Harsh Kasliwal

2003085

**EXPERIMENT 7**

**Q1.**

|  |  |
| --- | --- |
| 1. | Python program to demonstrate use of packages.    Create a file to import all the packages , calculate the salary and display all the details of the Employee  Salary = Basic +H.R.A – P.F. |

A package is a folder containing modules and maybe other folders that themselves may contain more folders and modules. Conceptually, it’s a namespace. This simply means that a package’s modules are bound together by a package name, by which they may be referenced.

A package must have a \_\_init\_\_.py file which tells the interpreter that this folder is a package. It may be empty, or it may contain some code to be executed upon initialization of the package.

A package may also contain subpackages, having their own \_\_init\_\_.py files and their own modules.

For example, say we have a package Game with a subpackage Sound and a module ‘load’

To import the module, we type the following:

import Game.Sound.load

We can also import it giving it an alias:

import Game.Sound.load as game

You can’t import a function using the dot operator(.) For that, you must type this:

from Game.Sound.load import volume\_up

**CODE:**

**main.py:**

from package.Account.salary import Salary

from package.Employee.profile import Profile

from package.Employee.Qualification import Qualification

n = int(input("Enter number of Employees: "))

emp\_list = []

for i in range(n):

name = input(f"Enter Name of Employee {i+1}: ")

age = input(f"Enter Age of Employee {i+1}: ")

dob = input(f"Enter Birth Year of Employee {i+1}: ")

degree = input(f"Enter Degree of Employee {i+1}: ")

experience = input(f"Enter Experience of Employee {i+1} (in years): ")

basic = int(input(f"Enter Basic Pay of Employee {i+1}: "))

hra = int(input(f"Enter HRA of Employee {i+1}: "))

pf = int(input(f"Enter PF of Employee {i+1}: "))

emp\_prof = Profile(name, age, dob)

emp\_q = Qualification(degree, experience)

emp\_s = Salary(pf, basic, hra)

t\_salary = basic + hra - pf

new\_employee = [emp\_prof, emp\_q, emp\_s, t\_salary]

emp\_list.append(new\_employee)

print()

print("Name\tAge\tDob\tDegree\tExp\tBasic\tHRA\tPF\tSalary")

print('-'\*75)

for emp in emp\_list:

print(emp[0].name, emp[0].age, emp[0].dob, emp[1].degree, emp[1].experience, emp[2].basicpay, emp[2].HRA, emp[2].PF, emp[3], sep='\t')

**pacakge/Account/salary.py:**

class Salary:

def \_\_init\_\_(self, PF, basicpay, HRA):

self.PF = PF

self.basicpay = basicpay

self.HRA = HRA

**pacakge/Employee/profile.py:**

class Profile:

def \_\_init\_\_(self, name, age, dob):

self.name = name

self.age = age

self.dob = dob

**package/Employee/Qualification.py:**

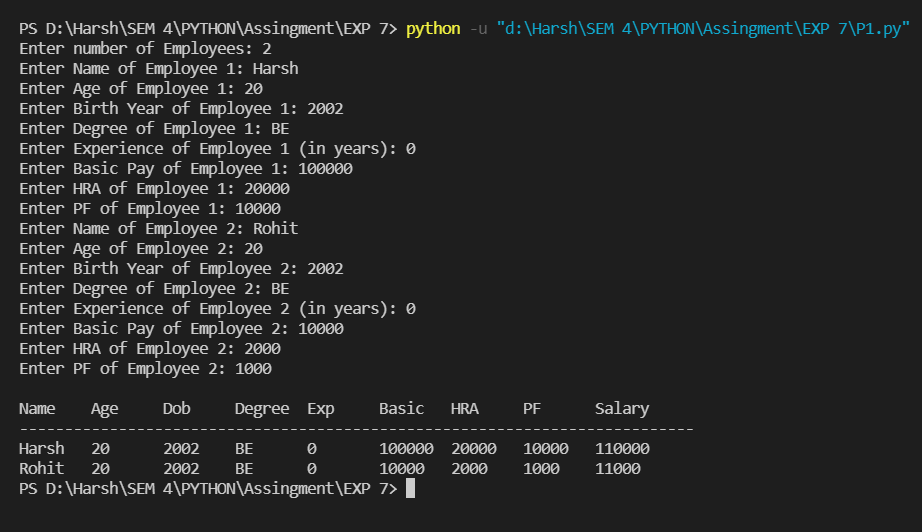
class Qualification:

def \_\_init\_\_(self, degree, experience):

self.degree = degree

self.experience = experience

OUTPUT



**Q2.**

|  |  |
| --- | --- |
| 2. | Python program to   * create directories using mkdir( ) and makedirs ( ) * remove directories using rmdir( ) and removedirs( ) * change current directory |

The OS module in Python provides functions for interacting with the

operating system. OS comes under Python’s standard utility modules. This

module provides a portable way of using operating system-dependent

functionality.

os.mkdir() method in Python is used to create a directory named path with

the speci ed numeric mode. This method raise FileExistsError if the directory

to be created already exists.

os.makedirs() method in Python is used to create a directory recursively. That

means while making leaf directory if any intermediate-level directory is

missing, os.makedirs() method will create them all.

os.rmdir() method in Python is used to remove or delete a empty directory.

OSError will be raised if the speci ed path is not an empty directory.

os.removedirs(): method in Python is used to remove directories recursively.

If the leaf directory in the speci ed path is successfully removed, then

os.removedirs() tries to successively remove every parent directory

mentioned in path until an error is raised.

os.chdir() method in Python used to change the current working directory to

speci ed path. It takes only a single argument as new directory path.

**CODE**

import os

def print\_directories(directory\_list):

i=0

print("Sr. No. \t Directory")

print("-" \* 40)

for directory in directory\_list:

print(i, "\t\t", directory)

i += 1

def print\_directory\_path\_and\_content(path):

print("The current working directory is:", path)

# Directories present in a specic path

print\_directories(os.listdir(path))

# Print Current

path = os.getcwd()

print\_directory\_path\_and\_content(path)

# Make Directory using mkdir

directory\_name = input("\nEnter Directory Name to be created: ")

os.mkdir(os.path.join(path, directory\_name))

# Print after creating Directories

print\_directory\_path\_and\_content(os.getcwd())

# Change Directory to the created one

os.chdir(directory\_name)

# Print after changing Directories

print\_directory\_path\_and\_content(os.getcwd())

path = os.getcwd()

# Make Directory using makedirs

path += r"/a/b/c"

os.makedirs(path)

# Print after creating Directories

print\_directory\_path\_and\_content(os.getcwd())

# Change Directory to the created one

os.chdir(r"a/b/c")

# Print after changing Directories

print\_directory\_path\_and\_content(os.getcwd())

os.chdir(r"../../..")

# Print after changing Directories

print\_directory\_path\_and\_content(os.getcwd())

path = os.getcwd()

path += r"/a/b/c"

# Remove directory

os.rmdir(path)

# Print after changing Directories

print\_directory\_path\_and\_content(os.getcwd())

OUTPUT

