

Généricité

Tristan Crolard

Laboratoire CEDRIC
Equipe « Systèmes Sûrs »

`tristan.crolard@cnam.fr`

`cedric.cnam.fr/sys/crolard`

Generics

- ▶ The built-in collection classes are **generic** types.
- ▶ Examples:
 - `list[E]`, `MutableSequence[E]` and `Sequence[E]`
 - `dict[K, V]`, `MutableMapping[K, V]` and `Mapping[K, V]`.
- ▶ E and K, V are called **type variables** or **type parameters**.
- ▶ Generic types have **one or more type parameters**, which represent arbitrary types.
- ▶ A generic type needs to be instantiated with actual types.
- ▶ For example, `list[E]` can be instantiated as `list[str]` where type variable E is replaced by `str`.
- ▶ Similarly, `dict[K, V]` can be instantiated as `dict[int, str]` where type variables K and V replaced by `int` and `str`.

Generic function

- ▶ A function can also be generic.
- ▶ The type variables need to be specified when the function is defined.
- ▶ Instantiation of generic functions is implicit.
- ▶ Example:

```
>>> def length[A](l: list[A]) -> int:  
    count: int = 0  
    for _ in l:  
        count += 1  
    return count
```

```
>>> length([1, 2, 3])
```

3

```
>>> length(["A", "B", "C", "D"])
```

4