

**D A T A**  
**S T R U C T U R E S**



- Conjunto o colecciones de datos / forma de guardar y gestionar datos
- Siendo parte de ellas: los valores, las relaciones y operaciones o funciones
- Python: Lists, Tuples, Sets & Dictionaries

[HTTPS://DOCS.PYTHON.ORG/3/TUTORIAL/DATASTRUCTURES.HTML#](https://docs.python.org/3/tutorial/datastructures.html#)

# LISTS

- Conjunto o colecciones ordenada de datos index-based
- Almacena los datos de manera contigua (uno al lado del otro)
- Métodos, interacciones, acceso a valores etc. también son index-based
- Coste-Búsqueda lineal
- Acceso constante
- "Un solo tipo de dato"

# LISTS COMPREHENSION

```
[__ for __ in __]  
[__ for __ in __ if __]
```



# LISTS COMPREHENSION

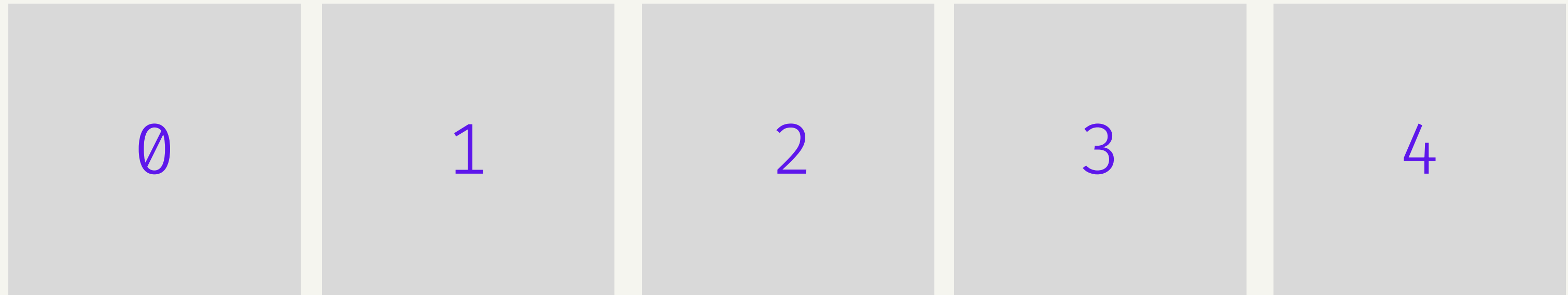


```
[element for element in elements_list]  
[element for element in elements_list if condition]
```

# DICTIONARIES

- Conjunto o colecciones de datos asociativa **key-value pairs**
- Almacena los datos en diferentes puntos de memoria
- Métodos, interacciones, acceso a valores etc. son convertidos a un índice/devuelven un **pointer**
- Acceso constante

