Control structures & Functions

GITAA
Transforming careers

In this lecture

- Control structures
    - If elif family
    - For
    - While
- Functions

Control Structures in Python

- Execute certain commands only when certain condition(s) is (are) satisfied (if-then-else)
- Execute certain commands repeatedly and use a certain logic to stop the iteration (for, while loops)

If else family of constructs

- If , If else and If-elif - else are a family of constructs where:
    - A condition is first checked, if it is satisfied then operations are performed
    - If condition is not satisfied, code exits construct or moves on to other options

If else family of constructs

Task	Command
• If construct:	• if expression: statements

• If – else construct:	• If expression:
	    statements
	else:
	    statements

• If – elif - else construct	• If expression1:
	    statements
	elif expression2:
	    statements
	else:
	    statements

For loop

- Execute certain commands repeatedly and use a certain logic to stop the iteration (for loop)

- Execute multiple commands repeatedly as per the specified logic (nested for loop)

Task	Command
for	for iter in sequence: statements

while loop

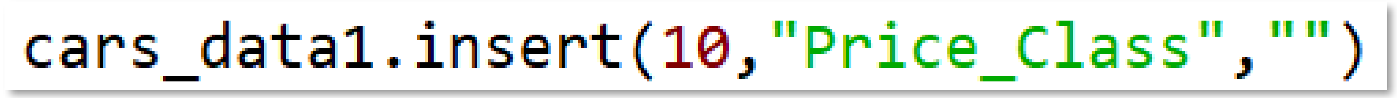
- A while loop is used when a set of commands are to be executed depending on a specific condition

Task	Command

while	while (condition is satisfied):
	        statements

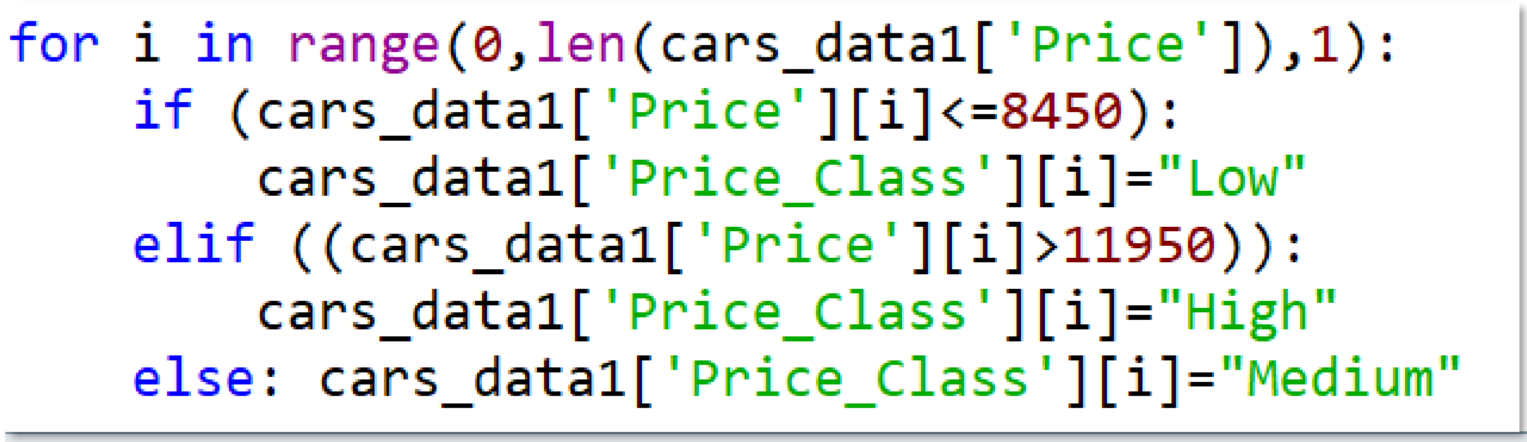
Example: if else and for loops

- We will create 3 bins from the 'Price' variable using *If Else and For Loops*
- The binned values will be stored as classes in a new column, 'Price Class'
- Hence, inserting a new column



```
cars_data1.insert(10,"Price_Class","")
```

