Methods and Parameters

In the dynamic realm of data science, effective data handling involves not just organizing data using classes but also utilizing methods and parameters within those classes.

Methods in Python Classes

Methods in Python classes are like little helpers that can perform specific tasks. They are functions tied to your data, bringing order and functionality.

The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.

Let's create a simple class to represent a dataset and add methods for basic data calculations.

Python3

```python
class DataSet:
    def __init__(self, data):
        self.data = data

    def calculate_mean(self):
        return sum(self.data) / len(self.dat

    def calculate_median(self):
        sorted_data = sorted(self.data)
        n = len(self.data)
        middle = n // 2
        if n % 2 == 0:
            return (sorted_data[middle - 1]
        else:
            return sorted_data[middle]
```

def __init__(self, data) is the initializer or constructor. It is executed when a new object of the class is created.

the 'calculate_mean' and calculate_median methods help us find the mean and median of our dataset.

Calling Python class methods

To call a class method, you use the class name, followed by a dot, and then the method name like this:

Python3

```python
ClassName.method_name()
```

Parameters

Python classes have methods, which are functions that are associated with the class. These methods can take parameters, allowing you to customize their behavior based on the specific instance of the class. Let's consider a simple example:

Python3

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def voice(self, loudness):
        print(f"{self.name} speaks {'loudly'

# Creating an instance of the person class
p1 = Person(name="John", age=40)

# Using the voice method with parameters
p1.voice(loudness=True)
```

```
John speaks loudly!
```

In this example:

- The __init__ method takes two parameters, name and age, which are used to initialize the attributes of the Person class (self.name and self.age).
- The voice method takes an additional parameter, loudness, allowing you to specify how loudly the person should speak.

Default Parameters

You can also assign default values to parameters in methods or constructors. If a value for a parameter is not provided during the method call or object creation, the default value will be used.

Dash
All
Articles
Videos
Probler
Quiz
« Prev
Next »

Python3

```python
class Circle:
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

# Creating instances with and without specif
small_circle = Circle()          # Uses defa
large_circle = Circle(radius=5)  # Specifies

print("Area of small circle:", small_circle.
print("Area of large circle:", large_circle.
```

```
Area of small circle: 3.14
Area of large circle: 78.5
```

In this example,

- the Circle class has a default parameter for the radius in its __init__ method.

Relevance to Data Science

As you step into the world of data science, methods and parameters become your crafty companions. They allow you to tailor your tools for specific data tasks and make your helpers adaptable and versatile. Let's extend our dataset class to have a method that detects outliers with a customizable threshold.

Python3

```python
class DataSet:
    def __init__(self, data):
        self.data = data

    def detect_outliers(self, threshold=2):
        mean_value = sum(self.data) / len(se
        std_dev = (sum((x - mean_value) ** 2
        return [x for x in self.data if abs(

# Usage
data_set = DataSet([15, 20, 25, 30, 100])
outliers = data_set.detect_outliers(threshol
```

Let's imagine you're dealing with different datasets, and you want to use the same method for different purposes.

Python3

```python
another_data_set = DataSet([50, 60, 70, 80,

# Using the Same Method with Different Parame
outliers_another = another_data_set.detect_ou
```

Here, the detect_outliers method is reused for another dataset, showcasing its adaptability.

Marked as Read