Modern Python

through

FastAPI

and friends

Who am I?

Sebastián Ramírez

tiangolo.com

Dev at Explosion

Berlin, Germany

github.com/tiangolo

linkedin.com/in/tiangolo

twitter.com/tiangolo

Explosion created:

spaCy prodigy THiNC

I created:

FastAPI Typer

Modern python™

Currently supported versions (3.6+)

📝 f-strings

🧙 Type annotations

⏱️ async / await

⚡ Performance

👥 Community

FastAPI

- Pydantic
- HTTPX
- Starlette
- Uvicorn
- Uvloop

Typer

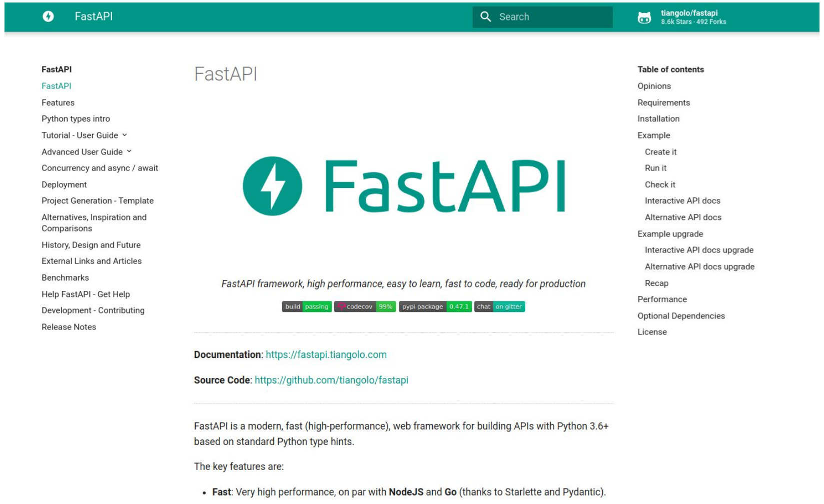
- Rich
- Click

@tiangolo

FastAPI

High performance, easy to learn, fast to code, ready for production

- Web API framework
- 23K GitHub stars (about 1K+ per month)
- Used by Microsoft, Uber, Netflix, etc.
- Performance in the top rank for Python


FastAPI
Search
tiangolo/fastapi
8.6k Stars · 492 Forks

FastAPI
- FastAPI
- Features
- Python types intro
- Tutorial - User Guide
- Advanced User Guide
- Concurrency and async / await
- Deployment
- Project Generation - Template
- Alternatives, Inspiration and Comparisons
- History, Design and Future
- External Links and Articles
- Benchmarks
- Help FastAPI - Get Help
- Development - Contributing
- Release Notes

FastAPI

FastAPI

FastAPI framework, high performance, easy to learn, fast to code, ready for production

build passing | codecov 99% | pypi package 0.47.1 | chat on gitter

Documentation: https://fastapi.tiangolo.com

Source Code: https://github.com/tiangolo/fastapi

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.

The key features are:

- **Fast**: Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic).

Table of contents

@tiangolo

Modern Python

📝 f-strings (formatted strings)

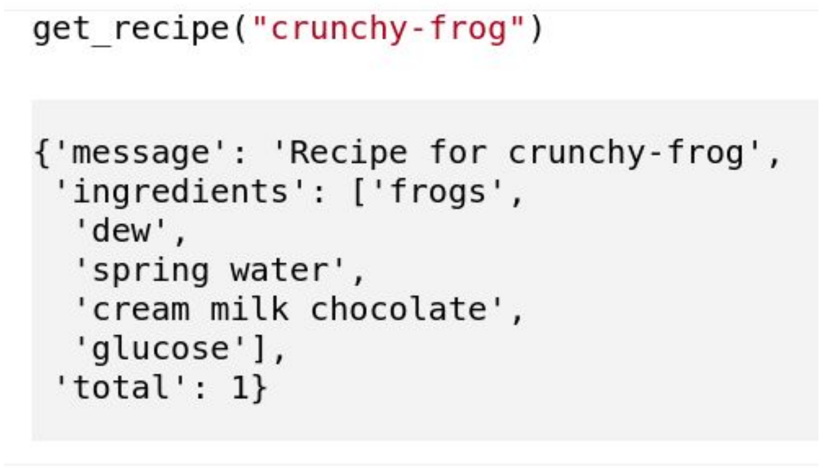
📝 f-strings (formatted strings)

```python
recipes = {
    "crunchy-frog": {
        "ingredients": [
            "frogs",
            "dew",
            "spring water",
            "cream milk chocolate",
            "glucose",
        ]
    },
    "albatross": {
        "ingredients": [
            "albatross"
        ]
    },
}
default_price = 1
```

```python
def get_recipe(name, quantity=1):
    recipe = recipes[name]
    return {
        "message": "Recipe for {name}".format(name=name),
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

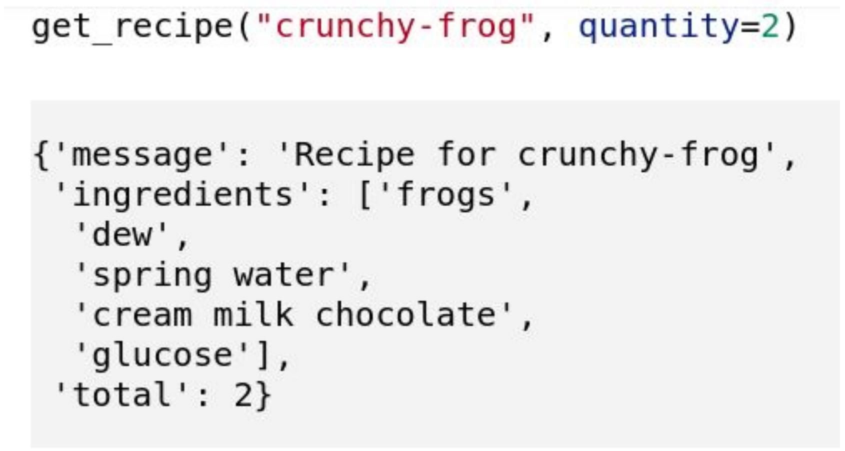
@tiangolo

📝 f-strings (formatted strings)



```python
get_recipe("crunchy-frog")
```

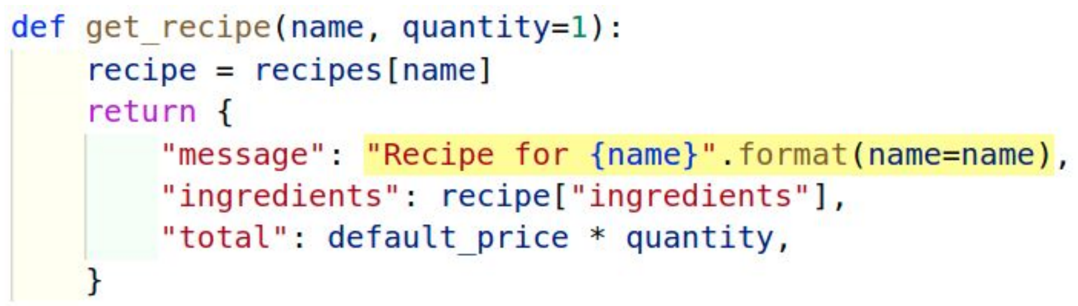
```
{'message': 'Recipe for crunchy-frog',
 'ingredients': ['frogs',
  'dew',
  'spring water',
  'cream milk chocolate',
  'glucose'],
 'total': 1}
```



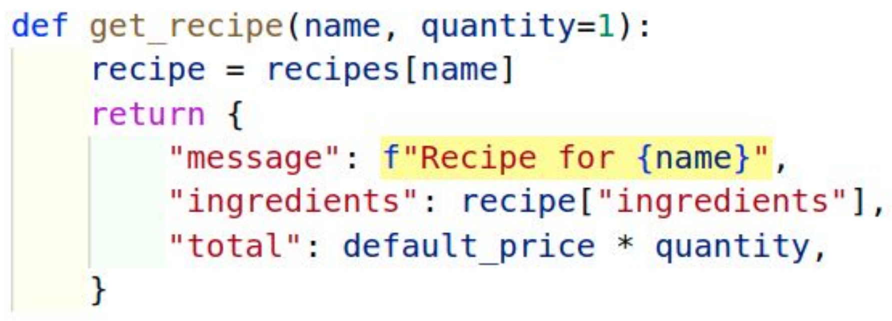
```python
get_recipe("crunchy-frog", quantity=2)
```

```
{'message': 'Recipe for crunchy-frog',
 'ingredients': ['frogs',
  'dew',
  'spring water',
  'cream milk chocolate',
  'glucose'],
 'total': 2}
```

📝 f-strings (formatted strings)



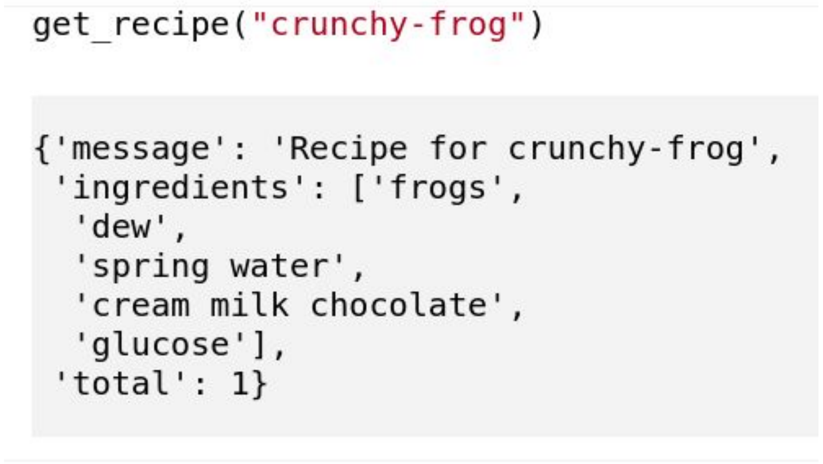
```python
def get_recipe(name, quantity=1):
    recipe = recipes[name]
    return {
        "message": "Recipe for {name}".format(name=name),
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



```python
def get_recipe(name, quantity=1):
    recipe = recipes[name]
    return {
        "message": f"Recipe for {name}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

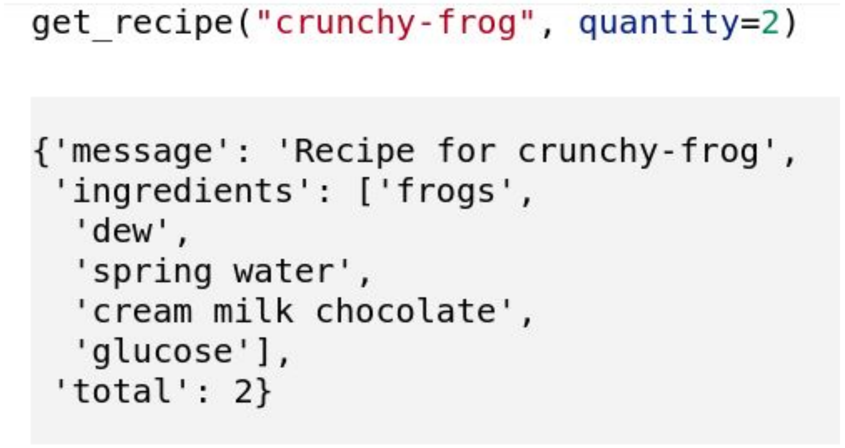
@tiangolo

📝 f-strings (formatted strings)



```python
get_recipe("crunchy-frog")
```

```
{'message': 'Recipe for crunchy-frog',
 'ingredients': ['frogs',
  'dew',
  'spring water',
  'cream milk chocolate',
  'glucose'],
 'total': 1}
```



```python
get_recipe("crunchy-frog", quantity=2)
```

```
{'message': 'Recipe for crunchy-frog',
 'ingredients': ['frogs',
  'dew',
  'spring water',
  'cream milk chocolate',
  'glucose'],
 'total': 2}
```

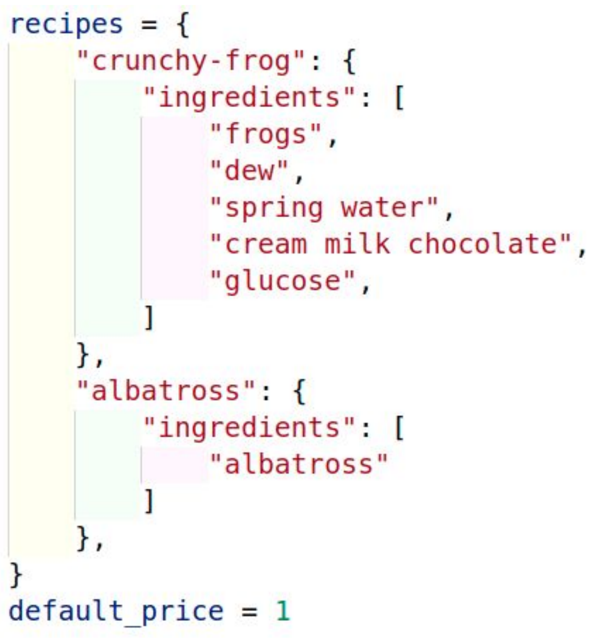
@tiangolo

Modern Python

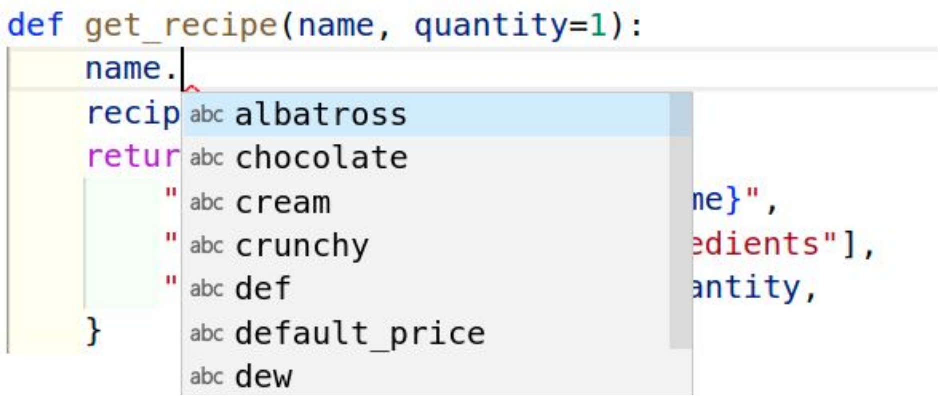
🧙‍♀️ Type annotations (type hints)

@tiangolo

🧙 Type annotations (type hints)



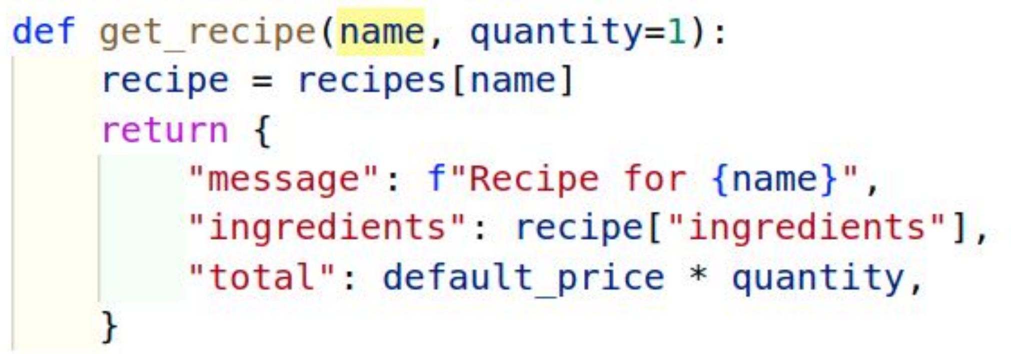
```python
recipes = {
    "crunchy-frog": {
        "ingredients": [
            "frogs",
            "dew",
            "spring water",
            "cream milk chocolate",
            "glucose",
        ]
    },
    "albatross": {
        "ingredients": [
            "albatross"
        ]
    },
}
default_price = 1
```



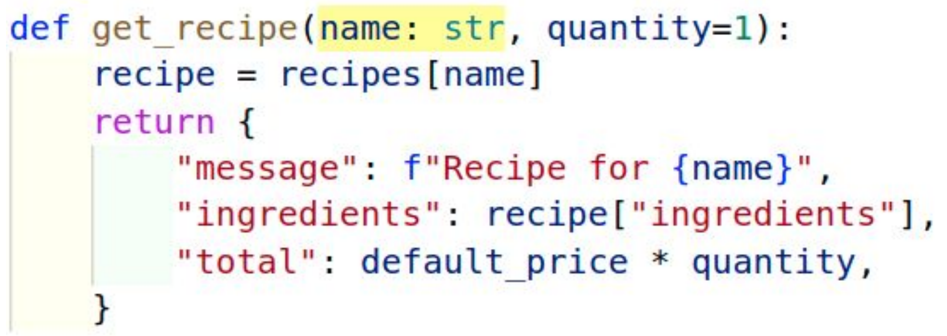
```python
def get_recipe(name, quantity=1):
    name.
    recip
    retur
        "                          me}",
        "                          edients"],
        "                          antity,
    }
```

@tiangolo

🧙‍♀️ Type annotations (type hints)



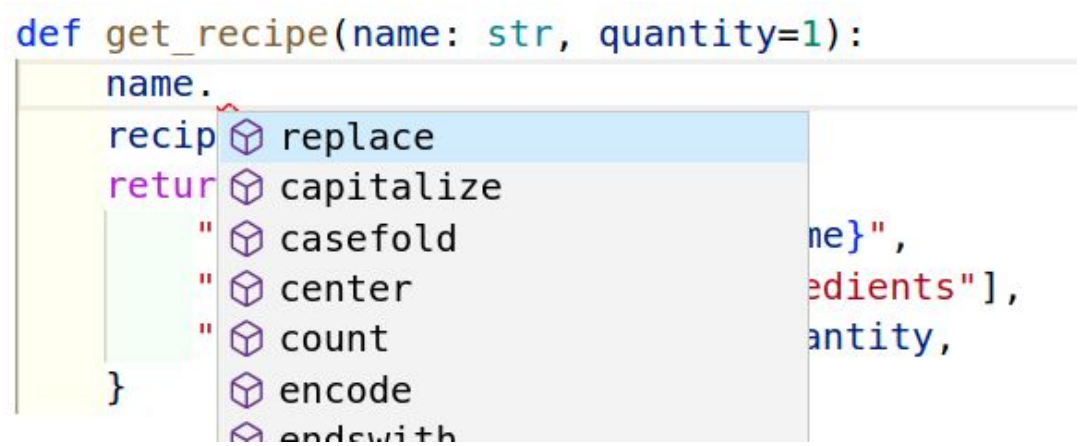
```python
def get_recipe(name, quantity=1):
    recipe = recipes[name]
    return {
        "message": f"Recipe for {name}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



```python
def get_recipe(name: str, quantity=1):
    recipe = recipes[name]
    return {
        "message": f"Recipe for {name}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

@tiangolo

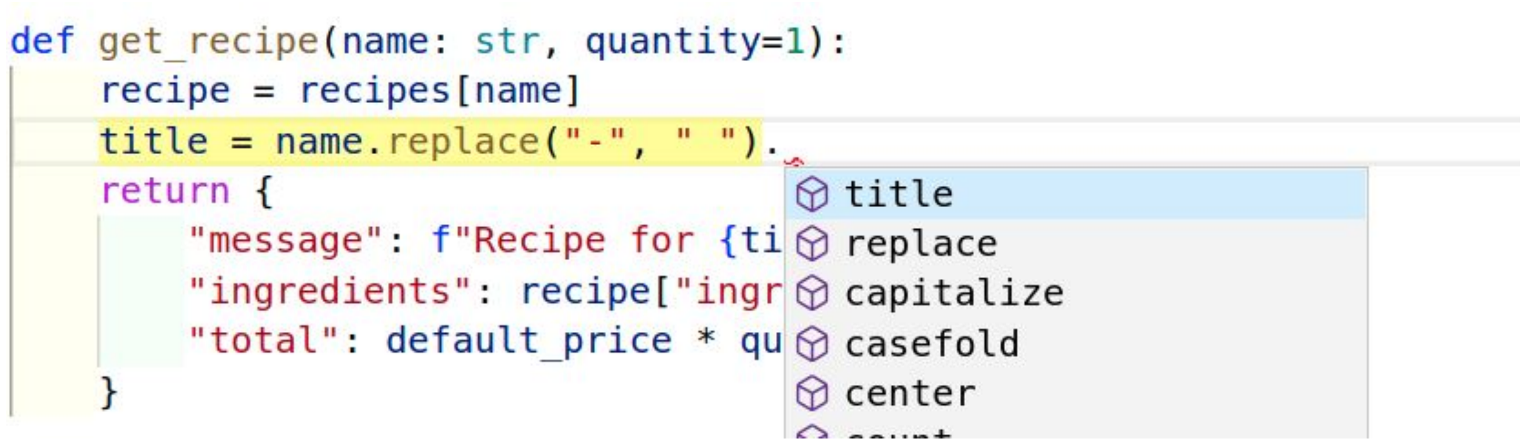
🧙 Type annotations (type hints) - autocompletion



