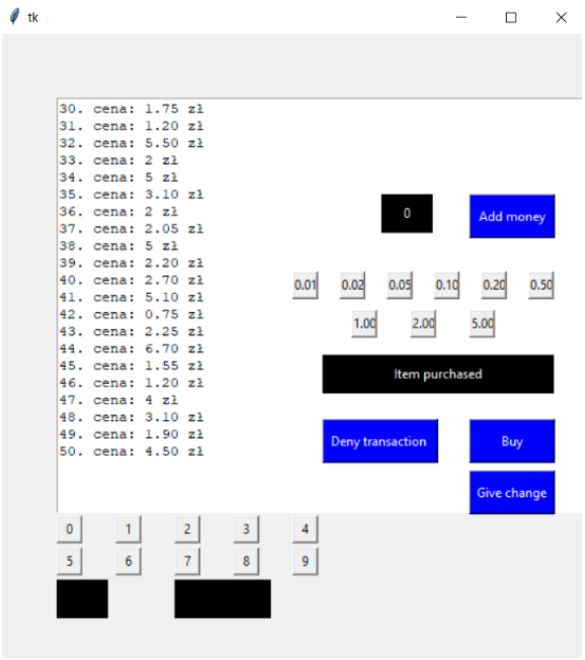


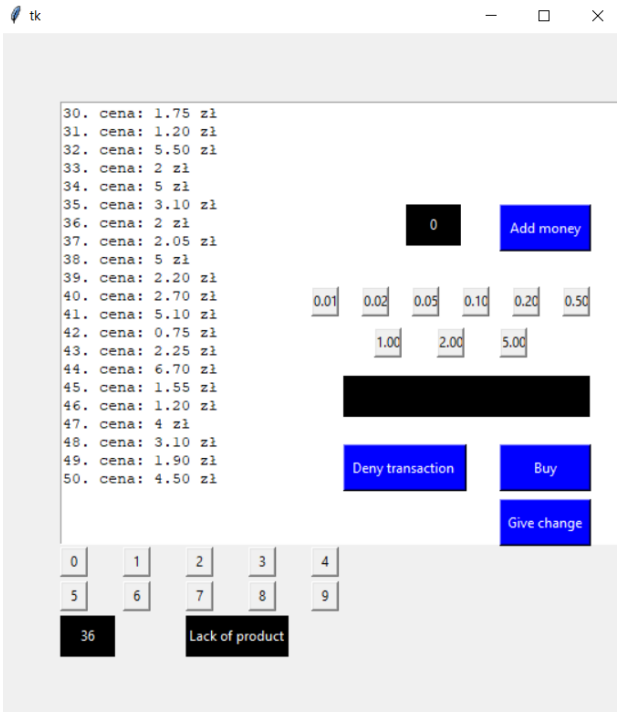




4. Wykupienie całego asortymentu, próba zakupu po wyczerpaniu towaru - oczekiwana informacja o braku.

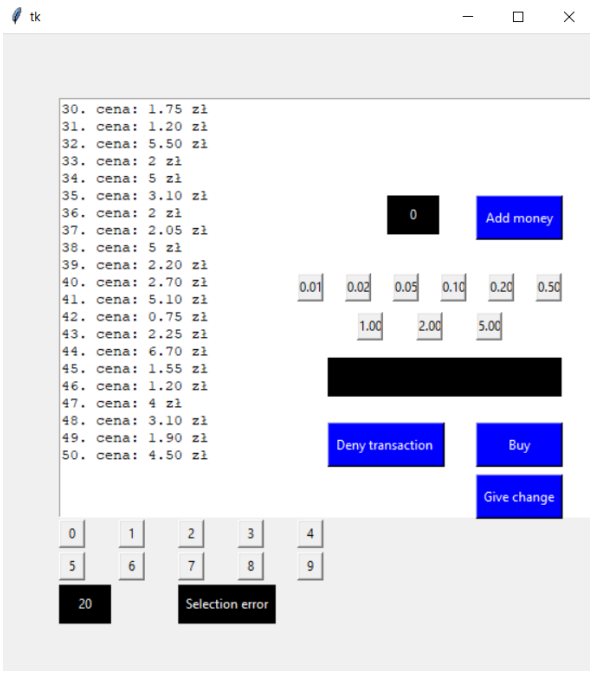


