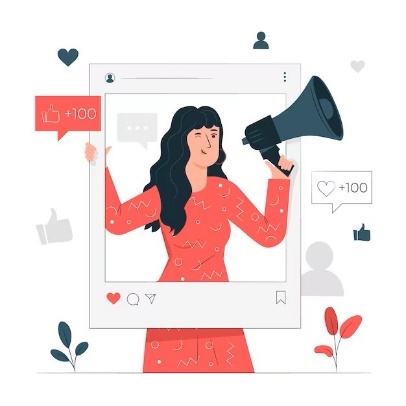
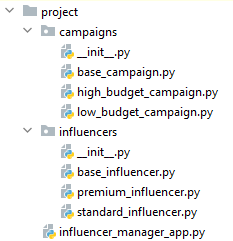
# Python OOP Exam - 6 April 2024



*In the bustling world of digital influence, our Influencer Manager App is designed to facilitate the coordination and management of influencer marketing campaigns. The app features a system for registering influencers, creating various types of campaigns with different budgets and engagement requirements, and tracking influencer participation and payments. It enables influencers to join campaigns, calculates payments based on specific rules, and provides functionality to initiate and monitor campaign statistics. The app serves as a centralized platform for orchestrating influencer marketing activities, ensuring smooth collaboration between brands and influencers.*

***Note: You are not allowed to change the folder and file structure and their names!***



**Judge Upload**

For the **first two problems**, create a **zip** file with the **project** **folder** and **upload it** to the judge system.

For the **last problem**, create a **zip** file with the **test folder** and **upload it** to the judge system.

You do not need to include **your venv, .idea, pycache, and \_\_MACOSX (for Mac users) in the zip file**, so you do not exceed **the maximum allowed size** of **16.00 KB**.

# Structure (Problem 1) and Functionality (Problem 2)

Our task is to implement all the classes' structure and functionality (properties, methods, inheritance, abstraction, etc.)

You are **free to add additional attributes** (instance attributes, class attributes, methods, dunder methods, etc.) to simplify your code and increase readability as long as it does not change the project's final result in accordance with its requirements so that the program works properly.

## Class BaseCampaign

In the **base\_campaign.py** file, the class **BaseCampaign** should be implemented. It is a **base class** for any **type of campaign**,and it **should not be able to be instantiated**.

### Structure

The class should have the following attributes:

* **campaign\_id: int**
  + An integer representing the **unique identifier** for each campaign.
  + If the **id is not a positive number** (**greater than zero**), raise a ValueError with the message:

**"Campaign ID must be a positive integer greater than zero."**

* Additionally, the **campaign id should be unique**. If it is not, a ValueError will be raised with the following message:

**"Campaign with ID {campaign\_id} already exists. Campaign IDs must be unique."**

* **Hint: You can keep the unique campaign IDs in a suitable collection as a class attribute and check if the new one is already there. Raise a** ValueError if it is there or add it to the collection if it is not.
* **brand: str**
  + A string representing the **name of the brand** associated with the campaign.
* **budget: float**
  + Representing the **budget required** for the campaign.
* **required\_engagement: float**
  + A **floating-point number** representing the **minimum engagement rate required** for an influencer to be eligible for the campaign.
* **approved\_influencers: list**
  + **Initially** set to an **empty list**. Will be **storing influencers objects** for the **campaign** if they meet the eligibility criteria.

### Methods

#### \_\_init\_\_(campaign\_id: int, brand: str, budget: float, required\_engagement: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### check\_eligibility(engagement\_rate: float)

* This method is designed to determine whether an **influencer is eligible to participate in a campaign** based on their **engagement rate**.
* Keep in mind that **each type of campaign** implements the method.

## Class HighBudgetCampaign

In the **high\_budget\_campaign.py** file, the class **HighBudgetCampaign** should be implemented. A high-budget campaign is a **type of campaign**. Each high-budget campaign has a budget of **$5000.0**.

### Methods

#### \_\_init\_\_(campaign\_id: int, brand: str, required\_engagement: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### check\_eligibility(engagement\_rate: float)

* This method takes an **engagement\_rate** parameter. The **influencer** is **eligible** for the **campaign** only if their **engagement rate** is **greater than or equal** to **120%** of the **required engagement rate**.
* **Returns** a **Boolean** value (**True** or **False**).