```
def get_recipe(name: str, quantity=1):
    name.
    recip
    retur
        "                                  me}",
        "                                  edients"],
        "                                  antity,
    }
```

@tiangolo

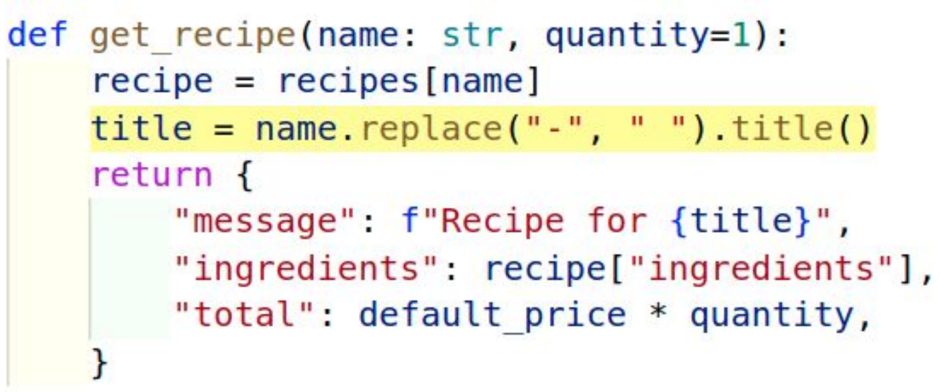
🧙 Type annotations (type hints) - autocompletion



```
def get_recipe(name: str, quantity=1):
    recipe = recipes[name]
    title = name.replace("-", " ").
    return {
        "message": f"Recipe for {ti
        "ingredients": recipe["ingr
        "total": default_price * qu
    }
```

@tiangolo

🧙 Type annotations (type hints)



```
def get_recipe(name: str, quantity=1):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

@tiangolo

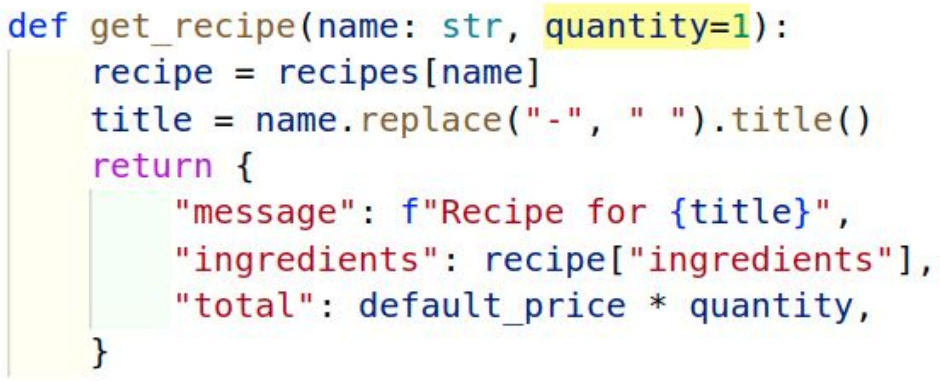
🧙 Type annotations (type hints)

```
get_recipe("crunchy-frog")
```

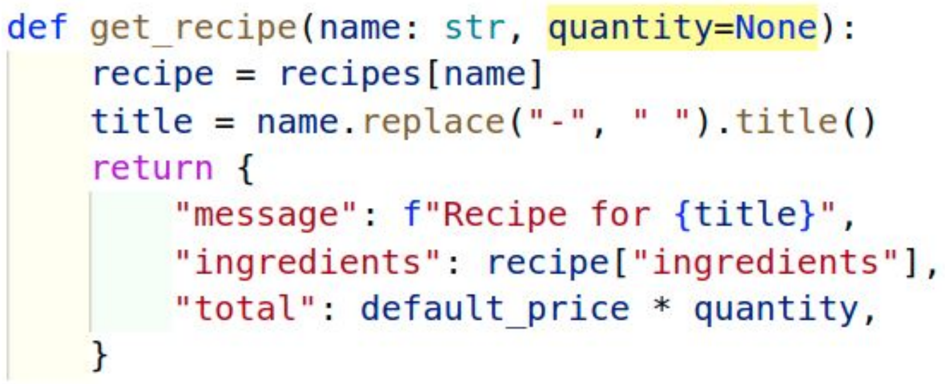
```
{'message': 'Recipe for Crunchy Frog',
 'ingredients': ['frogs',
  'dew',
  'spring water',
  'cream milk chocolate',
  'glucose'],
 'total': 1}
```

@tiangolo

🧙 Type annotations (type hints)



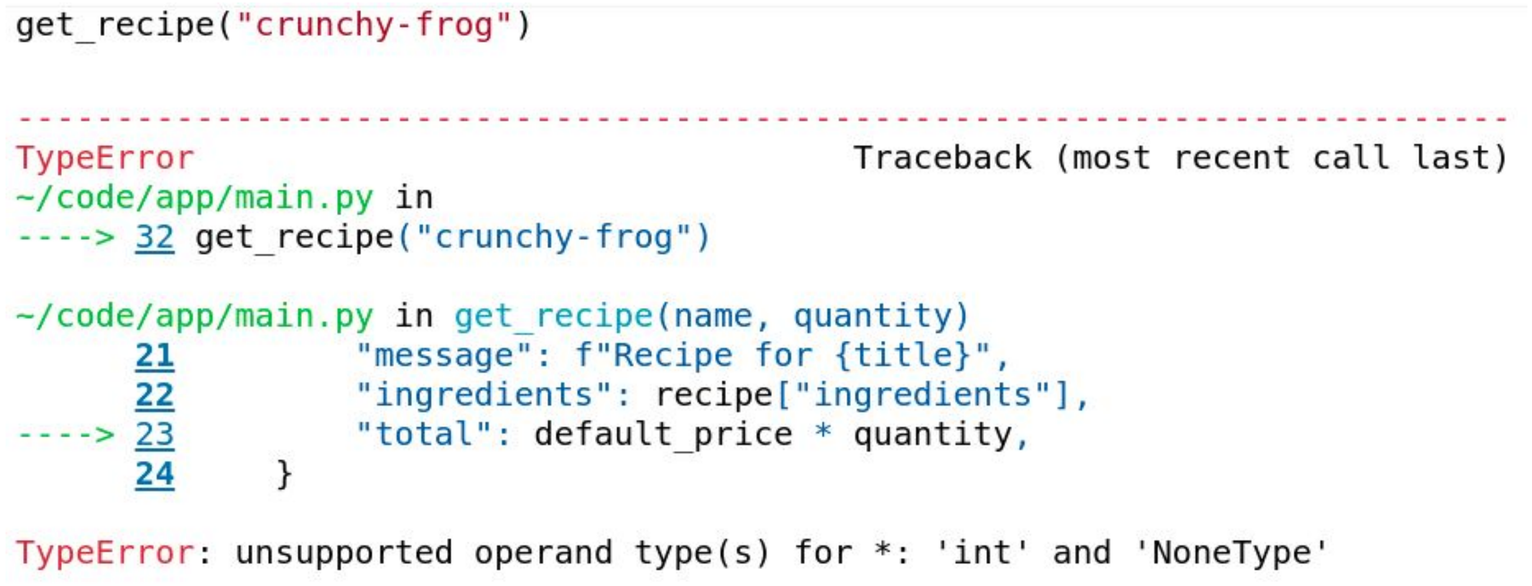
```python
def get_recipe(name: str, quantity=1):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



```python
def get_recipe(name: str, quantity=None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

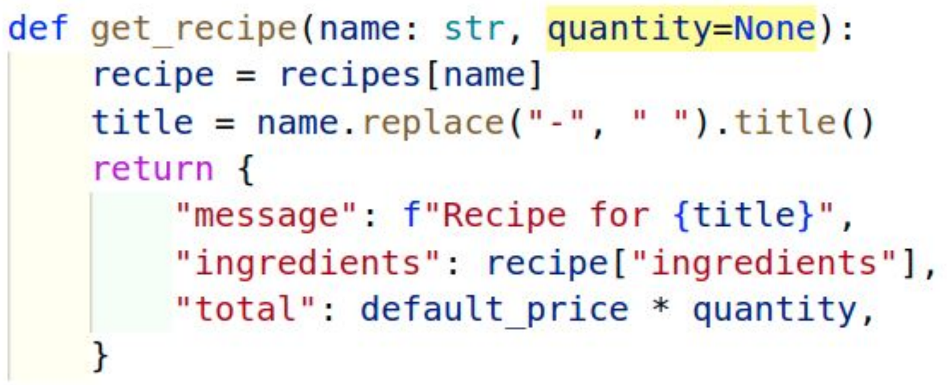
@tiangolo

🧙‍♀️ Type annotations (type hints)

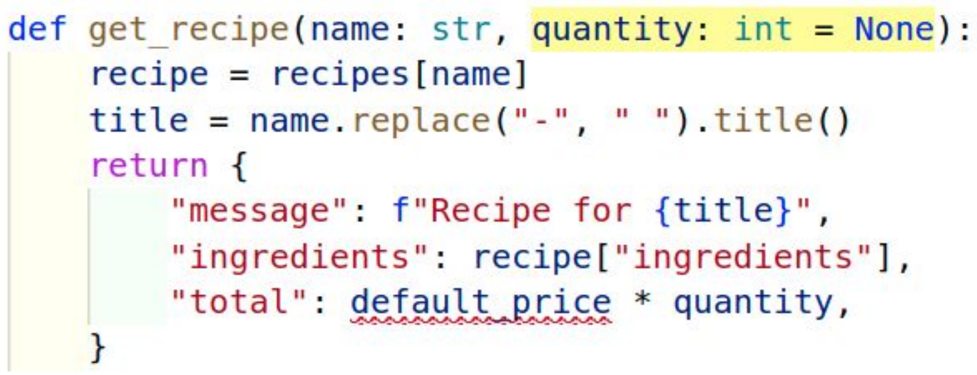


```
get_recipe("crunchy-frog")

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~/code/app/main.py in
----> 32 get_recipe("crunchy-frog")

~/code/app/main.py in get_recipe(name, quantity)
     21             "message": f"Recipe for {title}",
     22             "ingredients": recipe["ingredients"],
---> 23             "total": default_price * quantity,
     24         }

TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'
```

🧙 Type annotations (type hints) - error detection



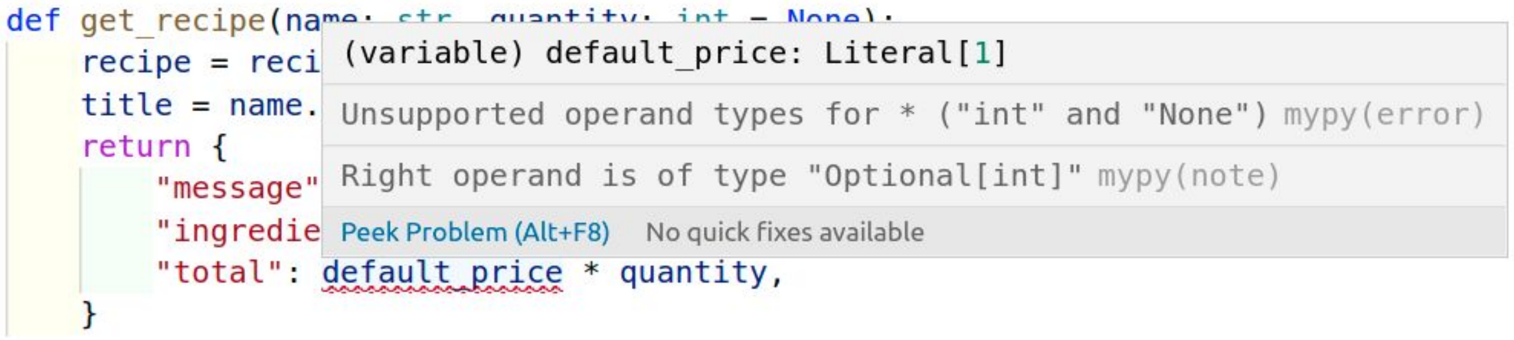
```python
def get_recipe(name: str, quantity=None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```

@tiangolo

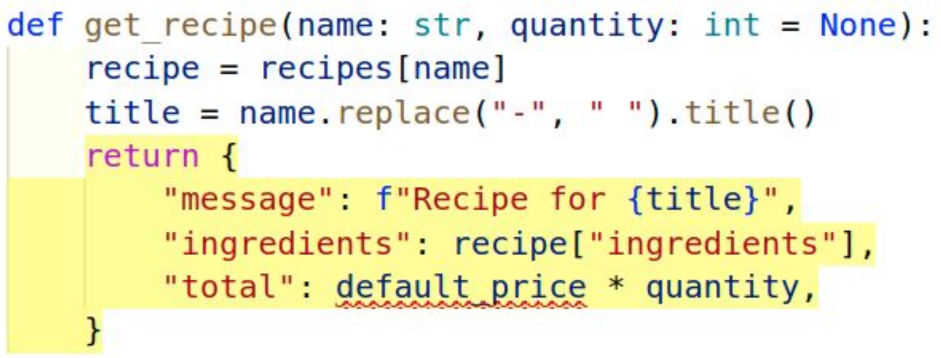
🧙 Type annotations (type hints) - error detection



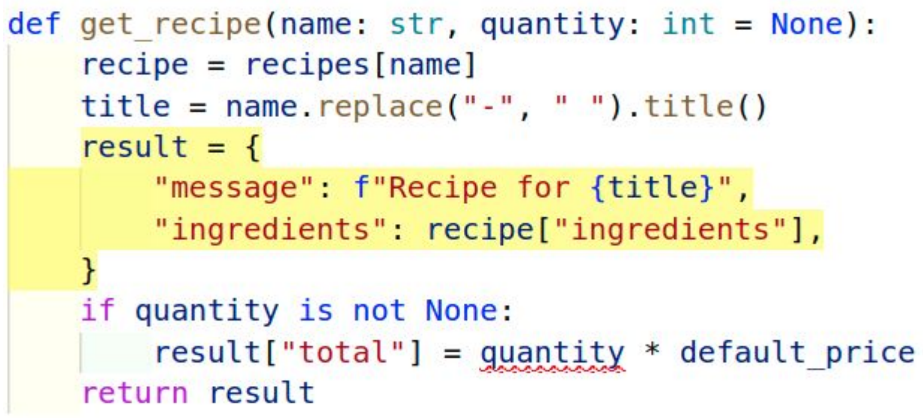
```
def get_recipe(name: str, quantity: int = None):
    recipe = reci
    title = name.
    return {
        "message"
        "ingredie
        "total": default_price * quantity,
    }
```

```
(variable) default_price: Literal[1]
Unsupported operand types for * ("int" and "None") mypy(error)
Right operand is of type "Optional[int]" mypy(note)
```

🧙 Type annotations (type hints) - error detection



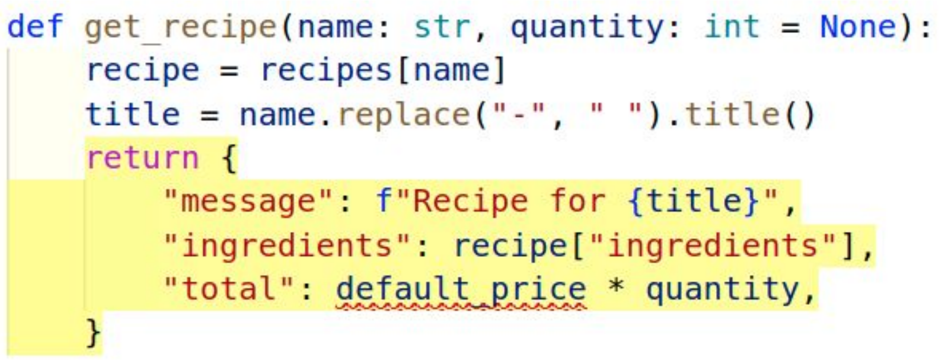
```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



