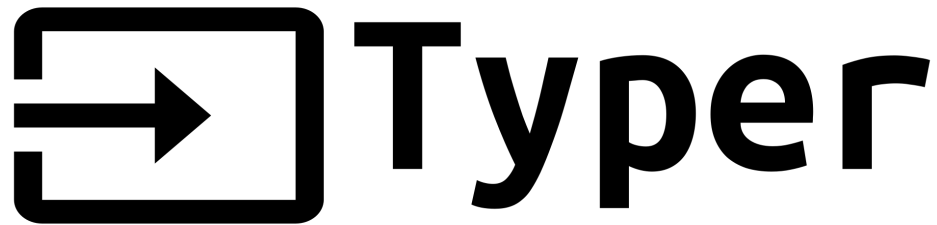


CLI apps with



type annotations and shell completion

Who am I?

Sebastián Ramírez

tiangolo.com

Berlin, Germany



github.com/tiangolo



linkedin.com/in/tiangolo



twitter.com/tiangolo

Senior Staff Software Engineer at



External Consultant
for other teams

I created:



FastAPI



Typer



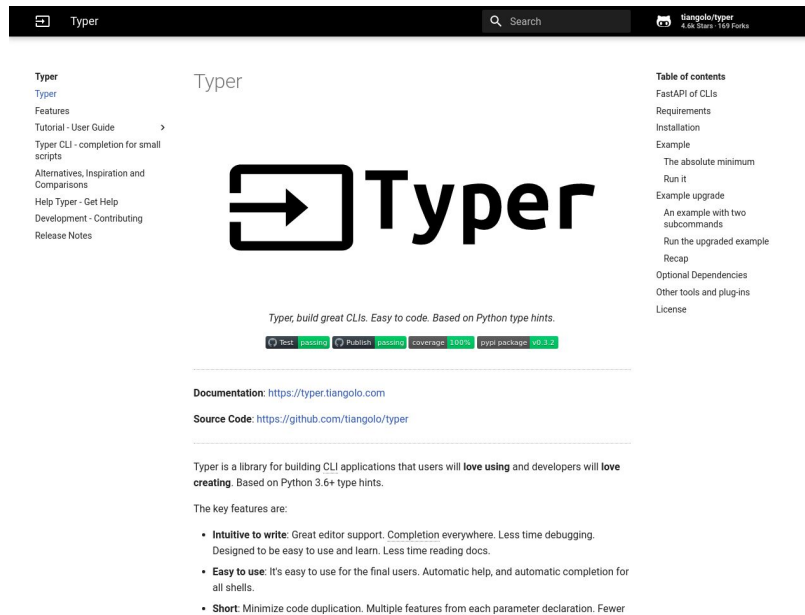
SQLModel



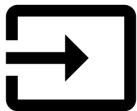
Asyncer

About Typer

- 8K+ GitHub stars
- **FastAPI's** little sibling
- Powered by Click

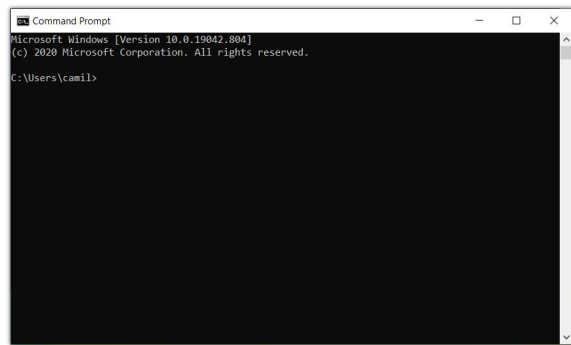
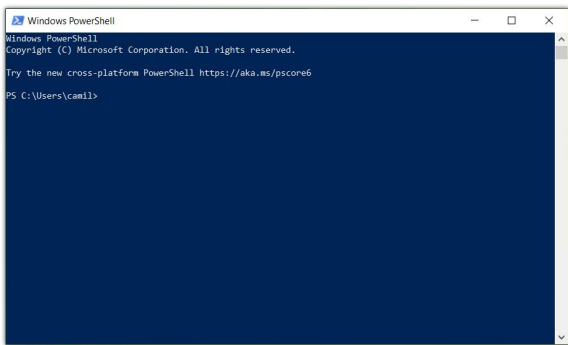
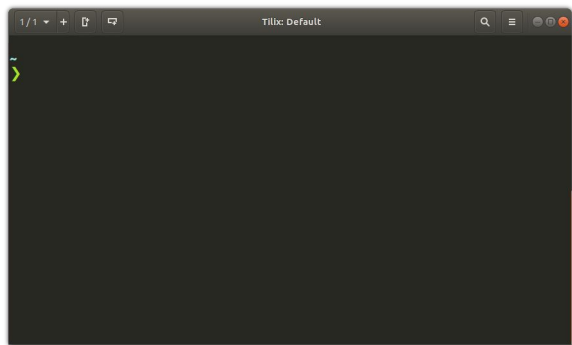
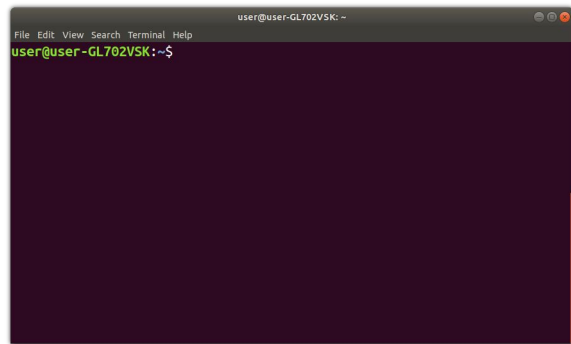


The screenshot shows the GitHub repository page for 'tiangolo/typer'. The page has a dark theme. At the top, there's a header with the repository name 'Typer', a search bar, and the repository statistics: '4.6k Stars · 169 Forks'. Below the header, the left sidebar contains a navigation menu with links: 'Typer', 'Features', 'Tutorial - User Guide', 'Typer CLI - completion for small scripts', 'Alternatives, Inspiration and Comparisons', 'Help Typer - Get Help', 'Development - Contributing', and 'Release Notes'. The main content area features the Typer logo, which consists of a square icon with a right-pointing arrow inside, followed by the word 'Typer' in a large, bold, sans-serif font. Below the logo, there's a tagline: 'Typer, build great CLIs. Easy to