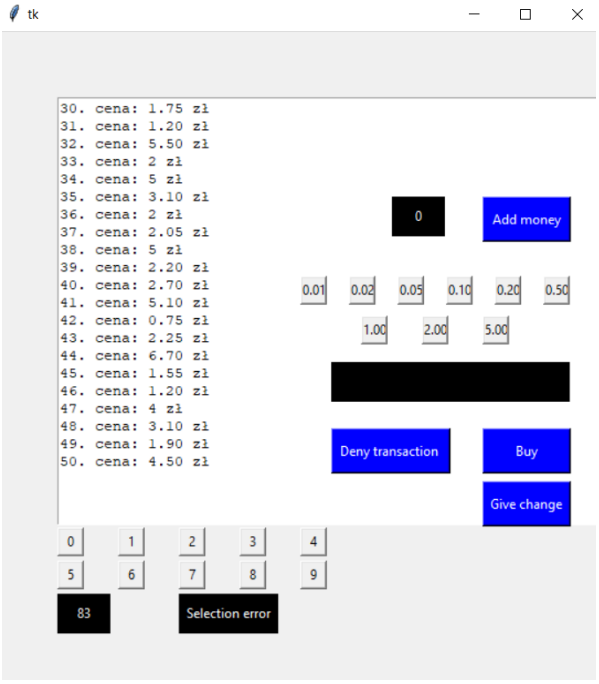




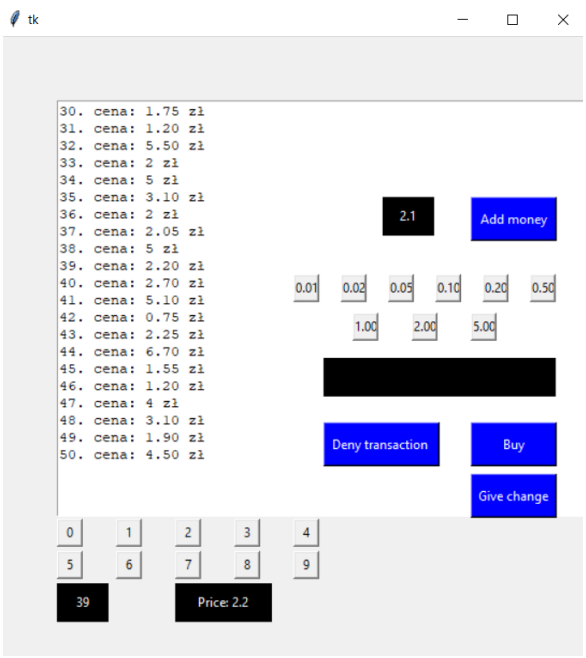
Lack of produkt - brak produktu

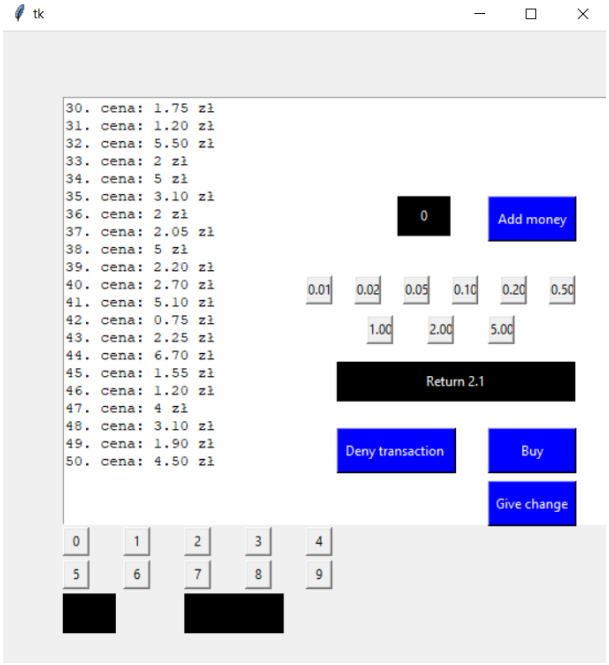
5. Sprawdzenie ceny towaru o nieprawidłowym numerze (<30 lub >50) - oczekiwana informacja o błędzie.



