Error Handling

For menu:

If-else and try-except statements are used to counter the issue of the option number being out of range for the menu function i.e. 1-6 or not a numeric option selected.

The user is given the option to try again in case of wrong selection made by them and by using try-except, only integer value is allowed to be selected for comparing.

Next the integer value is check if it lies in range 1-6 and if not, error message is displayed and the user is allowed to try again. Here the if-else is used.

```
def menu():
    while True:
        print("Select an Option:\n \n1.Create Habit")
        print("2.View Habit")
        print("3.Modify Habit")
        print("4.Analyze Habit")
        print("5.Delete Habit")
        print("6.Exit")
        try:
            x=int(input("Enter Your Choice... "))
            if x==1:
                create_habit_menu()
            elif x==2:
                view_habit()
            elif x==3:
                modify_habit()
            elif x==4:
                analyze_habit()
            elif x==5:
                delete_habit()
            elif x==6:
                print('Quiting...')
                break
        else:
            print("Invalid Input, Try Again\n")
        except:
            print("Invalid Input, Try Again")
```

Following figure highlights the error handling part.

For habit menu:

For the habit menu, again try-except is used to take an integer value and then again if-else is used to check for the options if ranging from 1-3 but is the option selected by the user is wrong and not in range then, the user is sent back to the main menu.

```
def create_habit_menu():
    print("Select an Option:\n \n1.Daily")
    print("2.Weekly")
    print("3.Monthly")
    try:
        x=int(input("Enter Your Choice... "))
        if x==1:
            create_habit('Daily')
        elif x==2:
            create_habit('Weekly')
        elif x==3:
            create_habit('Monthly')
        else:
            print("Invalid Input Returning Back...\n")
    except:
        print("Invalid Input Returning Back...\n")
```

Following figure highlights the error handling part.

For modify habit:

Again, first try-except is used to match if the user entered an integer value or not and again if the user entered value is greater than 1-3 an error message is displayed and the user is taken back to the main menu.

```
def modify_habit():
    y=[]
    f=open("tasks.txt",'r')
    print("Select an Option to Modify:\n \n1.Check Task")
    print("2.Uncheck Task")
    print("3.Modify Habit Name")
    try:
        sel=int(input("Your Choice... "))
    except:
        print("Invalid Choice, Returning to Main Menu\n")
        return
    if sel==1:
        mark_habit('Check')
    elif sel==2:
        mark_habit('Uncheck')
    elif sel==3:
        l=input("Enter New Habit Name:  ")
        mark_habit(l)
    else:
        print("Invalid Choice, Returning to Main Menu\n")
        return
    while True:
        x=f.readline()
        if x=='':
            break
        y.append(x)
```

Following figure highlights the error handling part.

For mark habit:

A check flag name ’found’ is created with initial value 0. First the name is match with all the other habit names and if none is found,

The ‘found’ flag remains 0 but if a match is found, it turns to one which at the end to the function prints the result to display a notification if the name entered existed in the data or not. And then displays the related error or success notification.

```
def mark_habit(x):
    y=[]
    found=0
    f=open("tasks.txt",'r')
    i=input("Enter the name of the habit to Modify: ")
    while True:
        o=f.readline()
        if o=='':
            f2=open("tasks.txt",'w')
            for j in y:
                f2.write(j)
            if found==0:
                print("No such habit with this name found...")
            else:
                print("Modification completed")
            return
        o=o.split(' ')
        if o[0]==i:
            found=1
            if x=='Check':
                o[4]=int(o[4])+1
                o[4]=str(o[4])
            elif x=='Uncheck':
                o[4]=str(0)
            else:
                o[0]=x
        li=o[0]+" "+o[1]+" "+o[2]+" "+o[3]+" "+o[4]+' '+o[5]
        y.append(li)
```

Following figure highlights the error handling part.

For habit entry:

There are many important checks kept for valid data in this function it also is connect to date entry function to receive values for the habit entry and ending date.

Try-except is used to get the valid integer dates data such as year, month and days and later the year is check to be greater than the current year, the month is kept to be checked if it is not greater than 12 and if the year is current than more the current month and if the month is also the same the day is checked not to be greater than 31 and not bigger than the current date for current year and month scenario. Finally, the check is kept for matching the current and the input date. Also, an infinite loop is kept so that the data has to be re-entered until correct data is not received.

```
def date_entry():
    while(1):
        y0=''
        y1=0
        y2=0
        y3=0
        while(1):
            try:
                y1=int(input("Please enter the year: "))
                if y1>1000:
                    break
                print("Invalid Input, Try Again")
            except:
                print("Invalid Input, Try Again")
        while(1):
            try:
                y2=int(input("Please enter the month: "))
                if y2<=12:
                    break
                print("Invalid Input, Try Again")
            except:
                print("Invalid Input, Try Again")
        while(1):
            try:
                y3=int(input("Please enter the day: "))
                if y3<=31:
                    break
                print("Invalid Input, Try Again")
            except:
                print("Invalid Input, Try Again")
        today=date.today()
        d4 = today.strftime("%Y-%m-%d")
        d4=d4.split('-')
        if int(d4[0])<y1:
            break
        elif int(d4[0])==y1 and int(d4[1])<y2:
            break
        elif int(d4[0])==y1 and int(d4[1])==y2 and int(d4[2])<=y3:
            break
        else:
            print("Date cannot be earlier than todays date... please enter again")
    y0=str(y1)+'-'+str(y2)+'-'+str(y3)
```

Following figure highlights the error handling part.