code. Based on Python type hints.' followed by a row of badges: 'Tests: passing', 'Publish: passing', 'Coverage: 100%', 'PyPI package: v0.3.1'. Below this, there are links for 'Documentation: https://typer.tiangolo.com' and 'Source Code: https://github.com/tiangolo/typer'. The bottom section of the page describes Typer as a library for building CLI applications that users will love using and developers will love creating, based on Python 3.6+ type hints. It lists key features: 'Intuitive to write' (Great editor support, Completion everywhere, Less time debugging, Designed to be easy to use and learn, Less time reading docs), 'Easy to use' (It's easy to use for the final users, Automatic help, and automatic completion for all shells), and 'Short' (Minimize code duplication, Multiple features from each parameter declaration, Fewer).



Command Line Interfaces (CLI) or Terminals

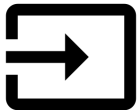
- Text based
- You send program and parameters
- Computer goes beep boop
- Maybe output
- Examples of CLI apps: git, ssh, pip



Simple function with type annotations

```
from typing import Optional

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")
```

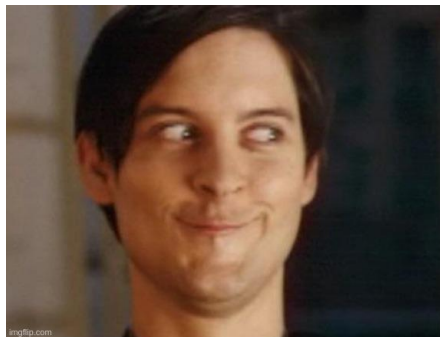


Simple function with type annotations

```
[2] main("Peter")
```



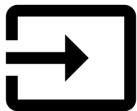
```
Hello Peter!
```



```
main(name="Logan", age=197)
```

```
Hello Logan!
```

```
It seems you are 197
```



