











# Pydantic y tipado

Una intro a lo nuevo en Python

#### Acerca del disertante



#### Francisco Maurino



Técnico electrónico y actual estudiante de último año de ingeniería en sistemas de información.

Amante de la electrónica y la programación en todas sus formas, actualmente se desempeña como desarrollador fullstack en Pabex.

Contacto: j.f.maurino@gmail.com

## Índice



- Python, tipado dinámico y Typescript
- Type hints y typing module
- Type checking con mypy
- Dataclasses y Pydantic
- Implementaciones



Python es un lenguaje de tipado dinámico, esto significa que:

- El intérprete se encarga de controlar los tipos de variables en tiempo de ejecución.
- Una variable puede cambiar de tipo durante su ciclo de vida.

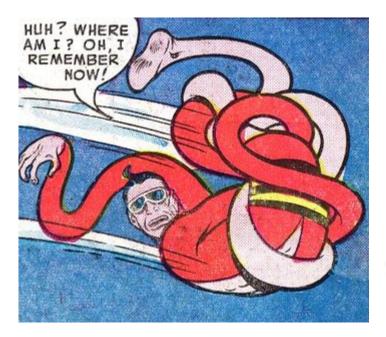




- Flexible
- Fácil o más sencillo
- Permite desarrollar más rápido

Pero...







"It hurts when I do this ... "



## "Tipado estático cuando sea posible, tipado dinámico cuando sea necesario"

- Erik Meijer & Peter Drayton - Microsoft

Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages



### **Typescript:**

Typescript se describe como "Javascript con sintaxis para tipado".

En pocas palabras, le da a Js soporte para una mejor integración con tu editor, detectando errores de forma temprana y fácil.

## Type hints y typing

## Type hints y typing



### Type hints

- No tienen ningún efecto en tiempo de ejecución
- Si se desea detectar errores de tipos se debe utilizar un type checker.
- Ayudan a documentar el código
- Mejorar la integración con IDEs y linters.

## Type hints y typing



### typing module

Este módulo proporciona soporte en tiempo de ejecución para anotaciones de tipado, tal y como se especifica en PEP 484, PEP 526, PEP 544, PEP 586, PEP 589, y PEP 591.

## Type checking con Mypy

### Type checking con Mypy



### Type checking con Mypy

Mypy es un static type checker compatible con Python.

```
def headline(align: bool):
    ...
headline("wrong")
```

## Type checking con mypy



### Otros módulos de static type checking

- Mypy
- Pycharm
- Pyre (Facebook)
- Pytype (Google)



#### **Dataclasses**

Este módulo nos provee con un decorador y funciones para agregar métodos especiales de forma automática a nuestras clases.



#### **Dataclasses**

#### Características:

- Generación de \_\_init\_\_, \_\_repr\_\_, eq, order, hash.
- Frozen data
- y más...



### **Pydantic**

Muy similar a dataclasses pero haciendo foco principalmente en:

- Data validation
- Data conversion

#### Otras características:

- Errores fáciles de leer
- (De)serialización
- Basado en clases
- JSON Schemas
- Soporte para ORM
- Parsing
- Config class



### **Pydantic**

Otras características interesantes:

- ORM mode
- Modelos recursivos
- Custom datatypes
- PyCharm plugin
- Soporte para Mypy
- Tipos de uso diario, por ejemplo: IPv4, IPv6, Email, PaymentCardNumber, PositiveInt, etc.

## Implementaciones

### **Implementaciones**

#### **FastAPI**

- MicroFramework, similar a
   Flask
- Generación automática de documentación OpenAPI
- Muy integrado a Pydantic
- Async

```
from typing import Optional
from fastapi import FastAPI
from pydantic import BaseModel
app = FastAPI()
class Item(BaseModel):
    name: str
    price: float
    is offer: Optional[bool] = None
@app.get("/")
def read root():
    return {"Hello": "World"}
@app.get("/items/{item id}")
def read item(item id: int, q: Optional[str] = None):
    return {"item id": item id, "q": q}
@app.put("/items/{item id}")
def update item(item id: int, item: Item):
    return {"item name": item.name, "item id": item id}
```

### **Implementaciones**

### **Django Ninja**

- "MicroFramework", o app complementaria para Django
- Generación automática de documentación OpenAPI
- Muy integrado a Pydantic

```
from django.contrib import admin
from django.urls import path
from ninja import NinjaAPI
api = NinjaAPI()
@api.get("/add")
def add(request, a: int, b: int):
    return {"result": a + b}
urlpatterns = [
    path("admin/", admin.site.urls),
    path("api/", api.urls),
```

## Conclusiones

## ¿Preguntas?



### Gracias









