Loops

Wednesday, October 7, 2020 2:41 PM

Loops

Datacamp

While Loops
Repeating an answer until a condition is met

```
while condition :
    expression
```

while_loop.py

```
error = 50.0
#     12.5
while error > 1:    # True
    error = error / 4
    print(error)
```

- When the condition is false, the code moves on

For Loop
Iterates through a variable in a sequence and ends when the loop reaches the end of the sequence

- Sequence: list, strings, dictionaries, pandas df

```
for var in seq :
    expression
```

family.py

```
fam = [1.73, 1.68, 1.71, 1.89]
for height in fam :
    print(height)
```

- Each item of the list seq is iterated over
- Stored into the variable "height"

You can use enumerate can produce two variable of the iteration: index value and the item

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

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

Loop Data Structure
To read through a dictionary, use the method .items() will generate a key and value for each iteration

```
for var in seq :
    expression
```

dictloop.py

```
world = { "afghanistan":30.55,
          "albania":2.77,
          "algeria":39.21 }
for k, v in world.items() :
    print(k + " -- " + str(v))
```

```
algeria -- 39.21
afghanistan -- 30.55
albania -- 2.77
```

2D Numpy Arrays
A for-loop will print out every series in the array

```
import numpy as np
np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])
np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])
meas = np.array([np_height, np_weight])
for val in meas :
    print(val)
```

```
[ 1.73  1.68  1.71  1.89  1.79]
[ 65.4  59.2  63.6  88.4  68.7]
```

To get every element in each array, you can use a numpy function np.nditer

```
import numpy as np
np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])
np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])
meas = np.array([np_height, np_weight])
for val in np.nditer(meas) :
    print(val)
```

```
1.73
1.68
1.71
1.89
1.79
65.4
...
```

Jupyter Notebook

Printing out a list of lists: i.e. "the hallway is 11.25 sqm"

```
# house list of lists
house = [["hallway", 11.25],
         ["kitchen", 18.0],
         ["living room", 20.0],
         ["bedroom", 10.75],
         ["bathroom", 9.50]]

# Build a for loop from scratch
for elements in house:
    for index, var in enumerate(elements):
        if index == 0:
            room = var
        elif index == 1 :
            sqm = var
            print("the " + str(room) + " is " + str(sqm) + " sqm")
```

Using a for loop to add a new column at the end of every row

```
1  # Import cars data
2  import pandas as pd
3  cars = pd.read_csv('cars.csv', index_col = 0)
4
5  # Code for loop that adds COUNTRY column
6  for lab, row in cars.iterrows():
7      cars.loc[lab, "COUNTRY"] = row['country'].upper()
8
9
10 # Print cars
11 print(cars)
```

For loops are less effective than indexing into the column you want to add using the apply method:

```
1  # Import cars data
2  import pandas as pd
3  cars = pd.read_csv('cars.csv', index_col = 0)
4
5  # Use a for loop to add a new entry
6  for lab, row in cars.iterrows() :
7      cars.loc[lab, "COUNTRY"] = row["country"].upper()
8
9  # Use .apply(str.upper)
10 cars["COUNTRY"] = cars["country"].apply(str.upper)
11
12 print(cars)
```

Dataframes

Select the label and the row item using iterrows()

```
import pandas as pd
brics = pd.read_csv("brics.csv", index_col = 0)
```

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

```
BR
country        Brazil
capital       Brasilia
area             8.516
population       200.4
Name: BR, dtype: object
...
RU
country       Russia
capital       Moscow
area            17.1
population     143.5
Name: RU, dtype: object
```

- Used in a for loop, every observation is iterated over and on every iteration the row label and actual row contents are available
  - Sort of like enumerate
- If you print out "row", it appears to be in dictionary form, giving you the column name + value

```
import pandas as pd
brics = pd.read_csv("brics.csv", index_col = 0)
brics["name_length"] = brics["country"].apply(len)
print(brics)
```

```
         country    capital    area  population
BR        Brazil   Brasilia   8.516      200.40
RU        Russia     Moscow  17.100      143.50
IN         India  New Delhi   3.286     1252.00
CH         China    Beijing   9.597     1357.00
SA  South Africa   Pretoria   1.221       52.98
```

- Us apply to conduct vectorized operations