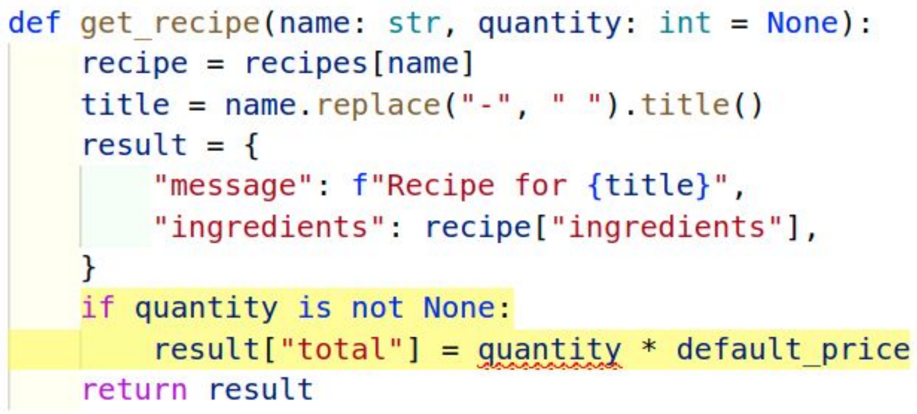
```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

@tiangolo

🧙 Type annotations (type hints) - error detection



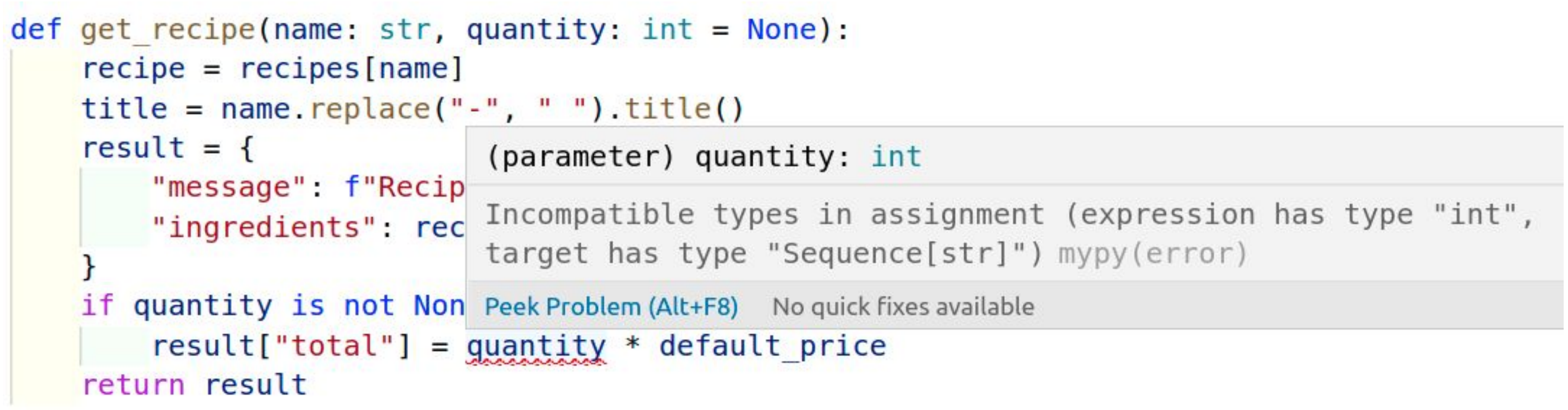
```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    return {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
        "total": default_price * quantity,
    }
```



```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

@tiangolo

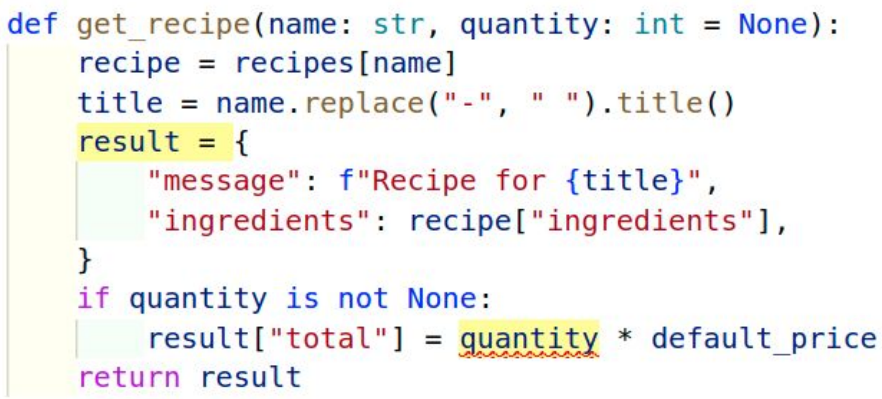
🧙 Type annotations (type hints) - error detection



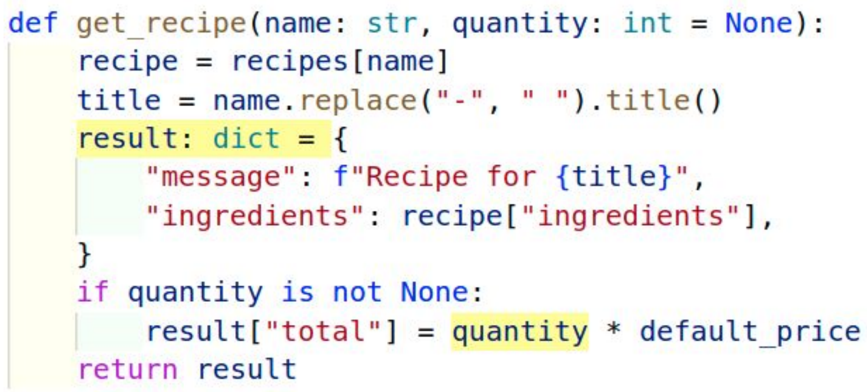
```
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result = {
        "message": f"Recip
        "ingredients": rec
    }
    if quantity is not Non
        result["total"] = quantity * default_price
    return result
```

```
(parameter) quantity: int
Incompatible types in assignment (expression has type "int",
target has type "Sequence[str]") mypy(error)
```

🧙‍♀️ Type annotations (type hints) - error detection

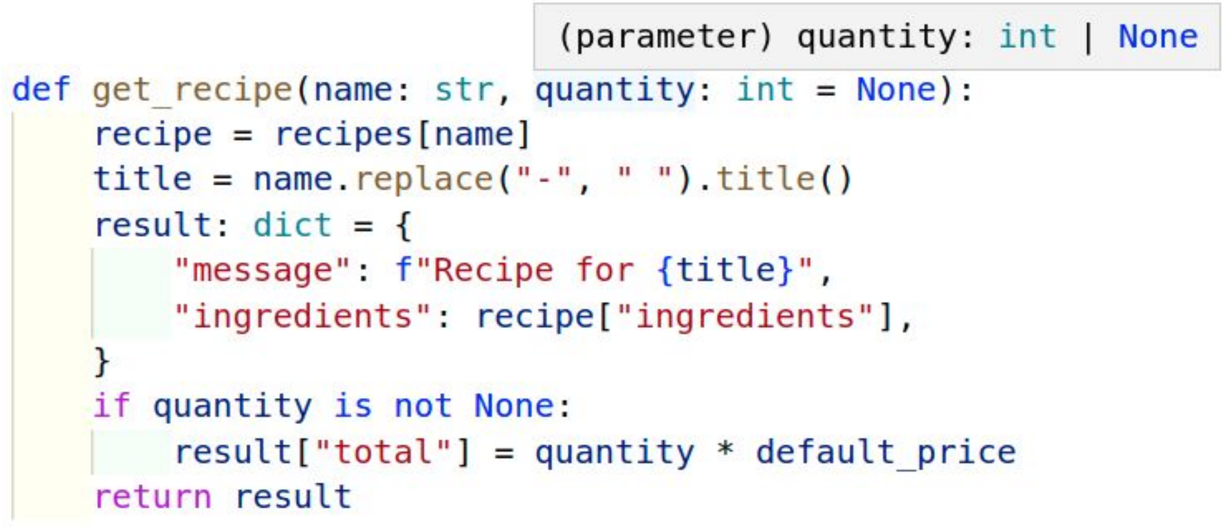


```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```



```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

🧙 Type annotations (type hints)

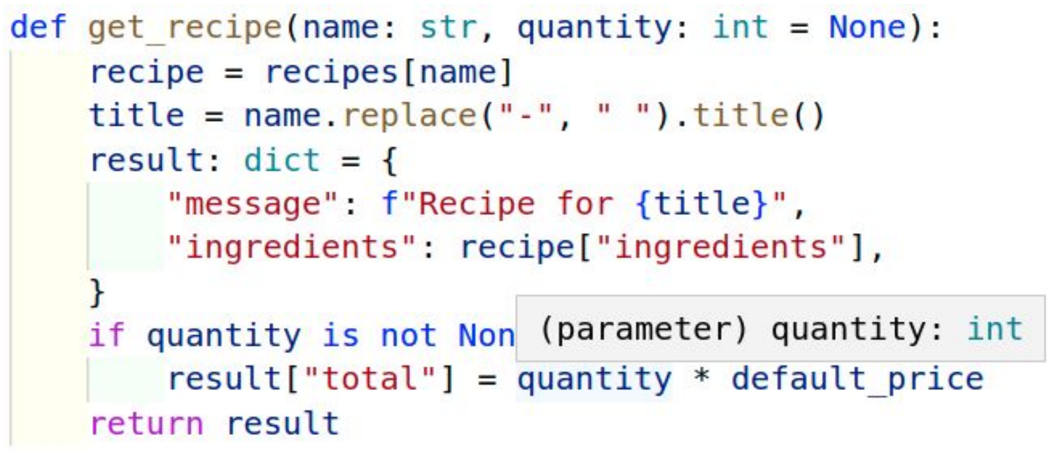


```
(parameter) quantity: int | None
```

```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

@tiangolo

🧙 Type annotations (type hints)

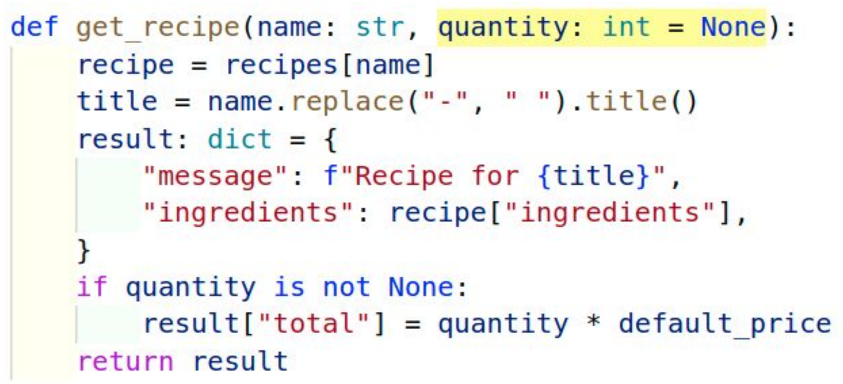


```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not Non
        result["total"] = quantity * default_price
    return result
```

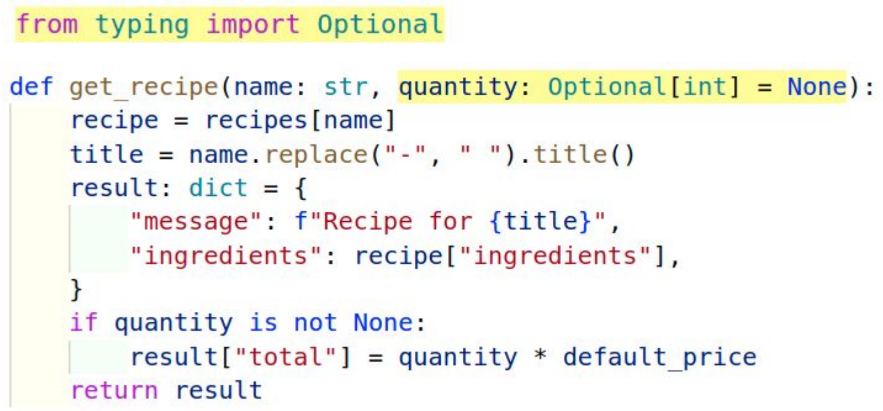
```
(parameter) quantity: int
```

@tiangolo

🧙 Type annotations (type hints)



```python
def get_recipe(name: str, quantity: int = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```



```python
from typing import Optional

def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

@tiangolo

🧙 Type annotations (type hints)

Some of the people behind:

Jukka Lehtosalo

@JukkaL

@JukkaLeh

Ivan Levkivskyi

@ilevkivskyi

@ILevkivskyi

Michael J. Sullivan

@msullivan

@msully4321

Modern Python

🧙 Type annotations with

FastAPI

@tiangolo

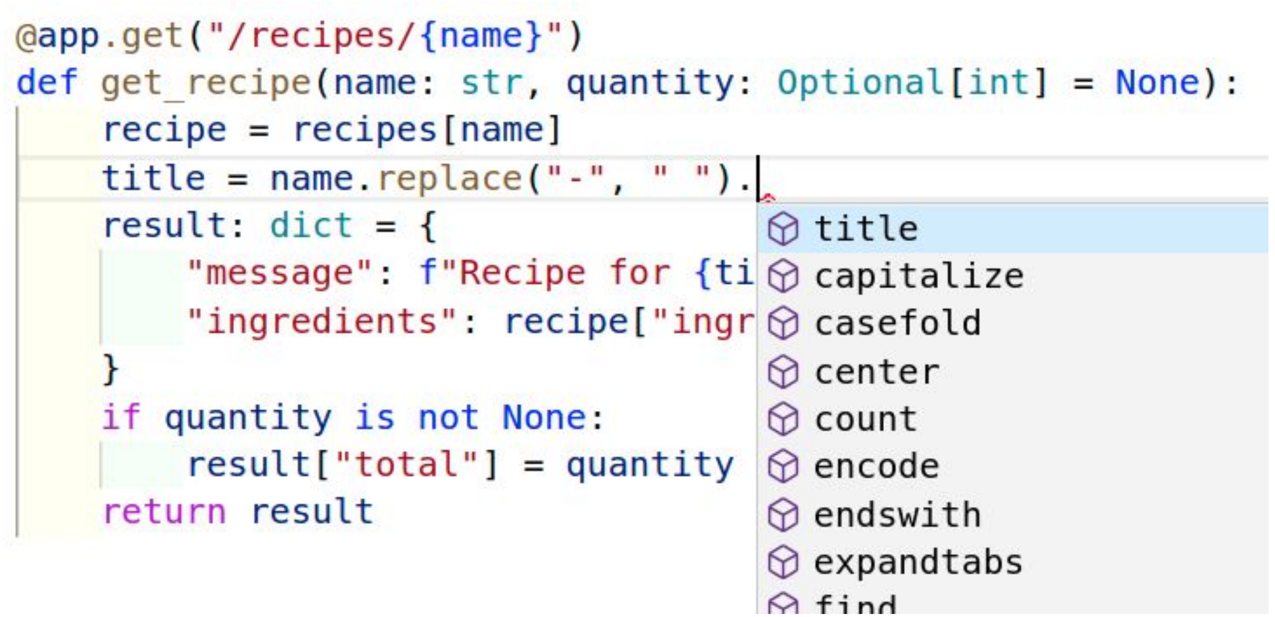
Type annotations with **FastAPI**

```python
from typing import Optional
from fastapi import FastAPI

recipes = {
    "crunchy-frog": {
        "ingredients": [
            "frogs",
            "dew",
            "spring water",
            "cream milk chocolate",
            "glucose",
        ]
    },
    "albatross": {
        "ingredients": ["albatross"]},
}
default_price = 1
```

```python
app = FastAPI()


@app.get("/recipes/{name}")
def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    return result
```

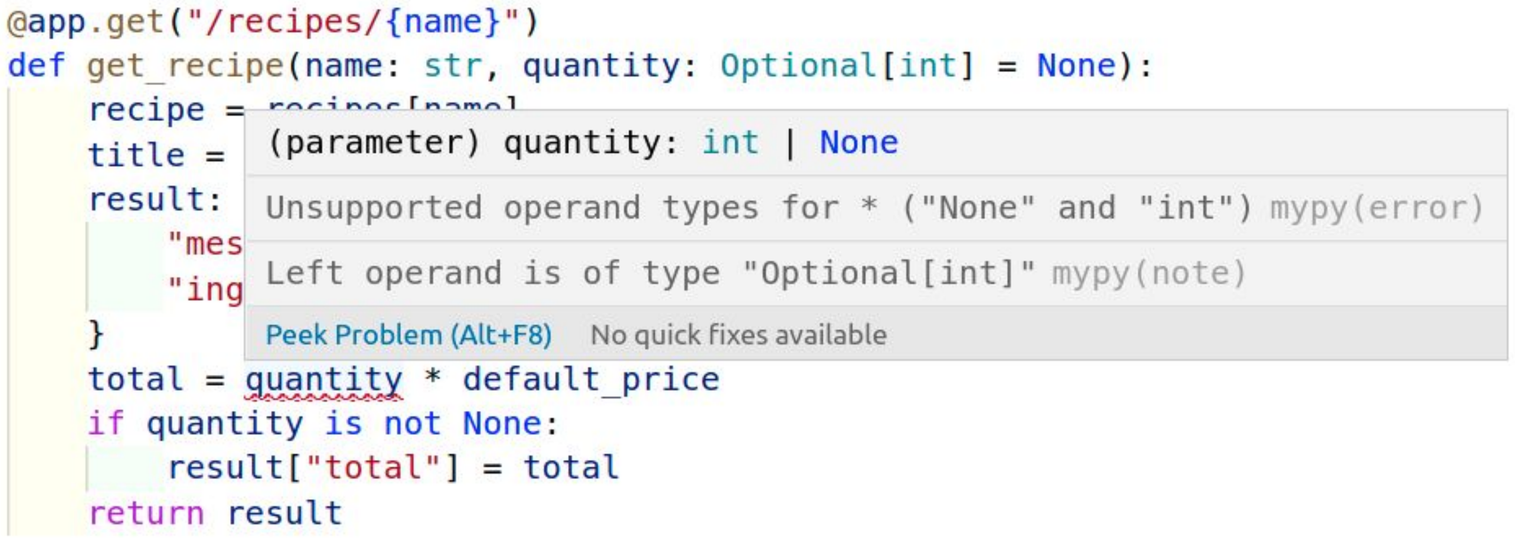
@tiangolo

🧙‍♀️ Type annotations with **FastAPI** - autocompletion



```
@app.get("/recipes/{name}")
def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title = name.replace("-", " ").
    result: dict = {
        "message": f"Recipe for {ti
        "ingredients": recipe["ingr
    }
    if quantity is not None:
        result["total"] = quantity
    return result
```

🧙 Type annotations with **FastAPI** - error detection



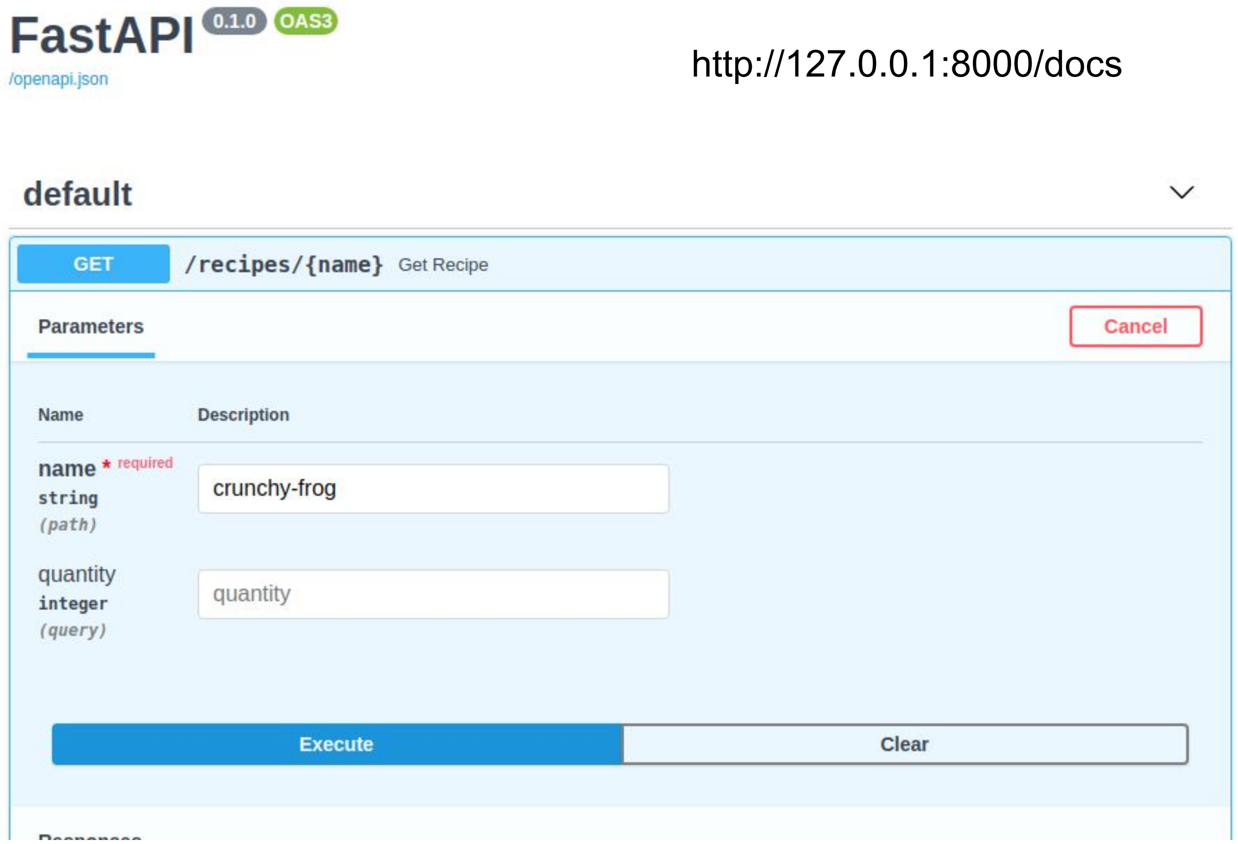
```
@app.get("/recipes/{name}")
def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title =
    result:
        "mes
        "ing
    }
    total = quantity * default_price
    if quantity is not None:
        result["total"] = total
    return result
```

