Topic 59 - Classes: Adding Methods to Classes

**What**

* Methods are functions defined within a class that allow instances to perform specific actions using their own data.
* By embedding functions as methods, classes become more functional, allowing instances to interact with their attributes seamlessly without needing to pass them as arguments each time.
* Methods use instance attributes directly and can also accept additional arguments when called.

**Why**

* **Encapsulation**: Methods encapsulate behavior specific to the class, making the class self-contained with both data (attributes) and functionality.
* **Ease of Use**: Methods make it simpler to call functions on an instance, as attributes are already included without requiring explicit argument passing.
* **Cleaner Code**: Built-in methods create readable code that is easier to maintain and understand, reducing repetitive function calls and parameters.

**How**

1. **Defining a Method in a Class**  
   Here’s an updated Patient class with an added method to check if a patient is a minor:

python

Copy code

class Patient:

def \_\_init\_\_(self, last\_name, first\_name, age):

self.last\_name = last\_name

self.first\_name = first\_name

self.age = age

# Method to check if patient is a minor

def say\_if\_minor(self):

if self.age < 21:

print(self.first\_name + " " + self.last\_name + " is a minor")

* + **Method Definition**: The say\_if\_minor method checks the age attribute and displays a message if the patient is under 21.
  + **Accessing Attributes**: Inside the method, self.first\_name, self.last\_name, and self.age are used to access instance-specific data.

1. **Calling a Method on an Instance**  
   Once the method is defined in the class, it can be easily called on an instance:

python

Copy code

pid4343 = Patient("Taleb", "Sue", 61)

pid4343.say\_if\_minor()

* + **Simple Call**: Unlike a freestanding function, say\_if\_minor is called directly on pid4343 without passing first\_name, last\_name, or age as arguments.
  + **Automatic Access**: The method has direct access to pid4343's attributes without needing explicit argument passing.

1. **Adding Arguments to Methods**  
   Methods can still take additional arguments when needed:

python

Copy code

def say\_if\_minor(self, month, insured=False):

if self.age < 21:

print(f"{self.first\_name} {self.last\_name} is a minor as of {month}.")

if insured:

print("Patient is insured.")

* + **Passing Arguments**: pid4343.say\_if\_minor("April", insured=True) passes April as a positional argument and insured=True as a keyword argument, enhancing method functionality.
  + **Flexible Arguments**: Additional arguments allow customization for specific needs without altering the method’s primary purpose.

**Things to Remember**

* **Define Methods Inside Classes**: Use def within the class to define methods, which operate on instance attributes.
* **Self Parameter**: The self keyword refers to the instance itself, allowing methods to access instance-specific data.
* **Call Methods with Dot Notation**: Access methods on instances using dot notation, e.g., instance\_name.method\_name(), to keep code clean and attribute handling automatic.