Why type annotations - error checks

```
from typing import Optional

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age > 18:
        print("Welcome to the adults section")
```

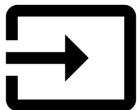
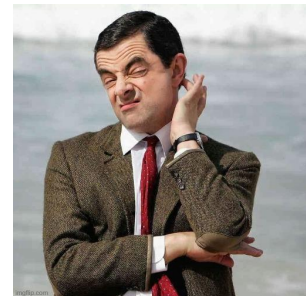


```
from typing import Optional
def main(name: str, age: int | None):
    print(f"Hello {name}!")
    if age > 18:
        print("Welcome to the adults section")
```

Left operand is of type "Optional[int]" mypy(note)

Peek Problem (Alt+F8) No quick fixes available

Unsupported operand types for < ("int" and "None") mypy(error)



Why type annotations - completion

```
1 from typing import Optional
2
3
4 def main(name: str, age: Optional[int] = None):
5     capitalized_name = name.cap
```

capitalize: () -> str × **capitalize**

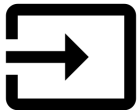
Return a capitalized version of the string.

More specifically, make the first character have upper case and the rest lower case.



```
from typing import Optional

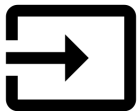
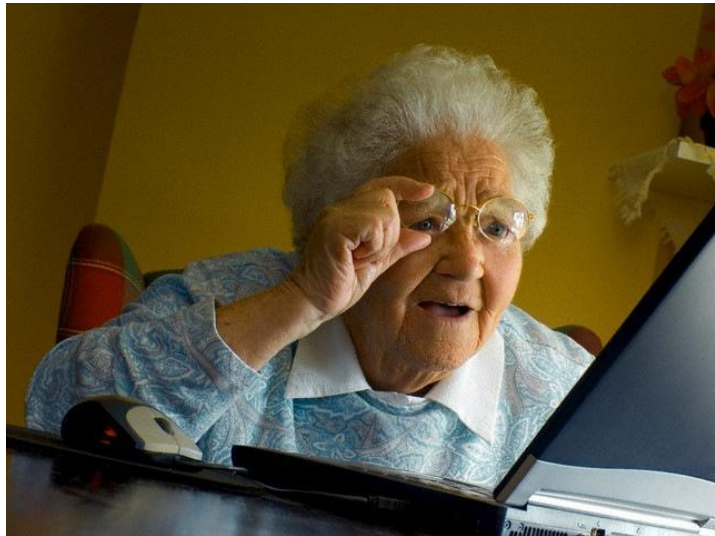
def main(name: str, age: Optional[int] = None):
    capitalized_name = name.capitalize()
    print(f"Hello {capitalized_name}!")
    if age:
        print(f"It seems you are {age}")
```



Simple function with type annotations - recap

```
from typing import Optional

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")
```

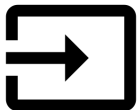
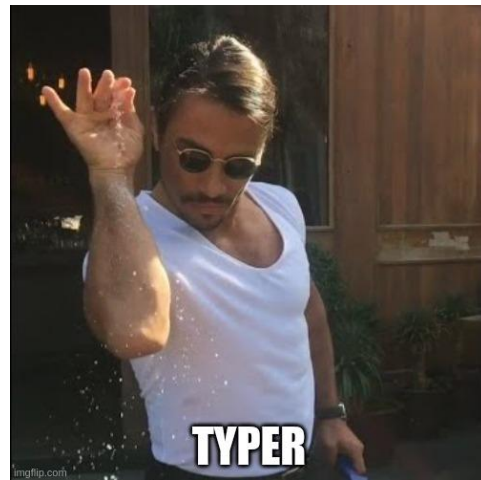


Simple scripts with **Typer**

```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```



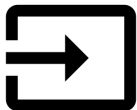
Simple scripts - required arguments

```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```

```
> python main.py Peter
Hello Peter!
```



Simple scripts with **Typer** - main block

```
from typing import Optional
import typer

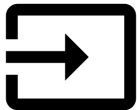
def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```

```
> python main.py Peter
Hello Peter!
```

```
from main import main

main(name="Wade")
```



Simple scripts - required arguments missing

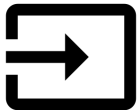
```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```

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

Error
Missing argument 'NAME'.
```



Simple scripts with --help

```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```



```
> python main.py --help
```

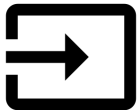
Usage: main.py [OPTIONS] NAME

Arguments

* name TEXT [default: None] [required]

Options

--age INTEGER [default: None]
--help Show this message and exit.