```
(parameter) quantity: int | None
Unsupported operand types for * ("None" and "int") mypy(error)
Left operand is of type "Optional[int]" mypy(note)
```

@tiangolo

FastAPI - API documentation

http://127.0.0.1:8000/docs



FastAPI 0.1.0 OAS3
/openapi.json

default

GET /recipes/{name} Get Recipe

Parameters

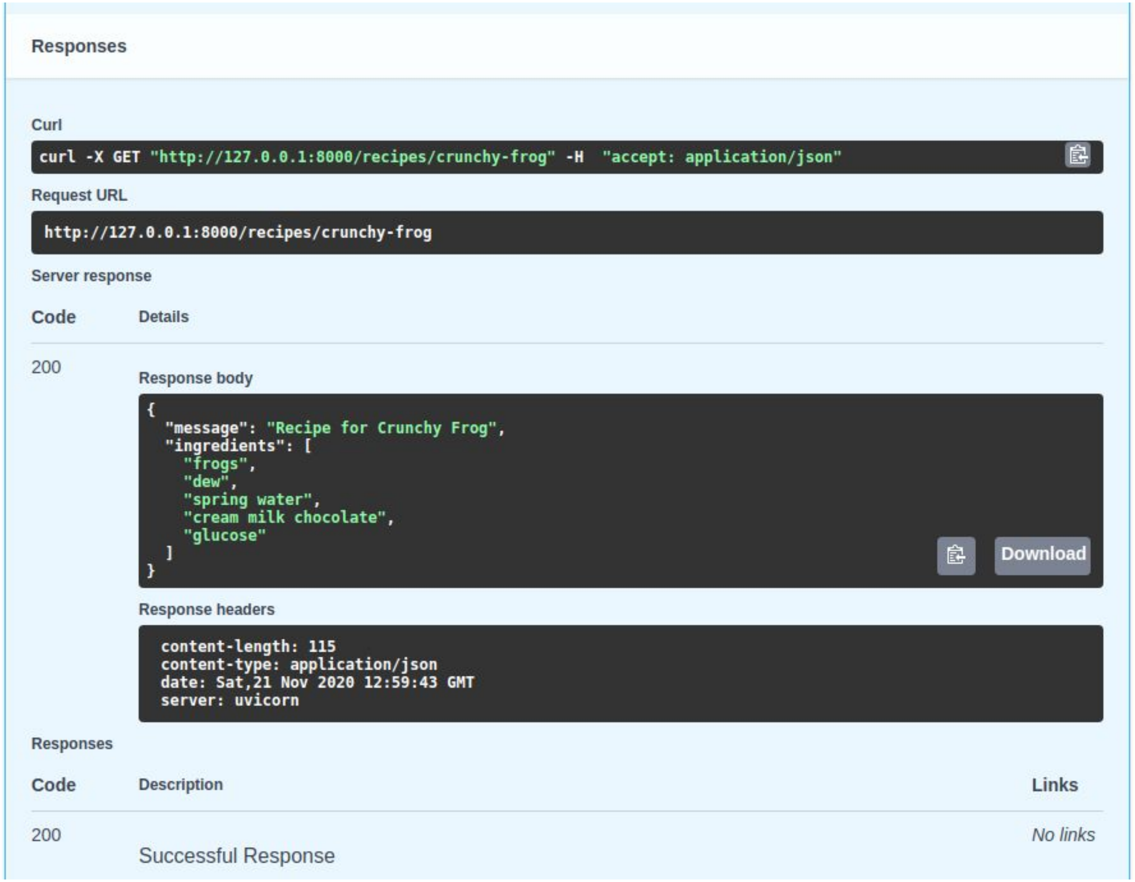
Cancel

Name	Description
name * required string *(path)*	crunchy-frog
quantity integer *(query)*	quantity

Execute Clear

@tiangolo

FastAPI - API documentation



Curl

```
curl -X GET "http://127.0.0.1:8000/recipes/crunchy-frog" -H  "accept: application/json"
```

Request URL

```
http://127.0.0.1:8000/recipes/crunchy-frog
```

Server response

Code	Details
200	Response body

```
{
  "message": "Recipe for Crunchy Frog",
  "ingredients": [
    "frogs",
    "dew",
    "spring water",
    "cream milk chocolate",
    "glucose"
  ]
}
```

Response headers

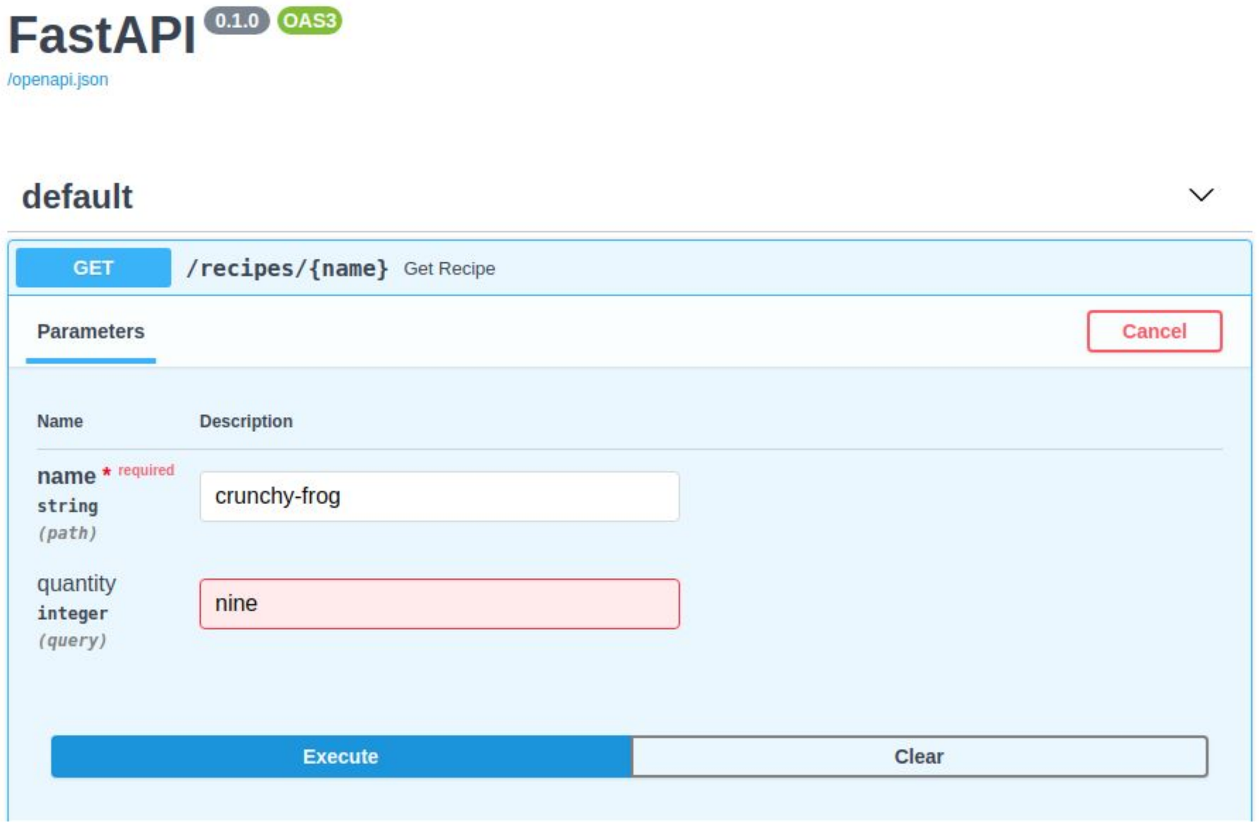
```
 content-length: 115
 content-type: application/json
 date: Sat,21 Nov 2020 12:59:43 GMT
 server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

@tiangolo

FastAPI - data validation



FastAPI 0.1.0 OAS3

/openapi.json

default

GET /recipes/{name} Get Recipe

Parameters

Cancel

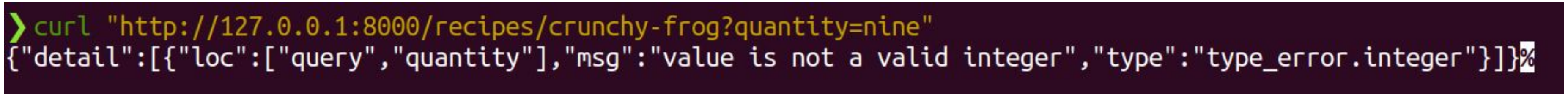
Name	Description
name * required string *(path)*	crunchy-frog
quantity integer *(query)*	nine

Execute

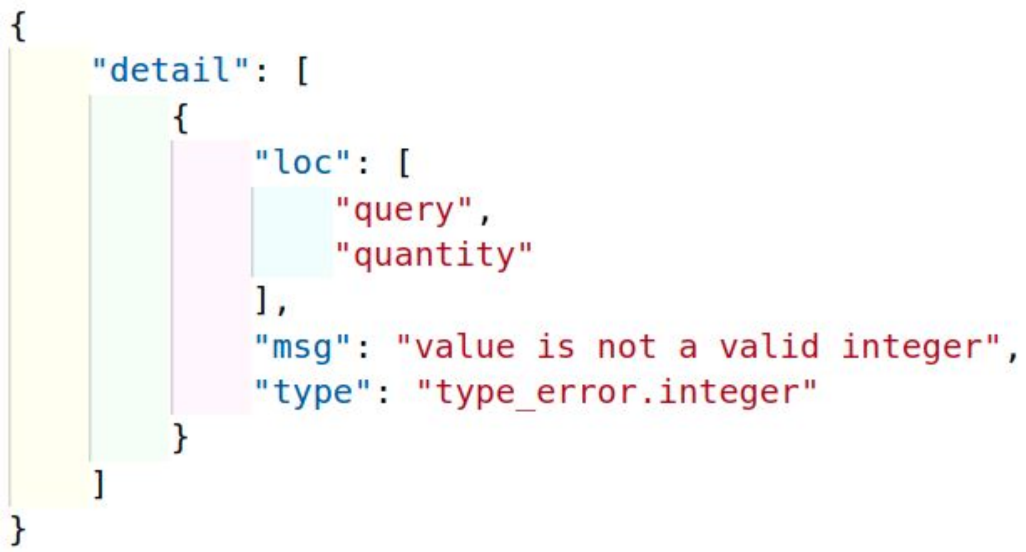
Clear

@tiangolo

FastAPI - data validation



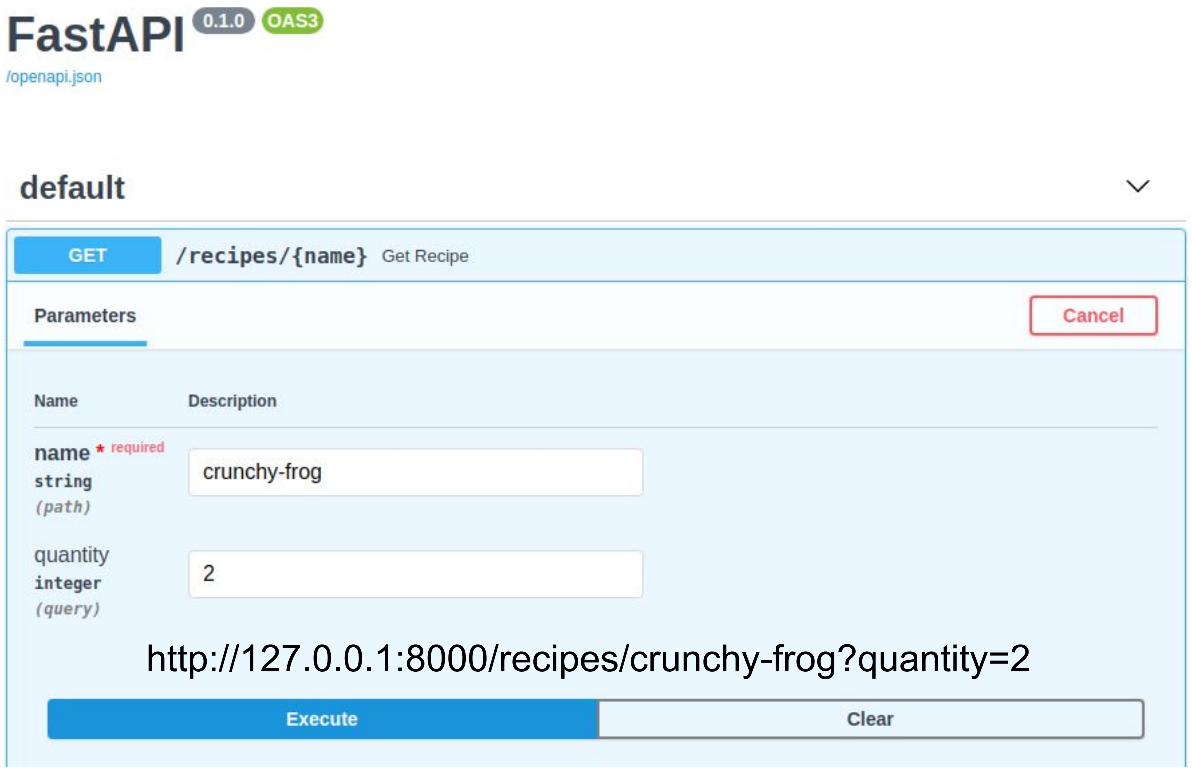
```
❯ curl "http://127.0.0.1:8000/recipes/crunchy-frog?quantity=nine"
{"detail":[{"loc":["query","quantity"],"msg":"value is not a valid integer","type":"type_error.integer"}]}%
```



```
{
    "detail": [
        {
            "loc": [
                "query",
                "quantity"
            ],
            "msg": "value is not a valid integer",
            "type": "type_error.integer"
        }
    ]
}
```

@tiangolo

FastAPI - data conversion


FastAPI 0.1.0 OAS3

/openapi.json

default

GET /recipes/{name} Get Recipe

Parameters

Cancel

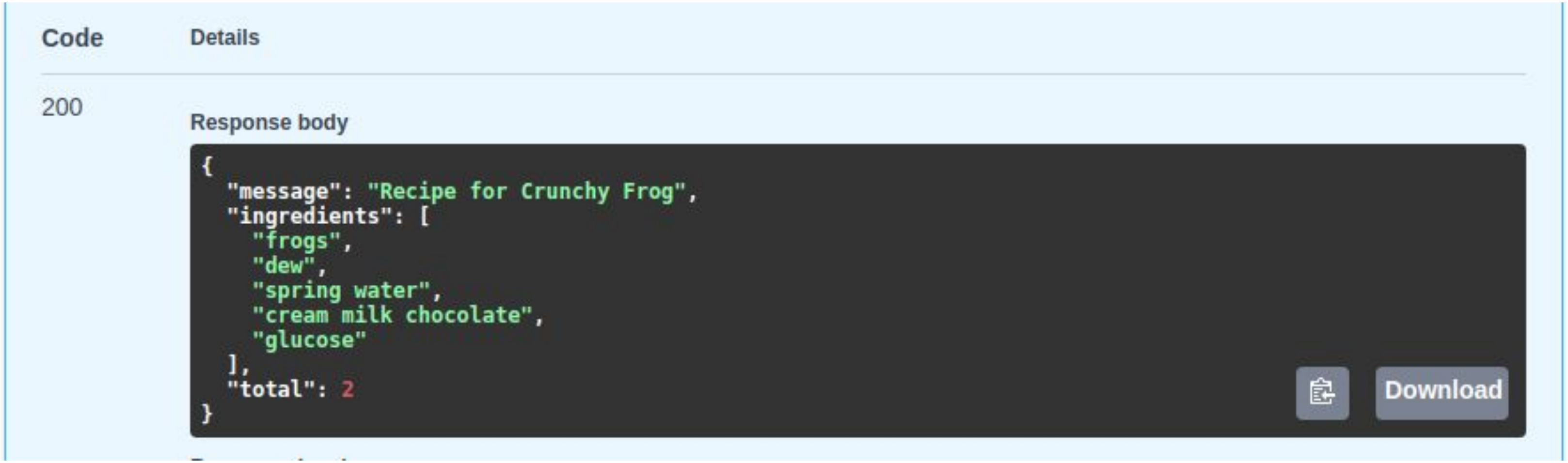
Name	Description
name * required string (path)	crunchy-frog
quantity integer (query)	2

http://127.0.0.1:8000/recipes/crunchy-frog?quantity=2

Execute Clear

@tiangolo

FastAPI - data conversion



```
{
  "message": "Recipe for Crunchy Frog",
  "ingredients": [
    "frogs",
    "dew",
    "spring water",
    "cream milk chocolate",
    "glucose"
  ],
  "total": 2
}
```

