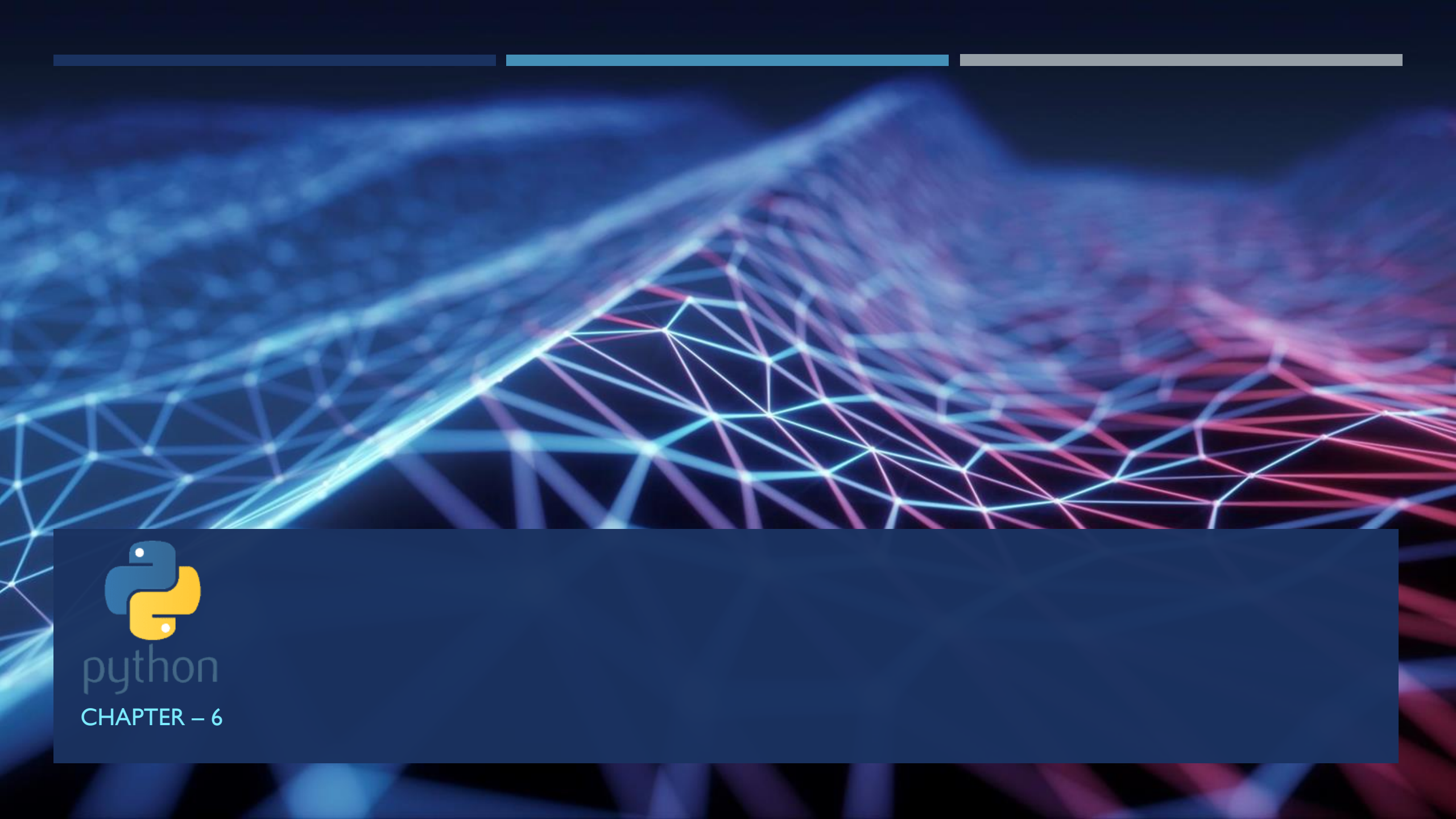


CHAPTER – 6

LOOPS IN PYTHON

LOOPS IN PYTHON

Python programming language provides following types of loops to handle looping requirements.

Python provides Two ways for executing loops:

- While Loop
- For Loop

While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

WHILE LOOP

- The while loop tells the computer to do something as long as the condition is met.
- Its construct consists of a block of code and a condition.
- The condition is evaluated, and if the condition is true, the code within the block is executed.
- The code is executed over and over again, as long as the condition is True.