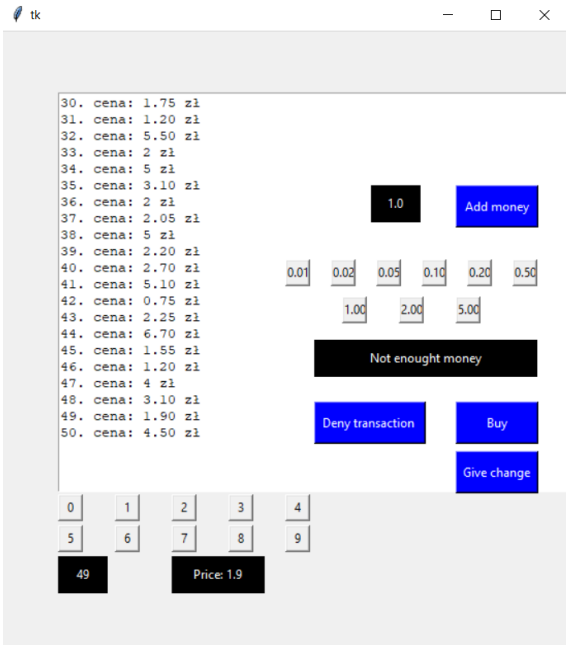


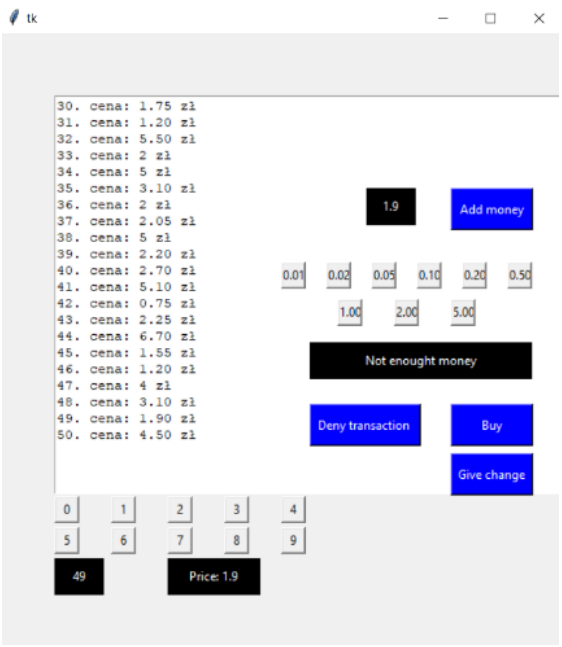
6. Wrzucenie kilku monet, przerwanie transakcji - oczekiwany zwrot monet.