FastAPI is based on standards

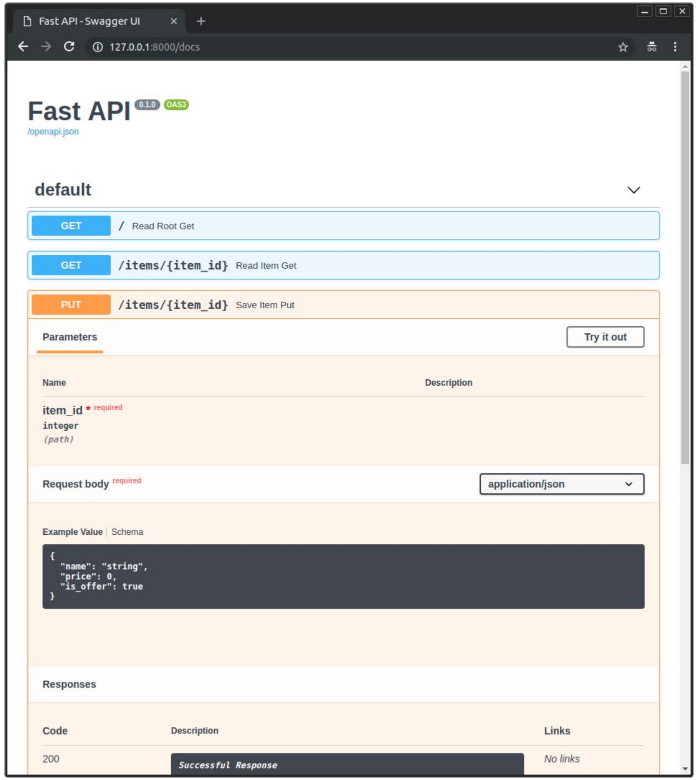
- OpenAPI
- JSON Schema
- OAuth2

📝 Automatic API docs ✨

- Standard type annotations

🔄 Automatic data conversion ✨

✔️ Automatic data validation ✨



```
{
  "name": "string",
  "price": 0,
  "is_offer": true
}
```

@tiangolo

Modern Python

🧙 Type annotations with

Typer

@tiangolo

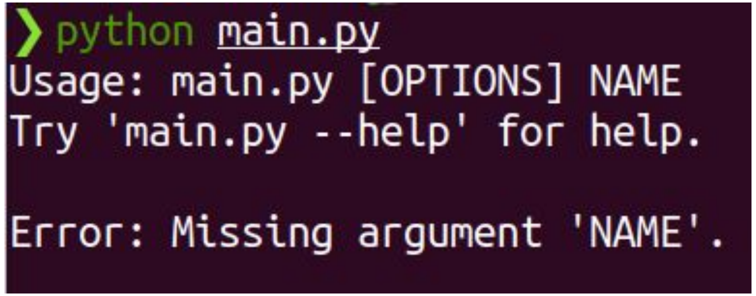
Type annotations with **Typer**

```python
from typing import Optional
import typer

recipes = {
    "crunchy-frog": {
        "ingredients": [
            "frogs",
            "dew",
            "spring water",
            "cream milk chocolate",
            "glucose",
        ]
    },
    "albatross": {
        "ingredients": ["albatross"]},
}
default_price = 1
```

```python
app = typer.Typer()


@app.command()
def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    result: dict = {
        "message": f"Recipe for {title}",
        "ingredients": recipe["ingredients"],
    }
    if quantity is not None:
        result["total"] = quantity * default_price
    typer.echo(result)
```

@tiangolo

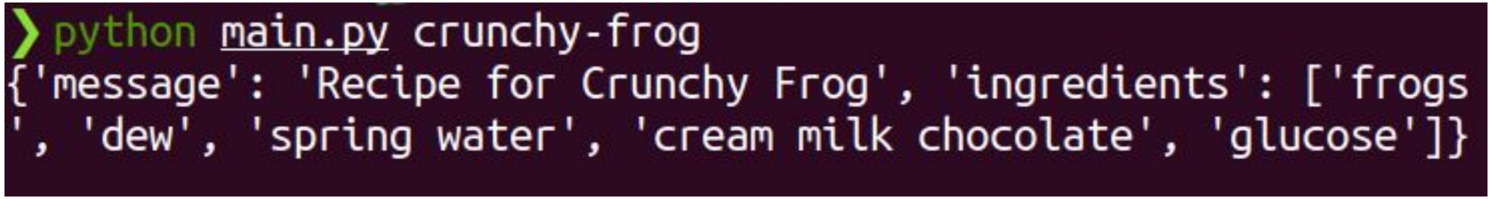
Typer - automatic help



```
❯ python main.py
Usage: main.py [OPTIONS] NAME
Try 'main.py --help' for help.

Error: Missing argument 'NAME'.
```

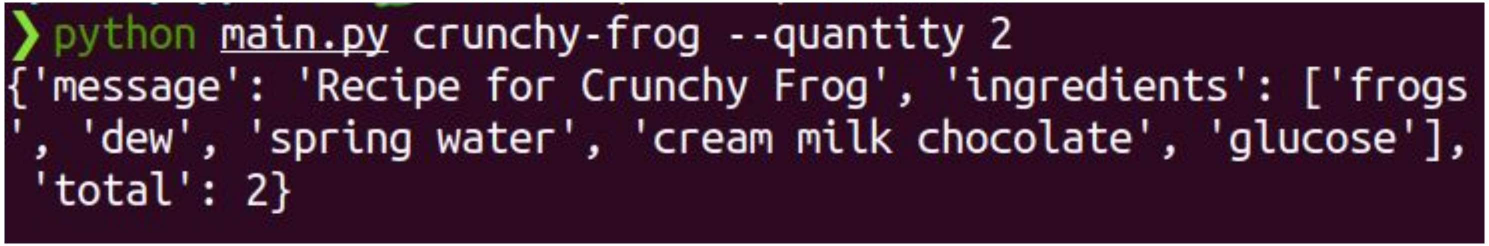
@tiangolo

Typer - CLI Arguments



```
❯ python main.py crunchy-frog
{'message': 'Recipe for Crunchy Frog', 'ingredients': ['frogs
', 'dew', 'spring water', 'cream milk chocolate', 'glucose']}
```

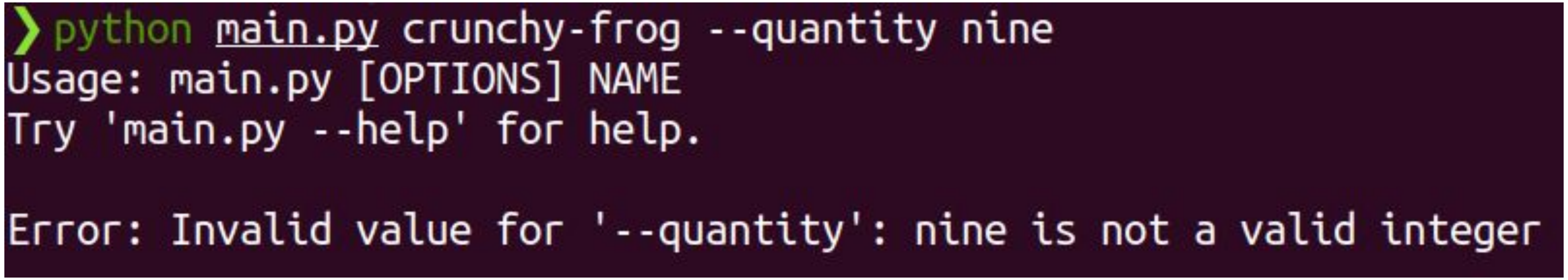
Typer - CLI Options



```
❯ python main.py crunchy-frog --quantity 2
{'message': 'Recipe for Crunchy Frog', 'ingredients': ['frogs
', 'dew', 'spring water', 'cream milk chocolate', 'glucose'],
 'total': 2}
```

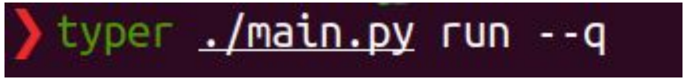
@tiangolo

Typer - validation



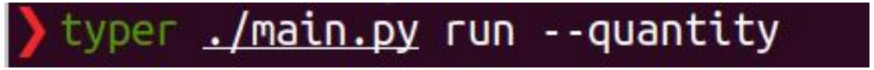
```
❯ python main.py crunchy-frog --quantity nine
Usage: main.py [OPTIONS] NAME
Try 'main.py --help' for help.

Error: Invalid value for '--quantity': nine is not a valid integer
```

Typer - shell (TAB) completion



```
❯ typer ./main.py run --q
```

<TAB>



```
❯ typer ./main.py run --quantity
```

@tiangolo

Modern Python

🧙 Type annotations with

Typer

and Rich

@tiangolo

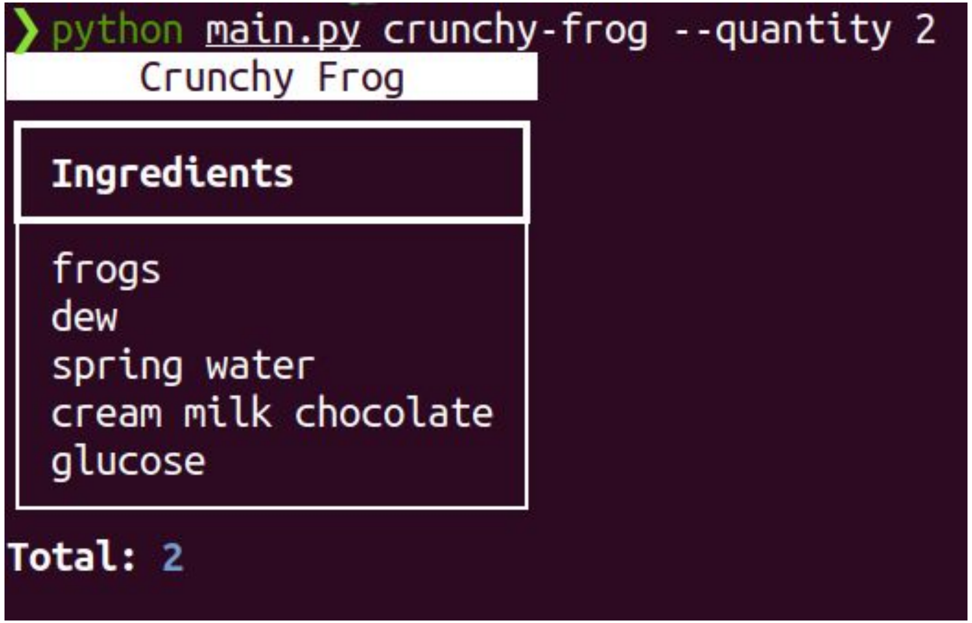
Typer with **Rich**

```python
from typing import Optional
from typer import Typer
from rich.table import Table
from rich.console import Console

recipes = {
    "crunchy-frog": {
        "ingredients": [
            "frogs",
            "dew",
            "spring water",
            "cream milk chocolate",
            "glucose",
        ]
    },
    "albatross": {
        "ingredients": ["albatross"]},
}
default_price = 1
```

```python
app = Typer()
console = Console()


@app.command()
def get_recipe(name: str, quantity: Optional[int] = None):
    recipe = recipes[name]
    title = name.replace("-", " ").title()
    table = Table("Ingredients", title=title)
    for ingredient in recipe["ingredients"]:
        table.add_row(ingredient)
    console.print(table)
    if quantity is not None:
        total = quantity * default_price
        console.print(f"Total: {total}", style="bold")
```

@tiangolo

Typer with Rich



```
❯ python main.py crunchy-frog --quantity 2
        Crunchy Frog
┌───────────────────────┐
│ Ingredients           │
├───────────────────────┤
│ frogs                 │
│ dew                   │
│ spring water          │
│ cream milk chocolate  │
│ glucose               │
└───────────────────────┘
Total: 2
```

@tiangolo

Typer and friends - the people behind

David Lord
Click maintainer

@davidism

@davidism

Will McGugan
Rich creator

@willmcgugan

@willmcgugan

Modern Python

🧙‍♀️ Type annotations with

FastAPI

and Pydantic

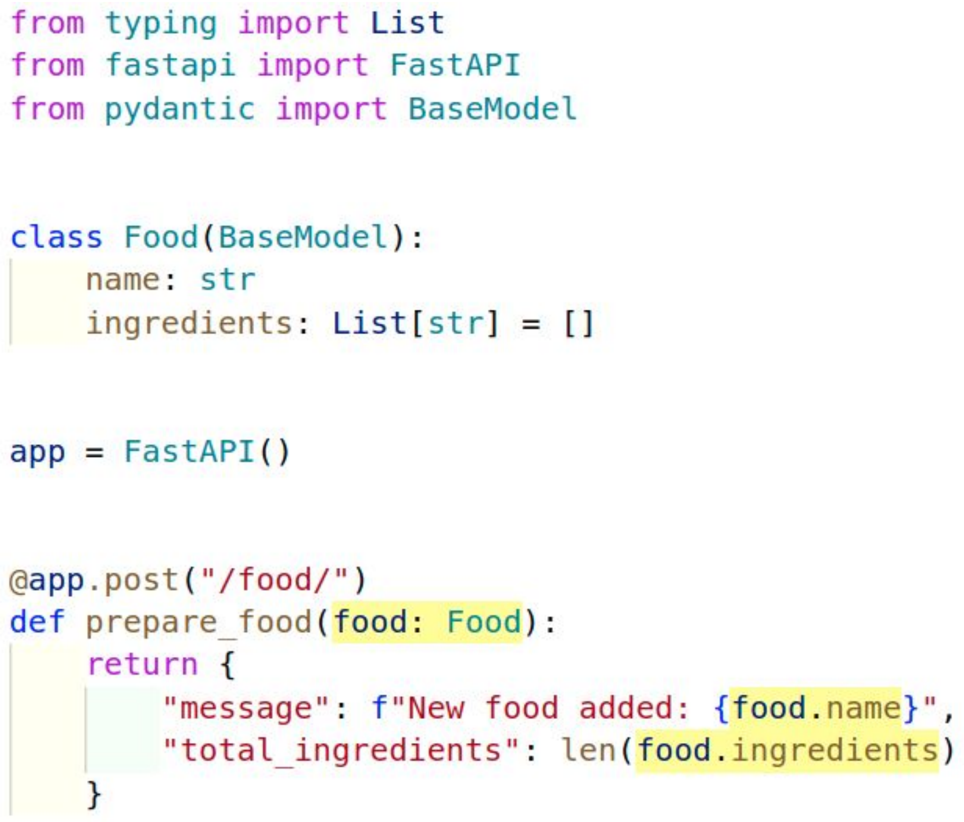
@tiangolo

FastAPI with Pydantic

```python
from typing import List
from fastapi import FastAPI
from pydantic import BaseModel


class Food(BaseModel):
    name: str
    ingredients: List[str] = []


app = FastAPI()


@app.post("/food/")
def prepare_food(food: Food):
    return {
        "message": f"New food added: {food.name}",
        "total_ingredients": len(food.ingredients)
    }
```

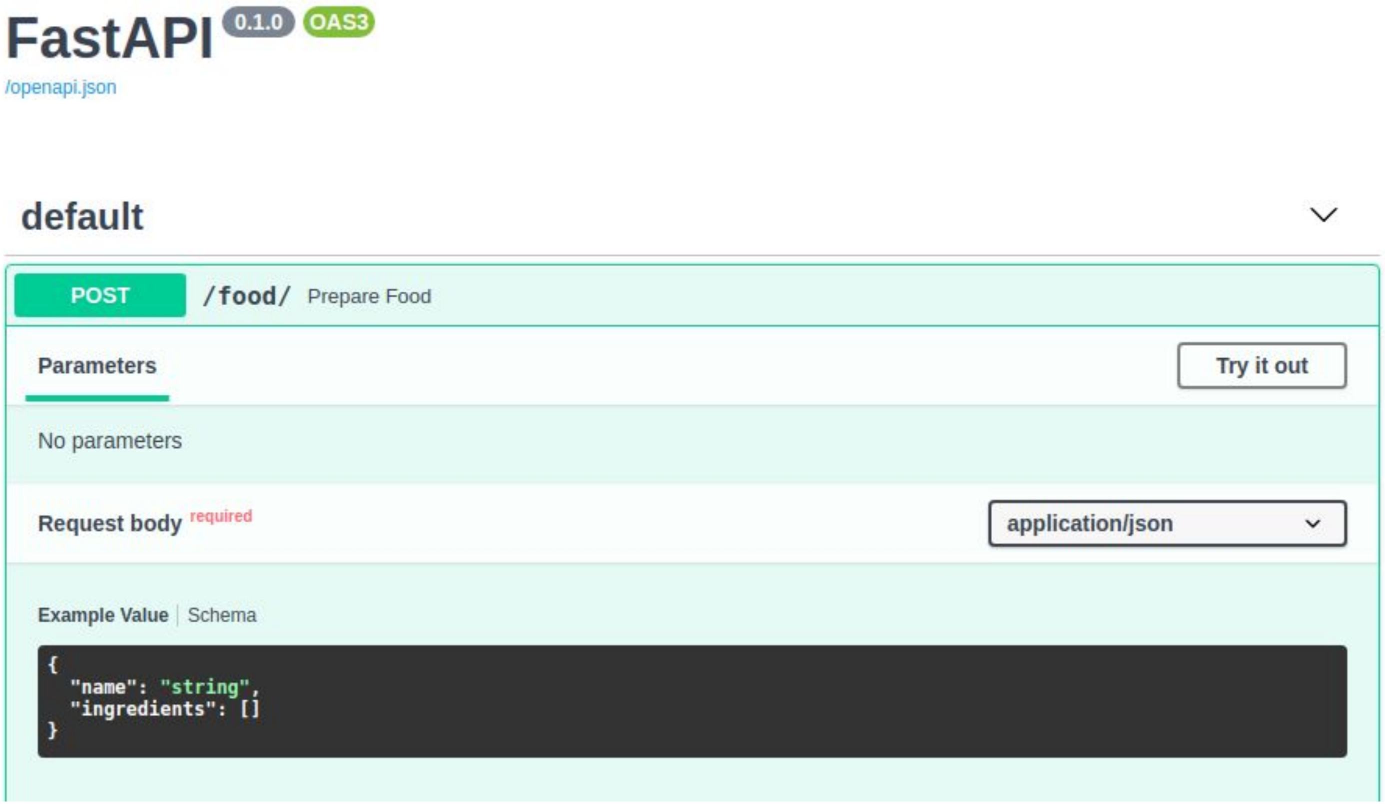
@tiangolo

FastAPI with **Pydantic**



```python
from typing import List
from fastapi import FastAPI
from pydantic import BaseModel


class Food(BaseModel):
    name: str
    ingredients: List[str] = []


app = FastAPI()


@app.post("/food/")
def prepare_food(food: Food):
    return {
        "message": f"New food added: {food.name}",
        "total_ingredients": len(food.ingredients)
    }
```

