LS 23 - Object Oriented Programming

Object Oriented Programming

Lets you create new objects in your program.

“Type” ~> “Class”

“Data/Variables” ~> “Attributes”

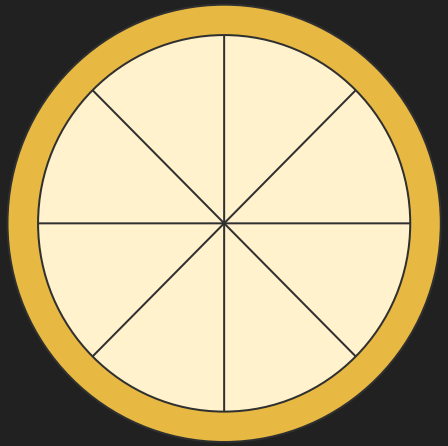
“Functions” ~> “Methods”

Example: Pizza

size: small

toppings: 0

gluten free: no

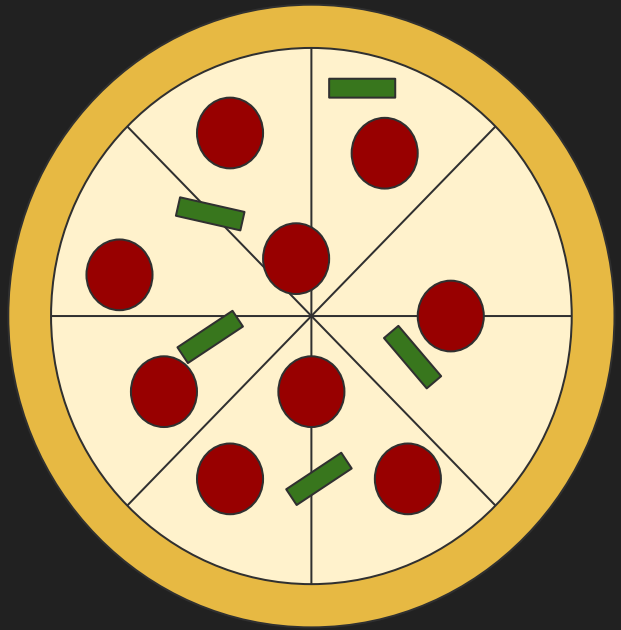


Example: Pizza

size: large

toppings: 2

gluten free: yes



Attributes

- variables that belong to each instantiation of the object
- Syntax:

    <attribute name> : <type>

    gluten_free : bool

Constructor

- Method that defines what happens when new object is created
- Signature Syntax:

```
def __init__(self, <other parameters>):
```

*Essentially returns self

- Instantiation:

```
<class name>(<arguments>)
```

Methods

- Functions that belong to an object
- Calling a method:

```
price(my_pizza) ~> my_pizza.price()
```

- Defining a method:

```
def <method_name>(self, <other parameters>) -> <return type>:
```

```
def price(self) -> float:
```

LS 24 - Classes in Memory

```python
class Pizza:


    size: str
    toppings: int
    gluten_free: bool

    def __init__(self, size_input: str, toppings_input: int, gf_input: bool):
        self.size = size_input
        self.toppings = toppings_input
        self.gluten_free = gf_input

    def price(self) -> float:
        cost: float = 0.0
        if self.size == "large":
            cost = 6.0
        else:
            cost = 5.0
        cost += .75 * self.toppings
        if self.gluten_free:
            cost += 1.0
        return cost

    def add_toppings(self, num_toppings: int) -> None:
        self.toppings += num_toppings

my_pizza: Pizza = Pizza("large", 0, False)
my_pizza.add_toppings(2)
print(my_pizza.price())
```