Περιγραφή των επιμέρους συναρτήσεων του προγράμματος:

Συναρτήσεις που αφορούν το γραφικό περιβάλλον:

```
def disappear(a):
    a.place(x=0,y=0,width=0,height=0)
```

Χρησιμοποιείται για την απομάκρυνση ενός αντικειμένου(entry,button, label) από το γραφικό περιβάλλον.

```
def disappear_all():
    global labels
    global entries
    global buttons
    for i in labels:
        disappear(i)
    for i in entries:
        disappear(i)
        i.delete(0,END)
    for i in buttons:
        disappear(i)
    disappear(submit_button)
```

Χρησιμοποιείται για την απομάκρυνση του συνόλου των αντικειμένων από το γραφικό περιβάλλον, κάνοντας χρήση της συνάρτησης ‘disappear’ με όρισμα καθένα εκ των αντικειμένων αυτών

```
def clear_all():
    for i in entries:
        i.delete(0,END)
```

Χρησιμοποιείται για να γίνει κενός ο χώρος των entries όπου ο χρήστης εισάγει την ποσότητα και την τιμή της πώλησης ή αγοράς

```
def appear():
    label1.place(x=50,y=80,width=150,height=20)
    label2.place(x=50, y=110, width=150, height=20)
    entry1.place(x=200,y=80,width=150,height=20)
    entry2.place(x=200, y=110, width=150, height=20)
```

Εμφανίζει τα labels και τα entries που θα χρησιμοποιηθούν για να καταχωρήσει χρήστης την αγορά ή πώληση(τιμή και ποσότητα)

Συναρτήσεις που αφορούν την καταχώρηση των αγορών και των πωλήσεων:

```
def FIFO():
    global k
    k='FIFO'
    disappear_all()
    clear_all()

def LIFO():
    global k
    k='LIFO'
    disappear_all()
    clear_all()

def MSTKf():
    global k
    k='MSTK'
    disappear_all()
    clear_all()
```

Κάθε μία από τις συναρτήσεις αυτές αποτελεί το command του αντίστοιχου button. Αφού ο χρήστης επιλέξει την μέθοδο παρακολούθησης που προτιμάει, καλείται η αντίστοιχη συνάρτηση και αποδίδει την αντίστοιχη τιμή στην μεταβλητή k που θα χρησιμοποιηθεί ως όρισμα επόμενων συναρτήσεων.

```
def agora():
    disappear_all()
    appear()
    submit_button.place(x=350,y=140,width=50,height=20)
    global litourgia
    litourgia='agora'

def polisi():
    disappear_all()
    appear()
    submit_button.place(x=350,y=140,width=50,height=20)
    global litourgia
    litourgia='polisi'
```

Οι συγκεκριμένες συναρτήσεις αποτελούν τα commands των αντίστοιχων επιλογών του menu. Αφού ο χρήστης επιλέξει αν θέλει να καταχωρήσει αγορά ή πώληση, καλείται η αντίστοιχη συνάρτηση(agora ή polisi) και εμφανίζονται τα entries και labels για την εισαγωγή της τιμής και ποσότητας. Κατόπιν εκχωρείται η αντίστοιχη τιμή στην μεταβλητή litourgia που θα χρησιμοποιηθεί ως όρισμα σε επόμενες συναρτήσεις.

```
def submit():
    q=int(entry1.get())
    p=float(entry2.get())
    clear_all()
    enimerosi_apothematos(q,p)
    if litourgia=='polisi':
        enimerosi_poliseon(q*p)

    disappear_all()
    clear_all()
```

Η συγκεκριμένη συνάρτηση αποτελεί το command του πλήκτρου submit_button που χρησιμοποιείται για εισαχθούν τα δεδομένα της αγοράς ή πώλησης που έχει πληκτρολογήσει ο χρήστης.

```
global inventory_data
global mstk
if litourgia=='agora':
    global agores
    agores=agores+q*p
    if k=='MSTK':
        if inventory_data.at[0, 'Τιμή'] == 0:
            inventory_data.at[0, 'Τιμή'] = p
            inventory_data.at[0, 'Ποσότητα']=q
            mstk=p
        else:
            mstk=(inventory_data.at[0,
'Ποσότητα']*mstk+p*q)/(inventory_data.at[0,'Ποσότητα']+q)
            inventory_data.at[0, 'Τιμή']=mstk
            inventory_data.at[0, 'Ποσότητα']+=q
    else:
        if len(inventory_data)==1:
            if inventory_data.at[0, 'Τιμή'] == 0:
                inventory_data.at[0, 'Τιμή'] = p
        a=0
        i=0
        while a==0 and i<=len(inventory_data)-1:
            if inventory_data.at[i, 'Τιμή'] == p:
                print('ok')
                inventory_data.at[i, 'Ποσότητα']+=q
                a=1
            i+=1
        if a==0:
            inventory_data.loc[len(inventory_data)]=[q,p]

if litourgia=='polisi':
    global kp
    if k=='FIFO':
        for i in range(0,len(inventory_data)):
            if inventory_data.at[i,'Ποσότητα']>q:
```

```
                inventory_data.at[i, 'Ποσότητα']-=q
                kp+=q*inventory_data.at[i, 'Τιμή']
                q=0
            else:
                kp+=inventory_data.at[i,
'Ποσότητα']*inventory_data.at[i,'Τιμή']
                q-=inventory_data.at[i,'Ποσότητα']

inventory_data=inventory_data.drop(i,axis=0)
        inventory_data.index = range(0,
len(inventory_data))
    elif k=='LIFO':
        for i in range(len(inventory_data)-1,-1,-1):
            if inventory_data.at[i,'Ποσότητα']>=q:
                inventory_data.at[i, 'Ποσότητα']-=q
                kp+=q*inventory_data.at[i, 'Τιμή']
                i=len(inventory_data)+1
            else:
                kp+=inventory_data.at[i,
'Ποσότητα']*inventory_data.at[i,'Τιμή']
                q-=inventory_data.at[i,'Ποσότητα']

inventory_data=inventory_data.drop(i,axis=0)
        inventory_data.index = range(0,
len(inventory_data))
    else:
        kp+=mstk*q
        inventory_data.at[0, 'Ποσότητα']-=q
```

