Modern Python in FastAPI

Type Hints: usati non per documentare ma per conversioni automatiche

Async / Await -> ASGI (Asynchronous Server Gateway Interface) servers -> scalabilita'

Classi Pydantic -> Data classes + validazioni

Integrazioni e facilitazioni con VsCode e PyCharm

Type Hints

Analisi esempio variabili e funzioni con/senza type hints

```
costo_massimo = None
vs
costo_massimo: Optional[int] = None

def calcola_costo(elementi):
vs
def calcola_costo(elementi: Iterable[Elemento]) -> int:
```

Diversa gestione di: *e.prezzo* nelle hints dell'editor

In FastAPI sono usate per le conversioni automatiche

Async / Await

Sfrutta la «potenza» della programmazione concorrente

- **Creazione task**
- **Funzioni async (con blocchi async/await)**
  - ... ***mentre aspetta, va avanti e gestisce nuovi comandi (richieste)***
- **Efficienza!**

WSGI vs ASGI

```
# WSGI (Web Service Gateway Interface, ex Gunicorn, MicroWSGI…)
def request(environ, start_response):
    r = start_response(environ)
    # ...
    return r

# ASGI
async def app(scope, receive, send):
    r = await receive(scope)
    # ...
    return await send(r, scope)
```

Sync -> WSGI
Async/await / requests in parallelo -> ASGI

Servers ASGI per sfruttare la potenza di async/await
uvicorn - https://www.uvicorn.org/

Altre risorse su ASGI

https://github.com/florimondmanca/awesome-asgi

Pydantic types

```
from pydantic import BaseModel
```

Data class in Pydantic
https://pydantic-docs.helpmanual.io/

Controlli e conversioni manuali vs automatiche usando le type annotations