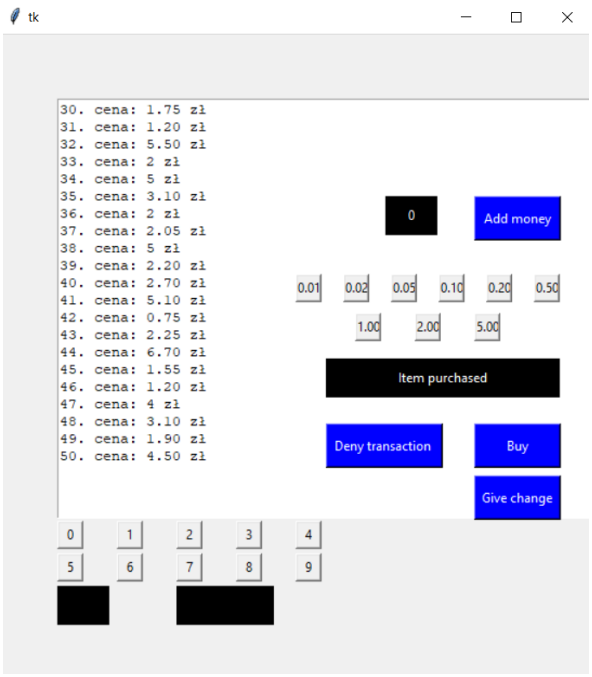




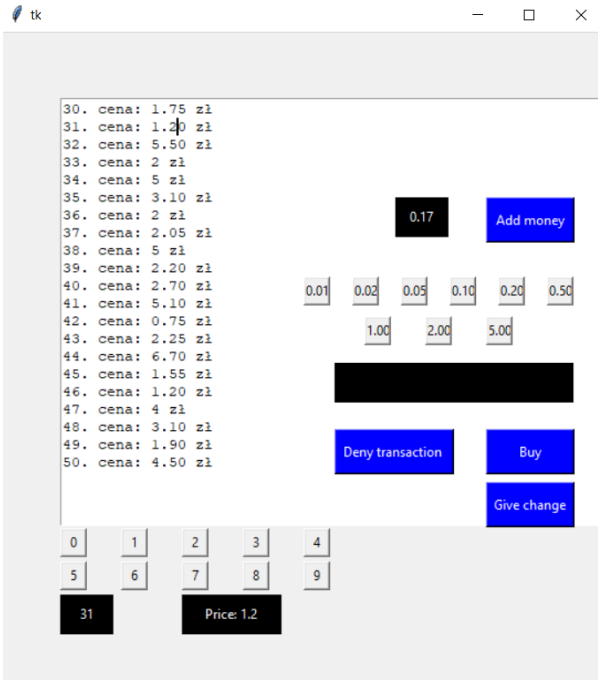
7. Wrzucenie za małej kwoty, wybranie poprawnego numeru towaru, wrzucenie reszty monet do odliczonej kwoty, ponowne wybranie poprawnego numeru towaru - oczekiwany brak reszty.