Η συγκεκριμένη συνάρτηση χρησιμοποιεί τις μεταβλητές k και litourgia που έχουν αρχικοποιηθεί σε προηγούμενες συναρτήσεις για να ενημερώσει το απόθεμα και το κόστος πωλήσεων. Τα δεδομένα του αποθέματος αποθηκεύονται σε δομή dataframe με διαφορετική γραμμή για κάθε παρτίδα στην περίπτωση των LIFO και FIFO.

Συναρτήσεις που αφορούν την αποθήκευση των δεδομένων:

```
def save_apothemata():
    global inventory_data
    global path
    inventory_data.to_excel(path +
'Απόθεμα.xlsx',index=False)
    print(inventory_data)
```

Αποθήκευση του dataframe όπου έχουν αποθηκευτεί τα δεδομένα του αποθέματος ως αρχείο excel.

```
def enimerosi_poliseon(polisi):
    month = date.today().month

    if month == 1:
        month = 'Ιανο.'
    elif month == 2:
        month = 'Φεβρ.'
    elif month == 3:
        month = 'Μαρτ.'
    elif month == 4:
        month = 'Απρι.'
    elif month == 5:
        month = 'Μαιο.'
    elif month == 6:
        month = 'Ιουν.'
    elif month == 7:
        month = 'Ιουλ.'
    elif month == 8:
        month = 'Αυγο.'
    elif month == 9:
        month = 'Σεπτ.'
    elif month == 10:
        month = 'Οκτο.'
    elif month == 11:
        month = 'Νοεμ.'
    elif month == 12:
        month = 'Δεκε.'
    global sales_data
    sales_data.loc[month]['Πωλήσεις'] += polisi
    plott(sales_data)
    save(sales_data)
    describe(sales_data)
```

Καθορισμός του μήνα κατά τον οποίον καταχωρείται η πώληση και ενημέρωση του dataframe όπου αποθηκεύονται αναλυτικά οι πωλήσεις ανά μήνα.

```
def plott(df):
    plt.plot(df['Πωλήσεις'], c='r')
    global months
    global path
    plt.savefig(path+'διάγραμμα.png')
```
Δημιουργία διαγράμματος με τα δεδομένα των πωλήσεων ανά μήνα

```
def save(df):
    global path
    df.to_excel(path+'Αναλυτικές Πωλήσεις ανά Μήνα.xlsx')
    print(df)
```
Αποθήκευση του αντίστοιχου dataframe

```
def describe(df):
    global path
    global kp
    global agores
    data2=df.describe()
    list=[]
    list.append(data2.at['mean','Πωλήσεις'])
    list.append(data2.at['min','Πωλήσεις'])
    list.append(data2.at['max', 'Πωλήσεις'])
    list.append(kp)
    list.append(agores)
    indexes=['μέσος αριθμός πωλήσεων ανά
μήνα:','ελάχιστος αριθμός πωλήσεων ανά μήνα:','μέγιστος
αριθμός πωλήσεων ανά μήνα:', 'κόστος πωλήσεων','αγορές']

data=pd.DataFrame(np.array(indexes).reshape((5,1)),column
s=['Στοιχείο'])
    data['Τιμή']=list
    print(data)
    writer=pd.ExcelWriter(path+'Στατιστικά
Στοιχεία.xlsx',engine='xlsxwriter')
    data.to_excel(writer,index=False,sheet_name='Sheet1')
    workbook=writer.book
    worksheet=writer.sheets['Sheet1']
    for i, col in enumerate(data.columns):
        width=max(data[col].apply(lambda
x:len(str(x))).max(),len(data[col]))
        worksheet.set_column(i,i,width)
    writer.close()
```
Χρήση της εντολής describe για τον προσδιορισμό των μέσων, ελάχιστων και μέγιστων μηνιαίων πωλήσεων και αποθήκευση των αντίστοιχων στοιχείων σε μορφή excel.