# week  5 exercises

# Ex4-1(continuation from last week)

download <Employee.py> (and <oop.py>) from last week's

use it this way:

-instantiate (create objects) 3 employees along with their name, and salary (decide your own)

-use the method displayEmployee to print them(3 employees) and also print total number of employee

USE it in 2 ways:

#1 create <as4-1.py> and import <Employee.py>

#2 modify <Employee.py> into <Employee1.py> (no separate file like #1)

Submit <as4-1.py> ,<Employee.py> and <Employee1.py>  zipped on SFS

Refer to  <oop.py> as an example of how to use this class

# Ex 4-2(continuation from last week)

Change salary variable to private

add “setters” and “getters” method to <Employee.py> to set and get these variables:

name, salary. Example:

```
 def setname(self, fullname):
    self.name = fullname
def getname(self):
    return self.name
```

# Ex 4-2(cont)

add a static method (use @staticmethod)
to define a method called
displayCountess that return empCount
then try to call it this way:
Employee.displayCountess()
This can be called before you even create any employee objects in the program

define a dictionary to hold names and salary of the employees in class variable(s)

This dictionary shall be populated with names (as key) and salary(as value) automatically as objects are created

Example end result:
{“korry”:2000,”jeet”:3000…….)}

hints: define an empty list in the beginning

keep adding the key and value inside __init__

Save the modified file as <example2.py> and submit on SFS

# Ex4-3(continuation from last week) -there is no Ex4-2!

review the attached files(<ex43py>).

Class Employees is derived from Person,
Manager from Employees

(ref: "inheritance" topic 2 weeks ago)

Use the classes this way:(import it from another file)

Create a total of 4 objects:

1 manager, 3 employees with different jobs, for example engineer, nurse, etc

# Try the below (example only)

```
myemployee = Employee('SFC Taro')
 myemployee.setsalary(2000)
 myemployee.setsex('Male')
 myemployee.setdob('1990/5/1')
 print myemployee
```

Task:

add a static method "created_jobs" like the below

```
def created_jobs():
    return Employee.created_employees
```

created_employees could be  a dictionary that contains job(key) and the number(value) of person who has that job,

for example

{"engineer":1, "nurse":2..... }

hints:

start by defining an empty dictionary

add the dictionary elements as each object get created

this is very similar to the last task that you did in class

## Bonus:(optional)

add a method that sorts the dictionary by the number of person on that job, i.e by its value

Do some reading at on how to sort a dictionary by "sorted"

http://www.pythoncentral.io/how-to-sort-python-dictionaries-by-key-or-value/

"sort" that we have learnt before is different to "sorted"

a={"a":3,"b":1,"c":4}

a.sort() will sort and update(return) a , so it cannot be assigned to a variable

b=a.sort() will not result in anything on b

b=sorted(a) will however work

# Homework

Infrared Remote Control API

define a class IRRemote
methods are:
VolumeUp, VolumeDown, ChannelUp,ChannelDown
Power(off/on)-->1 or 0
PlayStop(play/stop)-->1 or 0
ChannelNumber-->return channel number

## Homework(cont)

In the real life (API) those methods will talk to the hardware in your laptop. For this example, simply get the method to print something for example :”volume up”, etc

# Homework(cont)

test the class this way:

Power(1)
VolumeUp()
VolumeDown()
ChannelUp()
ChannelDown()
ChannelDown()
print the ChannelNumber
ChannelUp();
print ChannelNumber
PlayStop(1)
PlayStop(0)
Power(0)