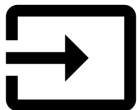
Simple scripts - options

```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```

```
> python main.py Logan --age 197
Hello Logan!
It seems you are 197
```



Simple scripts - invalid data

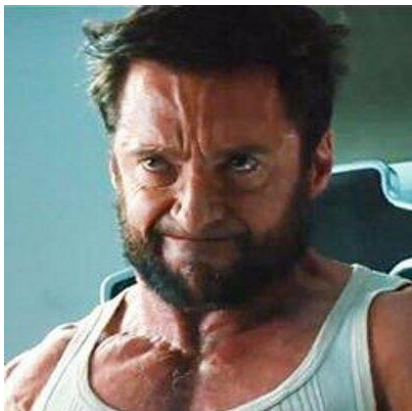
```
> python main.py Logan --age old
```

```
Usage: main.py [OPTIONS] NAME
```

```
Try 'main.py --help' for help.
```

Error

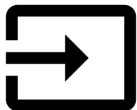
Invalid value for '--age': 'old' is not a valid integer.



```
from typing import Optional
import typer
```

```
def main(name: str, age: Optional[int] = None):
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")
```

```
if __name__ == "__main__":
    typer.run(main)
```



Simple scripts - more help

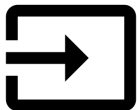
```
from typing import Optional
import typer

def main(name: str, age: Optional[int] = None):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```



Simple scripts - more help in the terminal

```
> python main.py --help
```

Usage: main.py [OPTIONS] NAME

Say hi to the user.

If the age is provided, mention it politely.

Then run away fast, in case the user is offended.

Arguments

* name **TEXT** [default: None] [required]

Options

--age **INTEGER** [default: None]

--help Show this message and exit.



Simple scripts - help for parameters

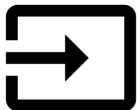
```
from typing import Optional
import typer

def main(
    name: str = typer.Argument(..., help="The name of the user"),
    age: Optional[int] = typer.Option(None, help="The age of the user"),
):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    typer.run(main)
```



Simple scripts - help for parameters in the terminal

```
> python main.py --help
```

Usage: main.py [OPTIONS] NAME

Say hi to the user.

If the age is provided, mention it politely.

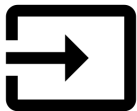
Then run away fast, in case the user is offended.

Arguments

*	name	TEXT	The name of the user [default: None] [required]
---	------	-------------	--

Options

--age	INTEGER	The age of the user [default: None]
--help		Show this message and exit.



A Typer app

```
from typing import Optional
import typer

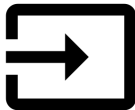
app = typer.Typer()

@app.command()
def main(
    name: str = typer.Argument(..., help="The name of the user"),
    age: Optional[int] = typer.Option(None, help="The age of the user"),
):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    app()
```



A Typer app - app object

```
from typing import Optional
import typer

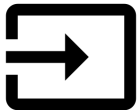
app = typer.Typer()

@app.command()
def main(
    name: str = typer.Argument(..., help="The name of the user"),
    age: Optional[int] = typer.Option(None, help="The age of the user"),
):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    app()
```



A Typer app - decorator

```
from typing import Optional
import typer

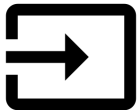
app = typer.Typer()

@app.command()
def main(
    name: str = typer.Argument(..., help="The name of the user"),
    age: Optional[int] = typer.Option(None, help="The age of the user"),
):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    app()
```



A Typer app - call the app

```
from typing import Optional
import typer

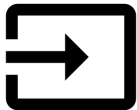
app = typer.Typer()

@app.command()
def main(
    name: str = typer.Argument(..., help="The name of the user"),
    age: Optional[int] = typer.Option(None, help="The age of the user"),
):
    """
    Say hi to the user.

    If the age is provided, mention it politely.

    Then run away fast, in case the user is offended.
    """
    print(f"Hello {name}!")
    if age:
        print(f"It seems you are {age}")

if __name__ == "__main__":
    app()
```



A Typer app - supergreet

```
> supergreet --help
```

Usage: supergreet [OPTIONS] NAME

Say hi to the user.