FastAPI with **Pydantic** - API documentation



FastAPI 0.1.0 OAS3

/openapi.json

default

POST /food/ Prepare Food

Parameters

Try it out

No parameters

Request body required

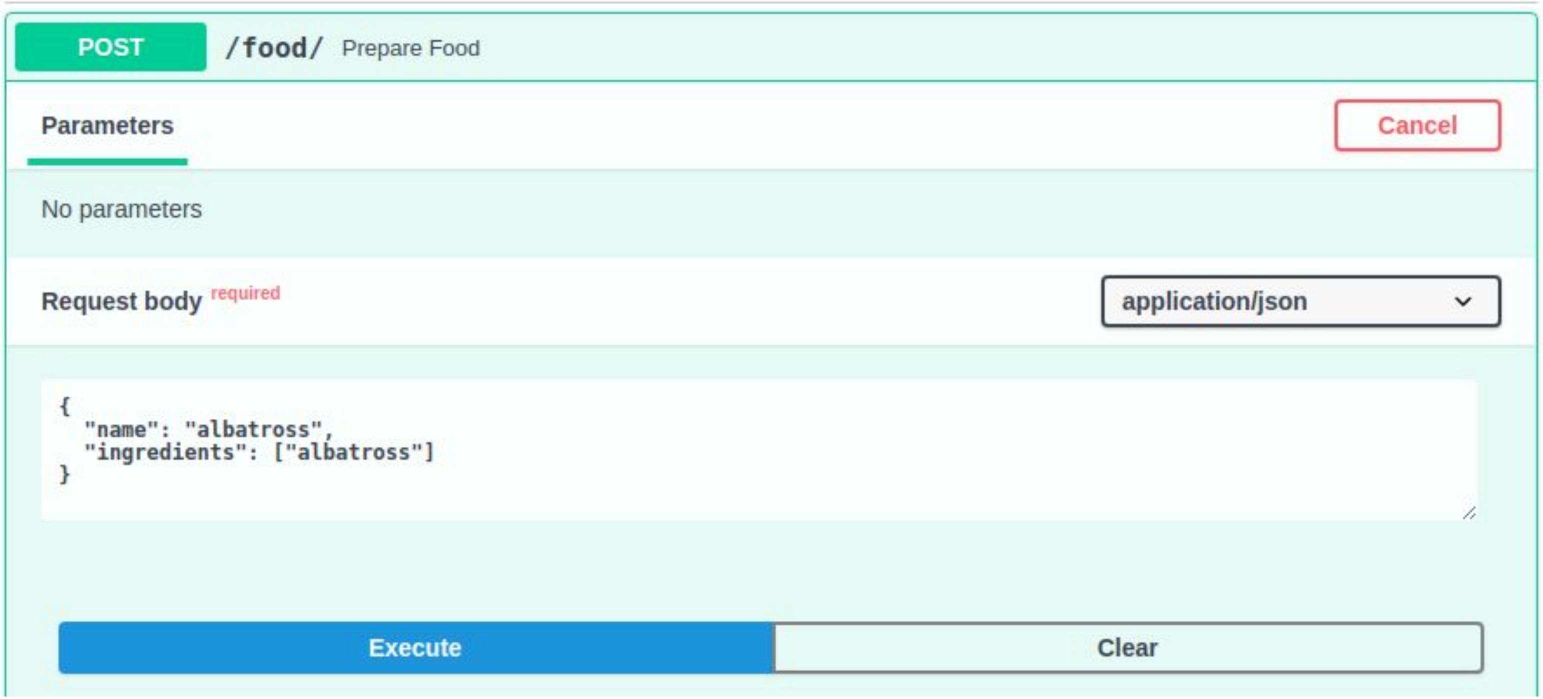
application/json

Example Value | Schema

```
{
  "name": "string",
  "ingredients": []
}
```

@tiangolo

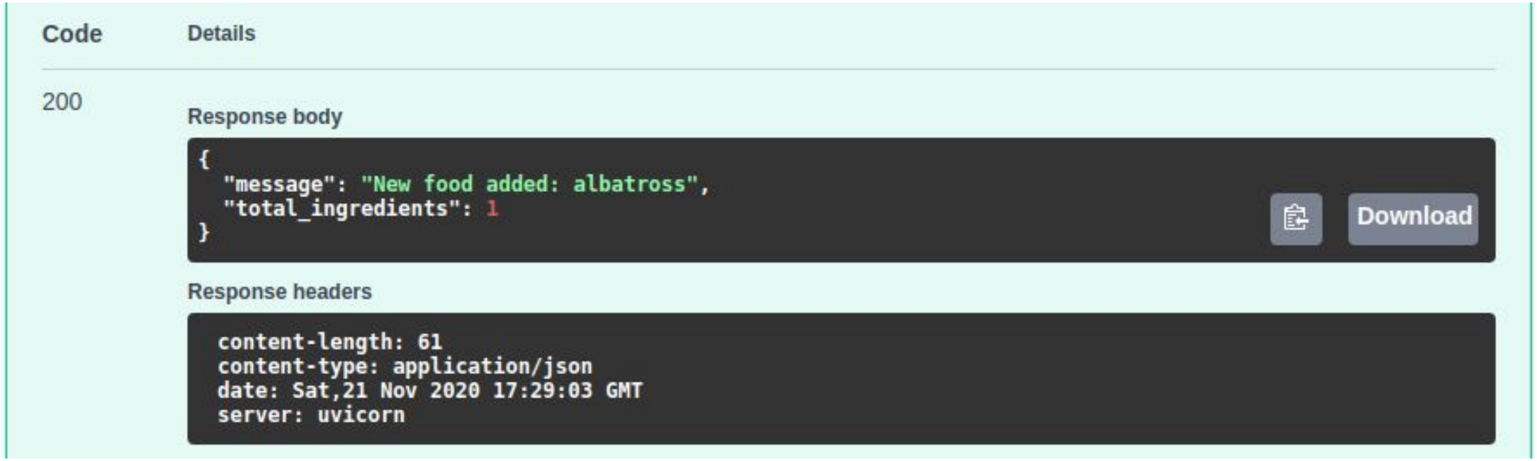
FastAPI with **Pydantic** - interactive API docs



```
{
  "name": "albatross",
  "ingredients": ["albatross"]
}
```

@tiangolo

FastAPI with **Pydantic** - interactive API docs



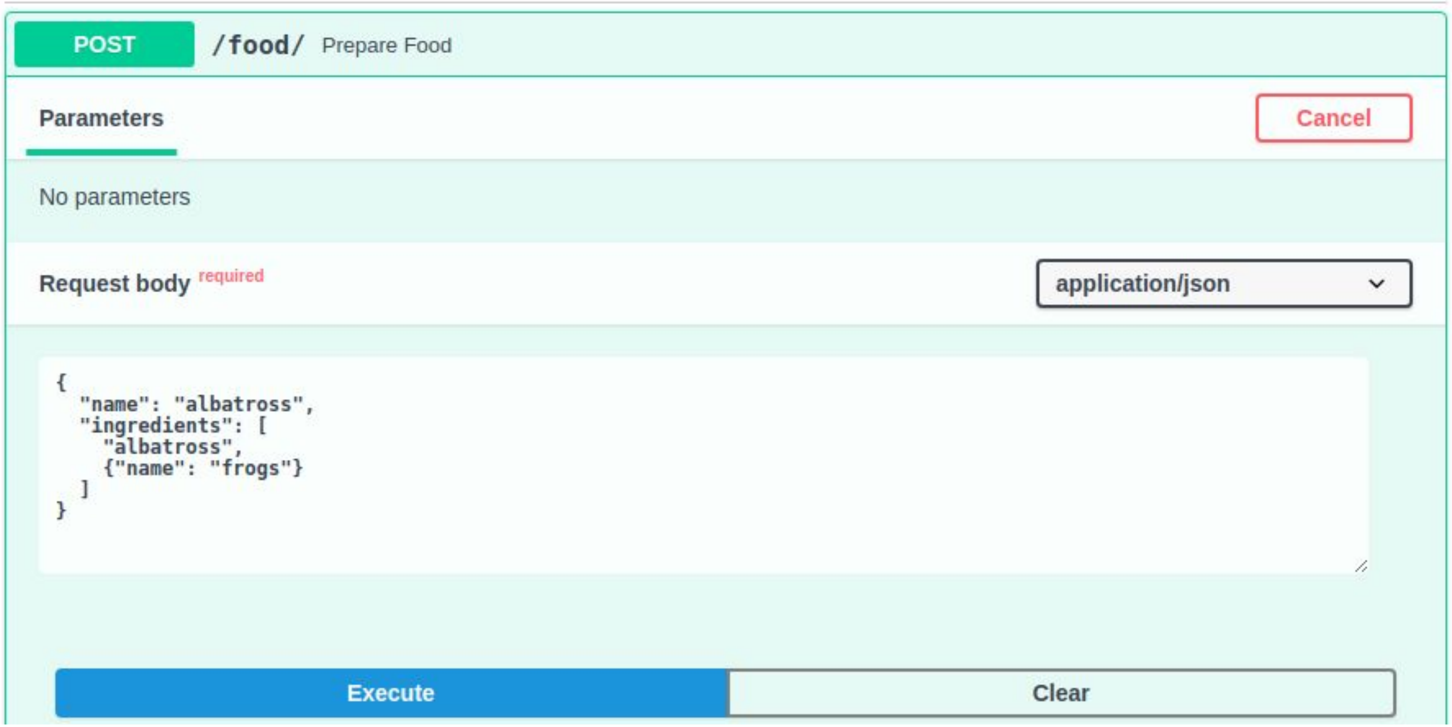
Response body

```
{
  "message": "New food added: albatross",
  "total_ingredients": 1
}
```

Response headers

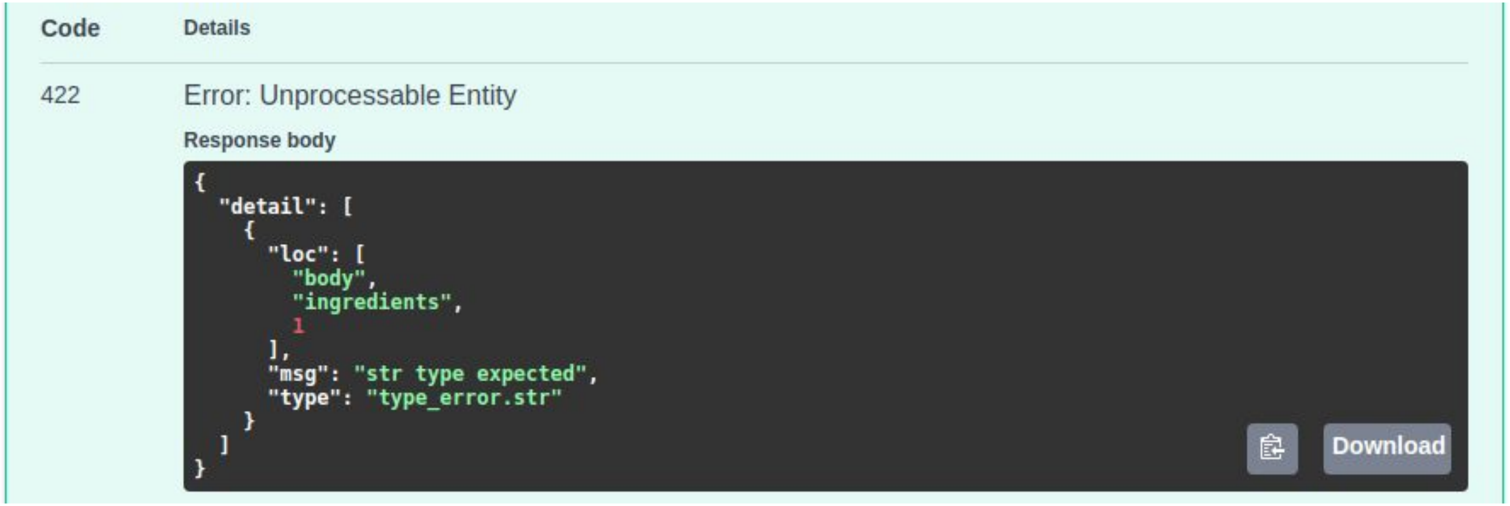
```
content-length: 61
content-type: application/json
date: Sat,21 Nov 2020 17:29:03 GMT
server: uvicorn
```

FastAPI with **Pydantic** - data validation



```
{
  "name": "albatross",
  "ingredients": [
    "albatross",
    {"name": "frogs"}
  ]
}
```

FastAPI with **Pydantic** - data validation



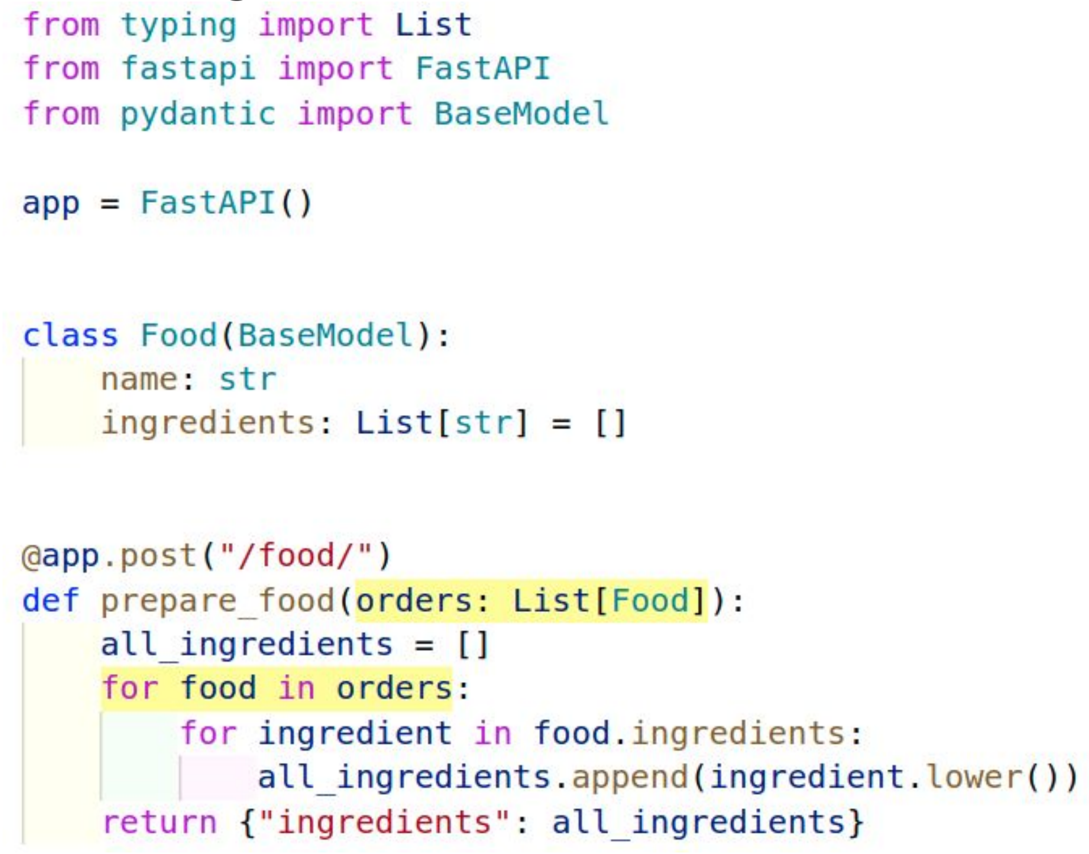
422 Error: Unprocessable Entity

Response body

```
{
  "detail": [
    {
      "loc": [
        "body",
        "ingredients",
        1
      ],
      "msg": "str type expected",
      "type": "type_error.str"
    }
  ]
}
```

@tiangolo

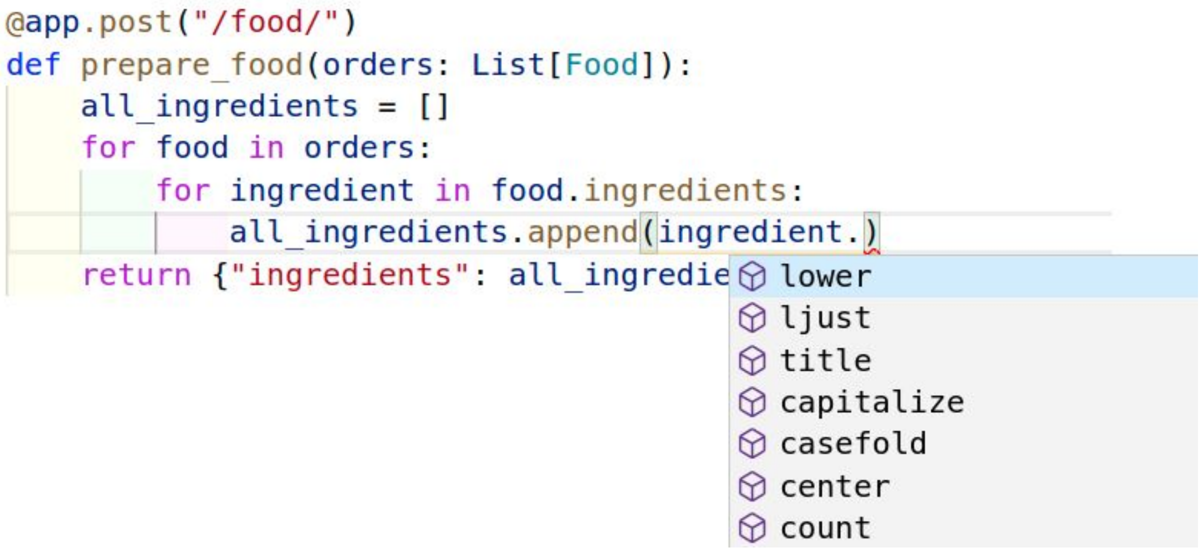
FastAPI with **Pydantic** - nested data



```python
from typing import List
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()


class Food(BaseModel):
    name: str
    ingredients: List[str] = []


@app.post("/food/")
def prepare_food(orders: List[Food]):
    all_ingredients = []
    for food in orders:
        for ingredient in food.ingredients:
            all_ingredients.append(ingredient.lower())
    return {"ingredients": all_ingredients}
```

@tiangolo

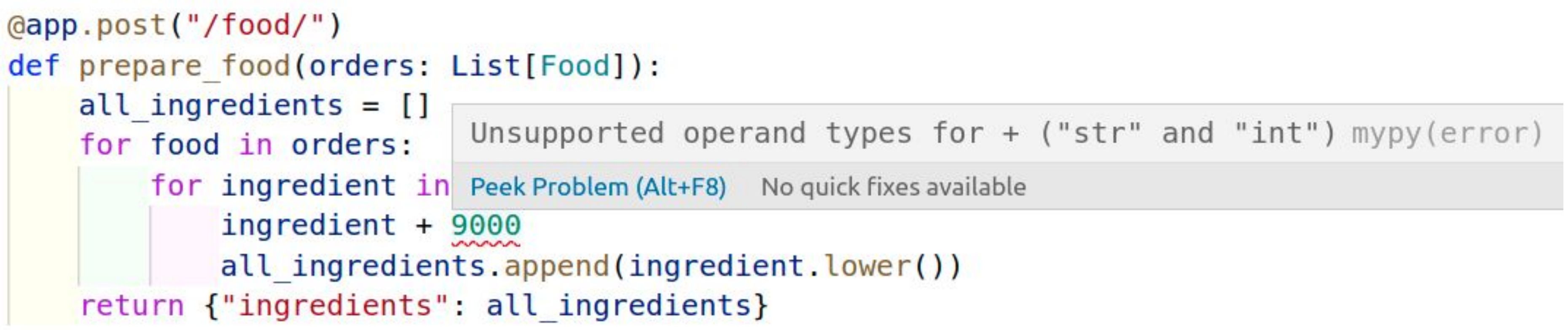
FastAPI with **Pydantic** - autocompletion



```
@app.post("/food/")
def prepare_food(orders: List[Food]):
    all_ingredients = []
    for food in orders:
        for ingredient in food.ingredients:
            all_ingredients.append(ingredient.)
    return {"ingredients": all_ingredie
```