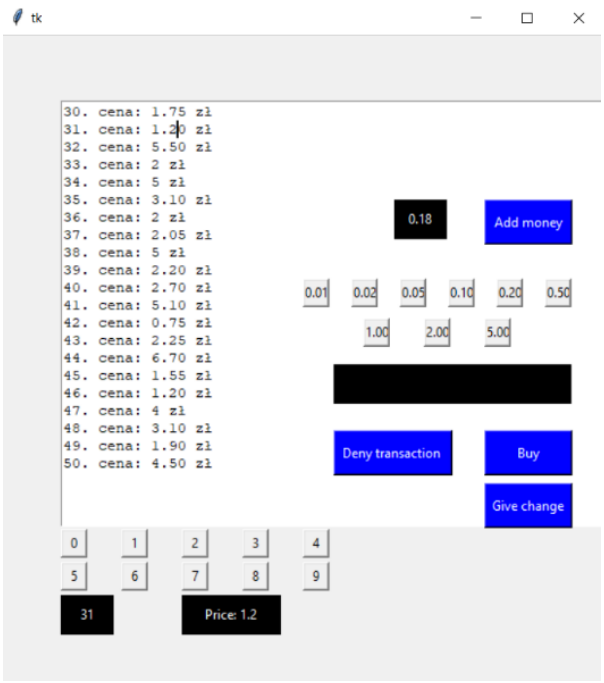


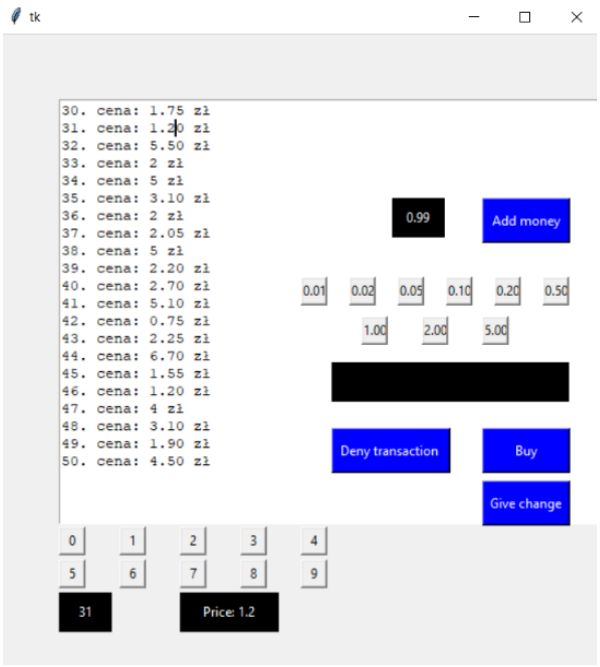


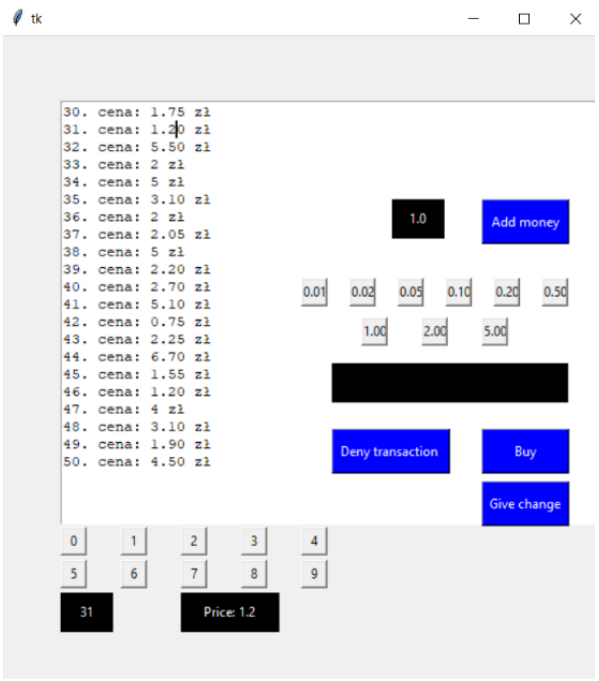


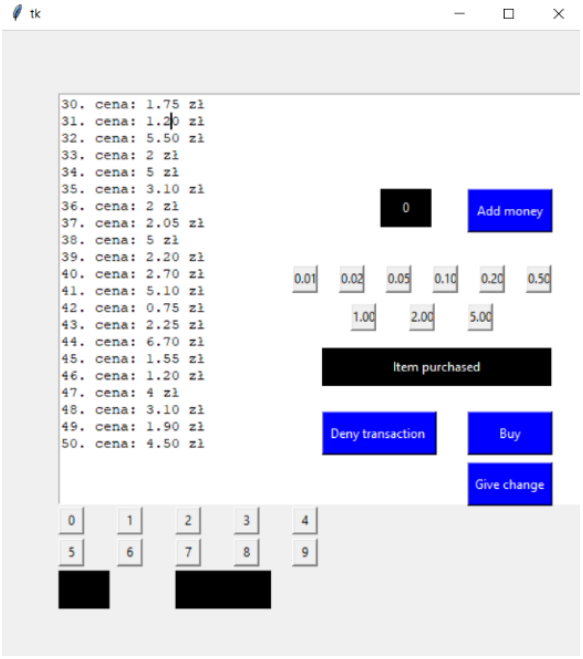
8. Zakup towaru płacąc po 1 gr - suma stu monet ma być równa 1 zł (dla floatów suma sto razy 0.01+0.01+0.01+...+0.01 nie będzie równa 1.0). Płatności można dokonać za pomocą pętli for w interpreterze.