Syntax:

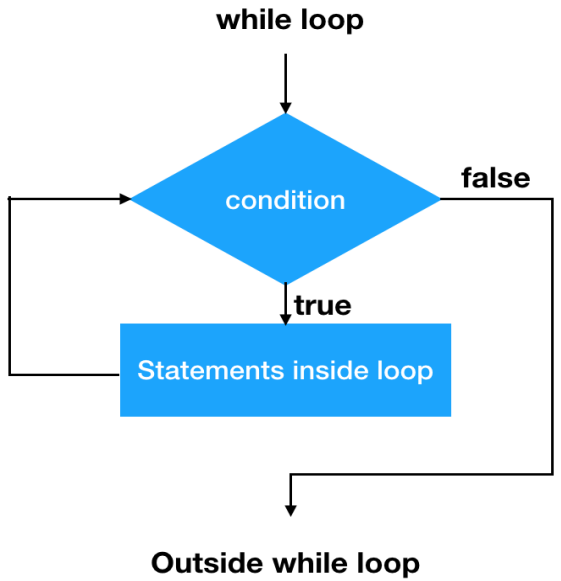
```
while condition :
    expression
```

Example

```
#Print x as long as x is less than 6
x = 1
while x < 6:
    print(x)
    x = x + 1
```

Out:

```
1
2
3
4
5
```



BASIC WHILE LOOP

Example

Creating an error variable that equals to 50.0, then divide it by 5 and print the variable as long as the it is greater than 1.

```
#variable, initially
error = 50.
#while loop condition
while error > 1 :
#divide error by 5
    error = error / 5
#print error
    print(error)
```

Out:

```
10.0
2.0
0.4
```

TASKS

Create a variable offset with an initial value of 8. Code a while loop that keeps running as long as the offset is not equal to 0. Print out the sentence "correcting...".

ANSWER

```
#creating a variable offset
offset=8
#adding a while loop
while offset!=0:
    print('correcting…')
    offset=offset-1
print(offset)
```

IF & ELSE WITHIN WHILE LOOP

Using an if & else statements within a while loop.

Example

Creating an error variable that equals to 40, if error equals to 10 divide error by 2, else divide error by 4 as long as error is greater than 0.

```
#variable, initially
error = 40
#while loop condition
while error > 1 :
 if error==10:
  error = error / 2
 else:
  error=error/4
#print error
 print(error)
```

Out
```
10.0
5.0
1.25
0.3125
```

TASKS

In the last task we created a variable offset, the while loop that corrects the offset is a good start, but what if the offset is negative?

Write a code where offset is initialized to -6.

ANSWER

```
# Initialize offset
offset = -6
# Code the while loop
while offset != 0 :
    print("correcting...")
    offset = offset + 1
    print(offset)
```

TASK- FINDINGS

This while loop will never stop running, because offset will be further decreased on every run.

offset != 0 will never become False and the while loop continues forever.

TASKS

Fix things by putting an if-else statement inside the while loop.

If offset is greater than zero, you should decrease offset by 1. Else, you should increase offset by 1.

Note: If your code is still taking too long to run (or your session is expiring), you probably made a mistake. Check your code and make sure that the statement offset != 0 will eventually evaluate to FALSE!

ANSWER

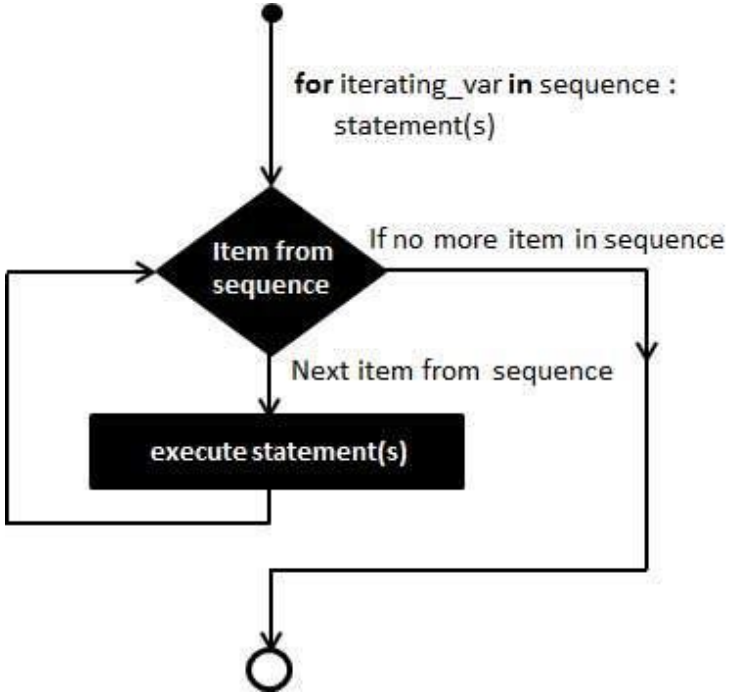
```
# Initialize offset
offset = -6
# Coding the while loop
while offset != 0 :
 print("correcting...")
 if offset>0 :
  offset=offset-1
 else :
  offset=offset+1
 print(offset)
```