## Class LowBudgetCampaign

In the **low\_budget\_campaign.py** file, the class **LowBudgetCampaign** should be implemented. A low-budget campaignis a **type of campaign**. Each low-budget campaign has a budget of **$2500.0**.

### Methods

#### \_\_init\_\_(campaign\_id: int, brand: str, required\_engagement: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### check\_eligibility(engagement\_rate: float)

* This method takes an **engagement\_rate** parameter. The **influencer** is **eligible** for the **campaign** only if their **engagement rate** is **greater than or equal** to **90%** of the **required engagement rate**.
* **Returns** a **Boolean** value (**True** or **False**).

## Class BaseInfluencer

In the **base\_influencer.py** file, the class **BaseInfluencer** should be implemented. It is a base class for any type of **influencer**, and it should not be able to be instantiated.

### Structure

The class should have the following attributes:

* **username:** str
  + The value represents the **username of the influencer**.
  + If the name is **an empty string or contains only white spaces**, raise a ValueError with the message: **"Username cannot be empty or consist only of whitespace!"**
* **followers: int** 
  + Represents the **number of followers** the influencer has.
  + Validation checks ensure that the **number** of **followers is greater than or equal to zero**. If not, a ValueError is raised with the message:

**"Followers must be a non-negative integer!"**

* **engagement\_rate: float**
  + Represents the **engagement rate of the influencer** as a floating-point number.
  + Validation checks ensure that the engagement rate falls **within the range [0.0, 5.0] inclusive**. If not, a ValueError is raised with the message:

**"Engagement rate should be between 0 and 5."**

* **campaigns\_participated: list**
  + Keeps **track** of the **campaigns** in which the **influencer has participated**.
  + **Initially** set to an **empty list**. Will store **campaigns objects**.

### Methods

#### \_\_init\_\_(username: str, followers: int, engagement\_rate: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### calculate\_payment(campaign: BaseCampaign)

* This method is responsible for **determining the payment** an influencer should **receive for participating** in a specific campaign.
* Keep in mind that each **type of influencer** has a **different payment percentage**, and it will implement the **method differently**.

#### reached\_followers(campaign\_type: str)

* The method returns the **calculated reached followers** based on the type of campaign the influencer participated in.
* Keep in mind that **each type of influencer** can implement the **method differently**.

#### display\_campaigns\_participated()

Returns a **string** with **information** about the **influencer**, including the **participated campaigns**:

* + If the influencer has **not participated** in any campaigns, it returns a message in the following format:  
    **"{username} has not participated in any campaigns."**
  + If the influencer has **participated** in campaigns, it returns a message in the following format (when you list the campaigns, **before the dash**, there are exactly **two spaces**):  
    **"{type\_of\_influencer} :) {username} :) participated in the following campaigns:**

**- Campaign ID: {campaign\_id1}, Brand: {brand1}, Reached followers: {reached\_followers}**

**- Campaign ID: {campaign\_id2}, Brand: {brand2}, Reached followers: {reached\_followers}**

**…**

**- Campaign ID: {campaign\_idn}, Brand: {brandn}", Reached followers: {reached\_followers}"**

* **Hint**: You can use the method **reached\_followers()**

## Class PremiumInfluencer

In the **premium\_influencer.py** file, the class **PremiumInfluencer** should be implemented. The Premium influencer is a **type of influencer**. Each influencer has an **initial payment percentage of 85%**.

### Methods

#### \_\_init\_\_(username: str, followers: int, engagement\_rate: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### calculate\_payment(campaign: BaseCampaign)

* This method takes a **campaign** parameter.
* The **payment** is calculated by multiplying the **campaign's budget** by the **influencer's payment percentage**.
* The method **returns** a **floating-point value** representing the calculated **payment**.

#### reached\_followers(campaign\_type: str)

This method takes a **campaign\_type** as a string parameter and **calculates** the **reached followers** based on the influencer's attributes and the specific adjustments associated with the given campaign type. The **campaign\_type** parameter is used to determine which type of campaign the influencer participated in.

* If the **campaign\_type** is **"HighBudgetCampaign"** the **reached followers** are **calculated** by **multiplying** the **influencer's followers** by their **engagement rate** and then applying a **1.5 multiplier**.
* If the **campaign\_type** is **"LowBudgetCampaign"** the **reached followers** are **calculated** by **multiplying** the **influencer's followers** by their **engagement rate** and then applying a **0.8 multiplier**.