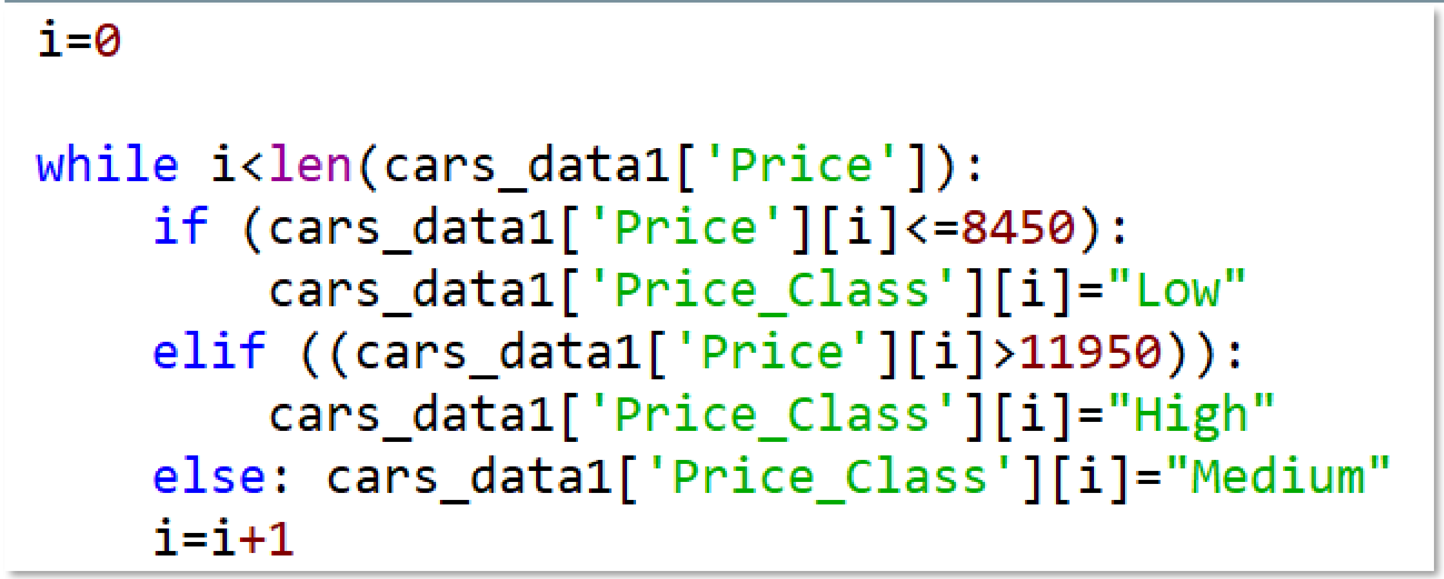
Example: if else and for loops



```
for i in range(0,len(cars_data1['Price']),1):
    if (cars_data1['Price'][i]<=8450):
        cars_data1['Price_Class'][i]="Low"
    elif ((cars_data1['Price'][i]>11950)):
        cars_data1['Price_Class'][i]="High"
    else: cars_data1['Price_Class'][i]="Medium"
```

- A for loop is implemented and the observations are separated into three categories:
  - Price
    - up to 8450
    - between 8450 and 11950
    - greater than 11950
- The classes have been stored in a new column 'Price Class'

Example: while loop



```
i=0

while i<len(cars_data1['Price']):
    if (cars_data1['Price'][i]<=8450):
        cars_data1['Price_Class'][i]="Low"
    elif ((cars_data1['Price'][i]>11950)):
        cars_data1['Price_Class'][i]="High"
    else: cars_data1['Price_Class'][i]="Medium"
    i=i+1
```

- A while loop is used whenever you want to execute statements until a specific condition is violated
- Here a while loop is used over the length of the column 'Price_Class' and an if else loop is used to bin the values and store it as classes

Example: while loop

- `Series.value_counts()` returns series containing count of unique values

```
cars_data1['Price_Class'].value_counts()
```

```
Out[14]:
Medium     751
Low        369
High       316
Name: Price_Class, dtype: int64
```

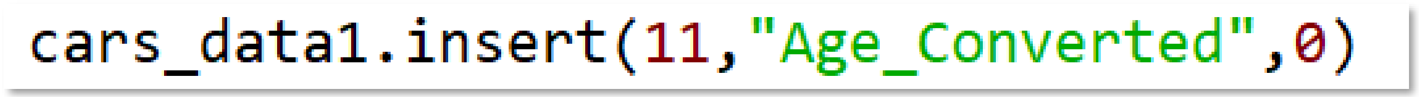
Functions in Python

- A function accepts input arguments and produces an output by executing valid commands present in the function
- Function name and file names need not be the same
- A file can have one or more function definitions
- Functions are created using the command def and a colon with the statements to be executed indented as a block
- Since statements are not demarcated explicitly, It is essential to follow correct indentation practises

```
def function_name(parameters):
      statements
```