FOR LOOP

- The for statement is used to iterate over the elements of a sequence.
- It's used when you have a piece of code which you want to repeat n number of time.
- The for loop is often distinguished by an explicit loop counter or loop variable.
- For loops are used for sequential traversal. For example: traversing a list or string or array etc.

Syntax:

```
for iterator_var in sequence:
    statements(s)
```



LOOP OVER A LIST - EXAMPLE

Using a for loop that iterates over all the values in height of the fam list and prints out every element.

```
#creating a list
fam = [1.73, 1.68, 1.71, 1.89]
#code the for loop
for height in fam :
    print(height)
```

Out:

```
1.73
1.68
1.71
1.89
```

TASKS

Write a for loop that iterates over all elements of the areas list and prints out every element separately.

Use the following code to solve this task:

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]
```

ANSWER

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]
# Code the for loop
for area in areas:
        print(area)
```

LOOP OVER INDEXES AND VALUES

Using a for loop to iterate over a list only gives you access to every list element in each run, one after the other.

If you also want to access the index information where the list element you're iterating over is located, you can use enumerate().

Example: have a look at how the for loop can be converted:

```
fam = [1.73, 1.68, 1.71, 1.89]
for index, height in enumerate(fam) :
    print("person " + str(index) + ": " + str(height))
```

Out:

```
person 0: 1.73
person 1: 1.68
person 2: 1.71
person 3: 1.89
```

TASKS

Using a for loop print the detail of the areas list so that your result looks like "room x: y". Where x is the index of the list element & y is the area.

Use the following code to solve this task:

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]
```

ANSWER

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]
# Change for loop to use enumerate() and update print()
for index, a in enumerate(areas) :
    print("room"+str(index)+":"+str(a))
```

TASKS

Use the print() function in the for loop so that the first printout becomes "room 1: 11.25", the second one "room 2: 18.0" and so on.

ANSWER

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]
# Code the for loop
for index, area in enumerate(areas) :
    print("room " + str(index+1) + ": " + str(area))
```

LOOP OVER DICTIONARY

We discussed dictionaries in the previous chapters your remember a dictionary holds key:value pair.

Iterate through all key, value pairs in a dictionary:

```
# pairs in a dictionary
statesandcapitals={'Alabama':'Montgomery','Alaska':'Juneau','Arizona':'Phoenix','Arkansas': 'Little Rock'}
# Iterating over values
for state, capital in statesandcapitals.items():
 print(state, ":", capital)
```

Out:

```
Alabama : Montgomery
Alaska : Juneau
Arizona : Phoenix
Arkansas : Little Rock
```

TASK

Write a for loop that goes through each key:value pair of europe.

On each iteration, "the capital of x is y" should be printed out, where x is the key and y is the value of the pair.

Use the europe dictionary to complete this task.

```
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo', 'italy':'rome', 'poland':'warsaw', 'austria':'vienna'}
```

ANSWER

```
# europe dictionary
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo', 'italy':'rome', 'poland':'warsaw', 'austria':'vienna'}
# Building a for loop
for country, capital in europe.items():
    print("The capital of " + country, "is", capital)
```

LOOP OVER A 1D NUMPY ARRAY

If you're dealing with a 1D Numpy array, looping over all elements can be as simple as:

```
#importing numpy
import numpy  as np
#creating an array
array=[0, 1,  2,  3  ,4,  5,  6,  7,  8,  9,  10,  11]
#adding a for loop
np_array=np.array(array)
for x in np_array:
    print(x)
```

Out:

```
0
1
2
3
4
5
6
7
8
9
10
11
```

LOOP OVER A 2D NUMPY ARRAY

- If you're dealing with a 2D Numpy array, it's more complicated. A 2D array is built up of multiple 1D arrays.

- To explicitly iterate over all separate elements of a multi-dimensional array, you'll need this

syntax: for x in np.nditer(my_array) :

LOOP OVER A 2D NUMPY ARRAY

Let's do an example,