The method **returns** the **integer** representation of the **reached followers**.

## Class StandardInfluencer

In the **standard\_influencer.py** file, the class **StandardInfluencer** should be implemented. The Standard influencer is a **type of influencer**. Each influencer has an **initial payment percentage of 45%**.

### Methods

#### \_\_init\_\_(username: str, followers: int, engagement\_rate: float)

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### calculate\_payment(campaign: BaseCampaign)

* This method takes a **campaign** parameter.
* The **payment** is calculated by multiplying the **campaign's budget** by the **influencer's payment percentage**.
* The method **returns** a **floating-point value** representing the **calculated payment**.

#### reached\_followers(campaign\_type: str)

This method takes a **campaign\_type** as a string parameter and **calculates** the **reached followers** based on the influencer's attributes and the specific adjustments associated with the given campaign type. The **campaign\_type** parameter is used to determine which type of campaign the influencer participated in.

* If the **campaign\_type** is **"HighBudgetCampaign"** the **reached followers** are **calculated** by **multiplying** the **influencer's followers** by their **engagement rate** and then applying a **1.2 multiplier.**
* If the **campaign\_type** is **"LowBudgetCampaign"** the **reached followers** are **calculated** by **multiplying** the **influencer's followers** by their **engagement rate** and then applying a **0.9 multiplier.**

The method **returns** the **integer** representation of the **reached** **followers**.

## Class InfluencerManagerApp

In the **influencer\_manager\_app.py** file, the class **InfluencerManagerApp** should be implemented. It will contain the functionality of the project.

### Structure

The class should have the following attributes:

* **influencers: list**
  + **Initially** an **empty list** to store all **influencer objects** registered with the Influencer Manager.
* **campaigns: list**
  + **Initially** an **empty list** to store all **campaign objects** that are scheduled to start.

### Methods

#### \_\_init\_\_()

* In the **\_\_init\_\_** method, all the needed attributes must be set.

#### register\_influencer(influencer\_type: str, username: str, followers: int, engagement\_rate: float):

The method **creates** an **influencer** of the given type and **adds** it to the **influencers** collection.

* If the influencer's type is **not valid**, return the following message:   
  **"{influencer\_type} is not an allowed influencer type."**
* If an influencer with the same **username** is already added to the list, do not duplicate records, return the following message:  
  **"{username} is already registered."**
* If none of the above cases is reached, the **influencer** is successfully created. Store the influencer in the appropriate collection and return it:  
  **"{username} is successfully registered as a {influencer\_type}."**
* Valid types of influencers are: **"PremiumInfluencer"** and **"StandardInfluencer"**

#### create\_campaign(campaign\_type: str, campaign\_id: int, brand: str, required\_engagement: float)

The method **creates** a campaign of the given type and **adds** it to the **campaigns** collection. The method is responsible for **allowing a new campaign to be created**.

* First, check if the **campaign type** is valid. If **not**, return the following message:  
  **"{campaign\_type} is not a valid campaign type."**
* If a **campaign** with the **same** **id** is **already added** to the **list**, do **not try to create** a new one, **return** the following message instead:  
  **"Campaign ID {campaign\_id} has already been created."**
* If the above case is not reached, create the correct type of **campaign** and **add** it to the appropriate collection. **Return** the following message:  
  **"Campaign ID {campaign\_id} for {brand} is successfully created as a {campaign\_type}."**
* Valid types of campaigns are: **"HighBudgetCampaign"** and **"LowBudgetCampaign"**

#### participate\_in\_campaign(influencer\_username: str, campaign\_id: int):

The method allows an **influencer to join a specific campaign**. It performs the **following checks**:

* **Influencer Validation:**
  + Check if the **influencer** with the given **username** is registered.
  + If **not found**, **return** the following message:  
    **"Influencer '{influencer username}' not found."**
* **Campaign Validation:**
* Verifies the existence of the **campaign** with the specified **campaign\_id**.
* If not found, **return** the following message:  
  **"Campaign with ID {campaign\_id} not found."**
* **Participation Check:**
  + Utilizes the **check\_eligibility** method to determine if the influencer is eligible to join.
  + If ineligible, **returns**:  
    **"Influencer '{influencer\_username}' does not meet the eligibility criteria for the campaign with ID {campaign\_id}."**
* **Payment Processing:**
  + Calculates the payment for the influencer.
  + If the **influencer payment** is **greater than zero** (**0.0**):
    - **Add** the **influencer** to the **approved influencers list** of the **campaign**
    - **Adjust** the **campaign's budget** (subtract the calculated payment)
    - **Add** the **campaign** to the **participated campaigns list** of the **influencer**
    - **Returns**:  
      **"Influencer '{influencer\_username}' has successfully participated in the campaign with ID {campaign\_id}."**
  + Otherwise do nothing.