If the age is provided, mention it politely.

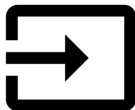
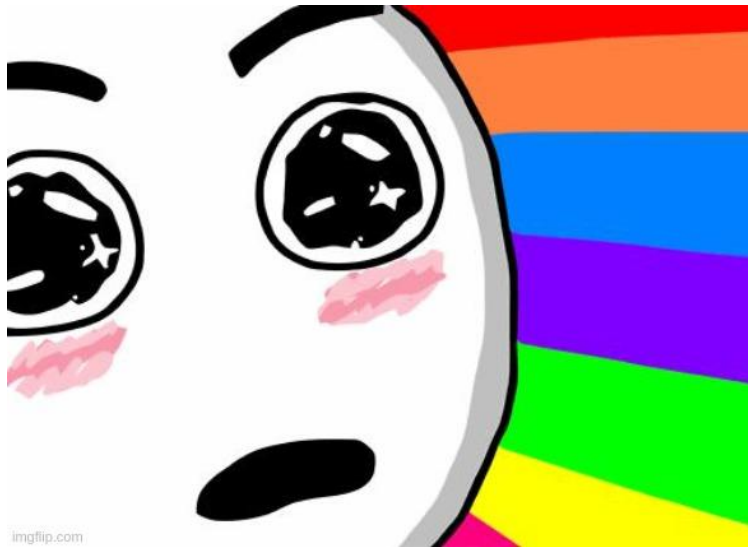
Then run away fast, in case the user is offended.

Arguments

* name **TEXT** The name of the user [default: None] **[required]**

Options

--age **INTEGER** The age of the user [default: None]
--install-completion Install completion for the current shell.
--show-completion Show completion for the current shell, to copy it or customize the installation.
--help Show this message and exit.

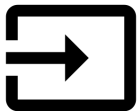


Install completion

```
> supergreet --install-completion  
zsh completion installed in /home/user/.zfunc/_supergreet  
Completion will take effect once you restart the terminal
```

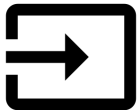
✨ Works for everything:

- Zsh
- Bash
- Fish
- PowerShell



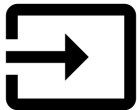
Use completion

```
> supergreet Logan --  
--age                -- The age of the user  
--install-completion -- Install completion for the specified she  
--show-completion    -- Show completion for the specified shell,
```



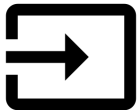
Use completion

```
> supergreet Logan --age
--age                -- The age of the user
--install-completion -- Install completion for the specified she
--show-completion    -- Show completion for the specified shell,
```



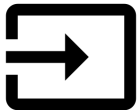
Use completion

```
> supergreet Logan --install-completion  
--age                -- The age of the user  
--install-completion -- Install completion for the specified she  
--show-completion    -- Show completion for the specified shell,
```



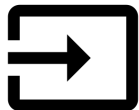
Use completion

```
> supergreet Logan --age
--age                -- The age of the user
--install-completion -- Install completion for the specified she
--show-completion    -- Show completion for the specified shell,
```



Use completion

```
> supergreet Logan --age 197  
Hello Logan!  
It seems you are 197
```



Completion

🧑‍🔧 Works by default and can be used for:

- Deeply nested commands
- CLI Options and CLI Arguments
- Custom data (e.g. remote, from an API)
- etc.



