HUMAITRIX

Igniting Ideas, Sparking Success

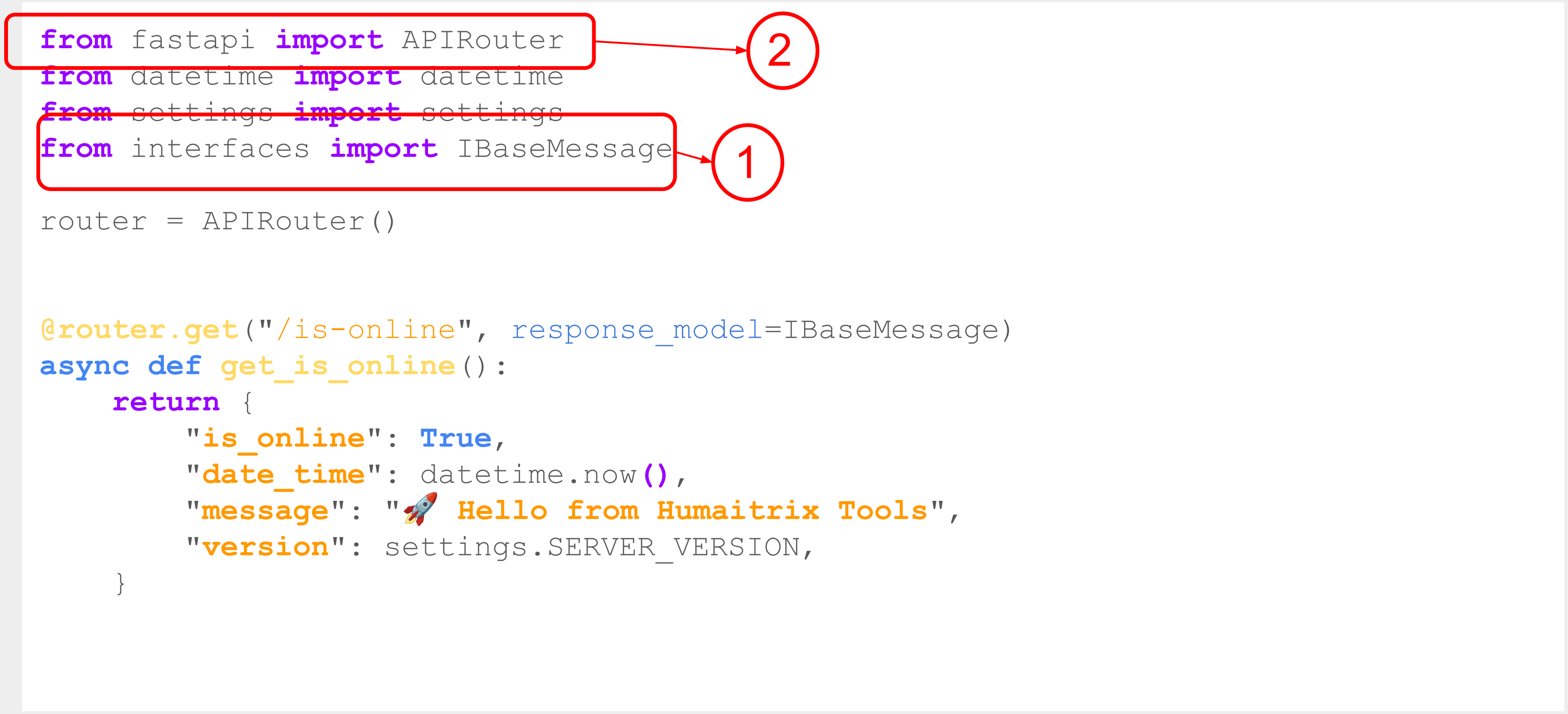
Python Class

Code Structure

HUMAITRIX

your B2B AI SaaS, workflow-free

How to Understand a Python Class



```python
from fastapi import APIRouter
from datetime import datetime
from settings import settings
from interfaces import IBaseMessage

router = APIRouter()


@router.get("/is-online", response_model=IBaseMessage)
async def get_is_online():
    return {
        "is_online": True,
        "date_time": datetime.now(),
        "message": "🚀 Hello from Humaitrix Tools",
        "version": settings.SERVER_VERSION,
    }
```

1. Importing from local packages

HUMAITRIX
your B2B AI SaaS, workflow-free

How To?