@tiangolo

FastAPI with **Pydantic** - error detection



```
@app.post("/food/")
def prepare_food(orders: List[Food]):
    all_ingredients = []
    for food in orders:
        for ingredient in
            ingredient + 9000
            all_ingredients.append(ingredient.lower())
    return {"ingredients": all_ingredients}
```

```
Unsupported operand types for + ("str" and "int") mypy(error)
Peek Problem (Alt+F8)    No quick fixes available
```

@tiangolo

FastAPI - the people behind

Samuel Colvin
Pydantic creator

@samuelcolvin

@samuel_colvin

Tom Christie
Creator of Starlette (and more)

@tomchristie

@_tomchristie

David Montague
Pydantic and FastAPI
notorious contributor

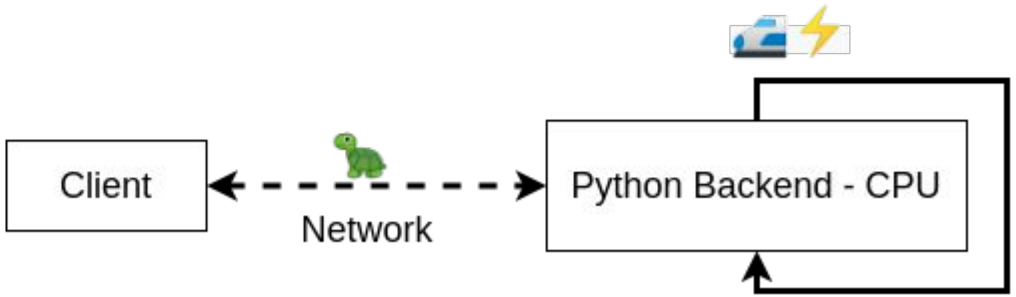
@dmontagu

Modern Python

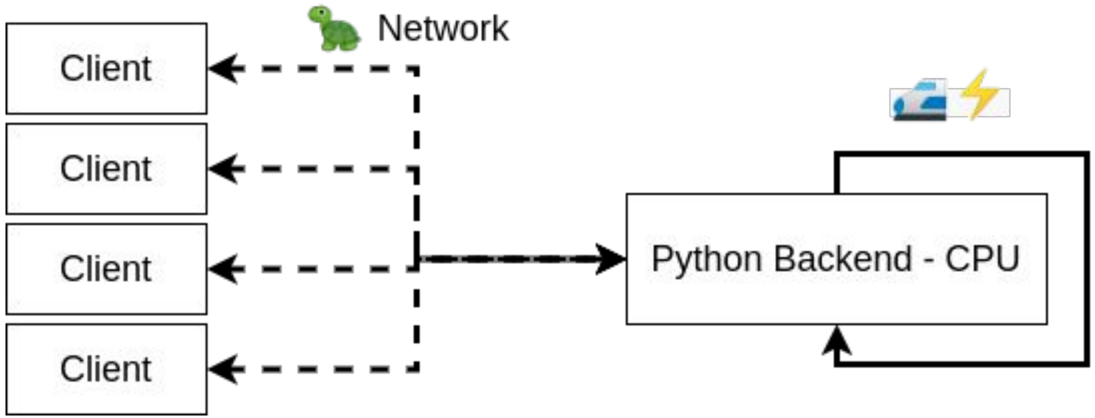
⏱️ async / await

@tiangolo

⏱️ async / await - concurrency



⏱️ async / await - concurrency

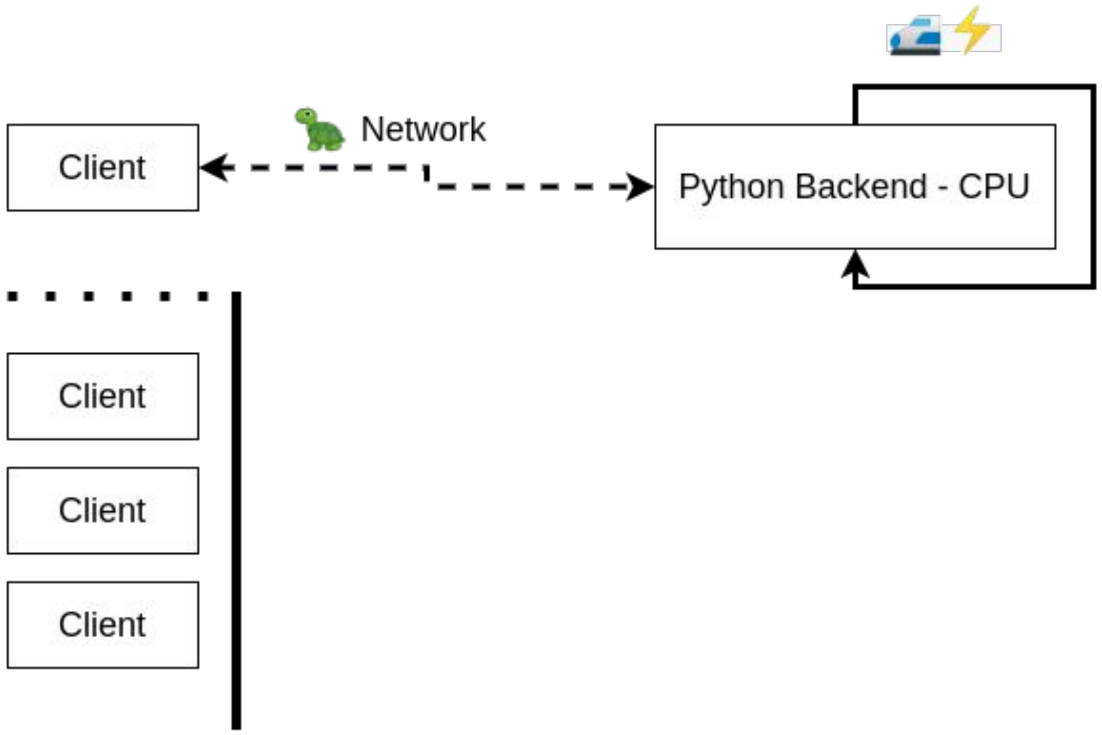


⏱️ async / await - concurrency

Photo by John Cameron: https://unsplash.com/@john_cameron

@tiangolo

⏱️ async / await - concurrency



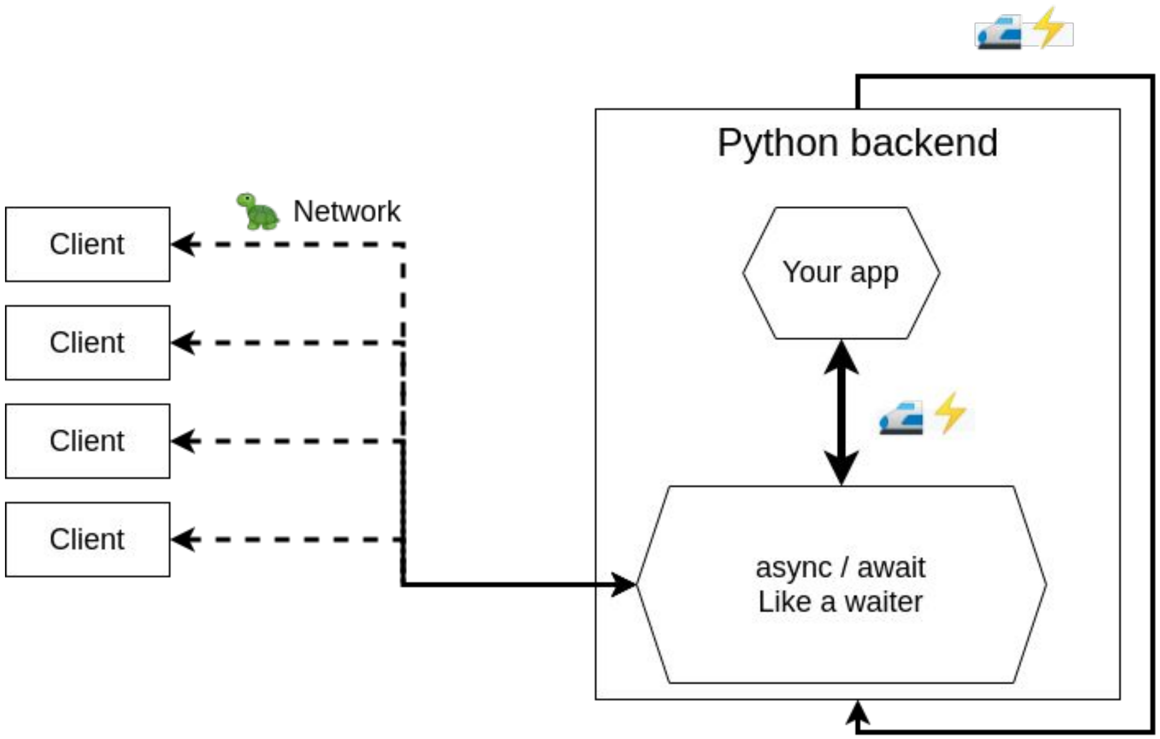
@tiangolo

⏱️ async / await - concurrency

Photo by Kate Townsend: https://unsplash.com/@k8townsend

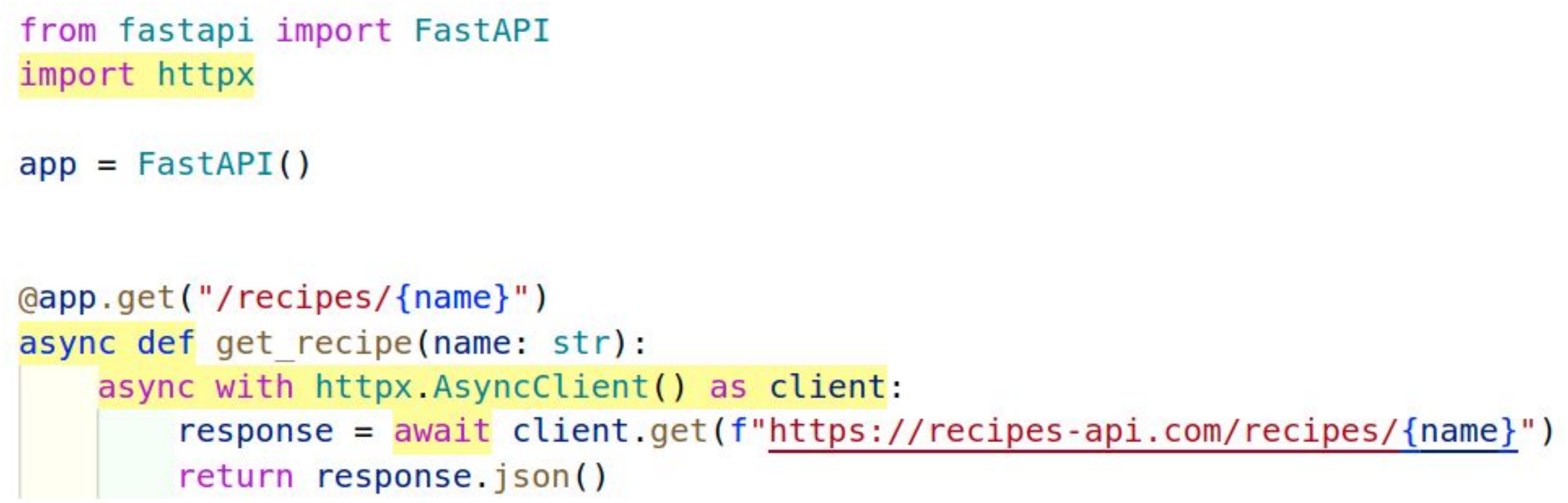
@tiangolo

⏱️ async / await - concurrency



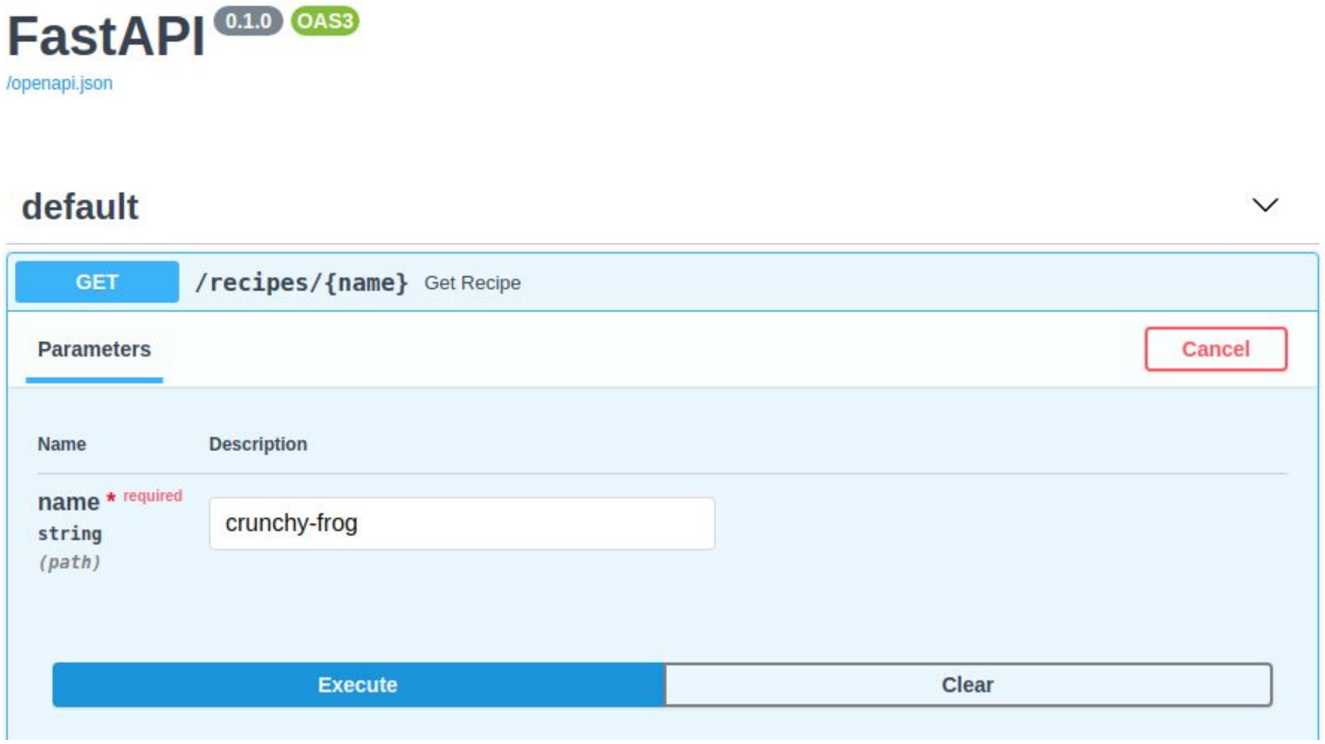
@tiangolo

⏱️ async / await - **FastAPI** and **HTTPX**



```python
from fastapi import FastAPI
import httpx

app = FastAPI()


@app.get("/recipes/{name}")
async def get_recipe(name: str):
    async with httpx.AsyncClient() as client:
        response = await client.get(f"https://recipes-api.com/recipes/{name}")
        return response.json()
```

⏱️ async / await - **FastAPI** and **HTTPX**


FastAPI 0.1.0 OAS3

/openapi.json

default

GET `/recipes/{name}` Get Recipe

Parameters

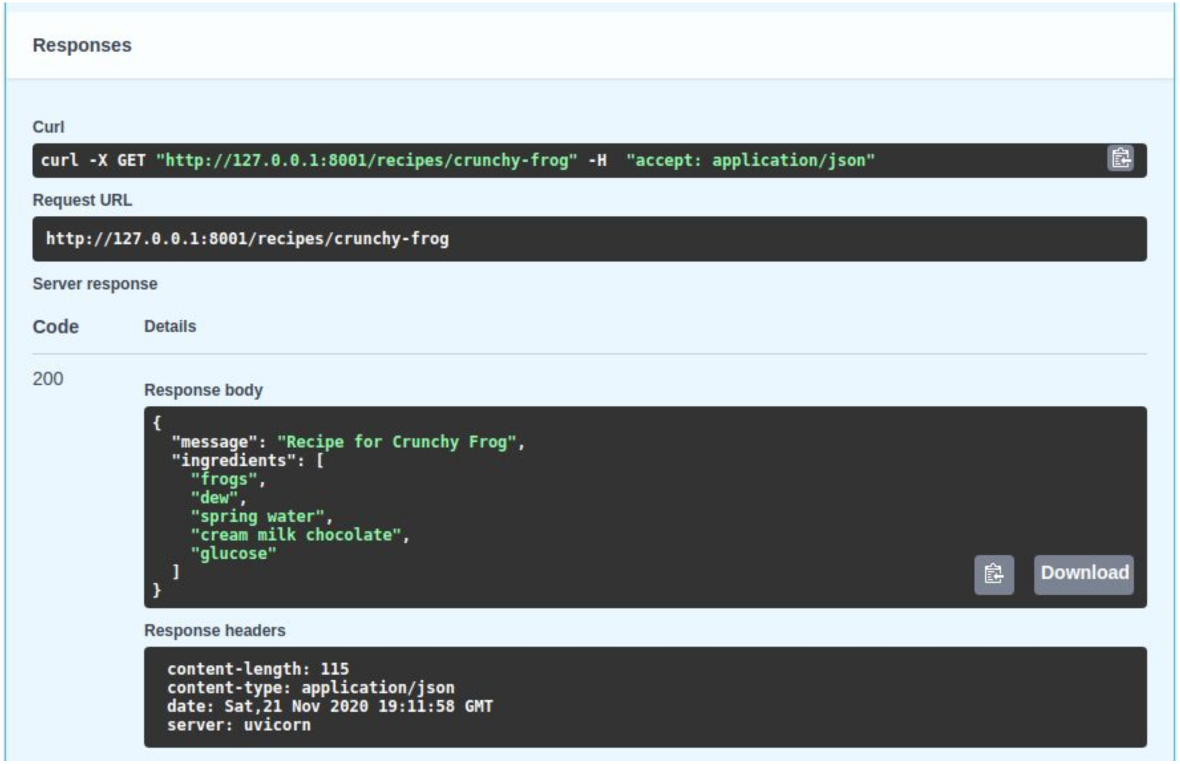
Cancel

Name	Description
name * required string *(path)*	crunchy-frog

Execute Clear

@tiangolo

⏱️ async / await - **FastAPI** and **HTTPX**



Responses

Curl

```
curl -X GET "http://127.0.0.1:8001/recipes/crunchy-frog" -H  "accept: application/json"
```

