**Assignment 2 for CS 603**

## Mohamed Fazmi Ahamed Fizer

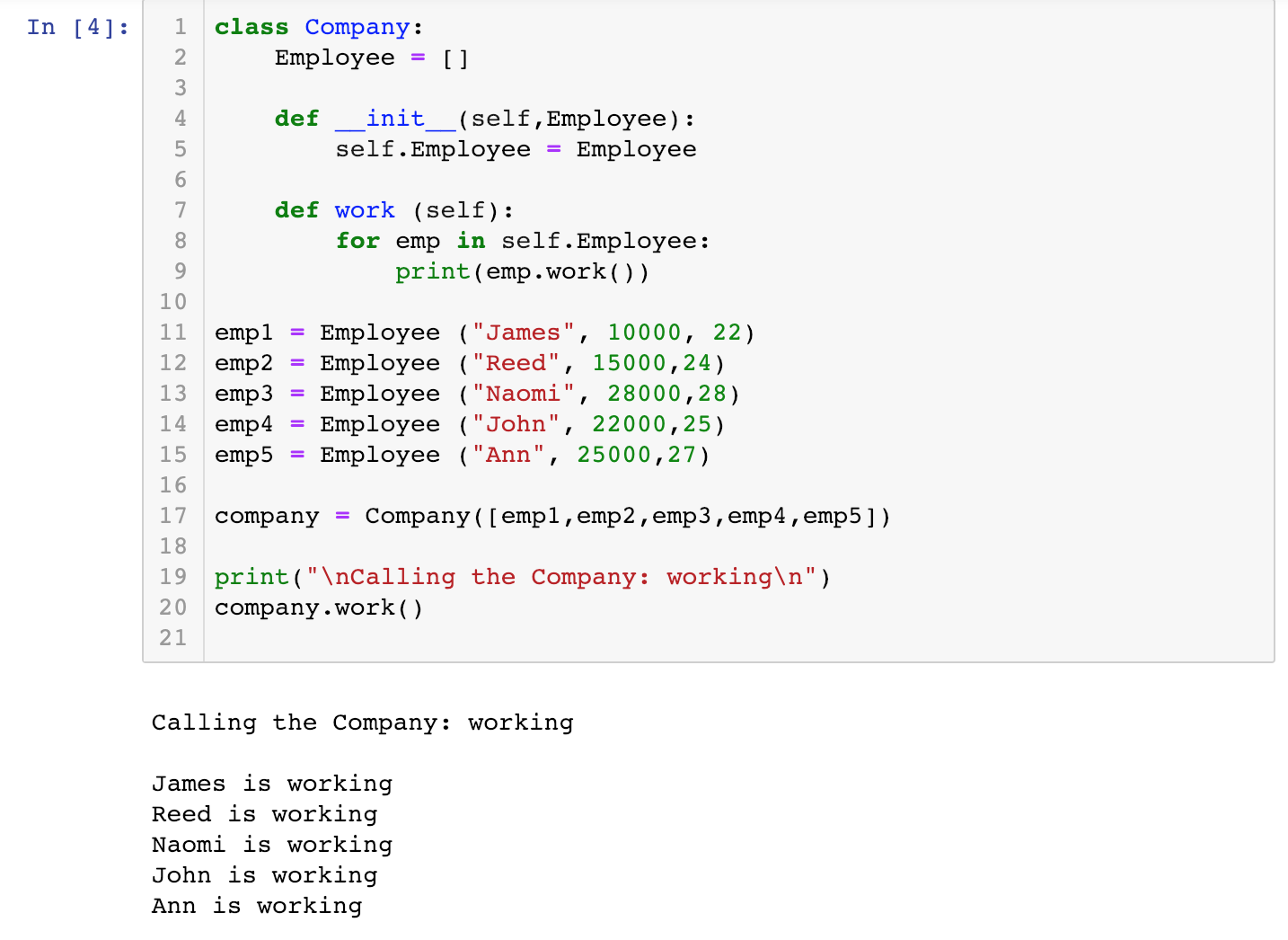
## Hari Charan Kaveri

## Tharinda Embuldeniya

**Using our class example unit2\_07 Employee class, create five different employees who earn different salary and are different age. Then use List comprehension to find the top three salary employees and display all of their information.**