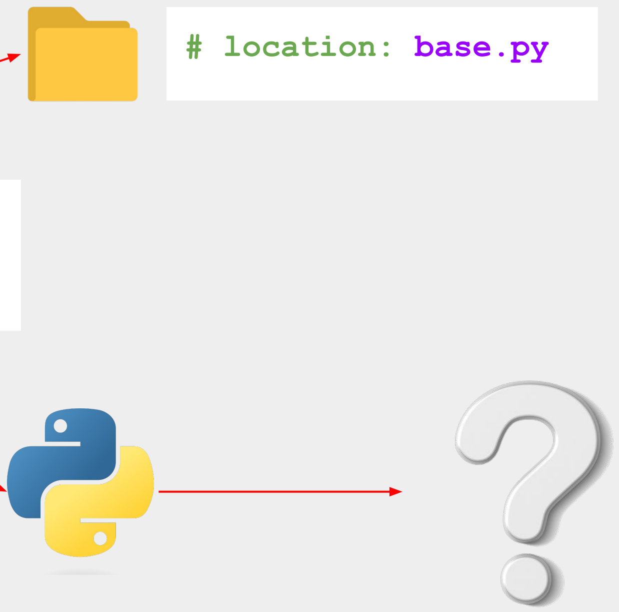
```python
# the import command...
from interfaces import IBaseMessage


    # ... is importing this:


    from datetime import datetime
    from typing_extensions import TypedDict


    class IBaseMessage(TypedDict):
        is_online: bool
        date_time: datetime
        message: str
        version: str
```

But IBaseMessage is located in base.py

How does Python do this mapping?

HUMAITRIX
your B2B AI SaaS, workflow-free

```
# command
from interfaces import IBaseMessage
```

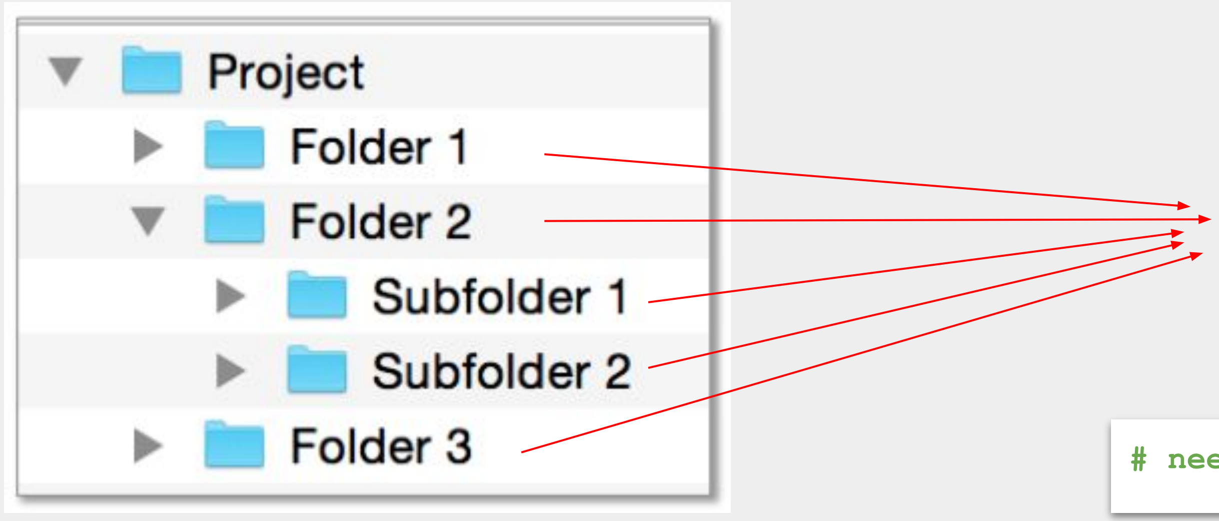


```
# location: base.py
```

HUMAITRIX
your B2B AI SaaS, workflow-free

Every subfolder is a Package

This is important to remember



packages

```
# need: __init__.py
```

__init__.py

A crazy file does the magic

HUMAITRIX
your B2B AI SaaS, workflow-free

```
# this file is super important and have the following purposes:

1.   To tell Python the folder is a package

     ○   To import from folder_name/file_name.py is necessary a __init__.py
              Example:
              your_project/
              │
              ├── apis/
              │   ├── __init__.py
              │   └── sample_api.py
              │
              ├── interfaces/
              │   ├── __init__.py
              │   └── sample_interface.py
              │
              ├── main.py
              └── requirements.txt
```