tk
30. cena: 1.75 zł
31. cena: 1.20 zł
32. cena: 5.50 zł
33. cena: 2 zł
34. cena: 5 zł
35. cena: 3.10 zł
36. cena: 2 zł
37. cena: 2.05 zł
38. cena: 5 zł
39. cena: 2.20 zł
40. cena: 2.70 zł
41. cena: 5.10 zł
42. cena: 0.75 zł
43. cena: 2.25 zł
44. cena: 6.70 zł
45. cena: 1.55 zł
46. cena: 1.20 zł
47. cena: 4 zł
48. cena: 3.10 zł
49. cena: 1.90 zł
50. cena: 4.50 zł
0.99
Add money
0.01
0.02
0.05
0.10
0.20
0.50
1.00
2.00
5.00
Deny transaction
Buy
Give change
0
1
2
3
4
5
6
7
8
9
31
Price: 1.2


tk
30. cena: 1.75 zł
31. cena: 1.20 zł
32. cena: 5.50 zł
33. cena: 2 zł
34. cena: 5 zł
35. cena: 3.10 zł
36. cena: 2 zł
37. cena: 2.05 zł
38. cena: 5 zł
39. cena: 2.20 zł
40. cena: 2.70 zł
41. cena: 5.10 zł
42. cena: 0.75 zł
43. cena: 2.25 zł
44. cena: 6.70 zł
45. cena: 1.55 zł
46. cena: 1.20 zł
47. cena: 4 zł
48. cena: 3.10 zł
49. cena: 1.90 zł
50. cena: 4.50 zł
1.0
Add money
0.01
0.02
0.05
0.10
0.20
0.50
1.00
2.00
5.00
Deny transaction
Buy
Give change
0
1
2
3
4
5
6
7
8
9
31
Price: 1.2


tk
30. cena: 1.75 zł
31. cena: 1.20 zł
32. cena: 5.50 zł
33. cena: 2 zł
34. cena: 5 zł
35. cena: 3.10 zł
36. cena: 2 zł
37. cena: 2.05 zł
38. cena: 5 zł
39. cena: 2.20 zł
40. cena: 2.70 zł
41. cena: 5.10 zł
42. cena: 0.75 zł
43. cena: 2.25 zł
44. cena: 6.70 zł
45. cena: 1.55 zł
46. cena: 1.20 zł
47. cena: 4 zł
48. cena: 3.10 zł
49. cena: 1.90 zł
50. cena: 4.50 zł
0
Add money
0.01
0.02
0.05
0.10
0.20
0.50
1.00
2.00
5.00
Item purchased
Deny transaction
Buy
Give change
0
1
2
3
4
5
6
7
8
9

