Function-oriented modeling and object-oriented modeling are two approaches to software design that are used to create software systems.

Here’s a brief explanation of each approach:

Function-oriented modeling:

Function-oriented modeling is a software design approach that focuses on identifying and modeling the functions or operations that a system performs. In function-oriented modeling, the system is decomposed into a set of functions or modules, each of which performs a specific task. The functions are then interconnected to create the overall system. The main goal of function-oriented modeling is to create a system that is efficient, reliable, and easy to maintain.

Object-oriented modeling:

Object-oriented modeling is a software design approach that focuses on identifying and modeling the objects that make up a system. In object-oriented modeling, a system is decomposed into a set of objects, each of which has its own set of attributes and behaviors. The objects are then interconnected to create the overall system. The main goal of object-oriented modeling is to create a system that is modular, extensible, and reusable.

Aspect	Function-oriented modeling	Object-oriented modeling
Approach	Decomposes system into functions	Decomposes system into objects
Abstraction	Achieved by breaking down the system into smaller functions	Achieved by grouping similar attributes and behaviors into objects

What is difference between function oriented and object oriented Modeling ? Explain in detail.

Aspect	Function-oriented modeling	Object-oriented modeling
Reusability	Generally less reusable than object-oriented modeling.	Generally more reusable than function-oriented modeling.
Extensibility	Generally less extensible than object-oriented modeling.	Generally more extensible than function-oriented modeling.
Modularity	Generally less modular than object-oriented modeling.	Generally more modular than function-oriented modeling.
Coupling	Functions are generally more tightly coupled.	Objects are generally more loosely coupled.
Inheritance	Not applicable.	Supports inheritance, allowing objects to inherit attributes and behaviors from other objects.
Polymorphism	Not applicable.	Supports polymorphism, allowing objects to have multiple forms or behaviors.
Encapsulation	Not as emphasized.	Emphasized, allowing objects to hide their internal details and expose only necessary information.
Examples	COBOL, Fortran, and Pascal.	Java, C++, and Python.