Commands

```
import typer

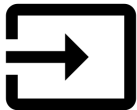
app = typer.Typer()

@app.command()
def hi(name: str):
    """
    Greet the user.
    """
    print(f"Hi {name}")

@app.command()
def bye(name: str, formal: bool = False):
    """
    Say bye to the user.

    Maybe formally. Then cry for the loss (of the leaving user). 🥲
    """
    if formal:
        print(f"Farewell dear {name}")
    else:
        print(f"Bye {name}")

if __name__ == "__main__":
    app()
```



Commands - hi

```
import typer

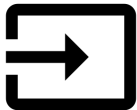
app = typer.Typer()

@app.command()
def hi(name: str):
    """
    Greet the user.
    """
    print(f"Hi {name}")

@app.command()
def bye(name: str, formal: bool = False):
    """
    Say bye to the user.

    Maybe formally. Then cry for the loss (of the leaving user). 🥲
    """
    if formal:
        print(f"Farewell dear {name}")
    else:
        print(f"Bye {name}")

if __name__ == "__main__":
    app()
```



Commands - bye

```
import typer

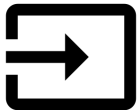
app = typer.Typer()

@app.command()
def hi(name: str):
    """
    Greet the user.
    """
    print(f"Hi {name}")

@app.command()
def bye(name: str, formal: bool = False):
    """
    Say bye to the user.

    Maybe formally. Then cry for the loss (of the leaving user). 😭
    """
    if formal:
        print(f"Farewell dear {name}")
    else:
        print(f"Bye {name}")

if __name__ == "__main__":
    app()
```



Help with commands

```
> supergreet --help
```

```
Usage: supergreet [OPTIONS] COMMAND [ARGS]...
```

Options

<code>--install-completion</code>	Install completion for the current shell.
<code>--show-completion</code>	Show completion for the current shell, to copy it or customize the installation.
<code>--help</code>	Show this message and exit.

Commands

<code>bye</code>	Say bye to the user.
<code>hi</code>	Greet the user.



Commands - hi again

```
import typer

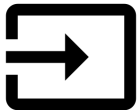
app = typer.Typer()

@app.command()
def hi(name: str):
    """
    Greet the user.
    """
    print(f"Hi {name}")

@app.command()
def bye(name: str, formal: bool = False):
    """
    Say bye to the user.

    Maybe formally. Then cry for the loss (of the leaving user). 🥲
    """
    if formal:
        print(f"Farewell dear {name}")
    else:
        print(f"Bye {name}")

if __name__ == "__main__":
    app()
```



Commands --help for hi

```
> supergreet hi --help
```

Usage: supergreet hi [OPTIONS] NAME

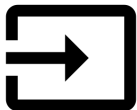
Greet the user.

Arguments

* name TEXT [default: None] [required]

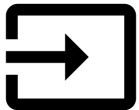
Options

--help Show this message and exit.



Command hi in terminal

```
> supergreet hi Peter  
Hi Peter
```



Commands - bye

```
import typer

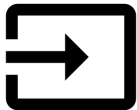
app = typer.Typer()

@app.command()
def hi(name: str):
    """
    Greet the user.
    """
    print(f"Hi {name}")

@app.command()
def bye(name: str, formal: bool = False):
    """
    Say bye to the user.

    Maybe formally. Then cry for the loss (of the leaving user). 😭
    """
    if formal:
        print(f"Farewell dear {name}")
    else:
        print(f"Bye {name}")

if __name__ == "__main__":
    app()
```



Commands --help for bye

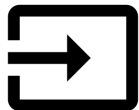
```
> supergreet bye --help

Usage: supergreet bye [OPTIONS] NAME

Say bye to the user.
Maybe formally. Then cry for the loss (of the leaving user). 😭

Arguments
*   name      TEXT [default: None] [required]

Options
--formal      --no-formal [default: no-formal]
--help        Show this message and exit.
```



More Typer features

- 🏠 Boolean flags
- ? Prompts
- 🗝️ Ask passwords
- 🚀 Launch apps (e.g. a browser)
- ✅ Easy testing
- ✨ More...





Sebastián Ramírez

tiangolo.com

Thank you!

typer.tiangolo.com



github.com/tiangolo



linkedin.com/in/tiangolo



twitter.com/tiangolo

