R to Python

Prithviraj Lakkakula

7/25/2022

Contents

Basics of Programming **1**

Creating a dataframe . . . 1
Imports . . . 2
Data Types . . . 3
Basic Math . . . 4
Comparisons and Boolean Operations . . . 6
Conditional Statements . . . 8
Lists . . . 9
Importing data from a variety of data formats . . . 12
How to call heads and tails? . . . 12
How to know the shape/length of the data? . . . 12
How to access single and multiple columns? . . . 12
Get summary stats of the data variables . . . 12
How to graph plots (scatter plot, histogram, boxplot, barplot etc.)? . . . 12
Handling missing values? . . . 12
Handling outliers? . . . 12
Checking class imbalance? . . . 12
Splitting the dataset into training and test sets for both cross section and time series data cases? . . . 12
Run a linear regression model along with predictions on test set, model evaluation, performance metrics? . . . 12
Run a logistic regression model along with predictions on test set, model evaluation, performance metrics? . . . 12

Basics of Programming

Creating a dataframe

R code

```r
# construct a dataframe
df <- data.frame(a = c(1, 2, 3, 4, 5),
                 b = c(3, 4, 5, 6, 7))
df
```

```
##   a b
## 1 1 3
## 2 2 4
## 3 3 5
## 4 4 6
## 5 5 7
```

Python code

```python
import pandas as pd
df = pd.DataFrame({'a': [1, 2, 3, 4, 5],
                           'b': [3, 4, 5, 6, 7]})
print(df)
```

```
##    a  b
## 0  1  3
## 1  2  4
## 2  3  5
## 3  4  6
## 4  5  7
```

Imports

R code

```r
sqrt(36)
```

```
## [1] 6
```

Python code

```python
import math
math.sqrt(36)

#importing a function
```

```
## 6.0
```

```python
from math import sqrt
sqrt(36)

# show all functions in math model
```

```
## 6.0
```

```
print(dir(math))
```

```
## ['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin'
```

Data Types

R code

```
class(5)
```

```
## [1] "numeric"
```

```
class(5.0)
```

```
## [1] "numeric"
```

```
class('Five')
```

```
## [1] "character"
```

```
class(FALSE)
```

```
## [1] "logical"
```

```
## check if an object is of a given type
is.numeric(5)
```

```
## [1] TRUE
```

```
is.character('Five')
```

```
## [1] TRUE
```

```
is.logical(FALSE)
```

```
## [1] TRUE
```

```
## Convert an object to a given type
as.character(5.5)
```

```
## [1] "5.5"
```

```
#as.numeric, as.factor etc to convert to numeric and factor types
```

Python code

```
type(5)
```

```
## <class 'int'>
```

```
type(5.0)
```

```
## <class 'float'>
```

```
type('five')
```

```
## <class 'str'>
```

```
type(False)

## check if an object is of a given type
```

```
## <class 'bool'>
```

```
isinstance(5.0, int)
```

```
## False
```

```
isinstance(5.0, (int, float))
```

```
## True
```

```
isinstance('Five', str)
```

```
## True
```

```
isinstance('Five', int)
```

```
## False
```

```
isinstance(False, bool)

## Convert an object to a given type
```

```
## True
```

```
str(5.5)
```

```
## '5.5'
```

Basic Math

R code

```r
5 + 4 #addition
```

```
## [1] 9
```

```r
5 - 4 #subtraction
```

```
## [1] 1
```

```r
5 * 4 #multiplication
```

```
## [1] 20
```

```r
5 ^ 4 #exponent
```

```
## [1] 625
```

```r
5 %% 4 # divides and remainder is the output. modulo
```

```
## [1] 1
```

```r
100 / 4 #division
```

```
## [1] 25
```

```r
floor(100 / 3) #returns floor
```

```
## [1] 33
```

```r
ceiling(100 / 3)
```

```
## [1] 34
```

Python code

```
import math
5 + 4 #addition
```

```
## 9
```

```
5 - 4 #subtraction
```

```
## 1
```

```
5 * 4 #multiplication
```

```
## 20
```

```
5 ** 4 #exponent
```

```
## 625
```

```
5 % 4 # divides and remainder is the output. modulo
```

```
## 1
```

```
100 / 4 #division
```

```
## 25.0
```

```
100 // 3 #returns floor #math.floor(100 / 3) also returns the same
```

```
## 33
```

```
math.ceil(100 / 3)
```

```
## 34
```

