9608/22/PRE/O/N/2020

The cell below declares the variables and arrays that are supposed to be pre-populated.

In [ ]:

```
ItemCode = ["1001", "6056", "5557", "2568", "4458"]
ItemDescription = ["Pencil", "Pen", "Notebook", "Ruler", "Compass"]
Price = [1.0, 10.0, 100.0, 20.0, 30.0]
NumberInStock = [100, 100, 50, 20, 20]

n = len(ItemCode)
```

TASK 1.4

Write program code to produce a report displaying all the information stored about each item for which the number in stock is below a given level. The planning and identifier table are in the pasudocode file and the markdown respectively.

In [ ]:

```
ThresholdLevel = int(input("Enter the minumum stock level: "))

for Counter in range(n):

    if NumberInStock[Counter] < ThresholdLevel:
        print("\nItem Code:", ItemCode[Counter])
        print("Item Description:", ItemDescription[Counter])
        print("Price:", Price[Counter])
        print("Number in stock:", NumberInStock[Counter])
```

TASK 2.2

Design an algorithm to input the four pieces of data about a stock item, form a string according to your format design, and write the string to the text file.
First draw a program flowchart, then write the equivalent pseudocode.

In [ ]:

```
RecordsFile = "Item Records.txt"
FileObject = open(RecordsFile, "a+")
WriteString = ""

NewItemCode = int(input("\nEnter item code: "))
WriteString = ':' + str(NewItemCode)

NewItemDescription = str(input("Enter item description: "))
WriteString += ':' + NewItemDescription

NewPrice = float(input("Enter new price: "))
WriteString += ':' + str(NewPrice)

NewNumberInStock = int(input("Enter the number of items in stock: "))
WriteString += ':' + str(NewNumberInStock) + '\n'

FileObject.write(WriteString)
FileObject.close()
```

TASK 2.4

The cell below defines the sub-routines which will be used by more than of the tasks.

In [ ]:

```
def GetItemCode():
    Valid = False

    TestItemCode = int(input("Enter the code of the item: "))

    while not (TestItemCode > 1000 and TestItemCode < 9999):
        TestItemCode = int(input("Re-enter the codeof the item: "))

    return TestItemCode


def ExtractDetails(RecordString, Details):
    Position = 0
    SearchString = RecordString + ':'

    if RecordString != "":
        for Counter in range(4):
            Position += 1
            CurrentCharacter = SearchString[Position : Position + 1]

            while CurrentCharacter != ':':
                Details[Counter] += CurrentCharacter
                Position += 1
                CurrentCharacter = SearchString[Position : Position + 1]
```

TASK 2.4 (1)

Add a new stock item to the text file. Include validation of the different pieces of information as appropriate. For example item code data may be a fixed format.

In [ ]:

```
WriteString = ""
WriteString = ':' + str(GetItemCode())

NewItemDescription = str(input("\nEnter item description: "))
WriteString += ':' + NewItemDescription

NewPrice = float(input("Enter the price of the item: "))
WriteString += ':' + str(NewPrice)

NewNumberInStock = int(input("Enter number of items in stock: "))
WriteString += ':' + str(NewNumberInStock) + '\n'

FileObject = open(RecordsFile, "a+")
FileObject.write(WriteString)
FileObject.close()
```

TASK 2.4 (2)

Search for a stock item with a specific item code. Output the other pieces of data together with suitable supporting text.

In [ ]:

```
Found = False
CurrentRecord = ""

print("\nEnter the code of the item you want to search for")
DesiredItemCode = GetItemCode()

FileObject = open(RecordsFile, "r+")
FileData = (FileObject.read()).split('\n')
FileObject.close()

for record in FileData:
    CurrentRecord = record
    if CurrentRecord[1:5] == str(DesiredItemCode):
        Found = True
        break

if Found:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(CurrentRecord, DetailsOfRecord)

    print("\nItem Code: " + str(DetailsOfRecord[0]))
    print("Item Description: " + DetailsOfRecord[1])
    print("Price of item: " + str(DetailsOfRecord[2]))
    print("Number of the item in stock: " + str(DetailsOfRecord[3]))

else:
    print("Item not found.")
```

TASK 2.4 (3)

Search for all stock items with a specific item description, with output as for task 2.

In [ ]:

```
DesiredItemDescription = str(input("\nEnter the description of the item you want to sea
rch for: "))

FileObject = open(RecordsFile, "r+")
FileData = (FileObject.read()).split('\n')
FileObject.close()

for record in FileData:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(record, DetailsOfRecord)

    if DetailsOfRecord[1] == DesiredItemDescription:
        print("\nItem Code: " + str(DetailsOfRecord[0]))
        print("Item Description: " + DetailsOfRecord[1])
        print("Price of item: " + str(DetailsOfRecord[2]))
        print("Number of the item in stock: " + str(DetailsOfRecord[3]))
```

TASK 2.4 (4)

Output a list of all stock items with a price greater than a given amount.

