# define your own function

def adder(x):  
 return x + x  
# map function  
map(fun,iter)  
# lambda function  
mapr = map(lambda x: x + x, mylist)  
print(list(mapr))

# dictionary  
mydict = {“a”:1,”b”:2}  
# adding to dictionary  
mydict[“c”] = 3  
# deleting from dictionary  
del mydict[“b”]

# check if an element is in dictionary  
print(“a” in mydict)  
# sort dictionary  
print(sorted(mydict.items()))  
# data exploratory  
print(dat.head()) # print first few lines with column names  
print(dat.dtypes) # print data types for each column  
print(dat.columns) # print column names  
# select column pandas data frame  
dat[[“height”,”weight”]] or dat.height

# pandas dropna  
mydat.dropna() # drop entries with NAs  
mydat.dropna(axis=”columns”) # drop cols with NAs

mydat.dropna(thresh=2) # drop entries if don’t have at least 2 non-NA values  
mydat.dropna(subset=[“first”,”last”]) # look for NAs in these cols  
# generate basic stats on cols  
print(mydat.describe())  
# plotting functions with matplotlib  
plt.plot(x, y, ‘r+’)  
plt.ylabel(“value”)  
plt.figure(1) # to make different plots  
# making subplots within a figure  
plt.subplot(311)  
plt.subplot(312)  
# add text inside plot  
plt.text(x,y,’$\mu=100,\ \sigma=15$’)  
# change x, y limits  
plt.axis([x1,x2,y1,y2])  
# add grid  
plt.grid(True)  
# print nice formatted output  
from pprint import pprint  
# know class of data  
print(type(x))  
# make a variable global, print outside of definition inside a function  
def foo():  
 global c  
 c = 1  
 return c  
foo()  
print(c)  
# using multiple functions at once in spark  
cleanrddnew = linesrdd.map(lambda x: wordclean(x)).map(lambda x: numclean(x))

# function that takes out anything that is not in the alphabet or numbers

import re  
def wordclean(x): return re.sub(“[^a-zA-Z0-9\s]+”,””,x).lower().strip()

# function takes out numbers  
def numclean(x): re.sub(“[0-9]+”,””,x)

**SPARK**

# create a rdd and do some basic word count  
rdd = spark.sparkContext.textFile(“C:/bible.txt”)  
splitrdd = rdd.flatMap(lambda x: x.split())  
reduced\_rdd = splitrdd.map(lambda x: (x,1)).reduceByKey(lambda x,y: x+y)  
sorted\_rdd = reduced\_rdd.sortBy(lambda x: x[1], ascending=False)  
# apply a function on a RDD  
cleanrdd = rdd.map(lambda x: function(x))