Comparisons and Boolean Operations

R code

```
## Assignment statement
x <- 10

## comparisons
x > 10
```

```
## [1] FALSE
```

```
x >= 10
```

```
## [1] TRUE
```

```
x != 10
```

```
## [1] FALSE
```

```
x == 7
```

```
## [1] FALSE
```

```
## Boolean operations
6 > 4 & 5 > 4
```

```
## [1] TRUE
```

```
6 > 4 | 5 > 4
```

```
## [1] TRUE
```

```
!FALSE
```

```
## [1] TRUE
```

```
FALSE | !FALSE & TRUE #evaluation order: ! (not), & (and), | (or)
```

```
## [1] TRUE
```

Python code

```
## Assignment statement
x = 10

## comparisons
x > 10
```

```
## False
```

```
x >= 10
```

```
## True
```

```
x != 10
```

```
## False
```

```
x == 7

## Boolean operations
```

```
## False
```

```
6 > 4 and 5 > 4
```

```
## True
```

```
6 > 4 or 5 > 4
```

```
## True
```

```
not False
```

```
## True
```

```
False or not False and True #evaluation order: ! (not), & (and), | (or)
```

```
## True
```

Conditional Statements

R code

```
## if statement
x <- 10
if (x > 0) {#condition
  print("success") #statement
}
```

```
## [1] "success"
```

```
## if/else statement
if (x > 0){
  print("success")} else {
  print("not success")
  }
```

```
## [1] "success"
```

```
## if/elif/else statement
if (x > 0){
  print("success")
} else if (x == 0) {
  print("not success")
} else {
  print("not so success")
}
```

```
## [1] "success"
```

Python code

```
## if statement
x = 10
if x > 0:
  print('success')

## if/else statement
```

```
## success
```

```
if x > 0:
  print('success')
else:
  print('not success')

## if/elif/else statement
```

```
## success
```

```
if x > 0:
  print('success')
elif x == 0:
  print('not success')
else:
  print('not so success')
```

```
## success
```

Lists

R code

```
### empty list
empty_list <- list()
empty_list
```

```
## list()
```

```
## creating a list
create_list <- list("he", "she", "him")
create_list
```

```
## [[1]]
## [1] "he"
##
## [[2]]
## [1] "she"
##
## [[3]]
## [1] "him"
```

```
## print 1st and 3rd element of create list
create_list[[1]] #R follows 1-indexing
```

```
## [1] "he"
```

```
create_list[[3]]
```

```
## [1] "him"
```

```
## length of the list
length(create_list)
```

```
## [1] 3
```

```r
## appending the list

create_list[[4]] <- "her"
create_list
```

```
## [[1]]
## [1] "he"
##
## [[2]]
## [1] "she"
##
## [[3]]
## [1] "him"
##
## [[4]]
## [1] "her"
```

```r
create_list <- append(create_list, c("me", "you"))
create_list
```

```
## [[1]]
## [1] "he"
##
## [[2]]
## [1] "she"
##
## [[3]]
## [1] "him"
##
## [[4]]
## [1] "her"
##
## [[5]]
## [1] "me"
##
## [[6]]
## [1] "you"
```

```r
str(create_list)
```

```
## List of 6
##  $ : chr "he"
##  $ : chr "she"
##  $ : chr "him"
##  $ : chr "her"
##  $ : chr "me"
##  $ : chr "you"
```

Python code

```
## empty list
empty_list = []
empty_list2 = list()
empty_list
```

```
## []
```

```
empty_list2

## create a list
```

```
## []
```

```
create_list = ['he', 'she', 'him']
create_list

## print 1st and 3rd element of create list
```

```
## ['he', 'she', 'him']
```

```
create_list[0] #python follows zero-indexing
```

```
## 'he'
```

```
create_list[2]

## length of a list
```

```
## 'him'
```

```
len(create_list)

## appending the list
```

```
## 3
```

```
create_list.append('her')
create_list.extend(['me', 'you'])
create_list
```

```
## ['he', 'she', 'him', 'her', 'me', 'you']
```

```
type(create_list)
```

```
## <class 'list'>
```

Importing data from a variety of data formats

How to call heads and tails?

How to know the shape/length of the data?

How to access single and multiple columns?

Get summary stats of the data variables

How to graph plots (scatter plot, histogram, boxplot, barplot etc.)?

Handling missing values?

Handling outliers?

Checking class imbalance?

Splitting the dataset into training and test sets for both cross section and time series data cases?

Run a linear regression model along with predictions on test set, model evaluation, performance metrics?

Run a logistic regression model along with predictions on test set, model evaluation, performance metrics?