Assessment on Python Training

Coding Section:

**(Any 5 Task out of 8 Task)**

Create a classes and functions with below functionalities

Task-1:

Write a class named ‘EmployeeClass’ which is having init () method with 3 arguments emp\_name, emp\_id, emp\_salary

Example:

E1 = EmployeeClass(‘emp-1’, 100, 1000)

Task-2:

Write instance methods to get emp\_name, emp\_id, emp\_salary

Expected output: E1.get\_emp\_name() Should print:

‘emp-1’

E1.get\_emp\_id() Should print: 100

E1.get\_emp\_salary() Should print:

1000

Task-3:

Write decorator function outside the above class, decorator function name will be ‘my\_company\_decorator’. Use this decorator on top of all 3 get-methods deﬁned inside the class. Details of the decorator has been provided below these examples. Please check

Expected output:

E1.get\_emp\_name() Should print:

Company Name Is: “XYZ Company” ‘emp-1’

Address: XYZ Address

E1.get\_emp\_id() Should print:

Company Name Is: “XYZ Company” 100

Address: XYZ Address

E1.get\_emp\_salary() Should print:

Company Name Is: “XYZ Company” 1000

Address: XYZ Address

Decorator Requirement: As we observed above, all get methods are using some common functionality which is

Company Name Is: “XYZ Company” Address: XYZ Address

Write a decorator for this common functionality and make use in all get methods

Task-4:

Make this ‘EmployeeClass’ iterable where if we iterate, in every iteration, it should return each character of emp\_name.

Example:

E1 = EmployeeClass(‘emp-1’, 100, 1000)

for c in E1:

print(“Each Char:”, c)

Then output should be Each Char: e

Each Char: m Each Char: p Each Char: - Each Char: 1

Task-5:

Write 2 class-methods where one method to set company head name and another method to get company head name

Example:

EmployeeClass.set\_company\_head\_name(‘head-1’) print(EmployeeClass.get\_company\_head\_name)  output ‘head-1’

Task-6:

Write variable-argument-static-method to compute average salary of employees. If we pass 2 or more salaries to methods, it should return the average salary.

Task-7:

Write new class called ‘NewEmployeeClass’ which is inheriting from ‘EmployeeClass’ and provide below functionality.

1. Add 2 instance-methods to set and get tax
2. Override get\_emp\_salary method to return (salary-tax)
3. Also write one more method called get\_old\_salary where inside this method, try to access super class method ‘get\_emp\_salary’ and return the super class method returned value.

Task-8:

Finally create below ﬁles,

1. Create new python ﬁle called ‘EmployeeModule.py’, Inside this ﬁle keep ONLY
   1. EmployeeClass
   2. my\_company\_decorator
2. Create another new python ﬁle called ‘NewEmployeeModule.py’, Inside this ﬁle keep ONLY
   1. NewEmployeeClass which is created in Task-6

NOTE: Since NewEmployeeClass is inheriting from ‘EmployeeClass’, import necessary module

1. Create new python ﬁle called ‘main\_program.py’, In this ﬁle import ‘NewEmployeeClass’ and test the following

 1: Create instance

E1 = NewEmployeeClass(‘emp-1’, 123, 1000)

 2: Add tax details E1.set\_emp\_tax(100)

 3: Access all methods

print(“Employee Name:”, E1.get\_emp\_name()) print(“Employee Salary:”, E1.get\_emp\_salary()) print(“Employee ID:”, E1.get\_emp\_id()) print(“Employee ID:”, E1.get\_emp\_tax())

1. Average Salary

avg\_sal = E1.get\_avg\_salary(1000, 2000, 3000) print(“avg\_sal:”, avg\_sal)  output=2000

1. Iterate for x in E1:

print(“Each Char: ”, x)

1. Get old salary

print(“Employee Old Salary :”, E1.get\_old\_salary())