#### calculate\_total\_reached\_followers()

* This method computes the **total number** of reached **followers (integer value)** for **each campaign** based on the **influencers** who **participated** in those **campaigns**.
* The method **returns** a **dictionary** with **key-value pairs** containing a **campaign object** as a **key** and the **total followers** as a **value**:

**{campaign\_object1: total\_followers, campaign\_object2: total\_followers, … }**

* If a **campaign** does **not have** any **participating influencers** (hence there are **no followers**), **do not include it** in theresult.
* **Hint**: You can use the designated **influencer's** method **reached\_followers()** and **campaigns\_participated** property

#### influencer\_campaign\_report(username: str):

The method generates a **report of campaigns** in which the specified influencer has participated. The username will always be a **valid string** of **existing** **influencer**.

* **Display details of the campaigns** they have taken part in in the following format:
  + If the influencer has **not participated** in any campaigns, it **returns** a message in the following format:  
    **"{username} has not participated in any campaigns."**
  + If the influencer has **participated** in campaigns, it returns a message in the following format (when you list the campaigns, **before the dash**, there are exactly **two spaces**):  
    **"{type\_of\_influencer} :) {username} :) participated in the following campaigns:**

**- Campaign ID: {campaign\_id1}, Brand: {brand1}, Reached followers: {reached\_followers}**

**- Campaign ID: {campaign\_id2}, Brand: {brand2}, Reached followers: {reached\_followers}**

**…**

**- Campaign ID: {campaign\_idn}, Brand: {brandn}", Reached followers: {reached\_followers}"**

* **Hint**: You can use the designated **influencer's** method **display\_campaigns\_participated()**

#### campaign\_statistics():

* **Returns** detailed information about the **Influencer Manager**.
* **Sort** the **campaigns by** the **total number** of **approved influencers**, **ascending**.
* If **multiple campaigns** have the **same** **number** of **approved influencers**, sort them in **descending order** by their **budgets**:  
  **"$$ Campaign Statistics $$  
   \* Brand: {brand1}, Total influencers: {approved\_influencers1}, Total budget: ${budget1}, Total reached followers: {total\_reached\_followers1}  
   \* Brand: {brand2}, Total influencers: {approved\_influencers2}, Total budget: ${budget2}, Total reached followers: {total\_reached\_followers2}  
  …  
   \* Brand: {brandn}, Total influencers: {approved\_influencersn}, Total budget: ${budgetn}, Total reached followers: {total\_reached\_followersn}"**
* There are exactly **two spaces before** the asterisk.
* The **budget** should be **formatted** to the **second decimal place**.