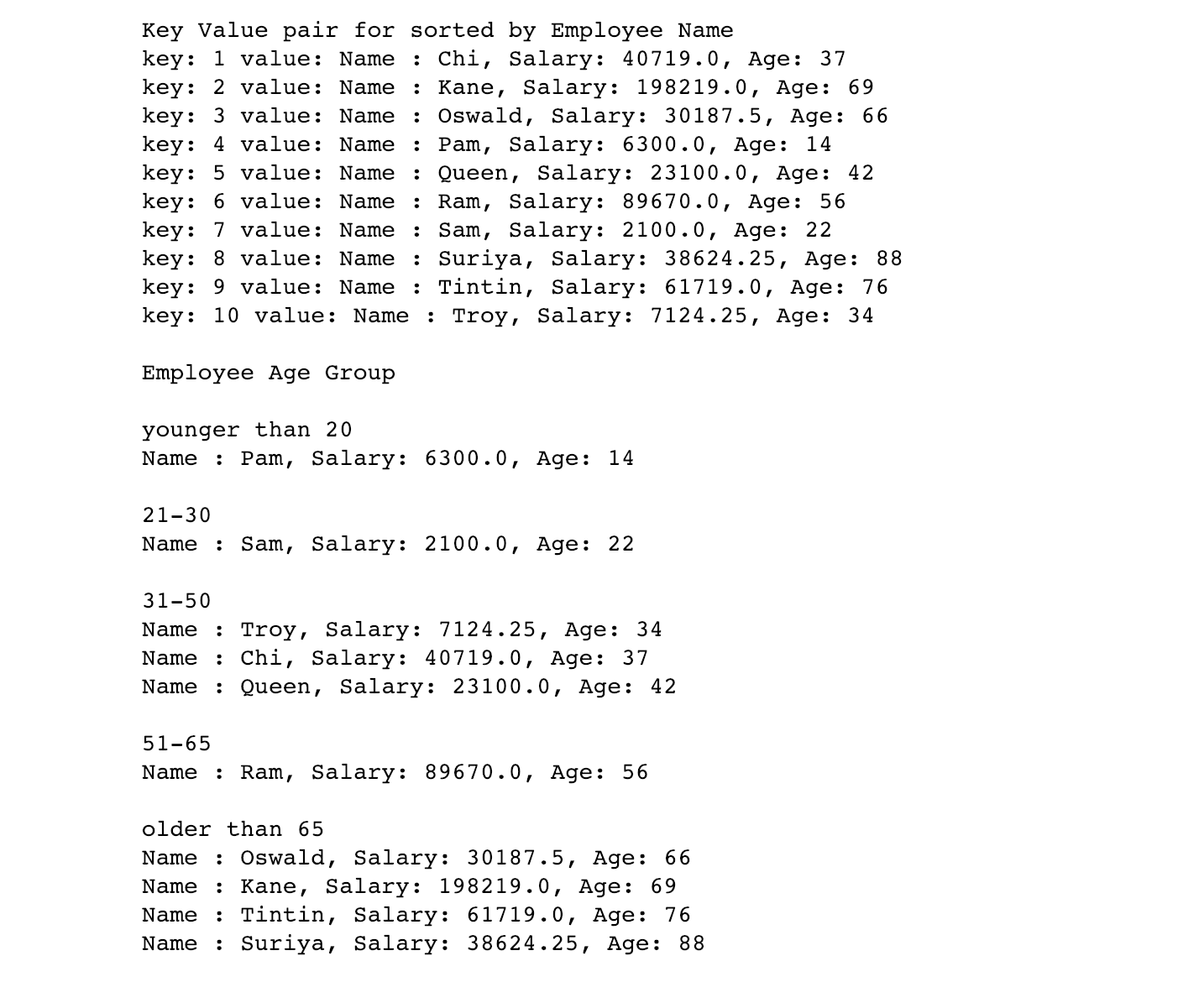


**Create a Company class that holds instances of employees. Then assign five employee instances to an instance of the Company class. Next, add a work() method to both the Company and Employee classes so that when you call the method on the Company class, each Employee instance assigned to the Employee class will work(). Test the Company’s work().**