__init__.py

A crazy file does the magic

HUMAITRIX

your B2B AI SaaS, workflow-free

```
2. To map classes in a single place

     # Without mapping:

          from interfaces.base import IBaseMessage

     # But to use classes this way:

          from .interfaces import IBaseMessage


          # We do this (in the __init__.py)

          from .base import IBaseMessage

          __all__ = ["IBaseMessage"]
```

2. Importing from external packages

HUMAITRIX
your B2B AI SaaS, workflow-free

How To?

```
# Where this is coming from?

from fastapi import APIRouter
```

Breaking down the installation process...

```
# python is the compiler
# pip is the package manager, meaning that pip itself is "just" another Python
  package to manage other packages
```

Breaking down the installation process...

```
python -m ensurepip --upgrade
python -m pip install --upgrade pip
```

Python and Pip Commands

Just EXE files

```
# Python

python is the language and the compiler

pip is the package manager


# When a package is installed, it's added to the PATH of the system running the app


pip manages this process by informing python what to do
```

Pip

The Package Manager

```
# Pip

pip uses a file named requirements.txt to understand what the app needs
```

Example

```
# Pip

from fastapi imports FastAPI
```

requirements.txt

```
fastapi
```

HUMAITRIX
your B2B AI SaaS, workflow-free

main.py

Breaking down the entry point of a python application

HUMAITRIX
your B2B AI SaaS, workflow-free

```
import os
import sys
from fastapi import FastAPI
from settings import settings
from apis import apis as api_router

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.append(os.path.join(BASE_DIR, "src"))

app = FastAPI()

if __name__ == "__main__":
    import uvicorn

    uvicorn.run("main:app", host="0.0.0.0", port=settings.SERVER_PORT, reload=True)
```

uvicorn

A modern IIS

HUMAITRIX
your B2B AI SaaS, workflow-free

```
# unicorn

basic HTTP Server to manage requests
```

Example

```
from settings import settings

app = FastAPI()

if __name__ == "__main__":
    import uvicorn

    uvicorn.run("main:app", host="0.0.0.0", port=settings.SERVER_PORT, reload=True)
```

```
# settings
```

```
# localhost
```

Uvicorn & python

Breaking down the app start

HUMAITRIX
your B2B AI SaaS, workflow-free

```
# Python runs from the package name

C:\> python main.py

# Check the IF again

     if __name__ == "__main__":
         import uvicorn

         uvicorn.run("main:app" ...)


     # main:app

          main: self
          app: a variable declared somewhere
```

App

The good part begins

HUMAITRIX

your B2B AI SaaS, workflow-free

```
# main.py

    import os
    import sys
    from fastapi import FastAPI
    from settings import settings
    from apis import apis as api_router

    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    sys.path.append(os.path.join(BASE_DIR, "src"))

    app = FastAPI()
    app.include_router(api_router, prefix="/api/v2")
```

```
# where the magic happens
```

What do we know about “apis”

HUMAITRIX
your B2B AI SaaS, workflow-free

How the Folder Structure Should Look Like?

apis:

- package
- If a folder, must have a __index__.py
- If a file, must export “api_router”

__index__.py:

- Will either export apis

  Or

- Will export “__all__”

apis

Mapping routes & error handling

HUMAITRIX
your B2B AI SaaS, workflow-free

```
# main.py

     from fastapi import APIRouter, Request
     from fastapi.responses import JSONResponse
     from starlette.exceptions import HTTPException
     from .base import router as base_router
     from .audio import router as audio_router
     from tools import logger

     apis = APIRouter()
     apis.include_router(base_router)
     apis.include_router(audio_router)

     DEFAULT_INTERNAL_ERROR = JSONResponse(
         {"status_code": 500, "detail": "Internal Server Error"}
     )
```

Questions

...

HUMAITRIX
your B2B AI SaaS, workflow-free

Homework

...

HUMAITRIX
your B2B AI SaaS, workflow-free

API:

- GET api/v2/ - Hello World
-