
# More Object Oriented Programming

# Object Oriented Programming

Lets you create new objects in your program.

“Type” ~> “Class”

“Data/Variables” ~> “Attributes”

“Functions” ~> “Methods”

```python
"""Define a Pizza class."""


class Pizza:

    # Attributes
    gluten_free: bool
    size: str
    num_toppings: int

    # Constructor
    def __init__(self, gf_input: bool, size_input: str, num_toppings_input: int):
        """Initialize the attribute values."""
        self.gluten_free = gf_input
        self.size = size_input
        self.num_toppings = num_toppings_input
        # this returns self

    # Cost Method
    def price(self) -> float:
        """Return the price of a Pizza."""
        cost: float = 0.0
        if self.size == "small":
            cost = 5.25
        else:
            cost = 7.50
        cost += .25 * self.num_toppings
        if self.gluten_free is True:
            cost += 1.00
        return cost
```

```python
"""Instantiate the Pizza class."""

from lessons.pizza_orders import Pizza, num_orders


alyssas_order: Pizza = Pizza(True, "small", 0)
lukes_order: Pizza = Pizza(False, "large", 2)
print(alyssas_order.price())
print(lukes_order.price())

team110_orders: list[Pizza] = [alyssas_order, lukes_order]

print(num_orders(team110_orders))
```

```python
# A function that uses an instance (or instances) of a class
def num_orders(pizzas: list[Pizza]) -> int:
    """Tells you how many elements are in pizzas."""
    return len(pizzas)
```

# Class Writing

- Write a class called Profile
- It should have two attributes, username: str and private: bool
- Write a *constructor* that takes two parameters: self and username_input: str. It should set the username attribute equal to username_input and set the private attribute to True.
- Write a method called tweet that takes as parameters self and msg: str. If self.private is False, then it should print msg

# Instantiation

- Create a new variable user1 that is reference to a Profile object with the username “110_rulez”
- Update user1’s private attribute to be False
- Use the tweet method call to tweet the message “OOP is cool!”

# Diagram

# Class Writing

- Write a class called Games
- It should have three attributes, collection: list[str], wishlist: list[str], and budget: float
- Write a *constructor* that takes four parameters: self, curr_collection: list[str], wish: list[str], and start_budget: float. It should update the collection, wishlist, and budget attributes accordingly.
- Write a method called purchase that takes as parameters self, name: str, and cost: float.
    - If cost is less than the budget attribute:
        - subtract cost from the budget
        - add name to collection
        - and if name is in wishlist, remove it from wishlist
    - Else (cost is greater than the budget attribute):
        - print “Sorry! Not enough money!” and do nothing else.

# Instantiation

- Create a new variable my_games that is reference to a Games object with the collection [“Sims”], the wishlist [“Witcher”], and the budget 50.75.
- Use the purchase method call to try and purchase “Stardew” at price 40.25
- Use the purchase method call to try and purchase “Witcher” at price 60.00
- Print out the collection, wishlist, and budget attributes