# LISTS



# DICTIONARIES





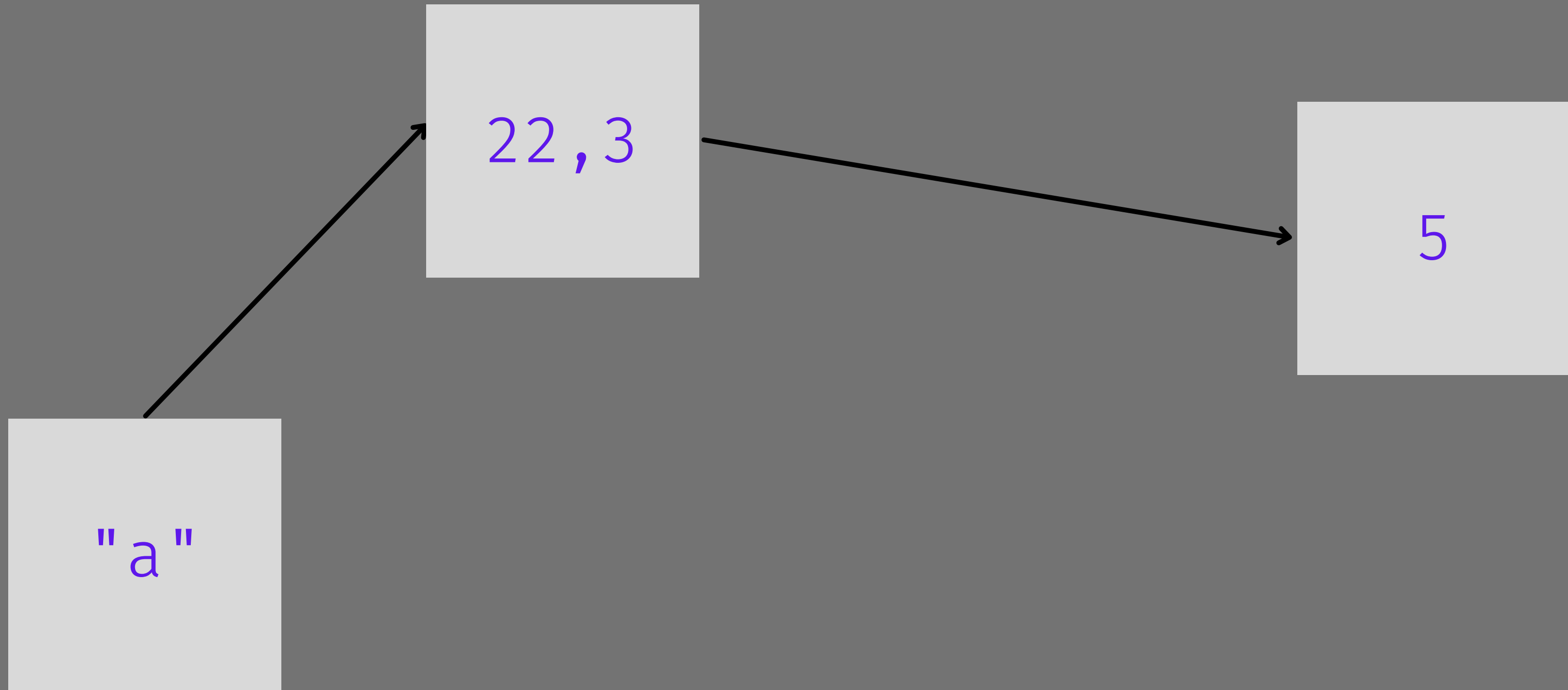
# TUPLES

- Conjunto o colecciones ordenada de datos e INMUTABLE
- Más rápida y segura
- Estática
- Coste-Búsqueda lineal
- Pueden ser utilizados como *keys* en un diccionario

# SETS

- Conjunto o colecciones desordenadas de datos
- NO admiten valores duplicados
- No soporta búsqueda por índices
- Iterables

# MEMORY



**M O D U L E S**

- Fichero Python que "permite" importar datos de un archivo a otro
- Permiten reducir el tamaño de los ficheros
- Esquematizan nuestros proyectos
- Modulos built-in & externos pip
- Se puede especificar qué queremos traer de cada uno de ellos from

```
import module_name  
import module_name as alias  
from module_name import specific
```



**pip** (3.4^ as default)



```
python3 -m pip install package_name
```

# venv

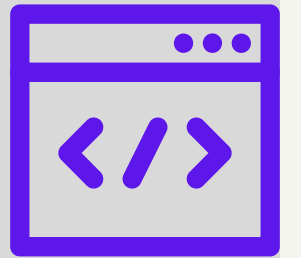
```
python3 -m venv name_environment  
source name_environment/bin/activate  
pip install requests
```





## venv W10

```
python -m venv name_environment  
source name_environment\Scripts\activate.bat  
pip install module_package
```



