Practice Lab: Selecting data in a Dataframe

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Use Pandas Library to create DataFrame and Series
- Locate data in the DataFrame using loc() and iloc() functions
- Use slicing

Exercise 1: Pandas: DataFrame and Series

Pandas is a popular library for data analysis built on top of the Python programming language. Pandas generally provide two data structures for manipulating data, They are:

- DataFrame
- Series

A **DataFrame** is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

- A Pandas DataFrame will be created by loading the datasets from existing storage.
- Storage can be SQL Database, CSV file, Excel file, etc.
- It can also be created from the lists, dictionaries, and from a list of dictionaries.

Series represents a one-dimensional array of indexed data. It has two main components :

1. An array of actual data.
2. An associated array of indexes or data labels.

The index is used to access individual data values. You can also get a column of a dataframe as a **Series**. You can think of a Pandas series as a 1-D dataframe.

```
# let us import the Pandas Library
import pandas as pd
```

Once you've imported pandas, you can then use the functions built in it to create and analyze data.

In this practice lab, we will learn how to create a DataFrame out of a dictionary.

Let us consider a dictionary 'x' with keys and values as shown below.

We then create a dataframe from the dictionary using the function pd.DataFrame(dict)

```
#Define a dictionary 'x'

x = {'Name': ['Rose','John', 'Jane', 'Mary'], 'ID': [1, 2, 3, 4],
'Department': ['Architect Group', 'Software Group', 'Design Team',
'Infrastructure'],
      'Salary':[100000, 80000, 50000, 60000]}

#casting the dictionary to a DataFrame
df = pd.DataFrame(x)

#display the result df
df
```

We can see the direct correspondence between the table. The keys correspond to the column labels and the values or lists correspond to the rows.

Column Selection:

To select a column in Pandas DataFrame, we can either access the columns by calling them by their columns name.

Let's Retrieve the data present in the ID column.

```
#Retrieving the "ID" column and assigning it to a variable x
x = df[['ID']]
x
```

Let's use the type() function and check the type of the variable.

```
#check the type of x
type(x)
```

The output shows us that the type of the variable is a DataFrame object.

Access to multiple columns

Let us retrieve the data for Department, Salary and ID columns

```
#Retrieving the Department, Salary and ID columns and assigning it to
a variable z

z = df[['Department','Salary','ID']]
z
```

Try it yourself

Problem 1: Create a dataframe to display the result as below:

```
#write your code here
```

Problem 2: Retrieve the Marks column and assign it to a variable b

```
#write your code here
```

Problem 3: Retrieve the Country and Course columns and assign it to a variable c

```
#write your code here
```

To view the column as a series, just use one bracket:

```
# Get the Student column as a series Object

x = df1['Student']
x

#check the type of x
type(x)
```

The output shows us that the type of the variable is a Series object.

Exercise 2: loc() and iloc() functions

loc() is a label-based data selecting method which means that we have to pass the name of the row or column that we want to select. This method includes the last element of the range passed in it.

Simple syntax for your understanding:

- loc[row_label, column_label]

iloc() is an indexed-based selecting method which means that we have to pass an integer index in the method to select a specific row/column. This method does not include the last element of the range passed in it.

Simple syntax for your understanding:

- iloc[row_index, column_index]

```
# Access the value on the first row and the first column

df.iloc[0, 0]

# Access the value on the first row and the third column

df.iloc[0,2]

# Access the column using the name

df.loc[0, 'Salary']
```

Let us create a new dataframe called 'df2' and assign 'df' to it. Now, let us set the "Name" column as an index column using the method set_index().

```
df2=df
df2=df2.set_index("Name")

#To display the first 5 rows of new dataframe
df2.head()

#Now, let us access the column using the name
df2.loc['Jane', 'Salary']
```

Try it yourself

Use the loc() function,to get the Department of Jane in the newly created dataframe df2.

```
#write your code here
```

Use the iloc() function to get the Salary of Mary in the newly created dataframe df2.

```
#write your code here
```

Exercise 3: Slicing

Slicing uses the [] operator to select a set of rows and/or columns from a DataFrame.

To slice out a set of rows, you use this syntax: data[start:stop],

here the start represents the index from where to consider, and stop represents the index one step BEYOND the row you want to select. You can perform slicing using both the index and the name of the column.

NOTE: When slicing in pandas, the start bound is included in the output.

So if you want to select rows 0, 1, and 2 your code would look like this: df.iloc[0:3].

It means you are telling Python to start at index 0 and select rows 0, 1, 2 up to but not including 3.

NOTE: Labels must be found in the DataFrame or you will get a KeyError.

Indexing by labels(i.e. using loc()) differs from indexing by integers (i.e. using iloc()). With loc(), both the start bound and the stop bound are inclusive. When using loc(), integers can be used, but the integers refer to the index label and not the position.

For example, using loc() and select 1:4 will get a different result than using iloc() to select rows 1:4.

We can also select a specific data value using a row and column location within the DataFrame and iloc indexing.

```
# let us do the slicing using old dataframe df

df.iloc[0:2, 0:3]
```

```
#let us do the slicing using loc() function on old dataframe df where index column is having labels as 0,1,2
df.loc[0:2,'ID':'Department']

#let us do the slicing using loc() function on new dataframe df2 where index column is Name having labels: Rose, John and Jane
df2.loc['Rose':'Jane', 'ID':'Department']
```

using loc() function, do slicing on old dataframe df to retrieve the Name, ID and department of index column having labels as 2,3

```
# Write your code below and press Shift+Enter to execute
```

Author(s):

Appalabhaktula Hema

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-03-31	0.1	Appalabhaktula Hema	Created initial version

© IBM Corporation 2022. All rights reserved.