



113- [PF] - Lab - Tipos de datos compuestos

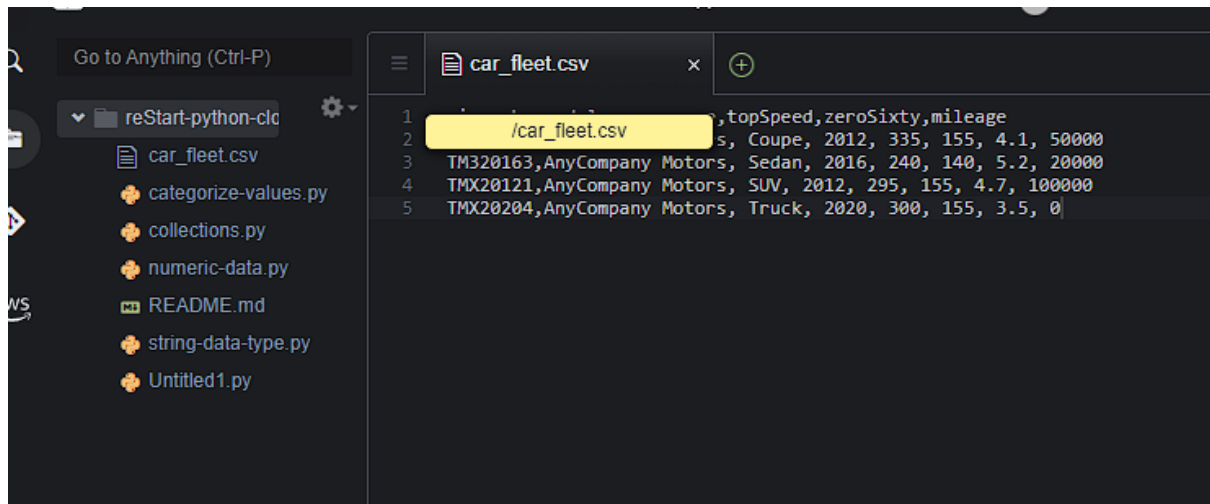
Creación de datos de un inventario de vehículos

Valores separados por comas (CSV) es un formato de archivo que se utiliza para almacenar datos tabulares, como los datos de una hoja de cálculo. Trabajaré con el archivo CSV del siguiente bloque.

1. En la barra de menú, seleccione **File > New File** (Archivo > Archivo nuevo).

Esta acción crea un archivo sin título.

1. Elija **File > Save As...** (Archivo > Guardar como...) y guarde el archivo como `car_fleet.csv`.
2. Copie y pegue el siguiente bloque de texto en el archivo **car_fleet.csv** y guarde el archivo.



```
vin,make,model,year,range,topSpeed,zeroSixty,mileage
TMX20122,AnyCompany Motors, Coupe, 2012, 335, 155, 4.1, 50000
TMX20163,AnyCompany Motors, Sedan, 2016, 240, 140, 5.2, 20000
TMX20121,AnyCompany Motors, SUV, 2012, 295, 155, 4.7, 100000
TMX20204,AnyCompany Motors, Truck, 2020, 300, 155, 3.5, 0
```

Sugerencia: Si se abre una ventana emergente con el mensaje *Native Clipboard Unavailable* (Portapapeles nativo no disponible), utilice el teclado, no el menú del navegador, para realizar acciones de copiar y pegar. Por ejemplo, en Windows, utilice CTRL+C y CTRL+V para copiar y pegar, respectivamente. En Mac, utilice Command+C y Command+V.

Creación de un programa de inventario de vehículos

Definición del diccionario

Leerá el archivo a través de un módulo denominado `csv`. Además, realizará una copia profunda de los datos para almacenarlos en la memoria mediante un módulo denominado `copy`.

1. En el panel de navegación del IDE, elija el archivo `.py` (haga doble clic sobre este) que creó en la sección anterior, *Creación del archivo de ejercicios de Python*.
2. Primero, importe los módulos que utilizará:

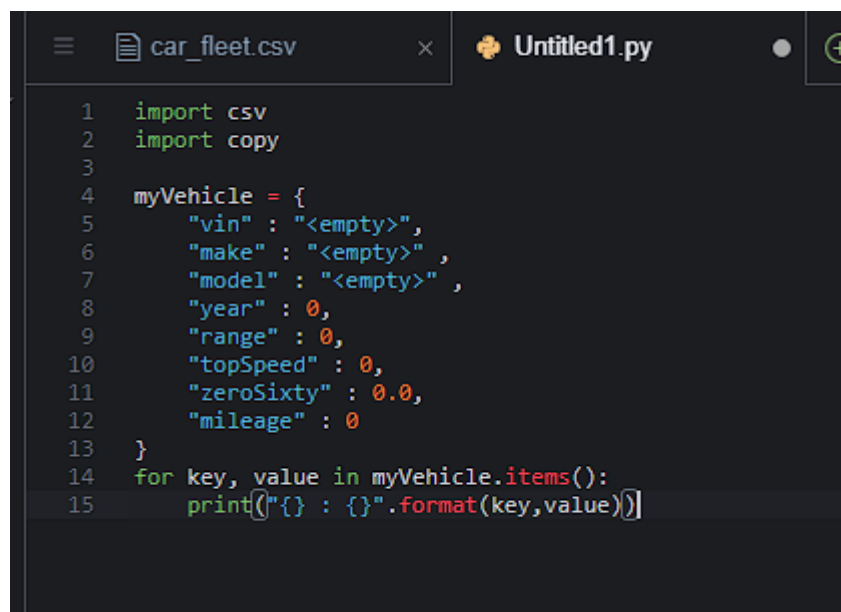
```
import csv
import copy
```

1. A continuación, defina el diccionario que funcionará como tipo compuesto para leer los datos tabulares:

```
myVehicle = {  
    "vin" : "<empty>",  
    "make" : "<empty>" ,  
    "model" : "<empty>" ,  
    "year" : 0,  
    "range" : 0,  
    "topSpeed" : 0,  
    "zeroSixty" : 0.0,  
    "mileage" : 0  
}
```

1. Utilizará un bucle `for` para recorrer las claves y valores del diccionario.

```
for key, value in myVehicle.items():  
    print("{} : {}".format(key,value))
```



The screenshot shows a code editor with two tabs: 'car_fleet.csv' and 'Untitled1.py'. The code in 'Untitled1.py' is as follows:

```
1 import csv  
2 import copy  
3  
4 myVehicle = {  
5     "vin" : "<empty>",  
6     "make" : "<empty>" ,  
7     "model" : "<empty>" ,  
8     "year" : 0,  
9     "range" : 0,  
10    "topSpeed" : 0,  
11    "zeroSixty" : 0.0,  
12    "mileage" : 0  
13 }  
14 for key, value in myVehicle.items():  
15     print("{} : {}".format(key,value))
```

Nota: La función `items()` pertenece al tipo de datos de diccionario. La función `items()` indica al bucle `for` que recorra la colección que pertenece al tipo de datos de diccionario.

1. Defina una lista vacía para almacenar el inventario de vehículos que leerá:

```
myInventoryList = []
```

1. Guarde el archivo.

Copia del archivo CSV en memoria

Leerá los datos del disco (disco duro) y realizará una copia en memoria (memoria de acceso aleatorio o RAM). En una computadora, se utiliza un *disco duro* para almacenar los datos a largo plazo, incluso cuando se apaga la alimentación. La sigla RAM hace referencia a la memoria temporal, que es más rápida, pero se borra cuando se apaga la alimentación de la computadora.

Se le presentará la instrucción de sintaxis `with open`, que mantiene un archivo abierto mientras lee datos. Cerrará automáticamente el archivo CSV cuando termine de ejecutarse el código dentro del bloque `with`.

También utilizará una nueva manera de formatear una cadena. En lugar de utilizar comillas dobles y `.format` para pasar las variables, puede utilizar comillas simples y escribir las variables entre los símbolos "{}".

Por último, `csv.reader()` es una función existente en la biblioteca `csv`, que ya ha importado con la instrucción `import csv`.

Debería estar familiarizado con la mayor parte del resto del código.

Ahora, regrese al archivo en Python:

1. Escriba el siguiente código:

```
with open('car_fleet.csv') as csvFile:
    csvReader = csv.reader(csvFile, delimiter=',')
    lineCount = 0
    for row in csvReader:
        if lineCount == 0:
            print(f'Column names are: {", ".join(row)}')
            lineCount += 1
        else:
            print(f'vin: {row[0]} make: {row[1]}, model: {row[2]}, year: {row[3]}, range: {row[4]}, topSpeed: {row[5]}, zeroSixty: {row[6]}, mileage: {row[7]}')
            currentVehicle = copy.deepcopy(myVehicle)
            currentVehicle["vin"] = row[0]
            currentVehicle["make"] = row[1]
            currentVehicle["model"] = row[2]
            currentVehicle["year"] = row[3]
            currentVehicle["range"] = row[4]
            currentVehicle["topSpeed"] = row[5]
            currentVehicle["zeroSixty"] = row[6]
            currentVehicle["mileage"] = row[7]
            myInventoryList.append(currentVehicle)
            lineCount += 1
    print(f'Processed {lineCount} lines.')
```

Aunque este código parece una gran cantidad de código para procesar, en su mayoría comprende instrucciones que vio en laboratorios anteriores. Contiene un bucle `for` con una instrucción `if-else`, seguido de una instrucción `print()` al final.