**Create ten different employees from a file: read employee data from a file(.txt or .json), then use generator to create a list of ten employees, sort the employee list by their name, use generator to raise every employee’s salary by 5%, then use generator to create a dictionary of the employees where the keys of the dictionary items are from range(10) and print them out. Furthermore, use itertools’s groupby() method to group employees by age group(younger than 20, 21-30, 31-50, 51-65, older than 65)**





**Create a EmployeeD class which is a subclass of UserDict and overrides \_\_getitem\_\_ and \_\_setitem\_\_ methods. Then read from your provided file to create a sequence of EmployeeD instances using generator function and print the total salaries.**

****

**Code:**

import json

import itertools as it

class Employee: #same as Employee(object)

'Common base class for all employees'

empCount = 0 #class attribute

#name, salary and \_age are instance attributes

def \_\_init\_\_(self, name, salary, age): #initializer must have at least argument beside self

self.name = name

self.salary = salary

self.\_\_age = age

Employee.empCount += 1

def displayCount(self):

print("Total Employee: {0}".format(Employee.empCount))

def displayEmployee(self):

print("Name : {0}, Salary: {1}, Age: {2}".format(self.name,self.salary, self.\_\_age))

def work(self):

return "%s is working" % (self.name)

def toString(self):

return "Name : {0}, Salary: {1}, Age: {2}".format(self.name,self.salary, self.\_\_age)

def getAge(self):

return self.\_\_age

emp1 = Employee ("James", 10000, 22)

emp2 = Employee ("Reed", 15000,24)

emp3 = Employee ("Naomi", 28000,28)

emp4 = Employee ("John", 22000,25)

emp5 = Employee ("Ann", 25000,27)

employees = [Employee ("James", 10000, 22),

Employee ("Reed", 15000, 24),

Employee ("Naomi", 28000,28),

Employee ("John", 22000,25),

Employee ("Ann", 25000,27)]

employees.sort (key=lambda x:x.salary ,reverse = True)

print("\nTop 3 Employees with Highest Salaries \n")

data = [emp.displayEmployee() for emp in employees[:3] ]

class Company:

Employee = []

def \_\_init\_\_(self,Employee):

self.Employee = Employee

def work (self):

for emp in self.Employee:

print(emp.work())

emp1 = Employee ("James", 10000, 22)

emp2 = Employee ("Reed", 15000,24)

emp3 = Employee ("Naomi", 28000,28)

emp4 = Employee ("John", 22000,25)

emp5 = Employee ("Ann", 25000,27)

company = Company([emp1,emp2,emp3,emp4,emp5])

print("\nCalling the Company: working\n")

company.work()

### path to file

with open ("/Users/techyouknow/Desktop/qfour.json") as file:

jdata = json.load(file)

file.close()

employ\_data = [itm for lines in jdata.values() for itm in lines]

#generator

def getEmployeeData(sqx):

for x in sqx:

yield x

emp\_list = []

for ent in getEmployeeData (employ\_data):

tmp = Employee(ent["name"],ent["salary"], ent["age"])

emp\_list.append(tmp)

# sort employees by their name

emp\_list.sort (key = lambda x :x.name)

# salary increase 5%

for e in getEmployeeData(emp\_list):

e.salary \*= 1.05

# create new dictionary with keys ranging to 10

emp\_dic = {}

for key , value in enumerate(getEmployeeData(emp\_list)):

emp\_dic.update ({key : value})

print ("\nKey Value pair for sorted by Employee Name")

for key , value in emp\_dic.items():

print ("key:", key +1, "value:", value.toString())

emp\_list.sort (key = lambda x :x.getAge())

print ("\nEmployee Age Group")

#age groups

def age\_group(age):

if age <= 20:

return "\nyounger than 20"

elif 20 < age <= 30:

return "\n21-30"

elif 31 < age <= 50:

return "\n31-50"

elif 50 < age <= 65:

return "\n51-65"

else:

return "\nolder than 65"

# using itertools groupby for age

for key, grp in it.groupby(emp\_list, lambda x: age\_group (x.getAge())):

print(key)

for e in grp:

print (e.toString())