In [ ]:

```
DesiredPrice = float(input("\nEnter the maximum threshold price: "))

FileObject = open(RecordsFile, "r+")
FileData = (FileObject.read()).split('\n')
FileObject.close()

for record in FileData:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(record, DetailsOfRecord)

    if float(DetailsOfRecord[2]) < DesiredPrice:
        print("\nItem Code: " + str(DetailsOfRecord[0]))
        print("Item Description: " + DetailsOfRecord[1])
        print("Price of item: " + str(DetailsOfRecord[2]))
        print("Number of the item in stock: " + str(DetailsOfRecord[3]))
```

Standalone Compliled Program

The above cells demonstrate how each individual aspect of each task works. The code in the cell below is for every task combined into one, and can run independently.

In [ ]:

```
## Arrays which are supposed to be pre-populated
ItemCode = ["1001", "6056", "5557", "2568", "4458"]
ItemDescription = ["Pencil", "Pen", "Notebook", "Ruler", "Compass"]
Price = [1.0, 10.0, 100.0, 20.0, 30.0]
NumberInStock = [100, 100, 50, 20, 20]

## Constant for the initial number of element (pre-defined)
n = len(ItemCode)

## Constant for the name of the file
RecordsFile = "Item Records.txt"

## Open file for "APPEND" and assign the I/O reference to a variable
FileObject = open(RecordsFile, "a")

## Subroutine to input a valid item code
def GetItemCode():
    Valid = False

    TestItemCode = int(input("\nEnter the code of the item: "))

    while not (TestItemCode > 1000 and TestItemCode < 9999):
        TestItemCode = int(input("Re-enter the codeof the item: "))

    return TestItemCode

## Subroutine to extract details of a given record string into an array
def ExtractDetails(RecordString, Details):
    Position = 0
    SearchString = RecordString + ':'

    if RecordString != "":
        for Counter in range(4):
            Position += 1
            CurrentCharacter = SearchString[Position : Position + 1]

            while CurrentCharacter != ':':
                Details[Counter] += CurrentCharacter
                Position += 1
                CurrentCharacter = SearchString[Position : Position + 1]


## TASK 1.4
ThresholdLevel = int(input("Enter the minumum stock level: "))

for Counter in range(n):

    if NumberInStock[Counter] < ThresholdLevel:
        print("\nItem Code:", ItemCode[Counter])
        print("Item Description:", ItemDescription[Counter])
        print("Price:", Price[Counter])
        print("Number in stock:", NumberInStock[Counter])


## TASK 2.2
WriteString = ""

NewItemCode = int(input("\nEnter item code: "))
WriteString = ':' + str(NewItemCode)
```

```
NewItemDescription = str(input("Enter item description: "))
WriteString += ':' + NewItemDescription

NewPrice = float(input("Enter new price: "))
WriteString += ':' + str(NewPrice)

NewNumberInStock = int(input("Enter the number of items in stock: "))
WriteString += ':' + str(NewNumberInStock) + '\n'

FileObject.write(WriteString)


## TAKS 2.4 (1)
WriteString = ""
WriteString = ':' + str(GetItemCode())

NewItemDescription = str(input("Enter item description: "))
WriteString += ':' + NewItemDescription

NewPrice = float(input("Enter the price of the item: "))
WriteString += ':' + str(NewPrice)

NewNumberInStock = int(input("Enter number of items in stock: "))
WriteString += ':' + str(NewNumberInStock) + '\n'

FileObject.write(WriteString)


## Close the file and save changes
FileObject.close()

## Open the file in "READ" mode
FileObject = open(RecordsFile, "r")

## Read data from the file into an array. They are also split using the newline delimit
er '\n'.
FileData = (FileObject.read()).split('\n')

## Remove last empty element
FileData.pop()

## Close the file
FileObject.close()

## TASK 2.4 (2)
Found = False

print("\nEnter the code of the item you want to search for")
DesiredItemCode = GetItemCode()

for record in FileData:
    if record[1:5] == str(DesiredItemCode):
        Found = True
        break

if Found:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(record, DetailsOfRecord)

    print("\nItem Code: " + str(DetailsOfRecord[0]))
```

```
    print("Item Description: " + DetailsOfRecord[1])
    print("Price of item: " + str(DetailsOfRecord[2]))
    print("Number of the item in stock: " + str(DetailsOfRecord[3]))

else:
    print("Item not found.")


## TASK 2.4 (3)
DesiredItemDescription = str(input("\nEnter the description of the item you want to sea
rch for: "))

for record in FileData:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(record, DetailsOfRecord)

    if DetailsOfRecord[1] == DesiredItemDescription:
        print("\nItem Code: " + str(DetailsOfRecord[0]))
        print("Item Description: " + DetailsOfRecord[1])
        print("Price of item: " + str(DetailsOfRecord[2]))
        print("Number of the item in stock: " + str(DetailsOfRecord[3]))


## TASK 2.4 (4)
DesiredPrice = float(input("\nEnter the maximum threshold price: "))

for record in FileData:
    DetailsOfRecord = ["" for i in range(4)]
    ExtractDetails(record, DetailsOfRecord)

    if float(DetailsOfRecord[2]) < DesiredPrice:
        print("\nItem Code: " + str(DetailsOfRecord[0]))
        print("Item Description: " + DetailsOfRecord[1])
        print("Price of item: " + str(DetailsOfRecord[2]))
        print("Number of the item in stock: " + str(DetailsOfRecord[3]))
```