Sin embargo, la siguiente línea necesita una explicación adicional:

```
currentVehicle = copy.deepcopy(myVehicle)
```

De forma predeterminada, Python realiza una *copia superficial* de tipos de datos complejos. Una copia superficial hace referencia o apunta a la ubicación del almacenamiento de la variable de diccionario `myVehicle`. Sin esta línea, contaría solo con una única caja de almacenamiento y solo el último elemento de la lista se copiaría en memoria. Esta línea garantiza que se creen nuevas cajas de almacenamiento en memoria para almacenar los nuevos datos tabulares que se están leyendo.

Impresión del inventario de vehículos

Finalizará el script de Python mostrando el inventario de automóviles de la variable `myInventoryList`.

1. Regrese al script en Python y escriba el siguiente código:

```
for myCarProperties in myInventoryList:
    for key, value in myCarProperties.items():
        print("{} : {}".format(key,value))
    print("-----")
```

1. Guarde el archivo.
2. Elija el botón **Run** (Ejecutar) en la barra de menú para ejecutar el programa.
3. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.
4. Revise una vez más el código que lee los datos tabulares del archivo CSV. Comprender esta sección del código es fundamental para este ejercicio.
- 5.

```
vin : <empty>
make : <empty>
```

```
model : <empty>
year : 0
range : 0
topSpeed : 0
zeroSixty : 0.0
mileage : 0
Column names are: vin, make, model, year, range, topSpeed, zeroSixty, mileage
vin: TMX20122 make: AnyCompany Motors, model: Coupe, year: 2012, range: 335,
topSpeed: 155, zeroSixty: 4.1, mileage: 50000
vin: TM320163 make: AnyCompany Motors, model: Sedan, year: 2016, range: 240,
topSpeed: 140, zeroSixty: 5.2, mileage: 20000
vin: TMX20121 make: AnyCompany Motors, model: SUV, year: 2012, range: 295,
topSpeed: 155, zeroSixty: 4.7, mileage: 100000
vin: TMX20204 make: AnyCompany Motors, model: Truck, year: 2020, range: 300,
topSpeed: 155, zeroSixty: 3.5, mileage: 0
Processed 5 lines.
vin : TMX20122
make : AnyCompany Motors
model : Coupe
year : 2012
range : 335
topSpeed : 155
zeroSixty : 4.1
mileage : 50000
```

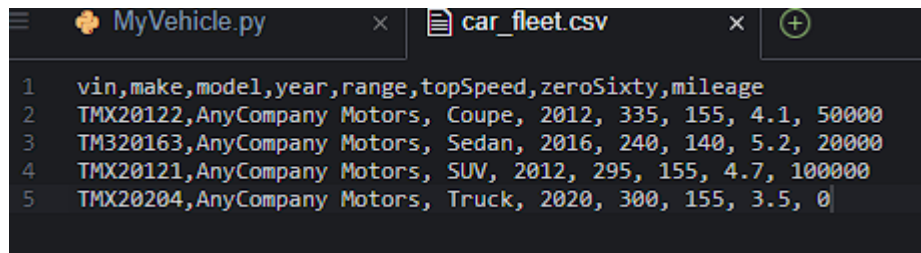
```
vin : TM320163
make : AnyCompany Motors
model : Sedan
year : 2016
range : 240
topSpeed : 140
zeroSixty : 5.2
mileage : 20000
```

```
vin : TMX20121
make : AnyCompany Motors
model : SUV
year : 2012
range : 295
topSpeed : 155
zeroSixty : 4.7
mileage : 100000
```

```
vin : TMX20204
make : AnyCompany Motors
model : Truck
year : 2020
range : 300
topSpeed : 155
```

```
zeroSixty : 3.5  
mileage : 0
```

Process exited with code: 0



```
1 vin,make,model,year,range,topSpeed,zeroSixty,mileage  
2 TMX20122,AnyCompany Motors, Coupe, 2012, 335, 155, 4.1, 50000  
3 TM320163,AnyCompany Motors, Sedan, 2016, 240, 140, 5.2, 20000  
4 TMX20121,AnyCompany Motors, SUV, 2012, 295, 155, 4.7, 100000  
5 TMX20204,AnyCompany Motors, Truck, 2020, 300, 155, 3.5, 0
```

¡Felicitaciones! Ha trabajado con tipos de datos compuestos en Python.