Example: functions

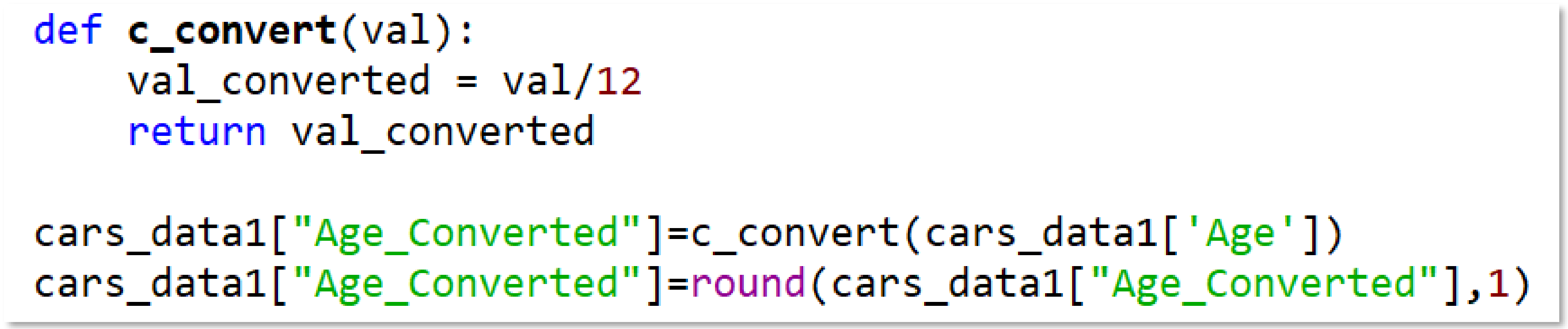
- Converting the Age variable from months to years by defining a function
- The converted values will be stored in a new column, ‘Age_Converted’
- Hence, inserting a new column



```
cars_data1.insert(11,"Age_Converted",0)
```

Example: functions

- Here, a function c_convert has been defined
- The function takes arguments and returns one value



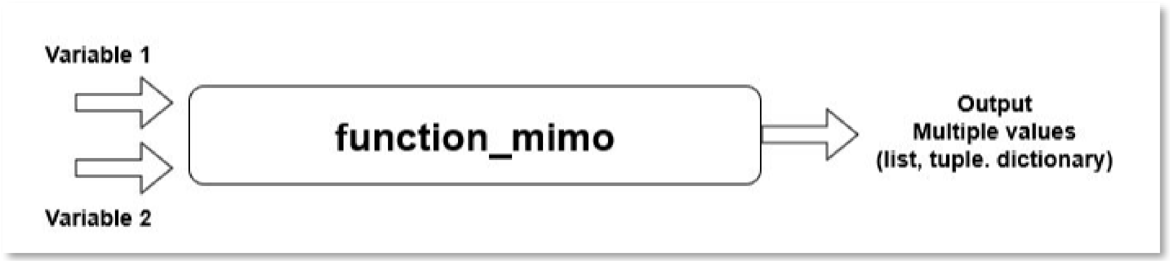
```python
def c_convert(val):
    val_converted = val/12
    return val_converted

cars_data1["Age_Converted"]=c_convert(cars_data1['Age'])
cars_data1["Age_Converted"]=round(cars_data1["Age_Converted"],1)
```

Function with multiple inputs and outputs

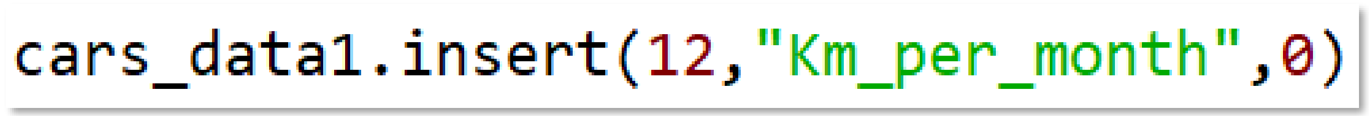
Function with multiple inputs and outputs

- Functions in Python takes multiple input objects but return only one object as output
- However lists, tuples or dictionaries can be used to return multiple outputs as required



Example: function with multiple inputs and outputs

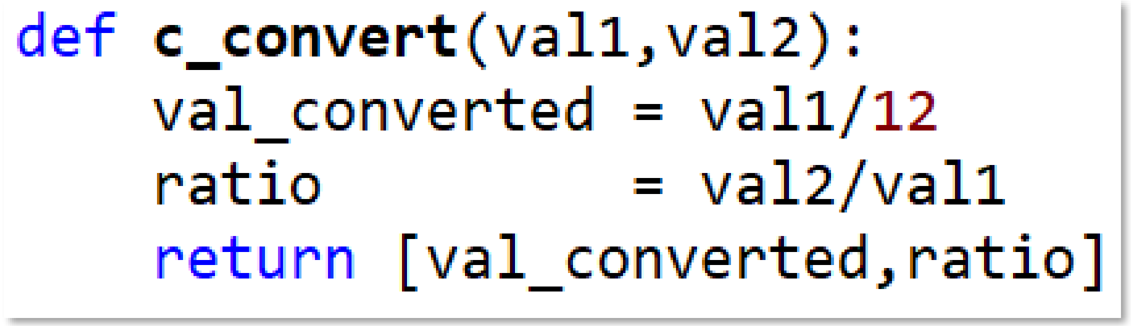
- Converting the Age variable from months to years and getting kilometers (KM) run per month
- The converted values of kilometer will be stored in a new column, 'km_per_month'
- Hence, inserting a new column



```
cars_data1.insert(12,"Km_per_month",0)
```