#### Examples

|  |
| --- |
| **Input** |
| **from project.influencer\_manager\_app import InfluencerManagerApp  manager = InfluencerManagerApp()**  *# Register influencers* **print(manager.register\_influencer("PremiumInfluencer", "JohnDoe", 50000, 4.2))**  **print(manager.register\_influencer("StandardInfluencer", "JaneSmith", 10000, 3.5))**  **print(manager.register\_influencer("PremiumInfluencer", "JohnDoe", 80000, 4.2))**  **print(manager.register\_influencer("InvalidInfluencer", "JohnDoe", 50000, 4.2))**  **print(manager.register\_influencer("StandardInfluencer", "AliceJohnson", 20000, 3.8))**  **print(manager.register\_influencer("PremiumInfluencer", "OliviaBennett", 80000, 4.5))**  **print(manager.register\_influencer("PremiumInfluencer", "DanielRodriguez", 90000, 4.8))**  **print(manager.register\_influencer("PremiumInfluencer", "EmilyTurner", 1000000, 5.0))**  *# Create campaigns*  **print(manager.create\_campaign("LowBudgetCampaign", 1, "TechGurus", 4.0))**  **print(manager.create\_campaign("HighBudgetCampaign", 2, "FashionTrendz", 3.0))**  **print(manager.create\_campaign("LowBudgetCampaign", 1, "FashionTrendz", 3.0))**  **print(manager.create\_campaign("LowBudgetCampaign", 3, "QuantumFusion", 3.0))**  **print(manager.create\_campaign("InvalidCampaign", 4, "FoodieDelights", 2.5))**  *# Participate in campaigns*  **print(manager.participate\_in\_campaign("JohnDoe", 1))**  **print(manager.participate\_in\_campaign("JaneSmith", 2))**  **print(manager.participate\_in\_campaign("AliceJohnson", 2))**  **print(manager.participate\_in\_campaign("AliceJohnson", 1))**  **print(manager.participate\_in\_campaign("NonExistentInfluencer", 1))**  **print(manager.participate\_in\_campaign("AliceJohnson", 3))**  **print(manager.participate\_in\_campaign("JohnDoe", 2))**  **print(manager.participate\_in\_campaign("JaneSmith", 4))**  **print(manager.participate\_in\_campaign("JaneSmith", 1))**  **print(manager.participate\_in\_campaign("OliviaBennett", 3))**  **print(manager.participate\_in\_campaign("DanielRodriguez", 3))**  **print(manager.participate\_in\_campaign("EmilyTurner", 3))**  *# Print influencer campaign reports and campaign statistics*  **print(manager.influencer\_campaign\_report("JohnDoe"))**  **print(manager.influencer\_campaign\_report("JaneSmith"))**  **print(manager.campaign\_statistics())** |

|  |
| --- |
| **Output** |
| **JohnDoe is successfully registered as a PremiumInfluencer.**  **JaneSmith is successfully registered as a StandardInfluencer.**  **JohnDoe is already registered.**  **InvalidInfluencer is not an allowed influencer type.**  **AliceJohnson is successfully registered as a StandardInfluencer.**  **OliviaBennett is successfully registered as a PremiumInfluencer.**  **DanielRodriguez is successfully registered as a PremiumInfluencer.**  **EmilyTurner is successfully registered as a PremiumInfluencer.**  **Campaign ID 1 for TechGurus is successfully created as a LowBudgetCampaign.**  **Campaign ID 2 for FashionTrendz is successfully created as a HighBudgetCampaign.**  **Campaign ID 1 has already been created.**  **Campaign ID 3 for QuantumFusion is successfully created as a LowBudgetCampaign.**  **InvalidCampaign is not a valid campaign type.**  **Influencer 'JohnDoe' has successfully participated in the campaign with ID 1.**  **Influencer 'JaneSmith' does not meet the eligibility criteria for the campaign with ID 2.**  **Influencer 'AliceJohnson' has successfully participated in the campaign with ID 2.**  **Influencer 'AliceJohnson' has successfully participated in the campaign with ID 1.**  **Influencer 'NonExistentInfluencer' not found.**  **Influencer 'AliceJohnson' has successfully participated in the campaign with ID 3.**  **Influencer 'JohnDoe' has successfully participated in the campaign with ID 2.**  **Campaign with ID 4 not found.**  **Influencer 'JaneSmith' does not meet the eligibility criteria for the campaign with ID 1.**  **Influencer 'OliviaBennett' has successfully participated in the campaign with ID 3.**  **Influencer 'DanielRodriguez' has successfully participated in the campaign with ID 3.**  **Influencer 'EmilyTurner' has successfully participated in the campaign with ID 3.**  **PremiumInfluencer :) JohnDoe :) participated in the following campaigns:**  **- Campaign ID: 1, Brand: TechGurus, Reached followers: 168000**  **- Campaign ID: 2, Brand: FashionTrendz, Reached followers: 315000**  **JaneSmith has not participated in any campaigns.**  **$$ Campaign Statistics $$**  **\* Brand: FashionTrendz, Total influencers: 2, Total budget: $412.50, Total reached followers: 406200**  **\* Brand: TechGurus, Total influencers: 2, Total budget: $206.25, Total reached followers: 236400**  **\* Brand: QuantumFusion, Total influencers: 4, Total budget: $4.64, Total reached followers: 4702000** |

# Task 3: Unit Tests (100 points)

You will **be provided with another skeleton** for this problem. **Open** the **new skeleton** as a **new project** and write tests for the **SocialMedia** class. The class will have some methods, fields, and one constructor, all of them working properly. You are **NOT ALLOWED** to change anything in the class code. Cover the whole class with unit tests to make sure that the class is working as intended. Submit **only the test** folder.