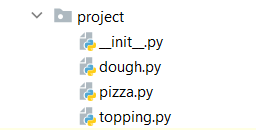
## Pizza Maker

Create separate file for each class as shown below and submit a zip file containing all files (zip the whole project folder/module) - it is important to include all files in project module to be able to make proper imports.



Create a class called **Topping**. Upon initialization it should receive:

* **topping\_type: str** - if the topping is an **empty string**, raise a **ValueError** with message **"The topping type cannot be an empty string"**
* **weight: float** - if the weight is **0 or less**, raise a **ValueError** with message **"The weight cannot be less or equal to zero"**

Hint: Use **Getters** and **Setters.**

Create a class called **Dough**. Upon initialization it should receive:

* **flour\_type: str** - if the flour typ'e is an **empty string**, raise a **ValueError** with message **"The flour type cannot be an empty string"**
* **baking\_technique: str** - if the topping is an **empty string**, raise a **ValueError** with message **"The baking technique cannot be an empty string"**
* **weight: float** - if the weight is **0 or less**, raise a **ValueError** with message **"The weight cannot be less or equal to zero"**

Create a class called **Pizza**. Upon initialization it should receive:

* **name: str** - if the name is an **empty string**, raise a **ValueError** with message **"The name cannot be an empty string"**
* **dough: Dough** - if the dough is **None**, raise a **ValueError** with message **"You should add dough to the pizza"**
* **toppings\_capacity: int** – represents the maximum number of toppings the pizza should have. If the capacity is **0 or less**, raise a **ValueError** with message **"The topping's capacity cannot be less or equal to zero"**
* **toppings: dict** – empty dictionary upon initialization that will contain the **topping type** **as a key** and the **topping's weight as a value**.

The class should have also 2 instance methods:

* **add\_topping(topping: Topping)**
  + **Add** a new topping to the dictionary
* If there is **no space left** for **a new topping**, raise a **ValueError**: **"Not enough space for another topping"**
* If the topping is **already in the dictionary**, **increase the value of its weight**.
* **calculate\_total\_weight()** - returns the total weight of the pizza (dough's weight and toppings' weight)

### Examples

|  |
| --- |
| **Test Code** |
| from project.dough import Dough  from project.pizza import Pizza  from project.topping import Topping  tomato\_topping = Topping("Tomato", 60)  print(tomato\_topping.topping\_type)  print(tomato\_topping.weight)  mushrooms\_topping = Topping("Mushroom", 75)  print(mushrooms\_topping.topping\_type)  print(mushrooms\_topping.weight)  mozzarella\_topping = Topping("Mozzarella", 80)  print(mozzarella\_topping.topping\_type)  print(mozzarella\_topping.weight)  cheddar\_topping = Topping("Cheddar", 150)  pepperoni\_topping = Topping("Pepperoni", 120)  white\_flour\_dough = Dough("White Flour", "Mixing", 200)  print(white\_flour\_dough.flour\_type)  print(white\_flour\_dough.weight)  print(white\_flour\_dough.baking\_technique)  whole\_wheat\_dough = Dough("Whole Wheat Flour", "Mixing", 200)  print(whole\_wheat\_dough.weight)  print(whole\_wheat\_dough.flour\_type)  print(whole\_wheat\_dough.baking\_technique)  p = Pizza("Margherita", whole\_wheat\_dough, 2)  p.add\_topping(tomato\_topping)  print(p.calculate\_total\_weight())  p.add\_topping(mozzarella\_topping)  print(p.calculate\_total\_weight())  p.add\_topping(mozzarella\_topping) |
| **Output** |
| Tomato  60  Mushroom  75  Mozzarella  80  White Flour  200  Mixing  200  Whole Wheat Flour  Mixing  260  340  ValueError: Not enough space for another topping |