#subclass

from collections import UserDict

class EmployeeD(UserDict):

def \_\_getitem\_\_(self, key):

return super().\_\_getitem\_\_(key)

def \_\_setitem\_\_(self,key,value):

return super().\_\_setitem\_\_(key,value)

empD\_list = []

for x in emp\_list:

temp = EmployeeD(name=x.name, salary=x.salary, age= x.getAge())

empD\_list.append(temp)

total\_sal = 0

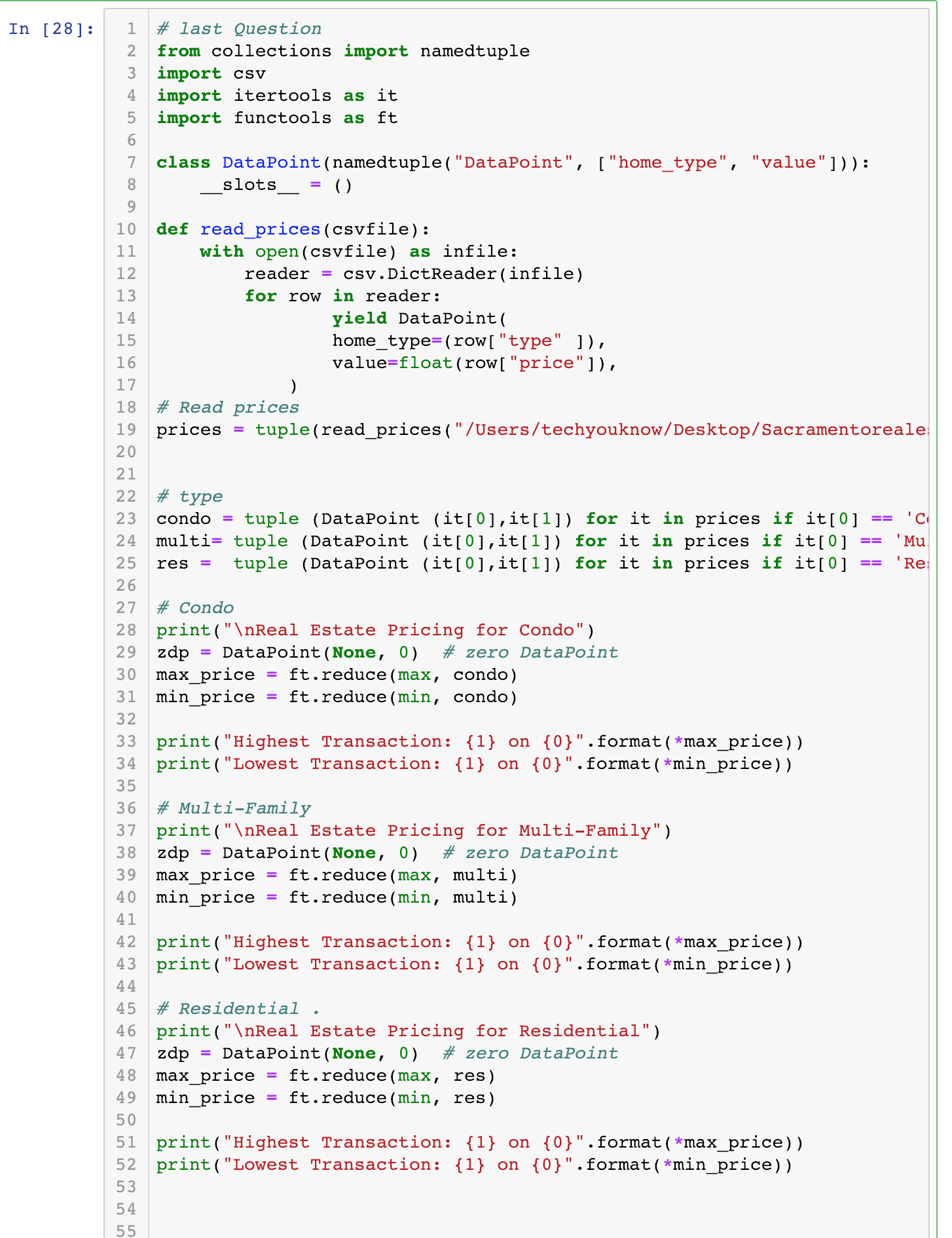
#using earlier generator function

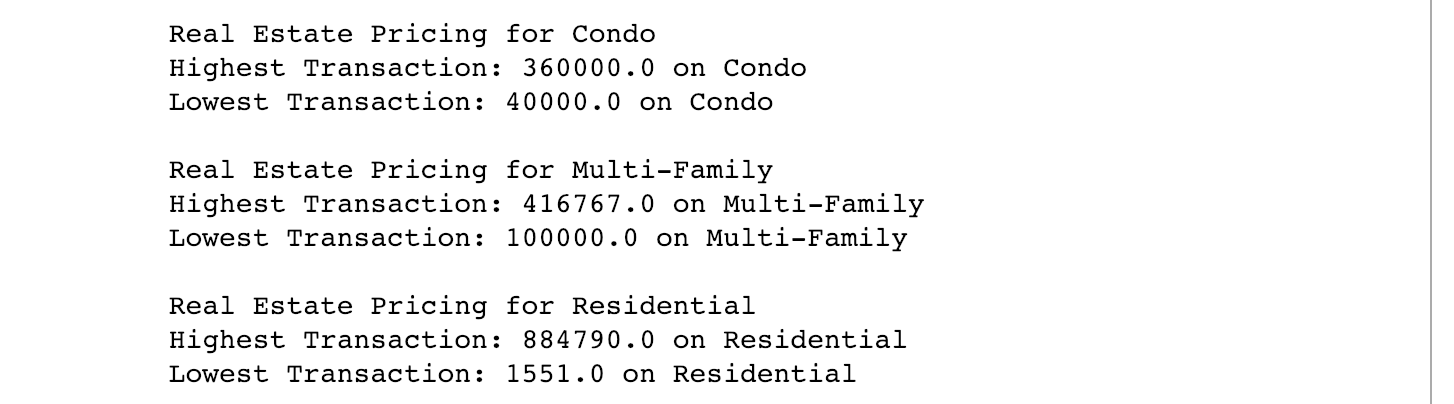
for emp in getEmployeeData(empD\_list):

total\_sal += emp.\_\_getitem\_\_("salary")

print("\n Total Salary for Employees :", total\_sal)

**Using what we have learned in Unit2 to analyze provided real estate transaction .csv file, you can use SP500 example as reference. Please find the highest and lowest transactions for each category of real estate: condo, residential and multi-family.**





**Code**

# last Question

from collections import namedtuple

import csv

import itertools as it

import functools as ft

class DataPoint(namedtuple("DataPoint", ["home\_type", "value"])):

\_\_slots\_\_ = ()

def read\_prices(csvfile):

with open(csvfile) as infile:

reader = csv.DictReader(infile)

for row in reader:

yield DataPoint(

home\_type=(row["type" ]),

value=float(row["price"]),