Pozostała część programu

```
readFromFileHowManyMoney()
readFile()
sum = 0
listAllMoneyFromPerson = []
numberDrink = ""
window = tk.Tk()
window.geometry('550x600')
textbox = tk.Text()
textbox.insert(tk.END, """30. cena: 1.75 zł\n31. cena: 1.20 zł\n32. cena: 5.50 zł\n33. cena: 2 zł\n34. cena: 5 zł\n35. cena: 3.10 zł
36. cena: 2 zł\n37. cena: 2.05 zł\n38. cena: 5 zł\n39. cena: 2.20 zł\n40. cena: 2.70 zł\n41. cena: 5.10 zł\n42. cena: 0.75 zł\n43. cena: 2.25 zł
44. cena: 6.70 zł\n45. cena: 1.55 zł\n46. cena: 1.20 zł\n47. cena: 4 zł\n48. cena: 3.10 zł\n49. cena: 1.90 zł\n50. cena: 4.50 zł""")
textbox.place(relx=0.1, rely=0.1)

addMoneyButton = tk.Button(text = "Add money", width=10, height=2, bg="blue", fg="white", command=showMoney)
addMoneyButton.place(relx=0.8, rely=0.25)

changeIsGivenButton = tk.Button(text = "Give change", width=10, height=2, bg="blue", fg="white", command=giveChange)
changeIsGivenButton.place(relx=0.8, rely=0.68)

addedMoney = tk.Label(text="0", width=6, height=2, bg="black", fg="white")

noDrink = tk.Label(text="", width=12, height=2, bg="black", fg="white")
noDrink.place(relx=0.3, rely=0.85)

changeLable = tk.Label(text="", width=30, height=2, bg="black", fg="white")
changeLable.place(relx=0.55, rely=0.5)

verificationTransactionButton = tk.Button(text = "Buy", width=10, height=2, bg="blue", fg="white", command=verificationTransactionFunction)
verificationTransactionButton.place(relx=0.8, rely=0.6)

denyTransactionButton = tk.Button(text = "Deny transaction ", width=14, height=2, bg="blue", fg="white", command=denyTransaction)
denyTransactionButton.place(relx=0.55, rely=0.6)

stuffNumberOfDrinks = tk.Label(text="", width=6, height=2, bg="black", fg="white")
stuffNumberOfDrinks.place(relx=0.1, rely=0.85)
```

```
button0 = tk.Button(text = "0", width=2, height=1, command=lambda:stuffNumber(0))
button0.place(relx=0.1, rely=0.75)
button1 = tk.Button(text = "1", width=2, height=1, command=lambda:stuffNumber(1))
button1.place(relx=0.2, rely=0.75)
button2 = tk.Button(text = "2", width=2, height=1, command=lambda:stuffNumber(2))
button2.place(relx=0.3, rely=0.75)
button3 = tk.Button(text = "3", width=2, height=1, command=lambda:stuffNumber(3))
button3.place(relx=0.4, rely=0.75)
button4 = tk.Button(text = "4", width=2, height=1, command=lambda:stuffNumber(4))
button4.place(relx=0.5, rely=0.75)
button5 = tk.Button(text = "5", width=2, height=1, command=lambda:stuffNumber(5))
button5.place(relx=0.1, rely=0.8)
button6 = tk.Button(text = "6", width=2, height=1, command=lambda:stuffNumber(6))
button6.place(relx=0.2, rely=0.8)
button7 = tk.Button(text = "7", width=2, height=1, command=lambda:stuffNumber(7))
button7.place(relx=0.3, rely=0.8)
button8 = tk.Button(text = "8", width=2, height=1, command=lambda:stuffNumber(8))
button8.place(relx=0.4, rely=0.8)
button9 = tk.Button(text = "9", width=2, height=1, command=lambda:stuffNumber(9))
button9.place(relx=0.5, rely=0.8)

button1coint = tk.Button(text = "0.01", width=2, height=1, command=lambda:addSum(1))
button2coint = tk.Button(text = "0.02", width=2, height=1, command=lambda:addSum(2))
button5coint = tk.Button(text = "0.05", width=2, height=1, command=lambda:addSum(5))
button10coint = tk.Button(text = "0.10", width=2, height=1, command=lambda:addSum(10))
button20coint = tk.Button(text = "0.20", width=2, height=1, command=lambda:addSum(20))
button50coint = tk.Button(text = "0.50", width=2, height=1, command=lambda:addSum(50))
button1papermoney = tk.Button(text = "1.00", width=2, height=1, command=lambda:addSum(100))
button2papermoney = tk.Button(text = "2.00", width=2, height=1, command=lambda:addSum(200))
button5papermoney = tk.Button(text = "5.00", width=2, height=1, command=lambda:addSum(500))

window.mainloop()
```