System Requirements Specification Index

For

Design Patterns – Python

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Problem Statement : Design patterns in python

Description : This assessment contains various subparts to test design pattern concepts in Python.

The solution contains the following folder structure.

Design_patterns |

|--singleton.py

|--factory.py

|--builder.py

|--decorator.py

|--tests – contains unit test cases for the solution

singleton.py:

Implement a Logger class using Singleton which logs messages. Logger accepts a value sink which can be either "console" or "file_name" in which it prints to the console in the former and to the given file in the later case. Here, for simplicity print to console in both cases.

factory.py:

1. Implement a class PF, which inherits class Investment and implements lockin_period(return 5) and interest_rate(return 5.5)
2. Implement a class PPF, which inherits class Investment and implements lockin_period(return 15) and interest_rate(return 8)
3. Implement a class ELSS, which inherits class Investment and implements lockin_period(return 3) and interest_rate(return 10)

Define Factory method that accepts a scheme, in this case ("pf", "ppf", "elss") and returns respective objects.

builder.py:

1. Implements all the methods of IBuilder interface and returns the stats as a list in StatsListBuilder class.
2. Implements all the methods of IBuilder interface and returns the stats as a json in StatsJsonBuilder class.

Implement your builder generator method stats in StatsGenerator class, which accepts builder objects and returns stats in respective format.

decorator.py:

1. Implement class AddGST by inheriting Bill class and implement generate_bill which returns bill by adding 2.5% GST.

2. Implement class AddDeliveryCharge by inheriting Bill class and implement generate_bill which returns bill by adding 5% of total bill(bill+ 2.5% of bill) after adding GST.

PS: Please follow class or function signatures and given naming conventions. Do not change any pre-existing methods.

Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
3. This editor Auto Saves the code
4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To setup environment:

   pip install requests

   pip install numpy

7. Launch application:

   python3 builder.py

   python3 decorator.py

   python3 factory.py

   python3 singleton.py

8. To run Test cases:

   python3 -m unittest

9. Before Final Submission also, you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository for code quality analysis graph.

-----x-----