)

# Read prices

prices = tuple(read\_prices("/Users/techyouknow/Desktop/Sacramentorealestatetransactions.csv"))

# type

condo = tuple (DataPoint (it[0],it[1]) for it in prices if it[0] == 'Condo')

multi= tuple (DataPoint (it[0],it[1]) for it in prices if it[0] == 'Multi-Family')

res = tuple (DataPoint (it[0],it[1]) for it in prices if it[0] == 'Residential')

# Condo

print("\nReal Estate Pricing for Condo")

zdp = DataPoint(None, 0) # zero DataPoint

max\_price = ft.reduce(max, condo)

min\_price = ft.reduce(min, condo)

print("Highest Transaction: {1} on {0}".format(\*max\_price))

print("Lowest Transaction: {1} on {0}".format(\*min\_price))

# Multi-Family

print("\nReal Estate Pricing for Multi-Family")

zdp = DataPoint(None, 0) # zero DataPoint

max\_price = ft.reduce(max, multi)

min\_price = ft.reduce(min, multi)

print("Highest Transaction: {1} on {0}".format(\*max\_price))

print("Lowest Transaction: {1} on {0}".format(\*min\_price))

# Residential .

print("\nReal Estate Pricing for Residential")

zdp = DataPoint(None, 0) # zero DataPoint

max\_price = ft.reduce(max, res)

min\_price = ft.reduce(min, res)

print("Highest Transaction: {1} on {0}".format(\*max\_price))

print("Lowest Transaction: {1} on {0}".format(\*min\_price))