## venv (W10)

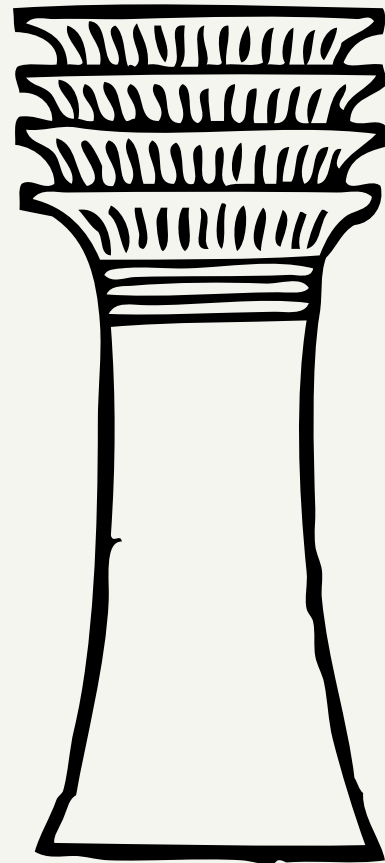
```
python -m venv name_environment  
name_environment\Scripts\activate.bat  
pip install module_package
```



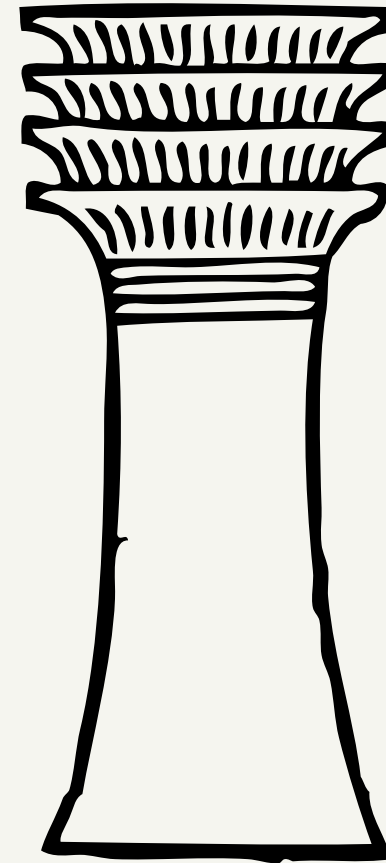
**O O P**

- Primer language OOP Simula (1967)
- Estructuración del código
- Reduce la inter-dependencia (SRP- SOLID principles)
- "De nodo a nodo"