Request URL

```
http://127.0.0.1:8001/recipes/crunchy-frog
```

Server response

Code	Details
200	Response body

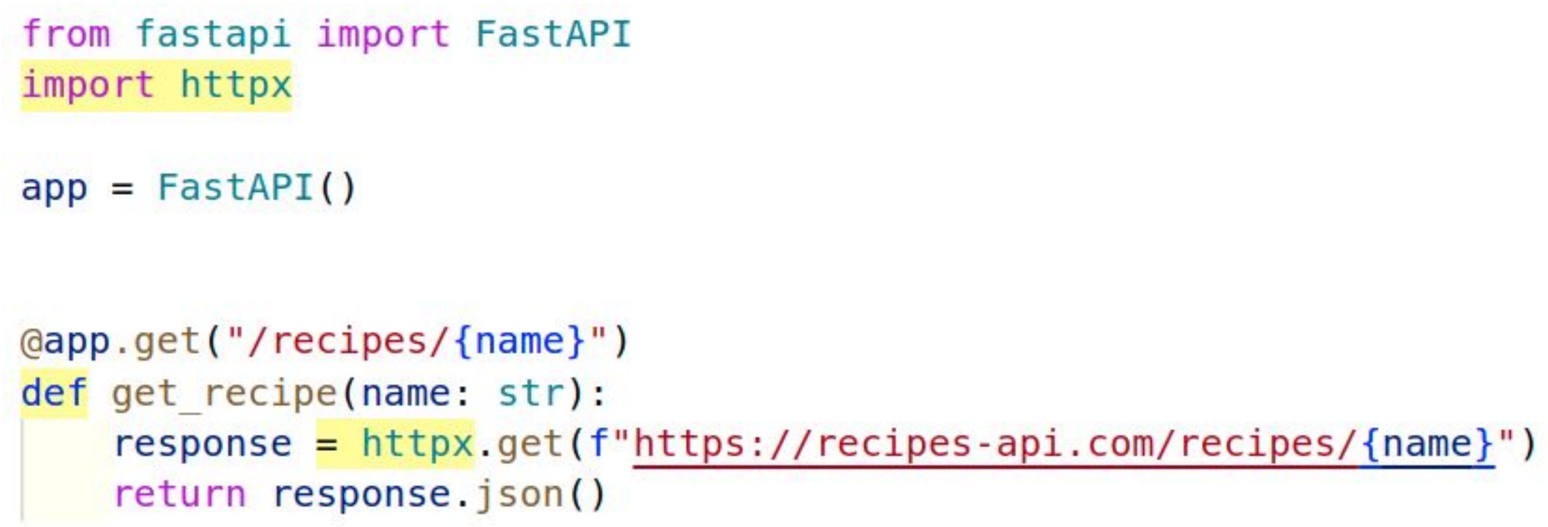
```
{
  "message": "Recipe for Crunchy Frog",
  "ingredients": [
    "frogs",
    "dew",
    "spring water",
    "cream milk chocolate",
    "glucose"
  ]
}
```

Response headers

```
 content-length: 115
 content-type: application/json
 date: Sat,21 Nov 2020 19:11:58 GMT
 server: uvicorn
```

@tiangolo

⏱️ async / await - optional with **FastAPI** and **HTTPX**



```python
from fastapi import FastAPI
import httpx

app = FastAPI()


@app.get("/recipes/{name}")
def get_recipe(name: str):
    response = httpx.get(f"https://recipes-api.com/recipes/{name}")
    return response.json()
```

FastAPI async / await - the people behind

Tom Christie
Creator of Starlette, HTTPX, and more

@tomchristie

@_tomchristie

Florimond Manca
Co-maintainer of HTTPX, and others

@florimondmanca

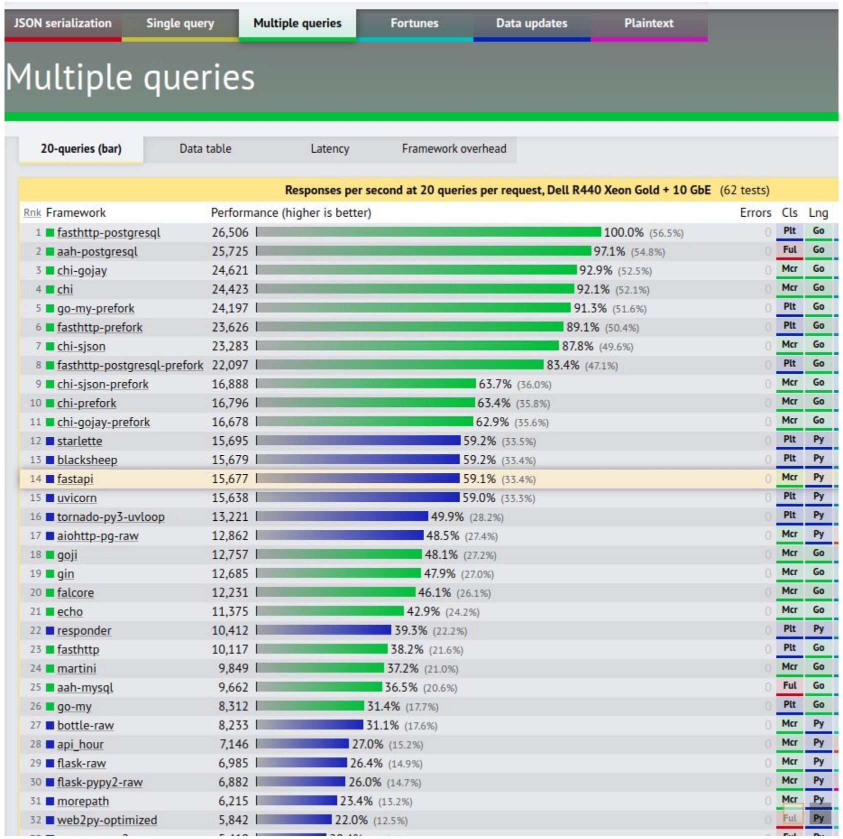
@florimondmanca

Andrew Godwin
Author of the ASGI specification

@andrewgodwin

@andrewgodwin

Performance with FastAPI



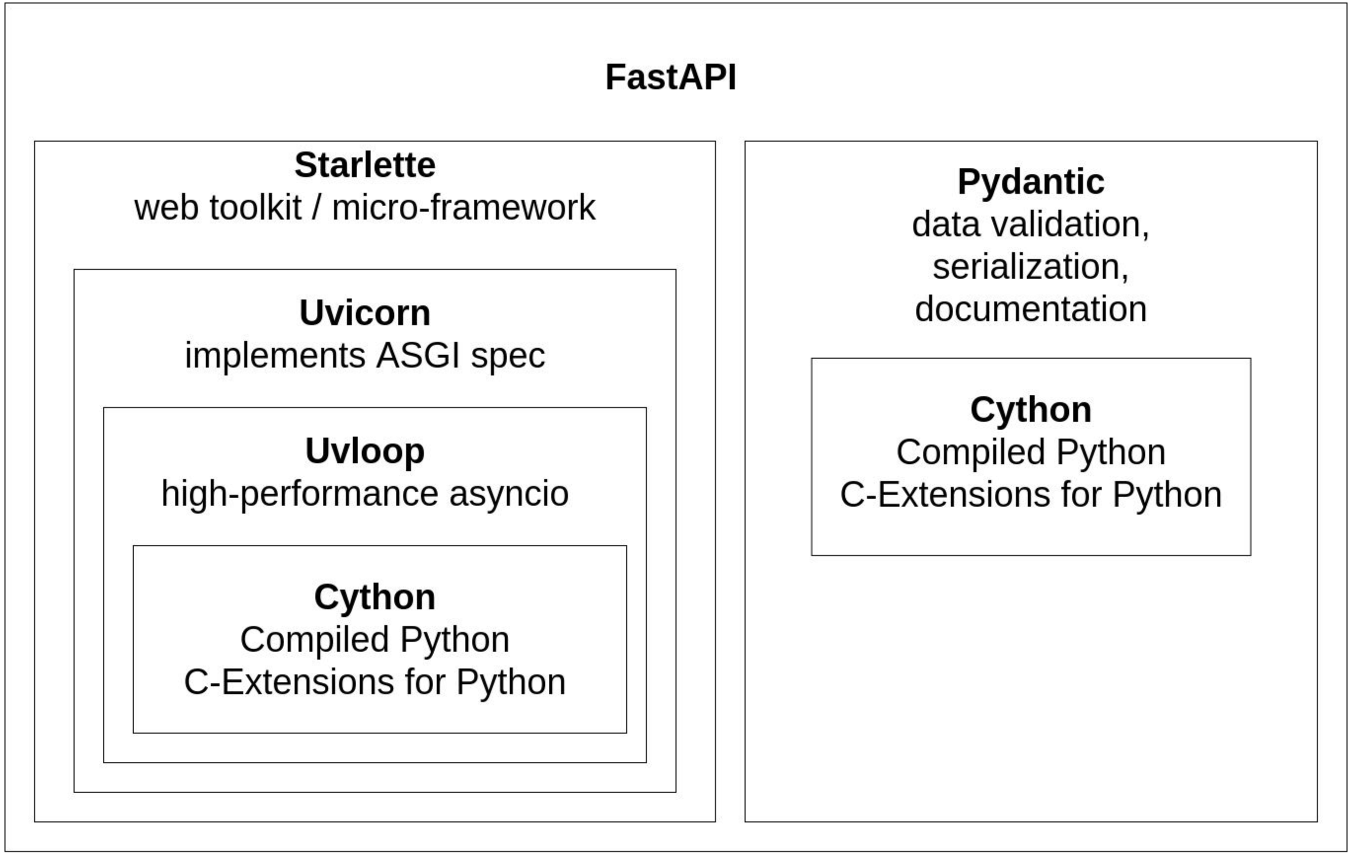
Multiple queries

Responses per second at 20 queries per request, Dell R440 Xeon Gold + 10 GbE (62 tests)

Rnk	Framework	Performance (higher is better)		Errors	Cls	Lng
1	🟩 fasthttp-postgresql	26,506	100.0% (56.5%)	0	Plt	Go
2	🟩 aah-postgresql	25,725	97.1% (54.8%)	0	Ful	Go
3	🟩 chi-gojay	24,621	92.9% (52.5%)	0	Mcr	Go
4	🟩 chi	24,423	92.1% (52.1%)	0	Mcr	Go
5	🟩 go-my-prefork	24,197	91.3% (51.6%)	0	Plt	Go
6	🟩 fasthttp-prefork	23,626	89.1% (50.4%)	0	Plt	Go
7	🟩 chi-sjson	23,283	87.8% (49.6%)	0	Mcr	Go
8	🟩 fasthttp-postgresql-prefork	22,097	83.4% (47.1%)	0	Plt	Go
9	🟩 chi-sjson-prefork	16,888	63.7% (36.0%)	0	Mcr	Go
10	🟩 chi-prefork	16,796	63.4% (35.8%)	0	Mcr	Go
11	🟩 chi-gojay-prefork	16,678	62.9% (35.6%)	0	Mcr	Go
12	🟦 starlette	15,695	59.2% (33.5%)	0	Plt	Py
13	🟦 blacksheep	15,679	59.2% (33.4%)	0	Plt	Py
14	🟦 fastapi	15,677	59.1% (33.4%)	0	Mcr	Py
15	🟦 uvicorn	15,638	59.0% (33.3%)	0	Plt	Py
16	🟦 tornado-py3-uvloop	13,221	49.9% (28.2%)	0	Plt	Py
17	🟦 aiohttp-pg-raw	12,862	48.5% (27.4%)	0	Mcr	Py
18	🟩 goji	12,757	48.1% (27.2%)	0	Mcr	Go
19	🟩 gin	12,685	47.9% (27.0%)	0	Mcr	Go
20	🟩 falcore	12,231	46.1% (26.1%)	0	Mcr	Go
21	🟩 echo	11,375	42.9% (24.2%)	0	Mcr	Go
22	🟦 responder	10,412	39.3% (22.2%)	0	Plt	Py
23	🟩 fasthttp	10,117	38.2% (21.6%)	0	Plt	Go
24	🟩 martini	9,849	37.2% (21.0%)	0	Mcr	Go
25	🟩 aah-mysql	9,662	36.5% (20.6%)	0	Ful	Go
26	🟩 go-my	8,312	31.4% (17.7%)	0	Plt	Go
27	🟦 bottle-raw	8,233	31.1% (17.6%)	0	Mcr	Py
28	🟦 api_hour	7,146	27.0% (15.2%)	0	Mcr	Py
29	🟦 flask-raw	6,985	26.4% (14.9%)	0	Mcr	Py
30	🟦 flask-pypy2-raw	6,882	26.0% (14.7%)	0	Mcr	Py
31	🟦 morepath	6,215	23.4% (13.2%)	0	Mcr	Py
32	🟦 web2py-optimized	5,842	22.0% (12.5%)	0	Ful	Py

@tiangolo

⚡ Performance with **FastAPI**



@tiangolo

⚡ **Performance** - the people behind

Tom Christie
Creator of Starlette, Uvicorn, and more

@tomchristie

@_tomchristie

Yury Selivanov
Creator of Uvloop

@1st1

@1st1

Stefan Behnel
Maintainer of Cython

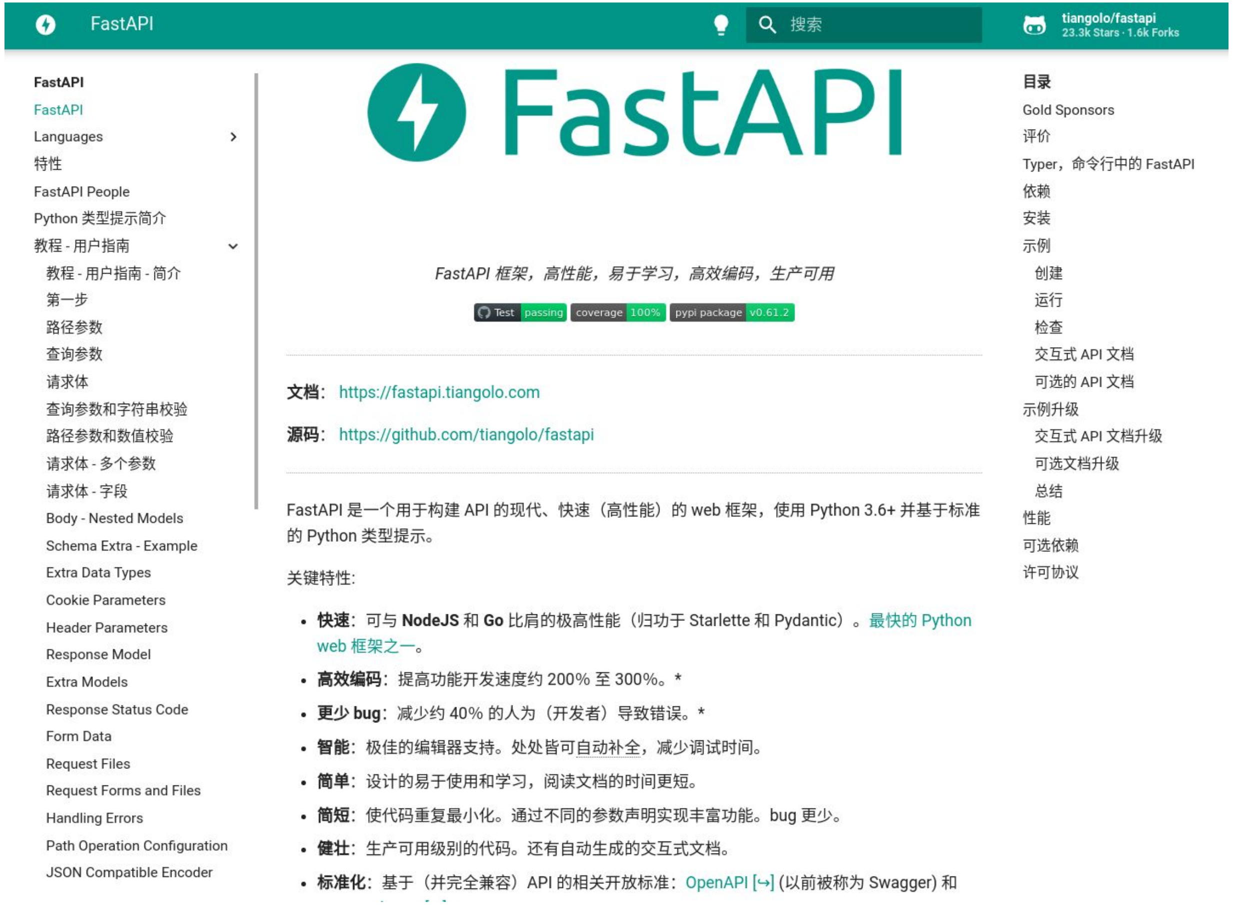
@scoder

👥 Community

- All by normal people 👥
- Mostly during free time ⏰
- Trying to help others ✨

@tiangolo

Community - FastAPI China



FastAPI
FastAPI
Languages
特性
FastAPI People
Python 类型提示简介
教程 - 用户指南
教程 - 用户指南 - 简介
第一步
路径参数
查询参数
请求体
查询参数和字符串校验
路径参数和数值校验
请求体 - 多个参数
请求体 - 字段
Body - Nested Models
Schema Extra - Example
Extra Data Types
Cookie Parameters
Header Parameters
Response Model
Extra Models
Response Status Code
Form Data
Request Files
Request Forms and Files
Handling Errors
Path Operation Configuration
JSON Compatible Encoder

FastAPI

FastAPI 框架，高性能，易于学习，高效编码，生产可用

Test passing | coverage 100% | pypi package v0.61.2

文档： https://fastapi.tiangolo.com

源码： https://github.com/tiangolo/fastapi

FastAPI 是一个用于构建 API 的现代、快速（高性能）的 web 框架，使用 Python 3.6+ 并基于标准的 Python 类型提示。

关键特性:

- **快速**：可与 **NodeJS** 和 **Go** 比肩的极高性能（归功于 Starlette 和 Pydantic）。最快的 Python web 框架之一。
- **高效编码**：提高功能开发速度约 200% 至 300%。*
- **更少 bug**：减少约 40% 的人为（开发者）导致错误。*
- **智能**：极佳的编辑器支持。处处皆可自动补全，减少调试时间。
- **简单**：设计的易于使用和学习，阅读文档的时间更短。
- **简短**：使代码重复最小化。通过不同的参数声明实现丰富功能。bug 更少。
- **健壮**：生产可用级别的代码。还有自动生成的交互式文档。
- **标准化**：基于（并完全兼容）API 的相关开放标准：OpenAPI [↪] (以前被称为 Swagger) 和

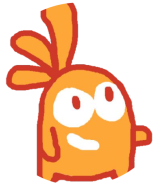
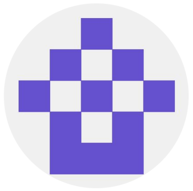
目录
Gold Sponsors
评价
Typer，命令行中的 FastAPI
依赖
安装
示例
创建
运行
检查
交互式 API 文档
可选的 API 文档
示例升级
交互式 API 文档升级
可选文档升级
总结
性能
可选依赖
许可协议

@tiangolo

👥 **Community** - FastAPI China

https://fastapi.tiangolo.com/fastapi-people/

China - FastAPI Experts, Translation contributors, Translation reviewers:

					
Henry Pan	**Dustyposa**	**Xie Wei**	**Laineyzhang55**	**yanever**	**Ikkyu**
@phy25	@Dustyposa	@waynerv	@Laineyzhang55	@yanever	@RunningIkkyu

@tiangolo

👥 **Community** - FastAPI China

You can help too! 🚀

- Translate a page 👥
- Review translations from others 🔍
- Answer questions from others 🗣️
- Help with bugs and features 🤓
- Help with other projects 👩‍💻👨‍💻

https://fastapi.tiangolo.com/help-fastapi/

https://fastapi.tiangolo.com/contributing/#translations

Thank you!

fastapi.tiangolo.com

Sebastián Ramírez

tiangolo.com

github.com/tiangolo

linkedin.com/in/tiangolo

twitter.com/tiangolo