Example: function with multiple inputs and outputs

- A multiple input multiple output function c_convert has been defined
- The function takes in two inputs
- The output is returned in the form of a list

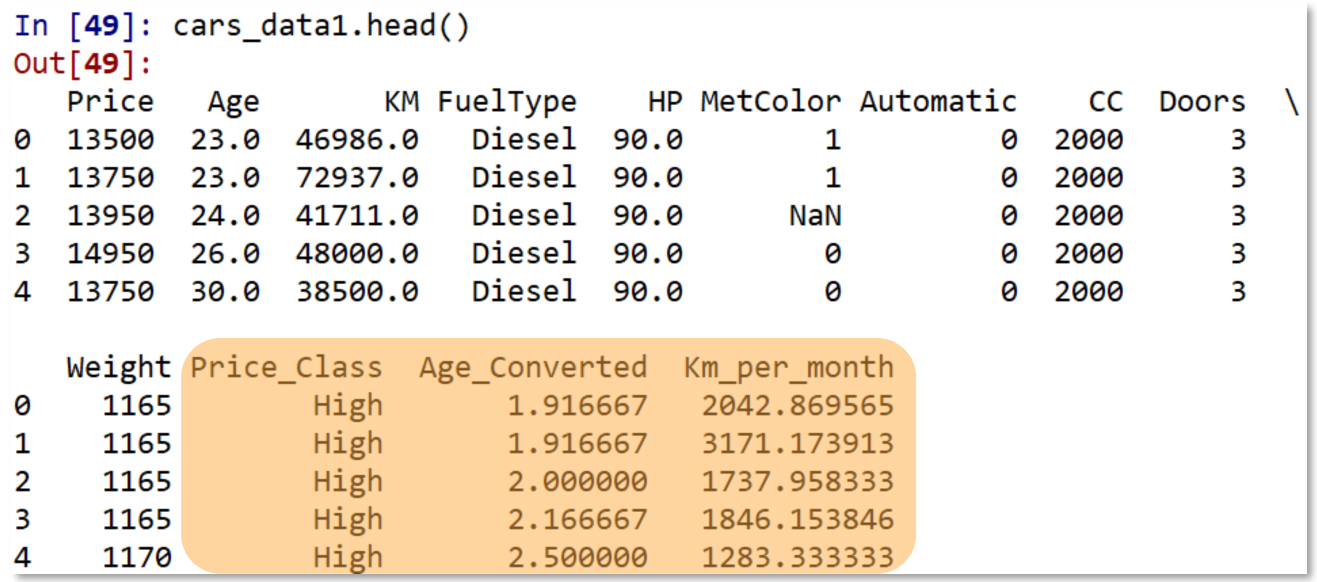


```
def c_convert(val1,val2):
    val_converted = val1/12
    ratio         = val2/val1
    return [val_converted,ratio]
```

Example: function with multiple inputs and outputs

- Here, Age and KM columns of the data set are input to the function
- The outputs are assigned to 'Age_Converted' and 'km_per_month'

```
cars_data1["Age_Converted"],cars_data1["Km_per_month"] = \
c_convert(cars_data1['Age'],cars_data1['KM'])
```



```
In [49]: cars_data1.head()
Out[49]:
   Price   Age       KM FuelType    HP MetColor Automatic    CC  Doors  \
0  13500  23.0  46986.0   Diesel  90.0        1         0  2000      3
1  13750  23.0  72937.0   Diesel  90.0        1         0  2000      3
2  13950  24.0  41711.0   Diesel  90.0      NaN         0  2000      3
3  14950  26.0  48000.0   Diesel  90.0        0         0  2000      3
4  13750  30.0  38500.0   Diesel  90.0        0         0  2000      3

   Weight Price_Class  Age_Converted  Km_per_month
0    1165        High       1.916667   2042.869565
1    1165        High       1.916667   3171.173913
2    1165        High       2.000000   1737.958333
3    1165        High       2.166667   1846.153846
4    1170        High       2.500000   1283.333333
```

Summary

- Control structures
    - If elif family
    - For
    - While
- Functions

```
rror_object
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
_operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
_operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

selection at the end -add
_ob.select= 1
ier_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
irror_ob.select = 0
= bpy.context.selected_ob
ata.objects[one.name].se

rint("please select exactly
```

THANK YOU