```
#creating an 2D array
array=[[0, 1], [2, 3] ,[4, 5], [6, 7], [8, 9], [10, 11]]
#using a for loop
np_array=np.array(array)
for x in np.nditer(np_array) :
    print(x)
```

Out:

```
0
1
2
3
4
5
6
7
8
9
10
11
```

TASK BACKGROUND

Two Numpy arrays that you might recognize from the previous classes:

- np_height, a Numpy array containing the heights of Major League Baseball players
- np_baseball, a 2D Numpy array that contains both the heights (first column) and weights (second column) of those players.

This data is loaded onto github

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/baseball_height_weight.csv

TASKS

Write a for loop that iterates over all elements in np_height and prints out "x inches" for each element, where x is the value in the array. Use the following data to create a 1D Numpy array from the height column

#baseball data

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/baseball_height_weight.csv

ANSWER

```
#importing data
import pandas as pd
import numpy as np
array=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/baseball_height_weight.csv")
#creating a for loop
np_array=np.array(array)
height=np_array[:,0]
for x in height:
        print(str(x)+' inches')
```

TASKS

Write a for loop that visits every element of the np_baseball array and prints it out.

ANSWER

```
# Import numpy as np
import numpy as np
import numpy as np
array=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/baseball_height_weight.csv")
#For loop over np_baseball
np_baseball = np.array(array)
for x in np.nditer(np_baseball):
        print(x)
```

LOOP OVER A DATAFRAME

Iterating over a Pandas DataFrame is typically done with the iterrows() method.

Used in a for loop, every observation is iterated over and on every iteration the row label and actual row contents are available:

Syntax:

```
for lab, row in dataframe.iterrows() :
        print(lab,row)
```

Example for loop on NBA dataset

```
# importing pandas module
import pandas as pd
# making data frame from csv file
data = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv")
 #Iterating over dataframe
for index, row in data.iterrows():
    print(index, row)
```

TASKS

Write a for loop that iterates over the rows on the cars dataset and print out the row label and the rows contents.

#cars data

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv

ANSWER

```
import pandas as pd
# Import cars data
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv', index_col = 0)
# Iterate over rows of cars
for lab,row in cars.iterrows():
    print(lab,row)
```

LOOP OVER A DATAFRAME

The row data that's generated by iterrows() on every run is a Pandas Series.

This format is not very convenient to print out. Luckily, you can easily select variables from the Pandas Series using square brackets:

Syntax: for index, row in dataframe.iterrows() :

print(row['column'])

Example name & ages of NBA players

```
# importing pandas
import pandas as pd
# making data frame from csv file
data = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv", index_col = 0)
# Iterate over rows of cars
for index, row in data.iterrows():
#print out player name with there age
    print(index + ": " + str(row['Age']))
```

Out:

```
Avery Bradley: 25.0
Jae Crowder: 25.0
John Holland: 27.0
R.J. Hunter: 22.0
Jonas Jerebko: 29.0
Amir Johnson: 29.0
Jordan Mickey: 21.0
```

TASKS

In the cars dataset using the iterators index and row, adapt the code in the for loop such that the first iteration prints out "US: 809", the second iteration "AUS: 731", and so on.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv', index_col = 0)
# Adapt for loop
for lab, row in cars.iterrows() :
    print(lab + ": " , str(row['cars_per_cap']))
```

LOOP - ADDING A NEW COLUMN

You can add a new column for length of the country names of the cars DataFrame by using the for loop.

```
#adding a for loop
for lab, row in cars.iterrows() :
    cars.loc[lab, "name_length"] = len(row["country"])
print(cars)
```

```
     cars_per_cap        country  drives_right  name_length
US            809  United States          True         13.0
AUS           731      Australia         False          9.0
JAP           588          Japan         False          5.0
IN             18          India         False          5.0
RU            200         Russia          True          6.0
MOR            70        Morocco          True          7.0
EG             45          Egypt          True          5.0
```

TASKS

Add a new column, named “COUNTRY”, that contains a uppercase version of the country names in the "country" column.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv', index_col = 0)
# Code for loop that adds COUNTRY column
for lab, row in cars.iterrows():
    cars.loc[lab, 'COUNTRY'] = row['country'].upper()
# Print cars
print(cars)
```

TASKS

Using a for loop add a new column, named “Drives”, that contains the values drives “right” or “left”.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv', index_col = 0)
# Code for loop that adds drives column
result = []
for value in cars['drives_right']:
    if value == 1:
        result.append("right")
    else:
        result.append("left")
cars["Drives"] = result
# Print cars
print(cars)
```

Q&A

python

THANK YOU