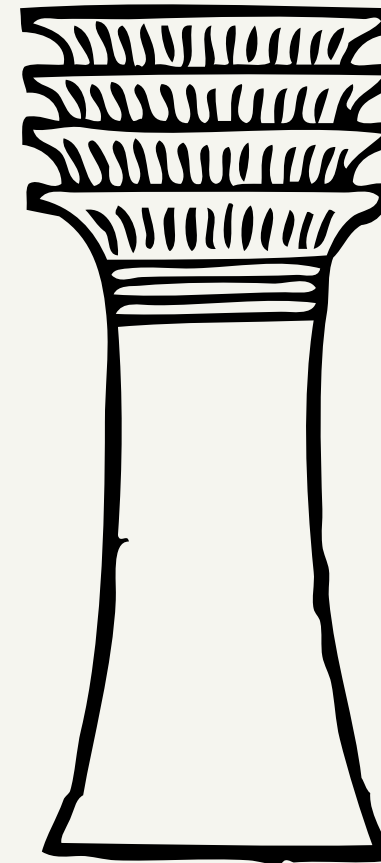
Abstraction



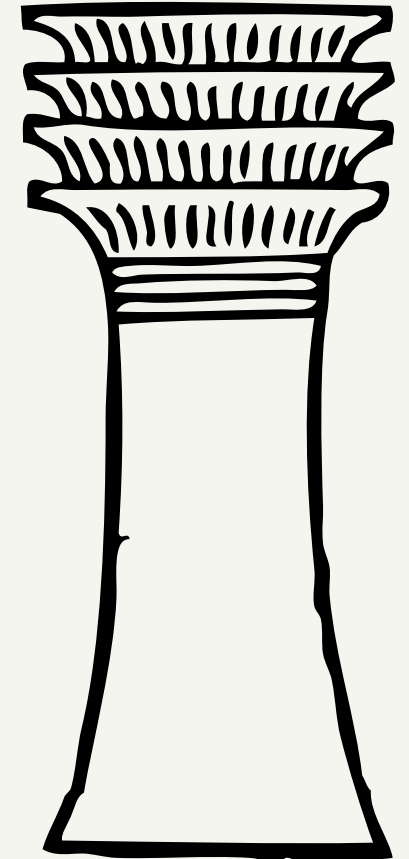
Encapsulation



Inheritance



Polymorphism



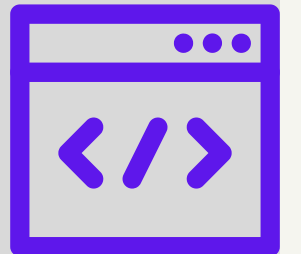
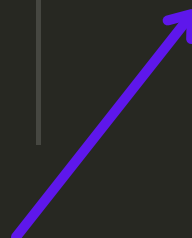
# INSTANCE ATTRIBUTES

```
class name_of_class:  
    def __init__(self, attribute):  
        self.attribute = attribute
```



# SELF

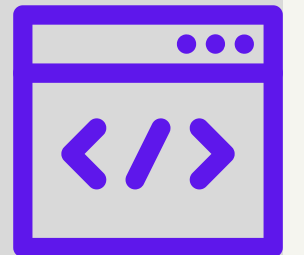
```
class Human:
    def __init__(self, specie, years_on_earth):
        self.specie = specie
        self.years_on_earth = years_on_earth
sapiens = Human("Sapiens", 250000)
```



```
sapiens.specie = "Sapiens"
sapiens.years_on_earth = 250000
```

## INSTANCE METHODS

```
class name_of_class:  
    def __init__(self, attribute):  
        self.attribute = attribute  
    def get_attribute(self):  
        return attribute
```





# CLASS ATTRIBUTES

```
class name_of_class:  
    class_attribute = attribute
```



# CLASS METHODS

```
class name_of_class:  
    @classmethod  
    def set_attribute(cls, value)  
        cls.attribute = value
```



# STATIC METHODS

```
class name_of_class:  
    @staticmethod  
    def is_true(value):  
        return true
```



# @property

```
class name_of_class:  
    @property  
    def work_with_attributes(self):  
        return properties
```



# INHERITANCE

```
class Child(Parent):  
    def __init__(self, PA, CA):  
        super().__init__(PA)  
        self.CA = CA
```



PA = parent attributes  
CA = child attributes

# DUNDER

- Nombre dado por double underscores
- Cuando son llamados dentro de una clase, modifican el comportamiento por defecto
- Los más utilizados son `__init__`, `__repr__` y `__str__`
- `repr` y `str` devuelven strings y modifican el cómo devolverlas al llamar `print`

```
def __init__(self, properties):  
    self.property = property  
  
def __str__(self):  
    return f"{self.property}"
```



**E R R O R S**

# TRY / CATCH

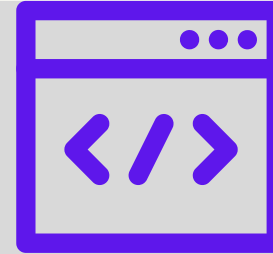
```
try:  
    an error  
catch type_of_error:  
    print("Something went wrong")
```



