# **🧠 Python OOP Guide: Instance Method vs Class Method vs Static Method**

## **🔧 What Are Utility / Helper Functions?**

A **utility/helper function** is a reusable, small function that performs **one specific task** and doesn't depend on any object (self) or class (cls) data.

### **✅ Example:**

class StringTools:

@staticmethod

def to\_upper(text):

return text.upper()

print(StringTools.to\_upper("gowtham")) # 👉 GOWTHAM

📌 This is a **pure utility** — doesn't need class or object data.

## **🧱 3 Types of Methods in Python OOP**

| **Method Type** | **Decorator** | **First Parameter** | **Can Access Object Data?** | **Can Access Class Data?** | **Use Case** |
| --- | --- | --- | --- | --- | --- |
| **Instance** | None | self | ✅ Yes | ✅ (indirectly via class) | Object-specific behavior |
| **Class** | @classmethod | cls | ❌ No | ✅ Yes | Class-wide configs, alt constructor |
| **Static** | @staticmethod | ❌ None | ❌ No | ❌ No | Utility/helper functions |

## **🔁 Example for Each Method Type**

### **✅ Instance Method — works on object (self)**

class Person:

def \_\_init\_\_(self, name):

self.name = name

def greet(self): # Instance method

print(f"Hello, I'm {self.name}")

p = Person("Gowtham")

p.greet() # 👉 Hello, I'm Gowtham

📌 Uses self.name (object-level data)

### **✅ Class Method — works on class (cls)**

class Company:

company\_name = "OpenAI"

@classmethod

def set\_company(cls, name):

cls.company\_name = name

Company.set\_company("Google")

print(Company.company\_name) # 👉 Google

📌 Uses cls to modify class-level data

### **✅ Static Method — general utility (no self or cls)**

class MathTools:

@staticmethod

def is\_even(n):

return n % 2 == 0

print(MathTools.is\_even(10)) # 👉 True

📌 No access to object or class — pure utility

## **❓Why Not Use Instance or Class Method Always?**

Because sometimes:

* You just need a **tool function** (like is\_even(), to\_upper())
* You don’t need to carry the weight of self or cls
* Static methods **make your class cleaner**

## **❌ Trying to access object data in classmethod or staticmethod**

class Test:

def \_\_init\_\_(self, name):

self.name = name

@classmethod

def show\_name\_cls(cls):

try:

print(self.name)

except Exception as e:

print("Class Method Error:", e)

@staticmethod

def show\_name\_static():

try:

print(self.name)

except Exception as e:

print("Static Method Error:", e)

t = Test("Gowtham")

t.show\_name\_cls() # ❌ Class Method Error: name 'self' is not defined

t.show\_name\_static() # ❌ Static Method Error: name 'self' is not defined

📌 self is not available in class/static methods!

## **🚀 Advantages of Inheritance**

1. **Code Reusability**: Common logic from the parent can be reused in multiple child classes.
2. **Scalability**: Easily extend or add new features by creating new subclasses.
3. **Maintainability**: Update code in one place (parent class) to reflect in all child classes.
4. **Logical Grouping**: Helps organize similar objects in a clear, hierarchical way.
5. **DRY Principle**: "Don’t Repeat Yourself" — reduces duplicate code.
6. **Supports Polymorphism**: Child classes can override methods for custom behavior.

## **🎯 Summary Table**

| **Feature** | **Instance Method** | **Class Method** | **Static Method** |
| --- | --- | --- | --- |
| Needs object | ✅ Yes | ❌ No | ❌ No |
| Needs class | ✅ Yes | ✅ Yes | ❌ No |
| Decorator required? | ❌ No | ✅ Yes | ✅ Yes |
| Common use | Object behavior | Alternate constructors / class config | Utilities (math, format) |

### **About the Author**

**Gowtham SB** is a **Data Engineering expert, educator,** **and content creator** with a passion for **big data technologies, as well as cloud and Gen AI** . With years of experience in the field, he has worked extensively with **cloud platforms, distributed systems, and data pipelines**, helping professionals and aspiring engineers master the art of data engineering.

Beyond his technical expertise, Gowtham is a **renowned mentor and speaker**, sharing his insights through engaging content on **YouTube and LinkedIn**. He has built one of the **largest Tamil Data Engineering communities**, guiding thousands of learners to excel in their careers.

Through his deep industry knowledge and hands-on approach, Gowtham continues to **bridge the gap between learning and real-world implementation**, empowering individuals to build **scalable, high-performance data solutions**.

𝐒𝐨𝐜𝐢𝐚𝐥𝐬

🎥𝐘𝐨𝐮𝐓𝐮𝐛𝐞 - https://www.youtube.com/@dataengineeringvideos

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - <https://instagram.com/dataengineeringtamil>

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - [https://instagram.com/](https://instagram.com/dataengineeringtamil)thedatatech.in

🤝𝐂𝐨𝐧𝐧𝐞𝐜𝐭 𝐟𝐨𝐫 𝟏:𝟏 - https://topmate.io/dataengineering/

💼𝐋𝐢𝐧𝐤𝐞𝐝𝐈𝐧 - https://www.linkedin.com/in/sbgowtham/

🌐𝐖𝐞𝐛𝐬𝐢𝐭𝐞 - https://codewithgowtham.blogspot.com

💻𝐆𝐢𝐭𝐇𝐮𝐛 - http://github.com/Gowthamdataengineer

💬𝐖𝐡𝐚𝐭𝐬 𝐀𝐩𝐩 - https://lnkd.in/g5JrHw8q

📧𝐄𝐦𝐚𝐢𝐥 - atozknowledge.com@gmail.com

📱𝐀𝐥𝐥 𝐌𝐲 𝐒𝐨𝐜𝐢𝐚𝐥𝐬 - <https://lnkd.in/gf8k3aCH>