ALevel CS C26 File processing and Exception

Records

pseudocode:

```
TYPE CarRecord
    DECLARE VehicleID : STRING
    DECLARE Registration  : STRING
    DECLARE DateOfRegistration : DATE
    DECLARE EngineSize : INTEGER
    DECLARE PurchasePrice  : CURRENC
ENDTYPE

// declare a variable
DECLARE ThisCar : CarRecord
```

python:

```
class CarRecord: # declaring a class without other methods
    def __init__(self): # constructor
        self.VehicleID = ""
        self.Registration = ""
        self.DateOfRegistration = None
        self.EngineSize = 0
        self.PurchasePrice = 0.00
ThisCar = CarRecord() # instantiates a car record
ThisCar.EngineSize = 2500 # assign a value to a field
Car = [CarRecord() for i in range(100)] # make a list of 100 car
records
Car[1].EngineSize = 2500 # assign value to a field of the 2nd car in
list
```

Random-access file processing

pseudocode:

```
// Saving a record
OPENFILE CarFile FOR RANDOM
Address ← Hash(ThisCar.VehicleID)
SEEK CarFile, Address
PUTRECORD CarFile, ThisCar
CLOSEFILE CarFile

// Retrieving a record
OPENFILE CarFile FOR RANDOM
Address ← Hash(ThisCar.VehicleID)
SEEK CarFile, Address
GETRECORD CarFile, ThisCar
CLOSEFILE CarFile
```

python:

```
import pickle # this library is required to create binary files
ThisCar = CarRecord()
CarFile = open('CarFile.DAT','rb+') # open file for binary read and write
Address = hash(ThisCar.VehicleID)
CarFile.seek(Address)
pickle.dump(ThisCar, CarFile) # write a whole record to the binary file
CarFile.close() # close file
#  to find a record from a given VehicleID:
CarFile = open('CarFile.DAT','rb')  # open file for binary read
Address = hash(VehicleID)
CarFile.seek(Address)
ThisCar = pickle.load(CarFile) # load record from file
CarFile.close()
```

Exception

python:

```
NumberString = input("Enter an integer: ")
try:
    n = int(NumberString)
    print(n)
except:
    print("This was not an integer")
```

Sequential file processing

pseudocode:

```
// Saving content of array
OPENFILE CarFile FOR WRITE
FOR i ← 1 TO MaxRecords
    PUTRECORD CarFile, Car[i]
NEXT i
CLOSEFILE CarFile

// Restoring contents of array
OPENFILE CarFile FOR READ
FOR i ← 1 TO MaxRecords
    GETRECORD CarFile, Car[i]
NEXT i
CLOSEFILE CarFile
```

python:

```
import pickle # this library is required to create binary files
Car = [CarRecord() for i in range(100)]

CarFile = open('CarFile.DAT', 'wb') # open file for binary write

for i in range(100): # loop for each array element
    pickle.dump(Car[i], CarFile) # write a whole record to the binary file

CarFile.close() # close file

CarFile = open('CarFile.DAT', 'rb') # open file for binary read

Car = [] # start with empty list
while True: # check for end of file
    Car.append(pickle.load(CarFile)) # append record from file to end of list

CarFile.close()
```