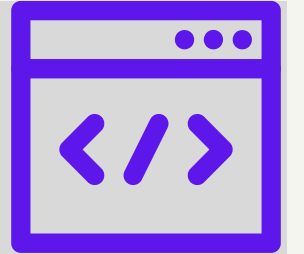


# COMMON ERRORS

IndexError



Not Found Index



KeyError

Not Found Key

NameError

Undefined

SyntaxError

Raised by Parser

ValueError

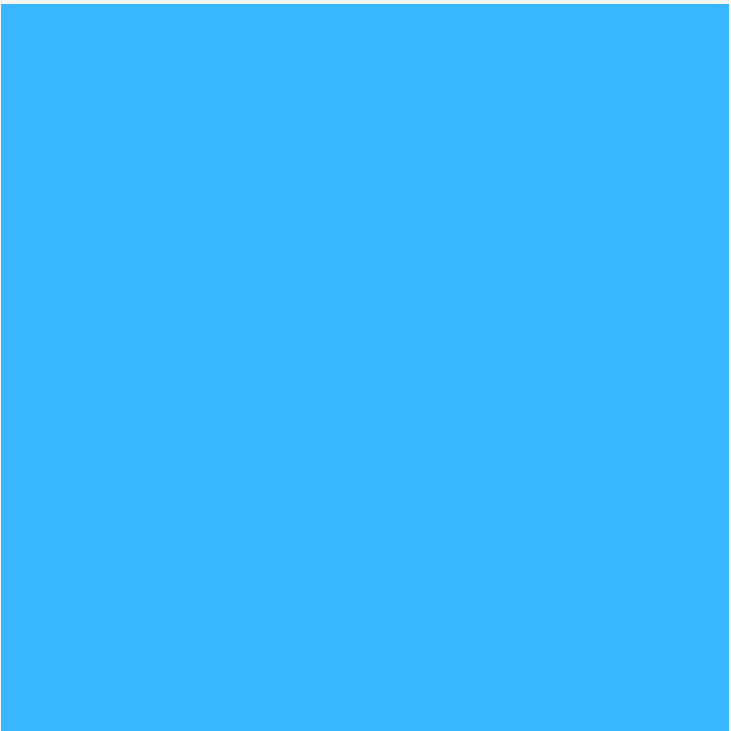
Improper Value

TypeError

Improper Type

DNS server

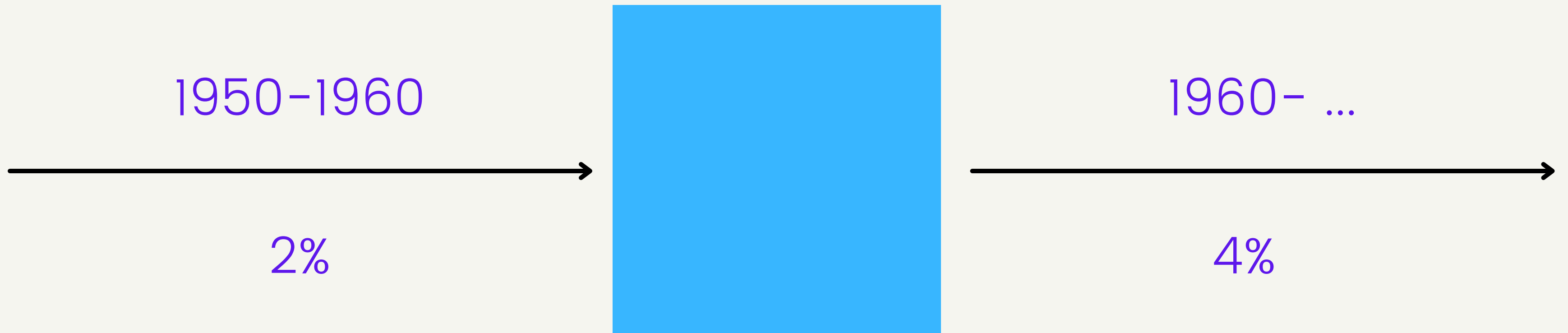
www.google.com



25.985.321.1



# Crecimiento anual



```
@classmethod  
mod_anual_growth()
```