

## Definition: Functor

### Functor

A **functor** is a structure-preserving mapping between [Category](#) structures.

### Formal Definition

Let  $\mathcal{C}$  and  $\mathcal{D}$  be categories. A functor  $F : \mathcal{C} \rightarrow \mathcal{D}$  consists of:

1. An **object mapping**: For each object  $A \in \text{Ob}(\mathcal{C})$ , an object  $F(A) \in \text{Ob}(\mathcal{D})$
2. A **morphism mapping**: For each morphism  $f : A \rightarrow B$  in  $\mathcal{C}$ , a morphism  $F(f) : F(A) \rightarrow F(B)$  in  $\mathcal{D}$

satisfying:

### Functor Laws

1. **Identity preservation**:  $F(\text{id}_A) = \text{id}_{F(A)}$  for all objects  $A$
2. **Composition preservation**:  $F(g \circ f) = F(g) \circ F(f)$  for all composable morphisms  $f$  and  $g$

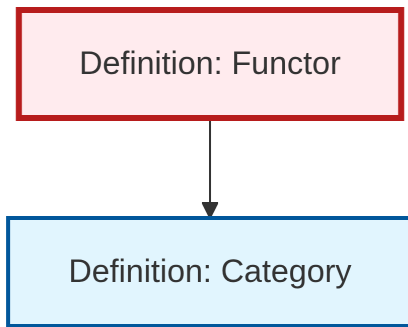
### Types of Functors

- **Covariant functor**: As defined above
- **Contravariant functor**: Reverses the direction of morphisms
- **Faithful functor**: Injective on morphism sets
- **Full functor**: Surjective on morphism sets
- **Fully faithful functor**: Both faithful and full
- **Essentially surjective**: Every object in  $\mathcal{D}$  is isomorphic to  $F(A)$  for some  $A$

### Examples

- The forgetful functor from **Grp** to **Set**
- The free group functor from **Set** to **Grp**
- The fundamental group functor from **Top** to **Grp**

## Dependency Graph



Local dependency graph